Руководство пользователя Free Pascal 2.4.2

Версия документа 2.4

Ноябрь 2010 (оригинал)
Октябрь 2011 (перевод на русский язык)

Информация об авторе:

Автор перевода:	Поляков Андрей Валерьевич
Web:	http://info-master.su
e-mail:	mail@info-master.su
Блог:	av-inf.blogspot.ru
В контакте:	vk.com/id185471101
Фейсбук:	facebook.com/100008480927503

Страница документа: http://av-mag.ru/doc/fpc-user-manual.htm

СОДЕРЖАНИЕ

ПРАВОВЫЕ ПОЛОЖЕНИЯ

Этот документ является бесплатным. Однако **это не означает**, что Вы можете делать с ним всё, что угодно. Приобретая этот документ, Вы автоматически соглашаетесь с правилами, описанными далее в этом разделе, а также в ЛИЦЕНЗИОННОМ СОГЛАШЕНИИ. Если Вы не согласны хотя бы с одним из этих правил - откажитесь от использования этого документа и удалите все копии документа со своего компьютера и всех Ваших носителей информации (включая облачные сервисы в Интернете).

ВАМ ЗАПРЕЩЕНО

- Продавать документ
- Изменять содержимое документа
- Присваивать авторство документа
- Использовать документ или любые его части в своих книгах или других инфо-продуктах без согласования с автором

ВАМ РАЗРЕШЕНО

- Использовать документ в личных целях (для самообразования и развития)
- Использовать цитаты из документа (объёмом не более 1-2 абзацев) на своих Интернет-ресурсах, книгах или в других инфо-продуктах **с обязательной ссылкой на первоисточник** (то есть на полное наименование документа и на официальный сайт документа: http://av-mag.ru/doc/fpc-user-manual.htm)
- Распространять документ любыми бесплатными способами (кроме спама)
- Включать в качестве подарка в комплект своих платных продуктов
- Раздавать за подписку (без нарушения правил данного раздела)

ОТВЕТСТВЕННОСТЬ АВТОРА

Автор гарантирует, что содержимое данного документа максимально соответствует оригиналу на английском языке (за исключением предисловия и введения).

Автор не предоставляет Вам никаких гарантий, явных или подразумеваемых, что документ не содержит ошибок, что он будет отвечать Вашим требованиям или ожиданиям, будет соответствовать Вашим целям и задачам.

Ни автор, ни другие юридические или физические лица, имеющие отношение к созданию, производству или распространению документа, не несут ответственности за прямые или косвенные убытки, которые могут возникнуть вследствие использования или невозможности использования документа, даже если Вы уведомили представителя автора о возможности возникновения таких убытков. Никакое другое письменное или устное соглашение, предоставленное Вам, не может расширить границы этой ответственности.

ПРЕДИСЛОВИЕ

ВАЖНО!

Этот документ является переводом официальной документации Free Pascal. Документ рассчитан на читателей, которые обладают базовыми знаниями о программировании. Если вы себя к таковым не относите, то есть если вы совсем ничего не понимаете в программировании, рекомендую предварительно прочитать книгу Основы программирования.

ПРИМЕЧАНИЕ

Перевод выполнен максимально близко к оригиналу. Однако в процессе работы автор перевода обнаружил несколько ошибок и печаток в оригинале. Поэтому текст перевода не на 100% соответствует содержимому оригинала.

ВАЖНО!

Для изучения программирования НЕ обязательно знать английский язык. Но очень желательно, поскольку это облегчает изучение. Поэтому, если вы не дружите с английским языком, то я рекомендую вам пройти хотя бы базовый бесплатный курс. Подробности см. здесь: http://info-master.su/mail/eng/.

О ссылках

В этом документе довольно часто будут встречаться ссылки, которые могут вести на сайт в Интернете или на какой-то раздел в самом документе.

Ссылки выглядят вот так:

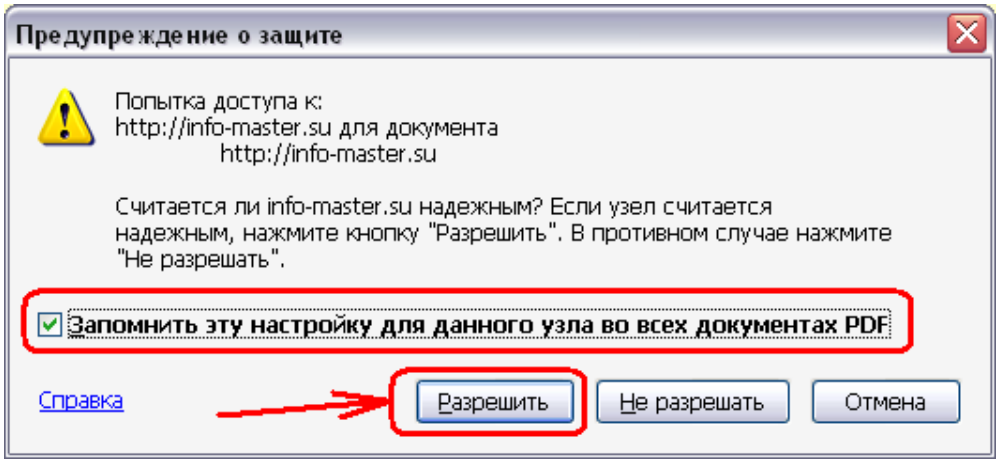
Основы программирования

или так:

http://info-master.su/programming/

Если вы щёлкните левой кнопкой мыши по такой ссылке, то вы перейдёте по соответствующему адресу.

Если ссылка ведёт на страницу в Интернете, то программа, с помощью которой вы читаете эту книгу, может запросить у вас разрешения на переход. В этом случае ответьте ДА (или YES, или OK, или Allow, или РАЗРЕШИТЬ, или что-то типа того – см. рисунок ниже).



Оглавление книги – это тоже ссылки. Если вы щёлкните по названию раздела в оглавлении, то вы перейдёте в соответствующий раздел.

Если ваша программа просмотра этой книги почему-то не даёт перейти по ссылке, то в случае, если ссылка выглядит как адрес в Интернете (например, так: http://info-master.su/programming/), вы можете просто скопировать её и вставить в адресную строку браузера или вручную переписать эту ссылку в адресной строке браузера, а затем нажать на клавишу ENTER.

Если вы плохо понимаете, о чём я говорил в предыдущем абзаце, то рекомендую почитать бесплатную книгу «Чайникам о компьютерах» здесь: http://av-mag.ru/user-comp/index.htm.

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Об этом документе

Этот документ представляет собой руководство пользователя Free Pascal.

Документ описывает установку и использование компилятора Free Pascal на различные поддерживаемые платформы. Документ не предоставляет исчерпывающий список всех поддерживаемых команд, и не описывает язык программирования Паскаль. Описание внутренней работы и возможностей компилятора см. в документе «Руководство программиста». В приложениях к этому документу вы найдёте список зарезервированных слов и сообщений об ошибках компиляции (с описаниями).

Документ описывает, как работать с компилятором. Первые рекомендации см. в файлах README и FAQ. Если информация в файлах README и FAQ расходится с информацией этого документа, то приоритет остаётся за файлами README и FAQ.

1.2. О компиляторе

Free Pascal – это 32 и 64-битный компилятор языка Паскаль. Текущая версия (2.2) может компилировать код для следующих процессоров:

- Intel i386 и выше (i486, семейство Pentium и выше)
- AMD64/x86_64
- PowerPC
- PowerPC64
- SPARC
- ARM
- Процессор m68K поддерживается старшими версиями.

Компилятор и библиотека времени выполнения (RTL – Run-Time Library) доступны для следующих операционных систем:

- DOS
- LINUX
- AMIGA (только версия 0.99.5)
- WINDOWS
- Mac OS X
- OS/2 (при использовании дополнительного пакета EMX, это также справедливо для DOS/Windows)
- FREEBSD
- BEOS
- SOLARIS
- NETBSD
- NETWARE
- OPENBSD
- MorphOS
- Symbian

Полный список всех версий доступен на сайте Free Pascal.

Насколько это возможно, Free Pascal был разработан максимально совместимым с Turbo Pascal 7.0 и Delphi 7 (несмотря на это, совместимость не является основной целью), но он также расширяет возможности этих языков благодаря таким элементам, как перегрузка операций. И, в отличие от своих прототипов, Free Pascal поддерживает множество платформ, то есть является кросс-платформенным.

Он также отличается тем, что вы не можете использовать модули, откомпилированные на одной системе в других системах, то есть вы не можете использовать откомпилированные модули Турбо Паскаль.

Также имеется текстовая версия Интегрированной Среды Разработки (Integrated Development Environment – IDE), доступная для Free Pascal. Пользователи, предпочитающие визуальную среду разработки, могут использовать Lazarus или MSIDE.

Free Pascal состоит из нескольких частей:

1. Компилятор.
2. Библиотека времени выполнения (RTL – Run-Time Library).
3. Пакеты. Это коллекция множества полезных модулей, основанных в целом на Windows 32 API и интерфейсе GTK-2.
4. Бесплатная библиотека компонентов (Free Component Library – FCL). Это набор основанных на классах полезных моделей, которые позволяют получить доступ к базам данных, обеспечивают поддержку изображений, Интернета, XML и т.п.
5. Полезные программы и модули.

Для работы с компилятором вам необходимы только первые две части. В этом документе описано использование компилятора и утилит. Язык программирования Паскаль описан в документе «Справочные материалы», а доступные процедуры и модули RTL и FCL описаны в соответствующих справочных руководствах.

1.3. Как получить больше информации

Если в документации вы не нашли ответы на все ваши вопросы, вы можете получить больше информации в Интернете по следующим адресам:

- http://www.freepascal.org – это основной сайт. Здесь содержатся адреса электронной почты и ссылки на другие источники
- http://community.freepascal.org:10000 – это форум, где вы можете задать интересующие вас вопросы.

Кроме этого существуют зеркала данных сайтов.

Наконец, если вы хотите дополнить этот документ, свяжитесь с разработчиком по адресу michael@freepascal.org.

2. УСТАНОВКА КОМПИЛЯТОРА

2.1. Перед установкой: системные требования

2.1.1. Требования к аппаратной части

Для работы компилятора необходим один из следующих процессоров:

1. Процессор Intel 80386 или выше. Сопроцессор не требуется, но отсутствие сопроцессора замедлит работу ваших программ, если вы будете использовать операции с плавающей точкой, так как в этом случае будет использована эмуляция.
2. Процессор AMD64 или EMT64.
3. Процессор PowerPC.
4. Процессор SPARC.
5. Процессор ARM.
6. Старшие версии FPC поддерживают процессор motorola 68000, но они пока не имеют документации.

Требования к памяти и дисковому пространству:

1. ОЗУ – 8 МБ. Этого достаточно для компиляции небольших программ.
2. Большие программы (такие как компиляторы) потребуют не менее 64 МБ памяти, но рекомендуется не менее 128 МБ (учтите, что компилируемые программы не требуют так много памяти для себя).
3. Не менее 80 МБ свободного места на диске, если исходные коды уже установлены. Иначе потребуется 270 МБ.

2.1.2. Требования к программной части

Для DOS

Дистрибутив для ДОС содержит в себе все файлы, которые требуются для работы компилятора и компиляции ваших программ.

Для UNIX

Для UNIX-подобных операционных систем (таких как LINUX или FreeBSD) вам потребуется установить следующие программы:

1. GNU as – GNU ассемблер.
2. GNU ld – GNU компоновщик.
3. Не обязательно (но настоятельно рекомендуется): GNU make. Для облегчения рекомпиляции библиотеки времени выполнения (Run-Time Library).

Для Windows

Дистрибутив Windows (как для 32-х, так и для 64-разрядной) содержит в себе все файлы, которые требуются для работы компилятора и компиляции ваших программ. Однако не помешает установить инструменты разработчика `mingw32` или `cygwin`, которые можно найти на сайте http://www.freepascal.org.

Для OS/2

Дистрибутив Free Pascal для OS/2 содержит в себе все файлы, которые требуются для работы компилятора и компиляции ваших программ. Однако не помешает дополнительно установить EMX для компиляции и запуска программ с компилятором Free Pascal. EMX можно найти здесь: ftp://hobbes.nmsu.edu/pub/os2/dev/emx/v0.9d.

Для Mac OS X

Потребуется ОС Mac OS X 10.1 или выше. Также должны быть установлены инструменты разработчика или XCode.

2.2. Установка компилятора

Установка компилятора Free Pascal не является сложной, но имеет отличия в зависимости от платформы.

2.2.1. Установка на Windows

Для установки компилятора на ОС Windows запустите программу установки `setup.exe`. Это обычная программа установки, которая предлагает обычные опции выбора, такие как выбор каталога установки и выбор компонентов для установки. Дополнительно вы можете определить, какие файлы связывать со средой разработки (`.pp` или `.pas`).

Не рекомендуется устанавливать компилятор в каталог, если в пути к каталогу имеются пробелы, так как некоторые внешние инструменты не поддерживают имена файлов с пробелами, и вы будете иметь проблемы при создании программ.

2.2.2. Установка на DOS или OS/2

Обязательные действия при установке

Сначала вы должны получить последнюю версию дистрибутива Free Pascal. Дистрибутив содержит файлы в zip-архиве, который вам нужно сначала распаковать, или вы можете загрузить компилятор как несколько отдельных файлов. Последнее может оказаться полезным, если у вас медленное соединение с Интернетом или вы хотите установить только выборочные компоненты. Распакованный zip-архив содержит файл установки `INSTALL.EXE`. Вы должны запустить эту программу для установки компилятора.

Экран при установке в DOS или OS/2 показан на рис. 2.1.

Программа установки позволяет выбрать:

- Какие компоненты вы будете устанавливать. Например, нужно ли устанавливать документацию, исходные коды и т.п. Если вы загружали компилятор в виде отдельных файлов, то пункты компонентов, которые вы не загрузили, вы не сможете установить, так как вы не сможете их выбрать.
- Каталог установки, то есть где вы хотите установить компилятор (например, `C:\PP`).

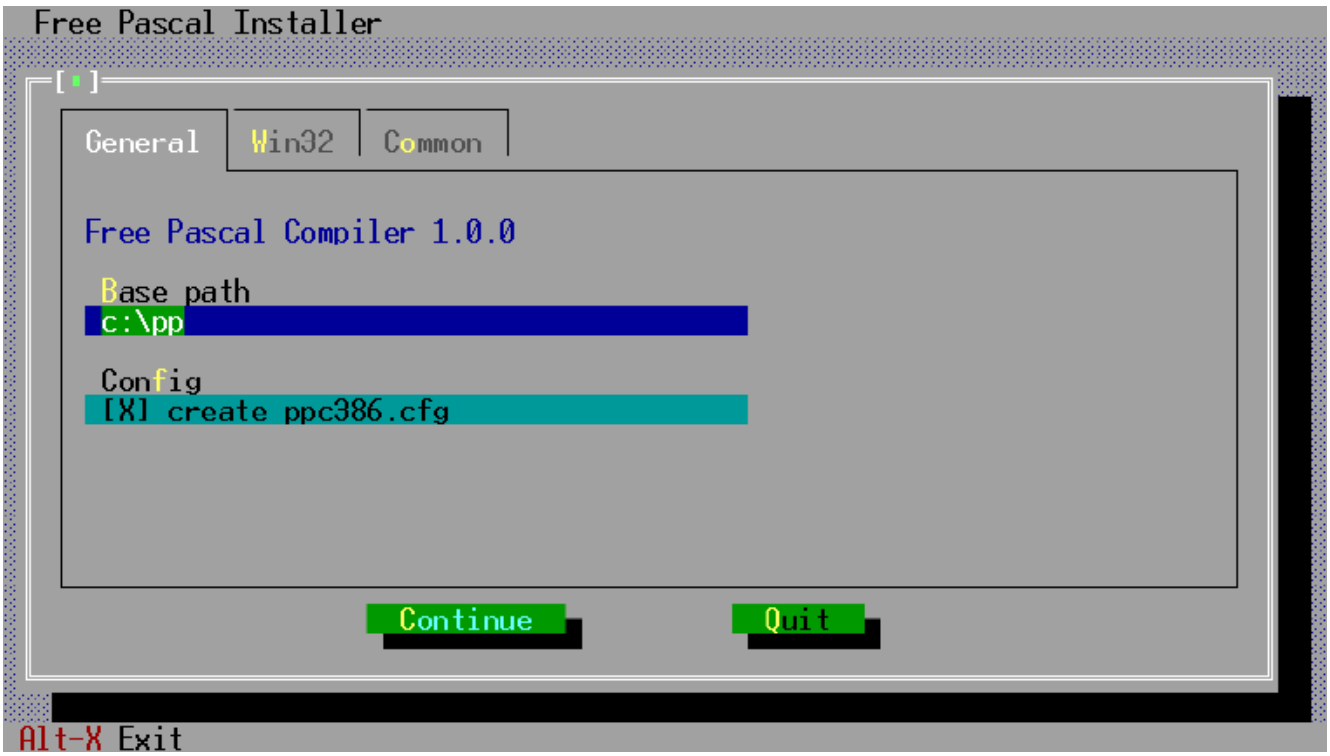


Рис. 2.1. Окно установки Free Pascal для DOS и OS/2.

Чтобы иметь возможность запуска компилятора Free Pascal из любой директории, вы должны к значению переменной `PATH` добавить путь папке, где установлен компилятор, которая находится в каталоге установки, например, `C:\PP\BIN`. Обычно это делается в файле `AUTOEXEC.BAT` примерно так:

```
SET PATH=%PATH%;C:\PP\2.2\BIN\i386-DOS
```

для DOS или

```
SET PATH=%PATH%;C:\PP\2.2\BIN\i386-OS2
```

для OS/2 (при условии, что вы установили компилятор в каталог по умолчанию).

На OS/2 Free Pascal устанавливает некоторые библиотеки из пакета EMX, если они ещё не установлены (если они будут установлены, то программа установки уведомит вас об этом). Библиотеки размещаются здесь:

```
C:\PP\DLL
```

Имя этого каталога должно быть добавлено в директиву `LIBPATH` файла `config.sys`:

```
LIBPATH=XXX;C:\PP\DLL
```

Разумеется, любые существующие каталоги в директиве `LIBPATH` (включая каталог `XXX` из примера выше) должны быть сохранены.

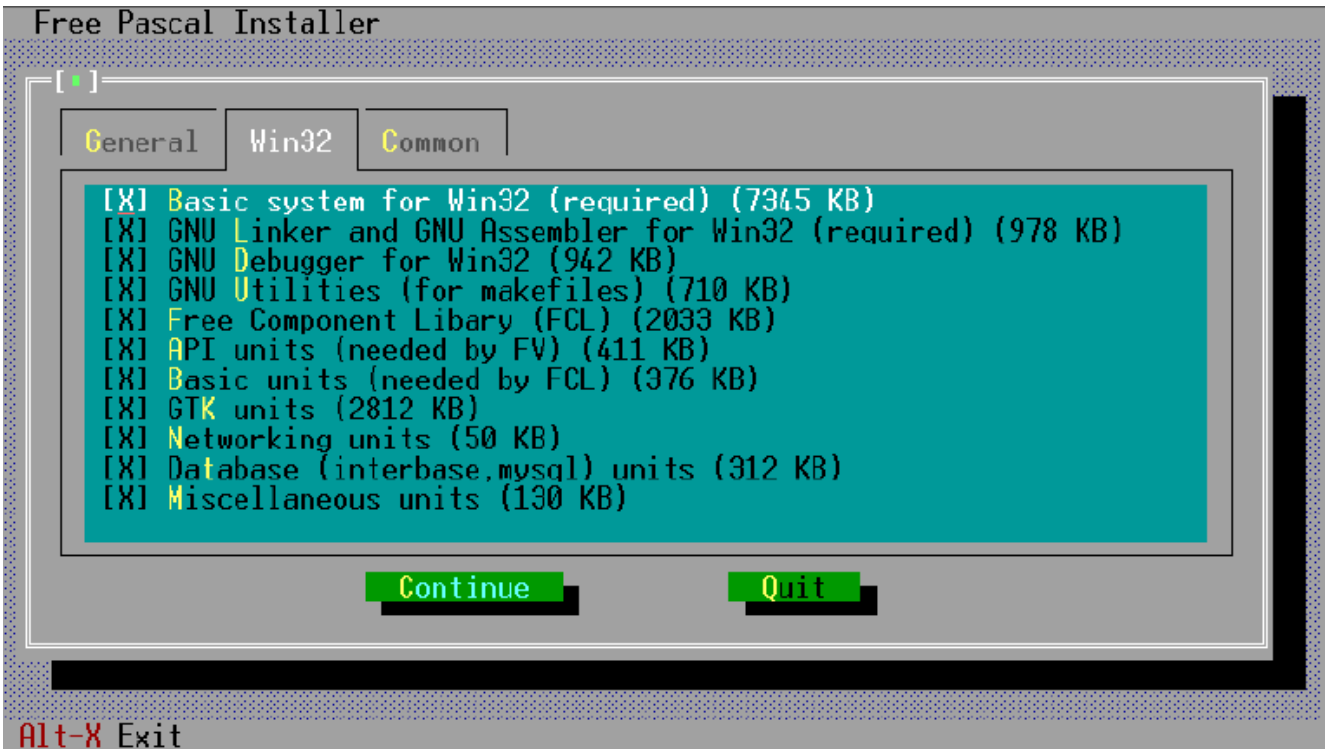


Рис. 2.2. Выбор компонентов Free Pascal для установки.

Опции установки: эмуляция сопроцессора

Если у вас старый процессор, который не имеет математического сопроцессора (i387), то необходимо установить эмулятор сопроцессора, так как Free Pascal использует сопроцессор для всех операций с плавающей точкой.

Установка эмуляции сопроцессора выбирается в программе установки (`INSTALL.EXE`) при установке на DOS и WINDOWS.

2.2.3. Установка на Linux

Обязательные действия при установке

Дистрибутив Free Pascal для Linux может быть трёх видов:

1. `tar.gz` версия, также доступная в виде отдельных файлов
2. `.rpm` (Red Hat Package Manager) версия
3. `.deb` (Debian) версия

Если вы используете формат `.rpm`, то установка сводится к команде:

```
rpm -i fpc-X.Y.Z-N.ARCH.rpm
```

где `X.Y.Z` – это версия файла `.rpm`, а `ARCH` – одна из поддерживаемых архитектур (i386, x86_64 и т.п.).

Если вы используете формат Debian, то установка сводится к команде:

```
dpkg -i fpc-XXX.deb
```

где `XXX` – это версия файла `.deb`.

Для установки вам потребуется доступ с правами пользователя `root`. Файл `.tar` позволит вам выполнить установку в вашу директорию, не имея прав доступа `root`.

Если вы загрузили дистрибутив в виде `.tar`-файла или в виде отдельных файлов, то установка будет более интерактивна.

В случае с `.tar`-файлом вам сначала потребуется распаковать архив в любой каталог, к которому вы имеете доступ. Используйте следующую команду:

```
tar -xvf fpc.tar
```

Здесь предполагается, что вы загрузили файл `fpc.tar` из Интернета (в реальности имя файла будет содержать номер версии, которая в примере не указана для простоты). Когда файл распакован, вам потребуется установить программу при помощи сценария.

Если вы загрузили компилятор в виде отдельных файлов, вам потребуется загрузить скрипт `install.sh` и библиотеки (в `libs.tar.gz`).

Теперь для установки программы всё, что вам потребуется сделать – это выполнить команду:

```
./install.sh
```

Затем вы должны ответить на несколько вопросов. В простейшем случае вы должны уточнить два момента:

1. Куда устанавливать программу
2. Определить, нужно ли устанавливать определённые компоненты (такие как исходные коды и демо-программы)

Сценарий автоматически определит, какие компоненты доступны для установки. Сценарий предлагает для установки только компоненты, которые были найдены. Поэтому вы должны сохранить оригинальные имена загруженных файлов.

Если вы запустили сценарий от имени пользователя `root`, то вы можете оставить все настройки установки по умолчанию. Если же запуск сценария выполнен от имени другого пользователя, то в качестве каталога установки вы должны указать директорию, к которой вы имеете доступ для записи, так как сценарий будет пытаться создавать директории от вашего имени. В принципе, вы можете установить компилятор в любом месте.

По окончании установки, программа инсталляции генерирует файл конфигурации (`fpc.cfg`) для компилятора Free Pascal. Это файл хранит выбранные вами настройки. Этот файл устанавливается в каталог `/etc` или в вашу домашнюю директорию (с именем `.fpc.cfg`), если вы не имеете прав для записи файлов в каталог `/etc`. Сценарий сделает копию в каталог, где вы установили библиотеки.

Компилятор первым делом просматривает файл `.fpc.cfg` в вашей домашней директории, перед тем как начать поиск файла в каталоге `/etc`.

2.3. Дополнительная конфигурация

В любой операционной системе после установки компилятора вы можете настроить несколько переменных среды окружения. Компилятор Free Pascal распознаёт следующие переменные:

- `PPC_EXEC_PATH` содержит имя директории, где могут находиться файлы поддержки компилятора.
- `PPC_CONFIG_PATH` определяет альтернативный путь для поиска файла `fpc.cfg`.
- `PPC_ERROR_FILE` определяет путь и имя файла ошибок.
- `FPCDIR` определяет корневой каталог установки Free Pascal (например : `C:\PP\BIN`).

Эти параметры, однако, устанавливаются в файле конфигурации, который создаётся в конце процесса установки, кроме переменной `PPC_CONFIG_PATH`, которую вы должны изменить самостоятельно, если вы не оставили место установки компилятора по умолчанию.

2.4. Перед компиляцией

В состав дистрибутива входит файл `README`. Он содержит последние инструкции по установке. Рекомендуется прочитать его перед установкой.

Кроме того, особенности установки на различные платформы и общие вопросы описаны в файле `FAQ`. Читайте его по мере необходимости при возникновении проблем.

2.5. Тестирование компилятора

После завершения установки и настройки переменных окружения, вы можете откомпилировать свою первую программу.

Демонстрационные программы, которые входят в дистрибутив Free Pascal**,** показывают, что может делать компилятор. Вы можете проверить правильность работы компилятора, пытаясь откомпилировать эти программы.

Компилятор вызывается следующим образом:

- `fpc.exe` под WINDOWS, OS/2 и DOS.
- `fpc` под большинством других операционных систем.

Для компиляции программы (например, `demo\text\hello.pp`), скопируйте эту программу в вашу рабочую директорию и просто наберите в командной строке:

```
fpc hello
```

Если у вас нет конфигурационного файла, вам может потребоваться указать компилятору местоположение модулей, например так:

```
fpc -Fuc:\pp\NNN\units\i386-go32v2\rtl hello
```

для DOS, а для LINUX вы должны напечатать:

```
fpc -Fu/usr/lib/fpc/NNN/units/i386-linux/rtl hello
```

(замените NNN номером версии Free Pascal, которую вы используете).

Конечно, здесь подразумевается, что вы установили Free Pascal в каталог по умолчанию (`C:\PP` или `/usr/lib/fpc/NNN`).

Если вы не получили сообщений об ошибках, то компилятор сгенерирует и выполнит вызов `hello.exe` в DOS, OS/2 или WINDOWS, или `hello` (без расширения) в UNIX и большинстве других операционных систем.

Для выполнения программы просто напечатайте:

```
hello или ./hello
```

Если всё правильно, вы должны увидеть на экране следующую строку:

```
Hello world
```

3. ИСПОЛЬЗОВАНИЕ КОМПИЛЯТОРА

Здесь описаны основные понятия компилирования программ и модулей. Более подробную информацию см. в разделе о конфигурации компилятора и в документе «Руководство программиста».

Описанные здесь примеры предполагают, что у вас есть правильно установленный файл конфигурации `fpc.cfg`, содержащий минимальные настройки путей для модулей RTL. В принципе, этот файл генерируется при установке программы. Вы можете проверить, что он находится в правильном месте (более подробно см. в разделе 5.2).

3.2. Поиск файлов

Перед началом компиляции программы или серии модулей важно знать, где компилятор ищет исходные файлы и другие файлы. Данный раздел содержит эту информацию и показывает, на что это влияет.

ПРИМЕЧАНИЕ
Используёте слеш (/) и обратный слеш (\) в качестве разделителя каталогов. Компилятор будет конвертировать этот символ в зависимости от операционной системы. Например, использование слеша вызовет проблемы на UNIX-подобных системах (таких как LINUX).

3.1.1. Файлы в командной строке

Файл, который вы ввели в командной строке, такой как

```
fpc foo.pp
```

компилятор будет искать ТОЛЬКО в текущей директории. Если вы указали директорию в имени файла, то компилятор будет просматривать эту директорию:

```
fpc subdir/foo.pp
```

и искать файл `foo.pp` в поддиректории текущей директории.

В чувствительных к регистру операционных системах (таких как LINUX и UNIX), имя файла должно учитывать регистр символов.

Для других операционных систем (таких как DOS, WINDOWS NT, OS/2) регистр символов значения не имеет.

3.1.2. Файлы модулей

Когда вы компилируете модуль или программу, которым требуются другие модули, компилятор будет искать откомпилированные версии модулей по следующим путям:

1. Текущий каталог
2. Каталог, где размещены исходные файлы
3. Каталог, где размещены исполняемые файлы компилятора
4. Каталог, указанный для поиска модулей

Вы можете добавить каталог для поиска модулей с помощью параметра `-Fu` (см. раздел 5.1.3). Каждый элемент этого параметра будет вставлять каталог в список путей поиска модулей таким образом, что последний указанный в командной строке путь будет просматриваться первым.

Компилятор добавляет несколько путей в путь поиска модулей:

1. Содержимое переменной окружения `XXUNITS`, где `XX` должно быть заменено на одно из поддерживаемых значений: GO32V2, LINUX,WIN32, OS2, BEOS, FREEBSD, NETBSD.
2. Стандартный каталог модулей. Этот каталог определяется переменной окружения `FPCDIR`. Если эта переменная не определена, то ей присваивается значение по умолчанию:
    a. Для LINUX:
        i. `/usr/local/lib/fpc/FPCVERSION` или
        ii. `/usr/lib/fpc/FPCVERSION` (какой будет найден первым).
    b. Для других операционных систем:
        i. Каталог исполняемых файлов компилятора с добавлением `'../'` к пути, если путь существует. Для Windows это будет: `C:\FPC\2.2\units\i386-win32` (это справедливо, если компилятор установлен в каталог `C:\FPC\2.2`).

После определения этих каталогов, следующие пути добавляются в путь поиска:

(a) `FPCDIR/units/FPCTARGET`
(b) `FPCDIR/units/FPCTARGET/rtl`

Здесь `TARGET` должно быть заменено на имя цели вашей компиляции – это комбинация процессора и операционной системы, например

```
/usr/local/lib/fpc/2.2/units/i386-linux/
```

или

```
/usr/local/lib/fpc/2.2/units/i386-win32/
```

Параметр `-Fu` принимает одиночный подстановочный символ `*`, который будет заменён всеми найденными каталогами, кроме самого указанного каталога. Например, передать каталоги

```
rtl/units/i386-linux
fcl/units/i386-linux
packages/base
packages/extra
```

можно командой

```
fpc -Fu"*/units/i386-linux"
```

и это будет иметь тот же эффект, что и команда

```
fpc -Furtl/units/i386-linux -Fufcl/units/i386-linux
```

так как оба каталога `rtl` и `fcl` содержат подкаталог `units/i386-linux`. Каталог `packages` не будет добавлен, так как он не содержит подкаталог `units/i386-linux`.

Следующая команда

```
fpc -Fu"units/i386-linux/*"
```

добавляет все подкаталоги, которые имеются в каталоге `units/i386-linux`, но не добавляет сам каталог `units/i386-linux`, поэтому вы должны добавить его вручную, если это потребуется:

```
fpc -Fu"units/i386-linux" -Fu"units/i386-linux/*"
```

Учтите, что (для оптимизации) компилятор будет отбрасывать все несуществующие пути из пути поиска, то есть существование пути (после подстановочного символа и расширения переменной окружения) будет проверяться.

Вы можете посмотреть пути, которые будут проверяться компилятором с помощью параметра `-vu`.

В операционных системах, где имена файлов чувствительны к регистру символов (таких как UNIX и LINUX), компилятор будет:

1. Искать оригинальное имя файла, то есть файл с именем в сохранённом регистре
2. Искать файл с именем в верхнем регистре
3. Искать файл с именем в нижнем регистре

Это необходимо, так как Паскаль нечувствителен к регистру и операторы `Uses Unit1` или `uses unit1` должны иметь одинаковый эффект.

Вначале выполняется поиск файлов с расширением `.ppu` (откомпилированный модуль), `.pp`, а затем с расширением `.pas`.

Например, предположим, что файлу `foo.pp` необходим модуль `bar`. Тогда команда

```
fpc -Fu.. -Fuunits foo.pp
```

укажет компилятору, что нужно искать модуль `bar` в следующих местах:

1. В текущем каталоге.
2. В каталоге, где расположены исполняемые файлы компилятора (не для LINUX).
3. В родительском каталоге текущей директории.
4. В подкаталоге `units` текущей директории
5. В стандартном каталоге модулей

Имена модулей, которые содержат более 8 символов, сначала будут искаться по полному имени. Если по полному имени модуль не будет найден, то имя будет обрезано до 8 символов, и компилятор снова попытается найти модуль во всех указанных директориях, но уже с сокращённым именем файла.

Если компилятор нашёл нужный модуль, он будет просматривать исходный файл этого модуля в той же директории, где находится модуль. Если исходный файл модуля найден, то компилятор сравнивает время последнего изменения файлов. Если исходный файл был изменён позднее, чем файл модуля, то компилятор попытается перекомпилировать модуль.

Если компилятор не нашёл откомпилированную версию модуля или был указан параметр `-B`, то компилятор в той же манере будет искать исходный файл и пытаться перекомпилировать его.

Рекомендуется устанавливать путь поиска модулей в конфигурационном файле `fpc.cfg`. Если вы это сделаете, то вам не придётся определять путь поиска модулей в командной строке каждый раз, когда вы выполняете компиляцию.

3.1.3. Подключаемые файлы

Если вы включаете файл в ваш исходный код `{$I filename}`, то компилятор будет искать этот файл в следующих местах:

1. По пути, указанному в имени файла в директиве.
2. В каталоге, где находится текущий исходный файл.
3. Во всех директориях, определённых в пути поиска подключаемых файлов.

Вы можете добавлять пути поиска подключаемых файлов с помощью параметров `-I` или `-Fi` (см. раздел 5.1.3).

Например, подключить файл `units/bar.inc` можно так:

```
{$i ../bar.inc}
```

Тогда следующая команда:

```
fpc -Iincfiles units/bar.inc
```

укажет компилятору, что файл `bar.inc` нужно искать в следующих директориях:

1. Родительский каталог текущей директории.
2. Подкаталог `units` в текущей директории.
3. Подкаталог `incfiles` в текущей директории.

3.1.4. Объектные файлы

Когда вы компонуете объектные файлы (используя директиву `{$L file.o}`), компилятор будет искать их разными путями, как и в случае с подключаемыми файлами:

1. Путь, указанный в имени объектного файла.
2. Каталог, где расположен текущий исходный файл.
3. Во всех директориях, определённых в пути поиска объектных файлов.

Вы можете добавлять пути поиска объектных файлов с помощью параметра `-Fo` (см. раздел 5.1.3).

3.1.5. Конфигурационный файл

Не обязательно все параметры передавать через командную строку. Компилятор может использовать конфигурационный файл, который может содержать несколько параметров командной строки. В каждой строке конфигурационного файла можно указать только один параметр.

За исключением случая, когда в командной строке указан параметр `-n` (см. раздел 5.1.1), компилятор всегда ищет конфигурационный файл `fpc.cfg` в следующих местах:

Для UNIX (например, LINUX):

1. Текущий каталог.
2. Ваша домашняя директория, определяемая как `.fpc.cfg`.
3. Директория, указанная в переменной окружения `PPC_CONFIG_PATH`, а если не установлена, то директория `etc` выше директории компилятора (например, если компилятор находится в `/usr/local/bin`, то поиск выполняется в `/usr/local/etc`).
4. Директория `/etc`.

Для всех других операционных систем:

1. Текущий каталог.
2. Каталог, указанный в переменной окружения `PPC_CONFIG_PATH` (если установлена).
3. Каталог, где находится компилятор.

Версии компилятора, предшествующие версии 1.0.6, используют конфигурационный файл `ppc386.cfg`. Этот файл до сих пор поддерживается, но использовать его не рекомендуется. Для совместимости вначале выполняется поиск файла `fpc.cfg`, а если он будет не найден, то будет выполнен поиск файла `ppc386.cfg`, который и будет использоваться (если будет найден).

ПРИМЕЧАНИЕ
Поиск файла `ppc386.cfg` будет удалён в версии компилятора 2.4.0. Для того чтобы обратить на это внимание, компилятор версии 2.3.1 выдаёт предупреждение, если используется конфигурационный файл `ppc386.cfg`.

3.1.6. Длинные имена файлов

Компилятор Free Pascal может работать с длинными именами файлов на всех платформах, кроме DOS. На Windows длинные имена поддерживаются, если это разрешено (это не всегда доступно на старых версиях Windows).

Если длинные имена не поддерживаются, то они обрезаются до 8 символов. Не рекомендуется использовать в именах каталогов и файлов пробелы, так как внешний компоновщик GNU не понимает такие имена.

3.2. Компиляция программы

Компиляция программы выполняется очень просто. Допустим, что у вас есть исходный код программы в файле `prog.pp`. Вы можете откомпилировать его следующей командой:

```
fpc [options] prog.pp
```

Квадратные скобки `[ ]` говорят о том, что параметры, находящиеся между ними, не являются обязательными.

Если исходный файл вашей программы имеет расширение `.pp` или `.pas`, вы можете его не указывать, и просто напечатать:

```
fpc [options] prog
```

Если всё выполнено правильно и исходный файл не содержит ошибок, то компилятор создаст исполняемый файл (готовое приложение).

Вы должны учесть, что после компиляции в вашей директории появятся также и другие файлы с расширением `.o`. Это объектные файлы вашей программы. Если программа уже откомпилирована, то вы можете удалить эти файлы. Однако не удаляйте их, если вы компилируете модуль. Это потому, что объектный файл содержит код модуля, и будет скомпонован в любой программе, которая его использует.

3.3. Компиляция модуля

Компиляция модуля не сильно отличается от компиляции программы. Отличие заключается только в том, что при компиляции модуля не вызывается компоновщик. Для компиляции модуля, который находится (например) в файле `foo.pp`, просто напечатайте:

```
fpc foo
```

См. также замечания о расширениях файлов в предыдущем разделе.

Если всё правильно, то вы получите два файла модуля:

1. `foo.ppu` – это файл с описанием откомпилированного вами модуля.
2. `foo.o` – это файл, содержащий актуальный код модуля. Этот файл и является конечным продуктом компиляции.

Оба файла необходимы, если вы планируете использовать модуль в нескольких программах. Не удаляйте их. Если вам нужен модуль для распространения, вы должны предоставлять оба файла – `.ppu` и `.o`. Один файл бесполезен без другого.

3.4. Модули, библиотеки и «умная» компоновка

Компилятор Free Pascal поддерживает «умную» компоновку и создание библиотек. «Умная» компоновка (Smartlink) – это технология, когда к проекту подключается только то, что необходимо, а не модуль целиком. Однако по умолчанию компилятор ведёт себя таким образом, что каждый модуль компилируется в один большой объектный файл, который будет целиком использоваться в вашей программе.

Общедоступные библиотеки могут быть созданы на большинстве платформ, несмотря на то, что версии FPC могут отличаться (например, они не поддерживаются GO32v2 и OS2). Также возможно поместить в библиотеку существующие модули (используя инструмент `ppumove`, см. раздел 8.7).

3.5. Уменьшение размера программы

При создании программы можно уменьшить размер исполняемого файла (вашей программы). Это возможно, поскольку компилятор помещает в программу всю информацию, которая, строго говоря, не всегда нужна для работы программы.

Излишняя информация может быть удалена с помощью программы, которая называется `strip`. Чтобы воспользоваться ею, просто напечатайте

```
strip prog
```

в командной строке и программа `strip` удалить всю ненужную информацию из вашей программы. Это может уменьшить размер до 30%.

Вы можете использовать переключатель `-Xs`, чтобы позволить компилятору автоматически сжимать программу во время компиляции (переключатель не имеет эффекта при компиляции модулей).

Другая технология уменьшения размера программы – это «умная» компоновка. Обычно модули (включая системные) компонуются целиком, то есть в программу включается весь код модуля. Однако возможно выполнить компиляцию таким образом, чтобы в программу были включены только те процедуры и функции модуля, которые используются в программе. Эта технология называется `smartlinking` («умная» компоновка).

Компилятор переключается в режим «умной» компоновки параметром `-XX` (см. раздел5.1.4). Эта технология подробно описана в документе «Руководство программиста».

4. ОШИБКИ КОМПИЛЯЦИИ

4.1. Основные ошибки

IO-error -2 at …

В ОС LINUX вы можете получить это сообщение при старте компилятора. Обычно оно означает, что компилятор не нашёл файл определений ошибок. Вы можете исправить это недоразумение с помощью параметра `-Fr` (см. раздел 5.1.3) для LINUX.

Error : File not found : xxx or Error: couldn't compile unit xxx

Это сообщение обычно появляется, когда путь к вашему модулю неправильно установлен. Помните, что компилятор ищет модули только в текущей директории и в директории, где находится сам компилятор. Если вы хотите, чтобы поиск выполнялся также в других местах, то вы должны явно указать эти места компилятору с помощью параметра `-Fu` (см. раздел 5.1.3). Или потребуется настроить конфигурационный файл.

4.2. Ошибки, которые могут встретиться в DOS

No space in environment

Эта ошибка может появиться, если вы вызываете файл `SET_PP.BAT` из `AUTOEXEC.BAT`. Для решения этой проблемы вы должны увеличить расширенную память для среды окружения. Чтобы это сделать, найдите в файле `CONFIG.SYS` строку, подобную этой:

```
SHELL=C:\DOS\COMMAND.COM
```

и измените её следующим образом

```
SHELL=C:\DOS\COMMAND.COM /E:1024
```

Если этот параметр уже установлен, то вы можете только увеличить это значение.

Coprocessor missing

Если компилятор выдаёт сообщение об отсутствии сопроцессора, установите режим эмуляции сопроцессора.

Not enough DPMI memory

Если вы хотите использовать компилятор с DPMI, вы должны иметь не менее 7-8 MB свободной памяти DPMI, но 16 Mb являются более предпочтительным объёмом.

5. КОНФИГУРАЦИЯ КОМПИЛЯТОРА

Работой компилятора можно управлять разными способами. По сути это два разных способа:

- Использование опций (параметров) командной строки
- Использование конфигурационного файла `fpc.cfg`.

Компилятор вначале читает конфигурационный файл. Только затем проверяются параметры командной строки. Это делает возможным установить несколько основных опций в конфигурационном файле, и тем самым сэкономить ваше время при компиляции нескольких модулей или программ.

Сначала рассмотрим список параметров командной строки, а затем – как устанавливать параметры командной строки в конфигурационном файле.

Читая этот документ, имейте в виду, что параметры командной строки являются чувствительными к регистру, то есть команда `HELP` или `Help` не будет работать, так как существует только команда `help`.

5.1. Использование параметров командной строки

Доступные параметры текущей версии компилятора перечислены по категориям. Также см. приложение A, где представлен список параметров в том виде, как его генерирует компилятор при вызове с параметром `-help`.

5.1.1. Основные параметры

Параметр	Описание	
-h	Выводит список всех параметров и выполняет выход из программы.	
-?	Аналогично параметру –h, но после вывода информации до конца экрана, ожидает нажатия клавиши ВВОД (ENTER) для продолжения.	
-i	Печатает авторские права и другую информацию. Вы можете расширить эту функцию, напечатав –ixxx, где xxx может принимать одно из следующих значений:	
	D	Возвращает дату компилятора.
	V	Возвращает сокращённую версию компилятора.
	W	Возвращает полную версию компилятора.
	SO	Возвращает операционную систему компилятора.
	SP	Возвращает процессор компилятора.
	TO	Возвращает целевую операционную систему.
	TP	Возвращает целевой процессор.
-l	Печатает логотип Free Pascal и номер версии.	
-n	Игнорирует конфигурационный файл при компиляции. Тем не менее вы можете поместить конфигурационный файл опцией @.	

5.1.2. Параметры обратной связи

Эти параметры используются для получения информации о процессе компиляции.

Параметр	Описание	
-vxxx	Вывод подробного описания. Здесь xxx может принимать одно из следующих значений:	
	e	Показывать ошибки. Эта опция установлена по умолчанию.
	i	Показывать общую информацию.
	w	Выдавать предупреждения.
	n	Выдавать замечания.
	h	Выдавать подсказки.
	l	Выдавать отчёт о количестве строк в процессе компиляции (каждые 100 строк).
	u	Показывать информацию о начале загрузки модулей.
	t	Показывать имена открываемых файлов.
	p	Показывать имена компилируемых процедур и функций.
	q	Показывать номера сообщений.
	c	Показывать состояние каждого процесса.
	mxxx	xxx – список сообщений, которые не нужно выводить. Сообщения разделены запятой. Этот параметр может быть определён несколько раз.
	d	Показывать дополнительную отладочную информацию.
	0	Не показывать сообщения. Это может быть полезным, когда приоритет имеют параметры конфигурационного файла.
	x	Показывать информацию о ходе выполнения (только для Win32).
	r	Форматировать ошибки в режиме совместимости с RHIDE/GCC.
	a	Показывать всю возможную информацию (это аналогично установке все параметров).
	b	Выводить в сообщениях полные имена файлов.
	v	Записывать подробную отладочную информацию в файл `fpcdebug.txt`.
	s	Записывать временные отметки. В основном предназначена для разработчиков компилятора.

Различия между понятиями «фатальные ошибки» и «ошибка/подсказка/предупреждение/замечание» следующие:

Термин	Перевод	Описание
Fatal	Фатальная ошибка	Компилятор обнаружил ошибку, и не может дальше продолжить компиляцию. Компиляция будет остановлена.
Error	Ошибка	Компилятор обнаружил ошибку, но может продолжить компиляцию (до конца текущего модуля).
Warning	Предупреждение	Это означает, что вероятно обнаружена ошибка, то есть что-то может быть неправильно в вашем коде.
Hint	Подсказка	Выводится, если компилятор предполагает, что код может быть написан лучше, но нет подозрения на ошибку.
Note	Замечание	Какая-либо заслуживающая внимания информация, но не ошибка.

Различия между подсказками и замечаниями не очень большие. И то и другое можно без риска игнорировать, однако предупреждения всегда нужно проверять.

5.1.3. Параметры, касающиеся файлов и каталогов

Параметр	Описание
-exxx	Здесь **xxx** – это каталог, содержащий исполняемые файлы **as** (Ассемблер) и **ld** (компоновщик).
-FaXYZ	Загружать модули **XYZ** после системных модулей, но перед любыми другими модулями. **XYZ** – это имена модулей, разделённые запятой. Это можно использовать только для программ. Применение параметра имеет тот же эффект, что и вставка имён **XYZ** в исходный код программы в разделе **uses**.
-FcXXX	Устанавливать вход кодовой страницы для XXX. Экспериментальный параметр.
-FCxxx	Установить бинарное имя RC-компилятора (компилятора ресурсов) в XXX.
-Fd	Отключить кэш внутренних каталогов компилятора. По умолчанию компилятор сохраняет имена всех файлов в каталог так быстро, насколько это возможно для одного файла в вышеупомянутой директории. Это гарантирует, что для всех имён файлов используется правильный регистр в отладочной информации, а также для создания откомпилированных файлов на чувствительных к регистру операционных системах и что всё это будет правильно выполняться на всех системах. Однако эта функция может замедлить работу на сетевых файловых системах, особенно когда выполняется компиляция программ в каталогах, содержащих много файлов. Это замедление может быть уменьшено отключением КЭШа с помощью этого переключателя.
-FD	То же самое, что и **–e**.
-Fexxx	Записывать ошибки, например в файл с именем **xxx**.
-FExxx	Сохранять исполняемые файлы и модули в каталог xxx вместо текущего каталога. Если этот параметр следует за параметром **–o** (см. раздел 5.1.4) и содержит путь компонента, то путь, указанный в параметре **–o** перепишет путь, указанный в параметре **–FE**.
-Ffxxx	Добавляет **xxx** в основной путь (только для Darwin).
-Fixxx	Добавляет **xxx** в путь поиска подключаемых файлов.
-Flxxx	Добавляет **xxx** в путь поиска библиотек (это также справедливо для компоновщика).
-FLxxx	Только для LINUX. Использовать **xxx** как динамический компоновщик. По умолчанию это /lib/ld-linux.so.2 или /lib/ldlinux.so.1, устанавливается тот, который найден первым.
-Fmxxx	Загрузить таблицу преобразования Юникод из файла **xxx**.txt в каталог, где размещён компилятор. Используется только совместно с параметром **–Fc**.
-Foxxx	Добавить **xxx** в путь поиска объектных файлов. Этот путь используется при просмотре файлов для компоновки.
-Frxxx	Определить **xxx** как файл, в котором содержатся сообщения компилятора. Установка этого параметра перепишет встроенные по умолчанию сообщения компилятора, которые выводятся на английском языке.
-FRxxx	Установить компоновщик ресурсов (.res) для **xxx**.
-Fuxxx	Добавить **xxx** в путь поиска модулей. Сначала выполняется поиск модулей в текущей директории. Если в текущем каталоге модули не найдены, то компилятор ищет их по пути поиска модулей. Вы должны всегда предоставлять путь к системным модулям. Путь **xxx** может содержать один подстановочный символ (*), который расширяет область поиска модулей. Учтите, что собственное местоположение не включается в список поиска (см. раздел «3.1.2. Файлы модулей»).
-FUxxx	Сохранять модули в каталоге **xxx** вместо текущего каталога. Этот параметр переписывается опцией **–FE**.
-Ixxx	Добавить **xxx** в путь поиска подключаемых файлов. Имеет тот же эффект, что и параметр **–Fi**.
-FWxxx	Записывать всю информацию, которая генерируется при оптимизации программы в файл **xxx**.
-Fwxxx	Читать всю информацию, сгенерированную при оптимизации программы из файла **xxx**.

5.1.4. Параметры, контролирующие результат компиляции

Список параметров приведён ниже. Более подробную информацию см. в документе «Руководство программиста».

Параметр	Описание	
-a	Не удалять файлы ассемблера (не применяется, когда используется встроенный ассемблер). Также применяется (когда возможно) для генерации пакетов сценариев.	
-al	Включать строки исходного кода в файл ассемблера в качестве комментариев.	
-an	Записывать информацию о контрольных точках в файл ассемблера (контрольные точки являются методом представления компилятором операторов или их составных частей). Это в основном предназначено для отладки кода, генерируемого компилятором.	
-ap	Использовать каналы вместо создания временных файлов ассемблера. Это может ускорить компиляцию на операционных системах OS/2 и LINUX, при условии, что используется ассемблер (такой как GNU, если внутренний ассемблер используется).	
-ar	Записывать информацию о заполнении и освобождении регистров в файл ассемблера. Это в основном предназначено для отладки кода, генерируемого компилятором.	
-at	Записывать информацию о временном заполнении и освобождении ресурсов в файл ассемблера.	
-Axxx	Указать, какого типа ассемблер должен генерироваться. Здесь **xxx** может принимать значения:	
	default	Использовать встроенный ассемблер по умолчанию.
	as	Ассемблер, использующий GNU as.
	nasmcoff	Файл Coff (Go32v2), использующий Nasm.
	nasmelf	Файл Elf32 (LINUX), использующий Nasm.
	nasmwin32	Файл Windows 32, использующий Nasm.
	nasmdosx	Файл Windows 32/DOSX, использующий Nasm.
	nasmobj	Объектный файл, использующий Nasm.
	masm	Объектный файл, использующий Masm (Microsoft).
	tasm	Объектный файл, использующий Tasm (Borland).
	elf	Elf32 (LINUX), использующий встроенный редактор.
	coff	Объектный файл Coff (Go32v2), использующий встроенный двоичный редактор.
	pecoff	Объектный файл PECoff (Win32), использующий встроенный двоичный редактор.
-B	Перекомпилировать все используемые модули, даже если исходный код модуля не изменился с момента последней компиляции.	
-b	Генерировать обзорную информацию. Эта информация может быть использована интегрированной средой разработки (IDE) для предоставления данных о классах, объектах, процедурах, типах и переменных в модуле.	
-bl	То же самое, что **–b**, но также генерирует информацию о локальных переменных, типах и процедурах.	
-Caxxx	Установить ABI (Application Binary Interface) для **xxx**. Параметр **–i** задаёт возможные значения для **xxx**.	
-Cb	Генерировать код с обратным порядком байтов.	
-Cc	Установить по умолчанию соглашения о вызовах, используемые компилятором.	
-CD	Создать динамическую библиотеку. Это используется для трансформации модулей в динамически связанные библиотеки на LINUX.	
-Ce	Эмулировать операции с плавающей точкой.	
-Cfxxx	Выбрать набор команд для операций с плавающей точкой. Возможные значения см. в описании параметра **–i**.	
-CFNN	Установить минимальную точность при операциях с плавающей точкой в **NN**. Возможные значения 32 или 64.	
-Cg	Включить генерацию PIC-кода. Это может потребоваться только при генерации библиотек на LINUX или других UNIX-подобных системах.	
-Chxxx	Зарезервировать **xxx** байтов в «куче». Значение **xxx** должно быть в диапазоне от 1020 до 67107840.	
-Ci	Генерировать проверку кода ввода/вывода. В случае одинакового кода входа/выхода ваша программа вернёт ошибку и будет завершена аварийно с ошибкой времени выполнения.	
-Cn	Пропускать стадию компоновки.	
-Co	Проверять переполнение при операциях с целыми числами. В случае переполнения в вашей программе будет генерироваться ошибка времени выполнения.	
-CO	Проверять возможность переполнения при операциях с целыми числами.	
-CpXXX	Установить тип процессора как **XXX**. Выбрать набор команд процессора, параметр **–i** выводит возможные значения.	
-CPX=N	Установить компоновку для **X** в **N**. Здесь **X** может принимать значения PACKSET, PACKENUM или PACKRECORD, а **N** может быть 1,2,4,8 или одним из ключевых слов DEFAULT или NORMAL.	
-Cr	Генерировать код проверки диапазона. Если ваша программа попытается получить доступ к элементу массива с неправильным индексом или увеличить перечисляемый тип с превышением его диапазона, то будет сгенерирована ошибка времени выполнения.	
-CR	Проверять вызываемый метод объекта на валидность.	

Параметр	Описание	
-Csxxx	Установить размер стека равным **xxx**.	
-Ct	Генерировать код проверки стека. Если ваша программа приведёт к переполнению стека, то будет сгенерирована ошибка времени выполнения.	
-CX	Создавать «умную» компоновку модуля при записи модуля. Эта технология будет включать в код только те части, которые используются программой. Весь неиспользуемый код будет отброшен. Это может уменьшить размер исполняемого файла.	
-dxxx	Определить имя символа **xxx**. Это может быть использовано для согласования частей компилятора с вашим кодом.	
-D	Генерировать DEF-файл (для OS/2).	
-Dd	Установить описание исполняемого файла или библиотеки (Windows).	
-Dv	Установить версию исполняемого файла или библиотеки (Windows).	
-E	То же самое, что и **–Cn**.	
-g	Генерировать отладочную информацию для отладки с **gdb**.	
-gc	Генерировать проверку для указателей. Это должно использоваться с параметром командной строки **–gh**. Если эти опции включены, то проверяется, что все указатели доступны в пределах «кучи».	
-gg	То же самое, что **–g**.	
-gh	Использовать модуль **heaptrc** (см. документ «Библиотека времени выполнения (RTL), раздел 19). Формирует отчёт об использования «кучи» после выхода из программы.	
-gl	Использовать модуль **lineinfo** (см. документ «Библиотека времени выполнения (RTL), раздел 22). Формирует файл с именем и номером строки, если программа была завершена с ошибкой.	
-goXXX	Установить опцию отладочной информации. Здесь **XXX** – это значение dwarfsets: оно включает DWARF – технологию отладки при произвольном формате записей отладочной (это не работает с **gdb** версии ниже 6.5).	
-gp	Сохранять регистр в символьных именах. По умолчанию все имена сохраняются в верхнем регистре.	
-gs	Записывать отладочную информацию.	
-gt	«Засорять» локальные переменные. Эта опция записывает в локальные переменные случайные значения при старте процедуры. Может использоваться для обнаружения инициализированных переменных.	
-gv	Формировать информацию для Valgrind.	
-gw	Формировать информацию для DWARF (версия 2).	
-gw2	Формировать информацию для DWARF (версия 2).	
-gw3	Формировать информацию для DWARF (версия 3).	
-kxxx	Поместить **xxx** в компоновщик.	
-Oxxx	Оптимизация выходного кода компилятора. Здесь **xxx** может иметь следующие значения:	
	aPARAM=VALUE	Определяет выравнивание структур и кода. **PARAM** определяет, что должно быть выровнено. **VALUE** определяет границы выравнивания. Описания возможных значений см. в документе «Руководство программиста».
	g	Оптимизировать размер. Пытаться сгенерировать наименьший код.
	G	Оптимизировать время. Пытаться генерировать быстрый код (по умолчанию).
	r	Сохранять точные переменные в регистрах (экспериментальная опция, использовать осторожно).
	u	Непостоянная оптимизация.
	1	1-й уровень оптимизации (быстрая оптимизация).
	2	2-й уровень оптимизации (**-O1** плюс некоторое замедление оптимизации).
	3	3-й уровень оптимизации (**-O2** плюс **-Ou**).
	oxxx	Определить специфическую оптимизацию. **xxx** может принимать значения: REGVAR – использовать регистрацию переменных STACKFRAME – пропускать фреймы стека LOOPUNROLL – разворачивать (уменьшать) циклы TAILREC – заменять очередь рекурсии для не рекурсионных циклов.
	pxxx	Выбрать процессор **xxx** для оптимизации. Команда **fpc –i** выводит все возможные наборы команд процессора.
	Wxxx	Генерирует информацию о полной программной оптимизации для функции **xxx**. Команда **fpc –i** выводит список возможных значений.
	wxxx	Выполняет полную программную оптимизацию для функции **xxx**. Команда **fpc –i** выводит список возможных значений.
	s	Оптимизировать в первую очередь размер, а затем скорость.
	Точный эффект от некоторых из описанных выше оптимизаций описан в документе «Руководство программиста».	
oxxx	Использовать **xxx** в качестве имени выходного (исполняемого) файла. Для использования только с данной программой. Выходное имя файла может содержать путь, и если это так, то данная опция перепишет любые предыдущие настройки, установленные параметром **–FE**. Если имя выходного файла не содержит пути, то используются настройки опции **–FE**.	

Параметр	Описание	
-pg	Генерировать код профайлера для **gprof**. Это определяет символ FPC_PROFILE, который может быть использован в соответствующих определениях.	
-s	Не вызывать ассемблер и компоновщик. Вместо этого компилятор записывает сценарий **PPAS.BAT** для ДОС или **ppas.sh** для LINUX, которые затем могут быть выполнены для генерации исполняемого файла. Это можно использовать для ускорения компиляции или для отладки выхода компилятора. Эта опция может принимать дополнительные параметры, в основном используемые для кросс-платформенной компиляции. Дополнительным параметром может быть одно из следующих значений:	
	h	Генерировать скрипт для компоновки на хосте. Сгенерированный скрипт может быть выполнен на платформе, где выполнялась компиляция (хост-платформа).
	t	Генерировать скрипт для компоновки на целевой платформе. Сгенерированный скрипт может быть выполнен на целевой платформе, то есть на ОС, на которой будет выполняться программа.
	r	Пропустить регистрацию распределения этапов (оптимизации будут отключены).
-Txxx	Определяет целевую операционную систему. Здесь **xxx** может быть одним из следующих значений:	
	emx	OS/2 через EMX (и DOS через дополнение EMX).
	freebsd	FreeBSD.
	Go32v2	DOS и версия 2 расширителя DJ DELORIE.
	linux	LINUX.
	netbsd	NetBSD.
	netware	Novell Netware Module (clib).
	netwlibc	Novell Netware Module (libc).
	openbsd	OpenBSD.
	os2	OS/2 (2.x), использующая расширитель EMX.
	sunos	SunOS/Solaris.
	watcom	Watcom-совместимый расширитель DOS.
	wdosx	Расширитель WDOSX.
	win32	32-разрядная Windows.
	wince	Windows для handhelds (ARM-процессор).
	Команда **fpc –i** выводит список поддерживаемых компилятором операционных систем.	
-uxxx	Неопределённый символ **xxx**. Эта опция является антиподом опции **–d**.	
-Ur	Генерировать выходные файлы модуля. Эти файлы не будут перекомпилированы, даже если доступны исходные файлы модуля. Может оказаться полезной при создании дистрибутива. Перезаписывается опцию **–B**.	
-W	Установить атрибуты Windows или OS/2 для генерации двоичного файла. Это может быть один или несколько параметров, перечисленных ниже:	
	Bhhh	Установить предпочтительный базовый адрес в **hhh** (в шестнадцатиричной системе).
	C	Генерировать консольное приложение (+) или GNU-приложение (-).
	D	Форсировать использование Def-файла для экспорта.
	F	Генерировать FS-приложение (+) или консольное приложение (-).
	G	Генерировать GNU-приложение (+) или консольное приложение (-).
	N	Не генерировать раздел перераспределения памяти.
	R	Генерировать раздел перераспределения памяти.
	T	Генерировать TOOL-приложение (+) или консольное приложение (-).
-Xx	Определить опции исполняемого файла. Параметр указывает компилятору, какой тип исполняемого файла должен быть сгенерирован. Здесь **x** может принимать следующие значения:	
	c	(Только для LINUX). Компоновать с библиотекой C. Вы должны использовать эту опцию только когда вы запустили Free Pascal для другой операционной системы.
	d	Не использовать стандартный путь библиотек. Это необходимо для кросс-платформенной компиляции, чтобы избежать компоновки с библиотеками хост-платформы.
	D	Компоновать с динамической библиотекой (определённой символом FPC_LINK_DYNAMIC).
	e	Использовать внешний (GNU) компоновщик.
	g	Создавать отладочную информацию в отдельном файле и добавлять раздел отладки компоновщика для исполняемого файла.
	i	Использовать встроенный компоновщик.
	MXXX	Установить имя входа в программу. По умолчанию это **main**.
	m	Генерировать мэп-файл компоновщика.
	PXXX	Присоединять спереди имена с **XXX** для кросс-платформенной компиляции.
	rXXX	Установить путь к библиотекам в **XXX**.
	Rxxx	Присоединять спереди **xxx** для всех путей поиска компоновщика (используется для кросс-платформенной компиляции).
	s	Отнимать символы из исполняемого файла.
	S	Компоновать модули статически (по умолчанию определяется символом FPC_LINK_STATIC).
	t	Компоновать библиотеки статически (поместить параметр **–static** в компоновщик).
	X	Выполнять «умную компоновку» модулей (по умолчанию определяется символом FPC_LINK_STATIC).

5.1.5. Параметры для исходных кодов (опции языка)

Более подробную информацию см. в документе «Руководство программиста».

Параметр	Описание	
-Mmode	Установить режим языка программирования. Здесь **mode** может принимать следующие значения:	
	delphi	Пытаться выполнять совместимость с Delphi. Этот режим более строгий, чем режим **objfpc**, потому что некоторые расширения Free Pascal будут отключены.
	fpc	Диалект Free Pascal (по умолчанию).
	macpas	Пытаться выполнять совместимость с диалектом Macintosh Pascal.
	objfpc	Подключает некоторые расширение Delphi. Этот режим отличается от Delphi, так как имеет некоторые специфические конструкции Free Pascal.
	tp	Пытаться выполнять совместимость с TP/BP 7.0. Это значит, что не будет работать перегрузка операций и т.п.
-Mfeature	Выбор функций языка в **feature**. Начиная с версии FPC 2.3.1 параметр командной строки **–M** может быть использован для выбора индивидуальных функций языка. В этом случае **feature** может принимать одно из следующих значений:	
	CLASS	Использовать классы Object Pascal.
	OBJPAS	Автоматически подключать модуль **ObjPas**.
	RESULT	Включить идентификатор **Result** для возвращения результата функциями.
	PCHARTOSTRING	Разрешить автоматически преобразовывать строки с нулевым окончанием в строки.
	CVAR	Разрешить использовать ключевое слово **CVAR**.
	NESTEDCOMMENTS	Разрешить использовать вложенные комментарии.
	CLASSICPROCVARS	Использовать классические процедурные переменные.
	MACPROCVARS	Использовать процедурные переменные в стиле **Mac**.
	REPEATFORWARD	Объявление и реализация функций должны совпадать.
	POINTERTOPROCVAR	Разрешить скрытно преобразовывать указатели в процедурные переменные.
	AUTODEREF	Автоматически (скрытно) получать значения типизированных указателей.
	INITFINAL	Разрешить использование **Initialization** и **Finalization**.
	POINTERARITHMETICS	Разрешить использовать арифметические операции с указателями.
	ANSISTRINGS	Разрешить использовать ANSI-строки.
	OUT	Разрешить использовать параметр **out**.
	DEFAULTPARAMETERS	Разрешить использовать по умолчанию значения параметра.
	HINTDIRECTIVE	Поддерживать директивы подсказок (deprecated, platform и т.п.).
	DUPLICATELOCALS	?
	PROPERTIES	Разрешить использовать глобальные свойства.
	ALLOWINLINE	Разрешить входные процедуры.
	EXCEPTIONS	Разрешить использовать исключения.
	Ключевое слово может сопровождаться знаком плюс (+) или минус (-) для включения или отключения функции.	
-Rxxx	Определяет, какой тип ассемблера вы используете для ассемблерных блоков в исходном коде программ. Здесь **xxx** может принимать значения:	
	att	Блоки **asm** содержат ассемблер стиля AT&T. Это стиль по умолчанию.
	intel	Блоки **asm** содержат ассемблер стиля Intel.
	default	Использовать ассемблер по умолчанию для указанной целевой платформы.
	direct	Блоки **asm** копируются в ассемблер «как есть», только заменяются определённые переменные.
-S2	Включить расширения Delphi 2 (режим **objfpc**). Не рекомендуется. Используйте вместо этого **–Mobjfpc**.	
-Sa	Включать описания операторов в компилируемый код.	
-Sc	Поддерживать операторы стиля C, например, *=, +=, /= и т.п.	
-Sd	Пытаться выполнять совместимость с Delphi. Не рекомендуется. Используйте вместо этого **–Mdelphi**.	
-SeN	Останавливать компилятор после N-ой ошибки. Обычно компилятор пытается продолжить работу после обнаружения ошибки, пока не накопится 50 ошибок или не случится фатальная ошибка, а затем останавливается. Если данная опция установлена, то компилятор остановится после N-ой ошибки (если N не указана, то она равна по умолчанию 1). Вместо числа может быть также указано **n**, **h** или **w**. В этом случае компилятор будет рассматривать замечания (n), подсказки (h) и предупреждения (w) как ошибки и остановится при превышении указанного количества.	

Параметр	Описание
-Sg	Поддерживать команды **label** и **goto**. По умолчанию эти команды не поддерживаются. Вы также должны указать этот параметр, если вы используете метки в операторах ассемблера (если вы используете стиль ассемблера AT&T).
-Sh	Использовать по умолчанию для строк ANSI-строки. Если эта опция указана, то компилятор будет интерпретировать ключевое слово **string** как **ansistring**. Иначе предполагается, что используются короткие строки (TP стиль).
-Si	Поддерживать стиль строки C++.
-SIXXX	Установить стиль интерфейса как **XXX**.
-Sk	Загрузить Kylix-совместимый модуль (fpcylix).
-Sm	Поддерживать макросы стиля C.
-So	Пытаться выполнять совместимость с Borland TP 7.0. Не рекомендуется. Используйте вместо этого **–Mtp**.
-Ss	Имена конструкторов должны быть **init**, а имена деструкторов должны быть **done**.
-St	Разрешить ключевое слово **static** в объектах.
-Sx	Включить ключевые слова исключений (по умолчанию в режиме Delphi/Objfpc). Эта опция помечает все ключевые слова исключений как ключевые слова также в режимах Turbo Pascal и FPC. Это может быть использовано для проверки кода, который независим от режимов настолько, насколько это возможно.
-Un	Не проверять имя модуля. Обычно имя модуля совпадает с именем файла. Эта опция позволяет сделать эти имена различными.
-Us	Компилировать системный модуль (system). Эта опция заставляет компилятор определять только некоторые очень важные типы.

5.2. Использование конфигурационного файла

Использование конфигурационного файла `fpc.cfg` является альтернативой применения параметров командной строки. Если конфигурационный файл найден, то выполняется его чтение, и строки этого файла обрабатываются таким образом, как будто вы напечатали их как параметры в командной строке. В каждой строке конфигурационного файла содержится один параметр командной строки. Параметры конфигурационного файла обрабатываются раньше, чем параметры командной строки.

Вы можете написать комментарии в конфигурационном файле, отмечая их знаком `#`. Всё, что следует за этим знаком, будет игнорироваться при обработке конфигурационного файла.

Алгоритм определения файла, который используется как конфигурационный, описан в разделе «3.1.5. Конфигурационный файл».

Когда компилятор заканчивает обработку конфигурационного файла, он начинает обрабатывать параметры командной строки.

Один из параметров командной строки позволяет вам указать другой конфигурационный файл. Например, если в параметре командной строки указать `@foo`, то в качестве конфигурационного будет открыт файл `foo`, опции которого в дальнейшем будут использоваться. Когда компилятор завершит чтение этого файла, он продолжит обрабатывать параметры командной строки.

Конфигурационный файл допускает несколько типов первичной обработки. Имеются в виду следующие директивы, которые вы должны размещать вначале строки:

```
#IFDEF
#IFNDEF
#ELSE
#ENDIF
#DEFINE
#UNDEF
#WRITE
#INCLUDE
#SECTION
```

Эти директивы работают по тому же принципу, что и их двойники в исходном коде Паскаля.

Все определённые по умолчанию директивы в исходном коде также являются определёнными по умолчанию в конфигурационном файле. Например, если компилятор является целевым для Intel 80x86 совместимым с платформой Linux, то параметры `cpu86` и `linux` будут определены во время интерпретации конфигурационного файла. Все возможные определения по умолчанию, которые имеют место быть во время компиляции, см. в приложении **G** документа «Руководство программиста».

В следующих разделах описываются различные директивы компилятора, используемые в конфигурационном файле.

5.2.1. IFDEF

Синтаксис:

```
#IFDEF Имя
```

Строки, которые следуют за этой директивой, будут прочитаны только в том случае, если ключевое слово `Имя` определено.

Чтение выполняется до тех пор, пока не будет найдено ключевое слово `#ELSE` или `#ENDIF`, после которого возобновится нормальный процесс компиляции.

Пример:

```
#IFDEF VER2_2_0
-Fu/usr/lib/fpc/2.2.0/linuxunits
#ENDIF
```

В этом примере `/usr/lib/fpc/2.2.0/linuxunits` будет добавлено в путь, если вы используете версию компилятора 2.2.0.

5.2.2. IFNDEF

Синтаксис:

```
#IFNDEF Имя
```

Строки, которые следуют за этой директивой, будут прочитаны только в том случае, если ключевое слово `Имя` НЕ определено.

Чтение выполняется до тех пор, пока не будет найдено ключевое слово `#ELSE` или `#ENDIF`, после которого возобновится нормальный процесс компиляции.

Пример:

```
#IFNDEF VER2_2_0
-Fu/usr/lib/fpc/2.2.0/linuxunits
#ENDIF
```

В этом примере `/usr/lib/fpc/2.2.0/linuxunits` будет добавлено в путь, если вы используете любую версию компилятора, кроме версии 2.2.0.

5.2.3. ELSE

Синтаксис:

```
#ELSE
```

Директива `#ELSE` может быть указана после директив `#IFDEF` или `#IFNDEF` как альтернатива. Строки, которые следуют за этой директивой, будут прочитаны только в том случае, если не выполнилось условие директив `#IFDEF` или `#IFNDEF`.

Чтение выполняется до тех пор, пока не будет найдено ключевое слово `#ENDIF`, после которого возобновится нормальный процесс компиляции.

Пример:

```
#IFDEF VER2_2_2
-Fu/usr/lib/fpc/2.2.2/linuxunits
#ELSE
-Fu/usr/lib/fpc/2.2.0/linuxunits
#ENDIF
```

В этом примере `/usr/lib/fpc/2.2.2/linuxunits` будет добавлено в путь, если вы используете версию компилятора 2.2.2, иначе будет добавлено `/usr/lib/fpc/2.2.0/linuxunits`.

5.2.4. ENDIF

Синтаксис:

```
#ENDIF
```

Директива `#ENDIF` отмечает конец блока, который начинается директивой `#IF(N)DEF`, возможно с директивой `#ELSE` между ними.

5.2.5. DEFINE

Синтаксис:

```
#DEFINE Имя
```

Директива `#DEFINE` определяет новое ключевое слово. Она имеет тот же эффект, что и параметр командной строки `-dname`.

5.2.6. UNDEF

Синтаксис:

```
#UNDEF Имя
```

Директива `#UNDEF` отменяет определение ключевого слова, если оно существует. Директива имеет тот же эффект, что параметр командной строки `-uname`.

5.2.7. WRITE

Синтаксис:

```
#WRITE Текст сообщения
```

Директива `#WRITE` выводит `Текст сообщения` на экран. Это может оказаться полезным для вывода предупреждений, если какие-либо опции являются установленными.

Пример:

```
#IFDEF DEBUG
#WRITE Setting debugging ON...
-g
#ENDIF
```

Если `DEBUG` определено, то этот пример выведет строку

```
Setting debugging ON...
```

а затем включит вывод отладочной информации.

5.2.8. INCLUDE

Синтаксис:

```
#INCLUDE ИмяФайла
```

Директива `#INCLUDE` даёт команду компилятору читать содержимое файла `ИмяФайла`, перед продолжением чтения параметров в текущем файле.

Это может оказаться полезным, если вы редко используете конфигурационный файл для проекта (или работаете на LINUX в вашей домашней директории), но в то же время хотите использовать глобальные параметры в глобальном конфигурационном файле.

Пример:

```
#IFDEF LINUX
#INCLUDE /etc/fpc.cfg
#ELSE
#IFDEF GO32V2
#INCLUDE c:\pp\bin\fpc.cfg
#ENDIF
#ENDIF
```

Если вы работаете на LINUX-машине, то будет подключен конфигурационный файл `/etc/fpc.cfg`, а если вы работает на DOS-машине, то будет подключен конфигурационный файл `c:\pp\bin\fpc.cfg`.

5.2.9. SECTION

Синтаксис:

```
#SECTION Имя
```

Директива `#SECTION` действует как директива `#IFDEF`, только не требует директивы `#ENDIF`. Специальное имя `COMMON` всегда существует, то есть строки, которые следуют за `#SECTION COMMON` всегда будут читаться.

5.3. Переменные замещения в путях

Чтобы избежать необходимости редактировать ваш конфигурационный файл слишком часто, компилятор позволяет вам определить следующие переменные в путях, которые вы можете передать в компилятор:

FPCFULLVERSION – замещает строку полной версии компилятора.
FPCVERSION – замещает строку версии компилятора.
FPCDATE – замещает дату компилятора.
FPCTARGET – замещает назначение компилятора (комбинация ПРОЦЕССОР-ОС).
FPCCPU – замещает целевой процессор компилятора.
FPCOS – замещает целевую ОС компилятора.

Для использования этих переменных просто вставьте их с символом `$` вначале, как показано ниже:

```
-Fu/usr/lib/fpc/$FPCVERSION/rtl/$FPCOS
```

Это равносильно следующей записи:

```
-Fu/usr/lib/fpc/2.2.2/rtl/linux
```

Если версия компилятора 2.2.2, а целевая ОС – это LINUX.

Эти замещения могут применяться как в командной строке, так и в конфигурационном файле. В командной строке LINUX вы должны иметь в виду, что использование символа `$` может иметь неблагоприятный эффект.

6. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ (IDE)

IDE (Integrated Development Environment – Интегрированная Среда Разработки) предоставляет для компилятора комфортабельный пользовательский интерфейс. В её состав входят редактор исходного кода с подсветкой синтаксиса, отладчик, таблица символов и др. IDE – это приложение, работающее в текстовом режиме, которое работает на всех поддерживаемых операционных системах. Она основана на IDE Turbo Pascal, к которой привыкли многие люди.

На текущий момент доступна IDE для DOS, WINDOWS и LINUX.

6.1. Первые шаги с IDE

6.1.1. Запуск IDE

Для запуска среды разработки просто напечатайте в командной строке:

```
fp
```

Среду разработки можно запустить также из графического интерфейса с помощью ярлыка, как это делается в Windows.

ПРИМЕЧАНИЕ
В Windows можно переключаться между оконным режимом и полноэкранным режимом с помощью комбинации клавиш `ALT+ENTER`.

6.1.2. Опции командной строки IDE

При запуске среды разработки в программу могут быть переданы параметры следующим образом:

```
fp [-Параметр] [-Параметр] ... <Имя файла> ...
```

Здесь `Параметр` – это один из следующих переключателей (все параметры чувствительны к регистру):

Параметр	Описание
-N	(Только для DOS). Не использовать длинные имена файлов. WINDOWS 95 и более поздние версии WINDOWS предоставляют интерфейс ДОС-приложениям для доступа к длинным именам файлов. IDE использует этот интерфейс доступа к файлам по умолчанию. Непосредственно в среде ДОС это может вызвать проблемы. Этот переключатель указывает IDE не использовать длинные имена файлов.
-Cfilename	Читать параметры IDE из файла **filename**. Между именем файла и **–C** не должно быть пробелов.
-F	Использовать альтернативные графические символы. Это может быть использовано для запуска IDE на LINUX в X-term или через сессию Телнет.
-R	После запуска IDE автоматически изменить директорию которая была активна при последнем выходе из IDE.
-S	Отключить мышь. Если используется эта опция, то мышь отключена, даже если физически она подключена к компьютеру.
-Tttyname	(Только для LINUX/UNIX). Отправить выходную программу в tty **ttyname**. Это позволяет избежать бесконечного переключения между программой и IDE.

Файлы, переданные в параметре, загружаются в окна редактора автоматически.

ПРИМЕЧАНИЕ
Для DOS/Win32 первый символ параметра командной строки может быть символом `/` вместо символа `-`. То есть `/S` эквивалентно `-S`.

6.1.3. Экран IDE

После запуска программы вы увидите окно среды разработки (см. рис. 6.1).

Рис. 6.1. Главное окно IDE Free Pascal.

В верхней части экрана находится главное меню, в нижней части – строка состояния. Свободное пространство между ними называется «Рабочий стол».

В строке состояния в виде подсказок выводятся наиболее часто применяемые комбинации клавиш. Щёлкая левой мышью по этим подсказкам, можно быстро выполнить указанную команду. Например, открыть файл можно через меню, нажав клавишу `F3` или щёлкнув мышью по надписи «`F3 Open`» в строке состояния. В правом углу строки состояния отображается текущее количество неиспользуемой памяти. Это только индикация, позднее IDE пытается получить больше памяти у операционной системы, если памяти не хватает.

Главное меню предоставляет доступ ко всем функциям IDE, а в правом углу меню отображается текущее время (не на всех версиях).

Выйти из среды разработки можно через меню `FILE-EXIT` или комбинацией клавиш `ALT+X`. Текст `FILE-EXIT` означает, что нужно сначала выбрать в меню `FILE`, а затем в открывшемся подменю выбрать `EXIT`.

ПРИМЕЧАНИЕ
Если файл `fp.ans` найден в текущей директории, то он будет загружен и использован для прорисовки заднего плана экрана (как это показано на рис. 6.1). Этот файл должен содержать команды рисования ANSI для прорисовки экрана.

6.2. Навигация в IDE

Навигация в среде разработки может выполняться как мышью, так и клавиатурой, если система работает с мышью.

6.2.1. Использование клавиатуры

Все функции IDE можно выполнить с помощью клавиатуры.

- Клавиатура используется для набора текста и навигации по исходным кодам.
- С помощью клавиатуры выполняются команды редактирования, такие как копирование и вставка текста.
- Перемещение и изменение размера окон.
- Клавиатуру можно использовать для доступа к меню. Для этого нужно нажать клавишу `ALT` и подсвеченную в пункте меню букву (см. рис. 6.1). Например, чтобы выбрать меню `FILE`, нужно нажать клавиши `ALT-F`. Также можно нажать клавишу `F10` и перемещаться по меню с помощью клавиш со стрелками. Более подробно меню описано в разделе 6.4.
- Многие команды IDE связаны с комбинациями клавиш. То есть типовые и специальные комбинации клавиш будут выполнять команду немедленно.

ПРИМЕЧАНИЯ

- При работе в LINUX X-Term или через сессию Телнет комбинации клавиш с `ALT` могут быть недоступны. Чтобы это исправить, нужно сначала нажать комбинацию клавиш `CTRL-Z`. Например, вместо `ALT-X` нужно нажать `CTRL-Z X`.
- В качестве альтернативы вы можете попробовать использовать комбинацию `ESC-X` вместо `ALT-X`, когда работает в LINUX.
- Полный список комбинаций клавиш можно найти в разделе 6.14.

6.2.2. Использование мыши

Если система оборудована мышью, то мышь можно использовать для работы в IDE. Левая кнопка используется для выбора пунктов меню, нажатия кнопок, выделения текста и т.п.

Правая кнопка используется для доступа к контекстному меню, если оно имеется. Удержание клавиши `CTRL` или `ALT` и щелчок правой кнопкой приведёт к выполнению определённой пользователем функции. См. раздел 6.12.4.

ПРИМЕЧАНИЯ

1. Иногда в руководствах используется понятие «перетаскивание мышью». Это означает, что мышь перемещается при нажатой левой кнопке.
2. Действие кнопок мыши может быть реверсивным (для левшей), то есть когда левая кнопка работает как правая, а правая работает как левая. В данном руководстве подразумевается, что мышь работает в обычном режиме, то есть не реверсирована.
3. Мышь не всегда доступна, даже если она установлена:
   - При запуске IDE на LINUX через соединение Телнет с Windows-машины.
   - При запуске IDE на LINUX в T-term под X-windows. В случае, если это определено на терминальной программе: под Konsole (KDE-терминал).
4. На Windows, если консоль имеет опцию «Быстрая вставка», позволяющую копировать текст в буфер обмена путём выбора в консольном окне. Если этот режим включен, мышь работать не будет. Опция «Быстрая вставка» должна быть отключена в свойствах окна, чтобы IDE могла отслеживать события мыши.

6.2.3. Навигация в диалогах

Диалоговые окна обычно имеют набор элементов, таких как кнопки, поля для ввода данных, списки и т.п. Для активации какого-либо из этих элементов можно воспользоваться следующими способами:

1. Щёлкнуть по элементу мышью.
2. Нажимать клавишу `TAB` до тех пор, пока фокус не попадёт на нужный элемент.
3. Нажать букву на клавиатуре, подсвеченную в наименовании элемента. Если фокус в текущий момент находится на элементе, который допускает редактирование, то клавиша `ALT` должна быть нажата одновременно с подсвеченной буквой. В случае с кнопкой будет выполнено действие, связанное с этой кнопкой.

В пределах поля для ввода данных, списков и редакторов, перемещение выполняется клавишами со стрелками.

6.3. Окна

В наше время работа с оконными приложениями не вызывает проблем у пользователей Windows или Linux. Тем не менее следующие разделы описывают работу с окнами, для того, чтобы вы могли максимально эффективно использовать Free Pascal IDE.

6.3.1. Общая информация об окнах

Обычное окно IDE показано на рис. 6.2.

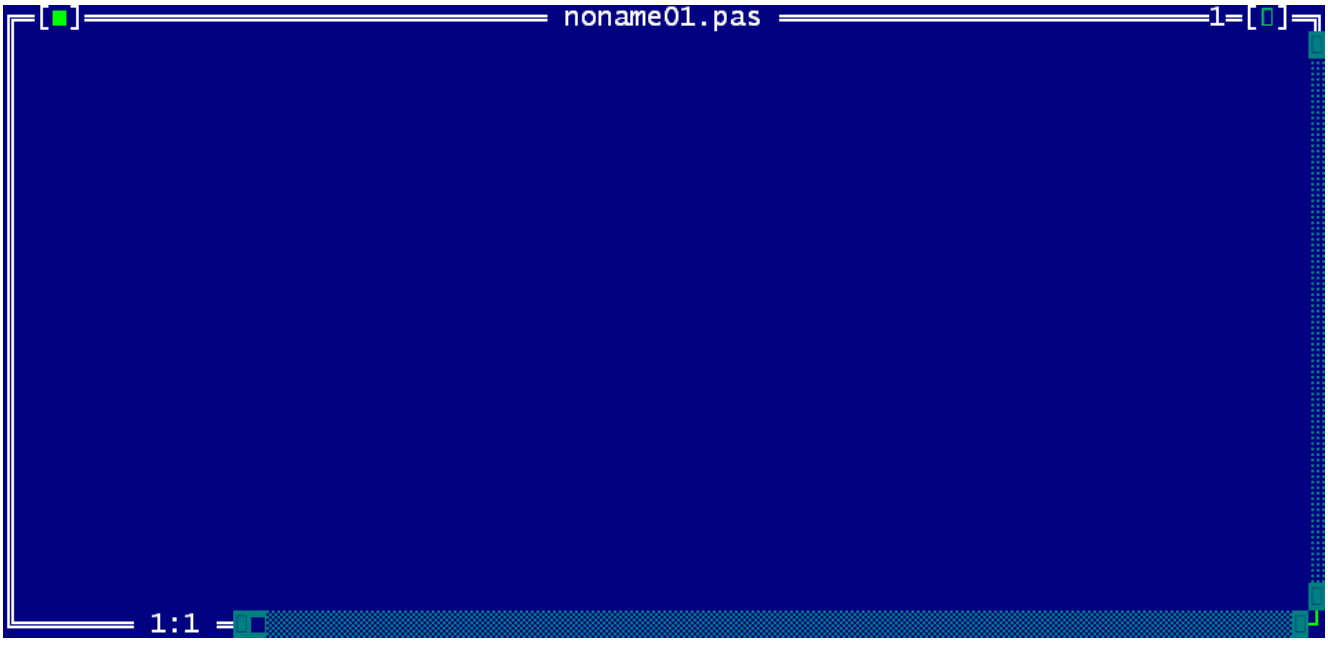


Рис. 6.2. Обычное окно IDE Free Pascal.

Это окно окружено так называемой рамкой – белая двойная линия вокруг окна.

В верхней части окна расположены четыре элемента:

- В верхнем левом углу окна расположен значок закрытия окна. Если щёлкнуть по этому значку, то окно закроется. Окно можно также закрыть комбинацией клавиш ALT-F3 или командой меню WINDOW – CLOSE. Все открытые окна можно закрыть командой меню WINDOW – CLOSE ALL.
- В середине верхней части отображается заголовок окна.
- Почти в правом верхнем углу видна цифра. Эта цифра является идентификатором окна редактора, нажимая комбинацию клавиш ALT-ЦИФРА можно быстро перейти к этому окну. Только первые 9 окон имеют такой идентификатор.
- В самом правом верхнем углу видна зелёная стрелка (в некоторых ОС вместо стрелки может отображаться зелёный квадрат, что означает неправильное определение таблицы символов). Щёлкните по этой стрелке, чтобы развернуть окно на весь рабочий стол. Щелчок по этой стрелке перезаписывает старый размер окна. Нажатие клавиши F5 имеет тот же эффект, что и щелчок по этой кнопке. Это же действие можно выполнить командой меню WINDOW – ZOOM. Окна и диалоги, которые имеют фиксированный размер, не могут быть увеличены.

Правая и нижняя стороны окна имеют полосы прокрутки. Их можно использовать для прокрутки содержимого окна по вертикали и горизонтали с помощью мыши. Щёлкая по стрелкам на концах полосы прокрутки, можно перемещать содержимое строка за строкой. Щёлкая по свободному пространству между стрелкой и прямоугольником полосы прокрутки, можно выполнять постраничное перемещение в окне. Непрерывное перемещение выполняется перетаскиванием прямоугольника полосы прокрутки мышью.

Звёздочка и цифры в нижнем левом углу отображают информацию о содержимом окна. Их назначение разъяснено в разделе, посвящённом редактору исходного кода, см. раздел 6.5.

6.3.2. Перемещение окон и изменение их размеров

Окна можно перемещать и изменять их размер с помощью мыши или клавиатуры.

Для перемещения окна:

- С помощью мыши: щёлкните мышью на заголовке окна и перетаскивайте окно.
- С помощью клавиатуры: перейдите в режим изменения размера/перемещения комбинацией клавиш CTRL-F5 или командой меню WINDOWS – SIZE/MOVE. Рамка окна окрасится в зелёный цвет, показывая, что IDE находится в режиме изменения размера/перемещения. Теперь клавиши управления курсором (клавиши со стрелками) можно использовать для перемещения окна по рабочему столу. Чтобы выйти из режима изменения размера/перемещения нажмите клавишу ENTER. В этом случае окно сохранит свои новые размеры и положение. Альтернативой является нажатие клавиши ESC. Эта команда возвращает окно в старые размеры и положение, и выполняет выход из режима изменения размера/перемещения.

Для изменения размеров окна:

- С помощью мыши: щёлкните мышью в нижнем правом углу окна и перетаскивайте мышь.
- С помощью клавиатуры: перейдите в режим изменения размера/перемещения комбинацией клавиш CTRL-F5 или командой меню WINDOWS – SIZE/MOVE. Рамка окна окрасится в зелёный цвет, показывая, что IDE находится в режиме изменения размера/перемещения. Теперь удерживайте клавишу SHIFT и используйте клавиши управления курсором для изменения размеров окна. Чтобы выйти из режима изменения размера/перемещения нажмите клавишу ENTER. В этом случае окно сохранит свои новые размеры и положение. Альтернативой является нажатие клавиши ESC. Эта команда возвращает окно в старые размеры и положение, и выполняет выход из режима изменения размера/перемещения.

Не все окна позволяют изменять свой размер. Например, диалоговые окна нельзя увеличивать или уменьшать (см. раздел 6.3.4).

Кроме этого окно можно скрыть. Для скрытия окна нажмите комбинацию клавиш CTRL-F6 или выберите команду меню WINDOW – HIDE. Чтобы снова отобразить на экране скрытое окно, нужно выбрать его в списке окон. Более подробно об этом в следующем разделе.

6.3.3. Работа с множеством окон

При работе над большим проектом возникает необходимость работы с множеством окон. Однако только одно из этих окон может быть активным, остальные будут неактивными.

Неактивные окна отображаются с серой рамкой. Неактивное окно можно сделать активным одним из следующих способов:

- Щёлкнуть мышью по неактивному окну, чтобы сделать его активным.
- Нажатие клавиши F6 будет выполнять поочерёдный переход по открытым окнам, делая их активными. Чтобы сделать активным предыдущее окно, нажмите SHIFT-F6.
- Командой меню WINDOW – NEXT можно сделать активным следующее окно в списке окон, командой WINDOW – PREVIOUS можно сделать активным предыдущее окно.
- Если окно имеет номер в верхнем правом углу, то его можно сделать активным комбинацией клавиш ALT-<НОМЕР>, где номер – это цифровая клавиша, соответствующая номеру окна.
- Нажатие клавиш ALT-0 вызывает диалоговое окно со списком доступных окон, где можно быстро активизировать окна, не имеющие номеров.

Окна могут быть упорядочены и размещены на рабочем столе IDE путём их разворачивания и изменения размера с помощью мыши или клавиатуры. Эта задача отнимает много времени, особенно при использовании клавиатуры. Вместо этого в меню есть команды WINDOW – TILE и WINDOW – CASCADE, которые используются следующим образом:

- TILE равномерно разделяет всё пространство рабочего стола между всеми окнами, размеры которых разрешено изменять.
- CASCADE помещает окна друг над другом, выравнивая их в виде каскада.

В редких случаях окна на экране могут перемешаться. В таком случае весь экран IDE можно обновить командой меню WINDOW – REFRESH DISPLAY.

6.3.4. Диалоговые окна

Во многих случаях IDE отображает диалоговые окна для получения данных от пользователя. Основное отличие диалоговых окон от обычных заключается в том, что другие окна не могут быть активными, пока активно диалоговое окно. Также меню недоступно, пока открыто диалоговое окно. Такое поведение окна называется модальным. Для доступа к другим окнам сначала нужно закрыть модальное окно.

Типичное диалоговое окно показано на рис. 6.3.

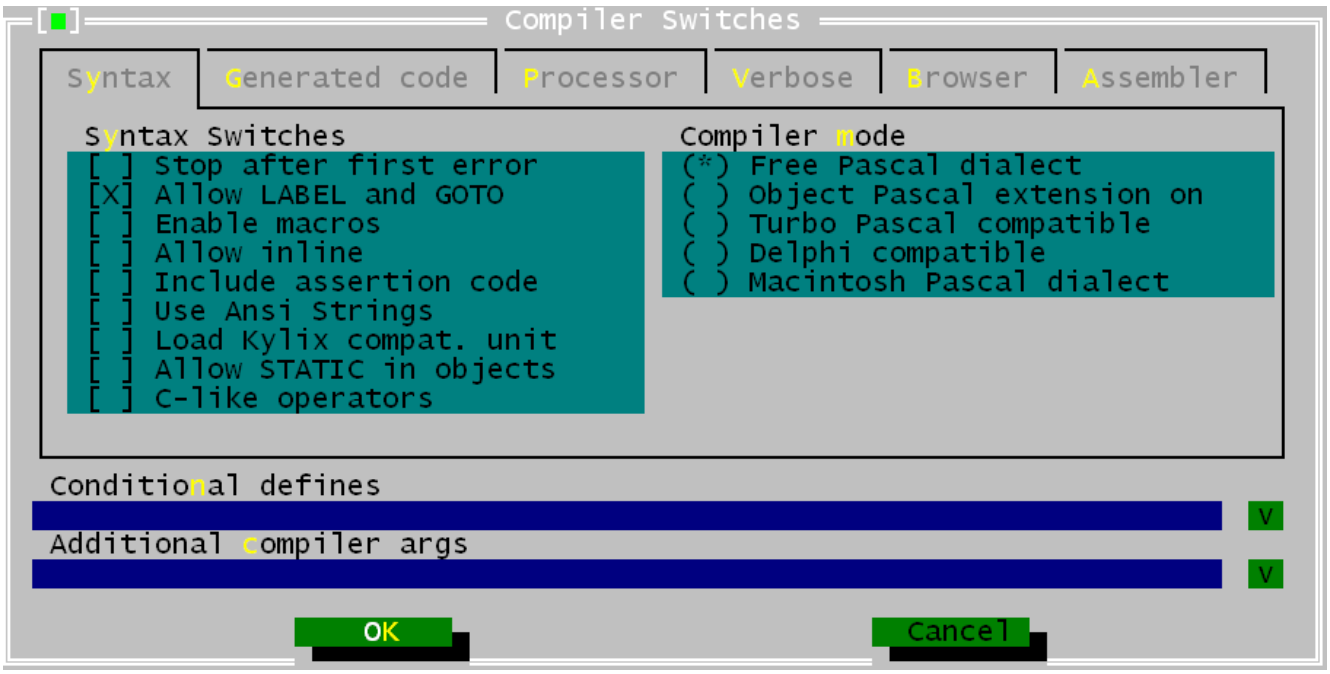


Рис. 6.3. Диалоговое окно.

6.4. Меню

Главное меню (серая полоса в верхней части экрана IDE) предоставляет доступ ко всем функциям IDE. В меню также отображаются часы, которые показывают текущее время (не во всех версиях). Меню всегда доступно, за исключением случаев, когда открыто диалоговое окно. Если диалоговое окно открыто, то его нужно закрыть, чтобы получить доступ к меню.

В некоторых окнах также доступно контекстное меню. Контекстное меню отображается там, где расположен курсор. Набор команд зависит от контекста.

6.4.1. Доступ к меню

Доступ к меню можно выполнить следующими способами:

1. Использовать мышь для выбора пункта меню. Требуется расположить курсор мыши над нужным пунктом меню и щёлкнуть любую кнопку мыши.
2. Нажать `F10`. Это действие переключит фокус на меню. Затем вы можете использовать клавиши со стрелками для навигации по меню. Клавиша `ENTER` используется для выбор пункта меню.
3. Прямой доступ к команде меню. Для этого нажмите сочетание клавиш `ALT-<ПОДСВЕЧЕННАЯ БУКВА МЕНЮ>`. Затем можно выбрать пункт подменю нажатием подсвеченной буквы, но уже без нажатия `ALT`. Например, нажатие `ALT-S`, а затем `G`, вызовет диалоговое окно `Goto Line` (перейти к строке).

Каждый пункт меню кратко описывается в строке состояния при выборе.

Если элемент имеет контекстное меню, то его можно вызвать щелчком правой кнопки мыши по элементу или комбинацией клавиш `ALT-F10`.

Для выхода из меню без выполнения каких-либо действий, нажмите клавишу `ESC`.

Далее описаны все элементы меню и действия, которые они выполняют.

6.4.2. Меню File

Меню **File** содержит все команды, которые позволяют пользователю загружать и сохранять файлы, а также выполнять выход из IDE.

Команда	Перевод	Описание
New	**Новый**	Открывает новое пустое окно в редакторе.
New from template	**Новый по шаблону**	Открывает окно для выбора шаблона. Затем можно указать некоторые параметры, которые будут подставлены в шаблон. Затем открывается новое окно в редакторе, где уже вставлен текст шаблона с указанными данными.
Open (F3)	**Открыть**	Вызывает диалог выбора файла и открывает выбранный файл в новом окне редактора.
Print	**Печать**	Выводит на печать содержимое текущего окна редактора.
Print Setup	**Настройки печати**	Вызывает окно настроек принтера.
Reload	**Перегрузка**	Перезагружает файл с диска.
Save (F2)	**Сохранить**	Сохраняет содержимое окна редактора в текущий файл. Если текущее окно еще не связано с файлом, то будет вызвано диалоговое окно для ввода имени файла.
Save as	**Сохранить как…**	Вызывает диалоговое окно для ввода имени файла. Текущее содержимое окна будет сохранено в указанный файл.
Save all	**Сохранить всё**	Сохранить содержимое всех открытых окон.
Change dir	**Сменить каталог**	Вызывает диалоговое окно для выбора каталога. Текущий рабочий каталог будет заменён на выбранную директорию.
Command shell	**Командная строка**	Переход в командный интерпретатор. После выхода из командной строки (обычно для этого надо напечатать команду EXIT) выполняется возврат в IDE. Какой командный интерпретатор будет вызван, определяется операционной системой.
Exit (ALT-X)	**Выход**	Выход из IDE. Если какие-либо открытые файлы не сохранены, IDE будет задавать вопрос о необходимости сохранения этих файлов.

Под элементом `Exit` в меню появляются несколько имён файлов, которые использовались в последнее время. Это можно использовать для быстрой загрузки данных файлов в редактор.

6.4.3. Меню Edit

Меню **Edit** содержит команды для доступа к буферу обмена и для отмены и возврата действий, связанных с редактированием. Многие из этих функций связаны с «горячими клавишами», которые выполняют аналогичные действия.

Команда	Перевод	Описание
Undo (ALT-BKSP)	**Отмена**	Отменяет последнее действие, выполненное в редакторе. Все действия, выполненные в редакторе, записываются в буфер. Выбор этого механизма будет перемещаться назад по этому буферу, то есть возможны множество уровней отмены.
Redo	**Возврат**	Повторяет последнее действие, которое было отменено командой Undo. Эта команда может вернуть множество отменённых действий.
Cut (SHIFT-DEL)	**Вырезать**	Удаляет выделенный в окне текст и копирует его в буфер обмена. Всё предыдущее содержимое буфера утрачивается. Новое содержимое буфера можно вставить в любом другом месте.
Copy (CTRL-INS)	**Копировать**	Копирует текущее выделение в буфер обмена. Всё предыдущее содержимое буфера утрачивается. Новое содержимое буфера можно вставить в любом другом месте.
Paste (SHIFT-INS)	**Вставить**	Вставляет содержимое буфера обмена в текст, начиная с позиции курсора. Содержимое буфера при этом не изменяется.
Clear (CTRL-DEL)	**Очистить**	Очищает (то есть удаляет) текущее выделение.
Select all	**Выбрать всё**	Выделить (выбрать) весь текст в текущем окне. Выбранный текст может быть удалён или скопирован в буфер обмена.
Unselect	**Отменить выделение**	Отменяет выделение.
Show clipboard	**Показать буфер обмена**	Открывает окно, где отображается текущее содержимое буфера обмена.

Если вы запустили IDE под Windows, то меню `Edit` будет иметь две дополнительных команды. IDE имеет собственный буфер обмена, который недоступен из Windows. Для обмена с буфером Windows существуют две команды:

Команда	Перевод	Описание
Copy to Windows	**Копировать в Windows**	Копирует выделение в буфер обмена Windows.
Paste from Windows	**Вставить из Windows**	Вставляет содержимое буфера обмена Windows (если он содержит текст) в окно редактора исходного кода в текущую позицию курсора.

6.4.4. Меню Search

Меню **Search** вызывает диалоговые окна для поиска и замены, а также таблицу символов IDE.

Команда	Перевод	Описание
Find (CTRL-Q F)	**Найти**	Вызывает диалоговое окно поиска. Искомый текст можно ввести, а затем закрыть окно кнопкой OK. Начнётся поиск введённого текста в активном окне, начиная с текущей позиции курсора. Если текст найден, он будет выделен.
Replace (CTRL-Q A)	**Заменить**	Вызывает диалоговое окно поиска и замены. Искомый текст можно ввести, а затем закрыть окно кнопкой OK. Начнётся поиск введённого текста в активном окне, начиная с текущей позиции курсора. Если текст найден, он заменён.
Search again (CTRL-L)	**Повторить поиск**	Повторяет последнюю операцию поиска или поиска и замены, используя те же параметры.
Go to line number (ALT-G)	**Перейти к строке по номеру**	Вызывает окно, где можно ввести номер строки. После нажатия кнопки OK выполняет переход к указанной строке.

Когда программа и модули откомпилированы с просматриваемой информацией, то в меню станут доступны следующие команды:

Команда	Перевод	Описание
Find procedure	**Найти процедуру**	Пока не применяется.
Objects	**Объекты**	Запрашивает имя объекта и открывает для просмотра окно для этого объекта.
Modules	**Модули**	Запрашивает имя модуля и открывает для просмотра окно для этого модуля.
Globals	**Глобально**	Запрашивает имя глобального символа и открывает для просмотра окно для этого глобального символа.
Symbol	**Символ**	Открывает окно со всеми известными символами, где символ можно выбрать. После выбора символа откроется окно обозревателя для просмотра этого символа.

6.4.5. Меню Run

Меню **Run** содержит команды, связанные с запуском программы. Ниже описано назначение этих команд.

Команда	Перевод	Описание
Run (CTRL-F9)	**Пуск**	Если исходный код был изменён, то выполняется компиляция программы. Если компиляция завершена без ошибок, то выполняется пуск программы. Если первичный файл был установлен, то он используется для определения программы, которая будет запускаться. Подробнее о первичном файле см. в разделе «6.4.6. Меню Compile».
Step over (F8)	**Пошаговое выполнение**	Выполняет пошаговое выполнение программы, то есть при выборе этой команды выполняется текущая строка в исходном коде программы. Если в текущей строке находится вызов подпрограммы, то эта подпрограмма выполняется как единый блок исходного кода.
Trace into (F7)	**Трассировка**	Выполняет текущую строку исходного кода. Если в текущей строке находится вызов подпрограммы, то выполняется вход в эту подпрограмму и далее выполнение этой команды будет выполнять пошаговое выполнение исходного кода подпрограммы. То есть команда Step over выполняет строку исходного кода программы без входа в процедуры, а команда Trace into делает то же самое, но с заходом в процедуры.
Goto cursor (F4)	**Перейти к курсору**	Запускает и выполняет программу до того места, где в исходном коде находится курсор.
Until return	**До выхода**	Выполняет текущую процедуру до выхода из процедуры.
Run directory	**Запуск из каталога**	Устанавливает (изменяет) рабочую директорию, когда выполняется программа.
Parameters	**Параметры**	Позволяет ввести параметры, которые будут переданы в программу в начале выполнения.
Program reset (CTRL-F2)	**Сброс программы**	Если программа находится в режиме отладки, то сеанс отладки завершается, а запущенная программа принудительно закрывается. Может оказаться полезным при зависании программы.

6.4.6. Меню Compile

Меню **Compile** содержит команды, связанные с компиляцией программ или модулей.

Команда	Перевод	Описание
Compile (ALT-F9)	**Компиляция**	Компилирует содержимое активного окна, независимо от установок первичного файла.
Make (F9)	**Создание**	Компилирует содержимое активного окна и все файлы модулей, определённые в программе, если они были изменены с момента последней компиляции. Если первичный файл был установлен, то вместо этого будет компилироваться первичный файл.
Build	**Сборка**	Компилирует содержимое активного окна и все файлы модулей, определённые в программе, не зависимо от того, были они изменены с момента последней компиляции или нет. Если первичный файл был установлен, то вместо этого будет компилироваться первичный файл.
Target	**Цель**	Позволяет определить целевую операционную систему, для которой будет компилироваться программа.
Primary file	**Первичный файл**	Устанавливает первичный файл. Если первичный файл установлен, то любые команды выполнения или компиляции программы будут действовать на первичный файл, вместо файла, загруженного в активное окно. Первичный файл нет необходимости загружать в IDE.
Clear primary file	**Сбросить первичный файл**	Сбрасывает первичный файл. После этого команды выполнения и компиляции программы будут действовать на содержимое активного окна.
Compiler messages (F12)	**Сообщения компилятора**	Отображает окно сообщений компилятора. Это окно будет показывать сообщения, генерируемые компилятором в процессе последней компиляции.

6.4.7. Меню Debug

Меню **Debug** содержит команды для помощи в отладке программ, такие как определение точек остановки и отслеживание состояния переменных.

Команда	Перевод	Описание
Output	**Выход**	Показывает результат работы программы в окне.
User screen (ALT-F5)	**Экран пользователя**	Позволяет переключаться между экраном программы и окном редактора.
Add watch (CTRL-F7)	**Добавить элемент наблюдения**	Добавляет элемент (например, переменную) в список отслеживания. Элемент наблюдения – это выражение, которое может быть вычислено в IDE и будет отображаться в специальном окне. Обычно это значения нескольких переменных.
Watches	**Список отслеживания**	Отображает текущий список отслеживания значений в отдельном окне.
Breakpoint (CTRL-F8)	**Точка останова**	Устанавливает точку останова на текущей строке. В процессе отладки программа будет остановлена на этой точке.
Breakpoint list	**Список точек останова**	Отображает текущий список точек останова в отдельном окне.
Evaluate	**Вычисление**	Вызывает окно для вычисления выражений.
Call stack (CTRL-F3)	**Стек вызовов**	Показывает обращения к стеку. Стек вызовов – это список адресов (а также имена файлов, номера строк, если эта информация была откомпилирована) процедур, которые в настоящий момент вызываются из программы.
Disassemble	**Дизассемблер**	Вызывает окно дизассемблера.
Registers	**Регистры**	Отображает текущее содержимое регистров процессора.
Floating point unit	**Модуль плавающей точки**	Отображает текущее содержимое регистров FPU.
Vector unit	**Обработка векторов**	Отображает текущее содержимое регистров MMX (или эквивалентных).
GDB window	**Окно GDB**	Отображает консоль отладчика GDB. Это может быть использовано для взаимодействия с отладчиком напрямую. В этой консоли можно напечатать любую команду GDB и результат будет отображён в этом окне.

6.4.8. Меню Tools

Меню **Tools** содержит несколько стандартных инструментов. Если пользователь определит новый инструмент, то он будет добавлен в это меню.

Команда	Перевод	Описание
Messages (F11)	**Сообщения**	Отображает окно сообщений. В этом окне отображается результат работы какого-либо инструмента. Подробности см. в разделе 6.10.1.
Coto next (ALT-F8)	**Перейти вперёд**	Переход к следующему сообщению.
Goto previous (ALT-F7)	**Перейти назад**	Переход к предыдущему сообщению.
Grep (SHIFT-F2)	**Поиск выражений**	Быстро просматривает регулярные выражения и опции, переданные в команду, и затем выполняет команду grep (аналогично одноимённой команде в Unix) с данным выражением или опцией. Для работы этой команды в системе должна быть установлена программа grep, а путь к ней указан в переменной окружения PATH. Подробности см. в разделе 6.10.2.
Calculator	**Калькулятор**	Отображает калькулятор. Подробности см. В разделе 6.10.4.
Ascii table	**Таблица ASCII**	Отображает таблицу ASCII-символов. Подробности см. в разделе 6.10.3.

6.4.9. Меню Options

Меню **Options** содержит команды для вызова всех диалоговых окон, которые применяются при установке настроек компилятора и IDE по усмотрению пользователя.

Команда	Перевод	Описание
Mode	**Режим**	Вызывает диалоговое окно для установки режима работы компилятора. Текущий режим отображается в меню справа от слова Mode. Подробности см. в разделе 6.11.8.
Compiler	**Компилятор**	Вызывает диалоговое окно для установки общих настроек компилятора. Эти настройки используются при компиляции программы или модуля.
Memory sizes	**Размер памяти**	Вызывает диалоговое окно для установки размера стека и размера «кучи» для программы. Эти параметры используются при компиляции программы.
Linker	**Компоновщик**	Вызывает диалоговое окно для установки некоторых настроек компоновщика. Эти параметры используются при компиляции программы или библиотеки.
Debugger	**Отладчик**	Вызывает диалоговое окно для установки настроек отладчика. Эти параметры используются при компиляции модуля или программы. Учтите, что отладчик не работает, пока генерируется отладочная информация для программы.
Directories	**Директории**	Вызывает диалоговое окно для определения различных каталогов, необходимых для работы компилятора. Эти каталоги используются, когда программа или модуль откомпилированы.
Browser	**Обозреватель**	Вызывает диалоговое окно для установки настроек обозревателя. Эти настройки определяют поведение обозревателя символов в IDE.
Tools	**Инструменты**	Вызывает диалоговое окно для конфигурации меню инструментов. Подробности см. в разделе 6.10.5.
Environment	**Окружение**	Вызывает диалоговое окно для конфигурации поведения IDE. Ниже представлен список команд подменю, которые используются для этого.
Preferences (Предпочтения)		Основные настройки, такие как нужно ли автоматически сохранять файл или нет и какие файлы должны сохраняться. Также здесь можно установить видеорежим.
Editor (Редактор)		Управление различными настройками окон редактора.
CodeComplete (Завершение кода)		Используется для установки слов, которые могут быть автоматически дописаны при наборе этих слов в окне редактора.
CodeTemplates (Шаблоны кода)		Используется для определения шаблонов кода, которые могут быть автоматически вставлены в окно редактора.
Desktop (Рабочий стол)		Используется для управления поведением рабочего стола, то есть некоторые функции могут быть включены или выключены.
Keyboard & Mouse (Клавиатура и мышь)		Может использоваться для выбора условий вырезания/копирования/вставки, управления действиями мыши и назначения команд для различных действий мыши.
Learn keys (Обучаемые клавиши)		Позволяет IDE запомнить нажатия клавиш, для того, чтобы назначить им различные команды. Это полезно для Linux и Unix-подобных платформ, где актуальные нажатия клавиш отправляются в IDE путём терминальной эмуляции.
Open	**Открыть**	Вызывает диалоговое окно, где можно выбрать файл, содержащий настройки редактора. Если файл выбран, то после закрытия окна файл будет прочитан и хранящиеся в нём настройки применены.
Save	**Сохранить**	Сохраняет текущие настройки в файл по умолчанию.
Save as	**Сохранить как**	Сохраняет текущие настройки в альтернативный файл. Команда открывает диалоговое окно, где можно указать файл для сохранения настроек.

Учтите, что настройки не сохраняются автоматически. Вы должны сохранять настройки явно, с помощью команды `OPTIONS – SAVE`.

6.4.10. Меню Window

Меню **Window** предоставляет доступ к некоторым функциям окна редактора. Подробности см. в разделе «6.3. Окна».

Команда	Перевод	Описание
Tile	**Черепица**	Равномерно распределяет открытые окна по рабочему столу.
Cascade	**Каскад**	Располагает открытые окна каскадом.
Close all	**Закрыть всё**	Закрывает все открытые окна.
Size/move (CTRL-F5)	**Изменить размер/ переместить**	Переводит IDE в режим изменения размера/перемещения. После выполнения этой команды активное окно может быть перемещено с помощью кнопок со стрелками.
Zoom (F5)	**Увеличить**	Увеличивает или уменьшает текущее окно.
Next (F6)	**Следующее**	Активирует следующее окно в списке окон.
Previous (SHIFT-F6)	**Предыдущее**	Активирует предыдущее окно в списке окон.
Hide (CTRL-F6)	**Скрыть**	Скрывает активное окно.
Close (ALT-F3)	**Закрыть**	Закрывает активное окно.
List (ALT-0)	**Список**	Отображает список открытых окон. В этом списке любое окно можно активировать, закрыть, отобразить или скрыть.
Refresh display	**Обновить экран**	Перерисовывает экран.

6.4.11. Меню Help

Меню **Help** предоставляет доступ ко всем функциям помощи по IDE, а также возможность настройки справочной системы.

Команда	Перевод	Описание
Contents	**Содержание**	Отображает содержание справочной системы.
Index (SHIFT-F1)	**Индекс**	Индексы справочной системы.
Topic search (CTRL-F1)	**Поиск раздела**	Выполняет переход к разделу, связанному с текущим подсвеченным текстом.
Previous topic (ALT-F1)	**Предыдущий раздел**	Переход в раздел, который был просмотрен перед текущим.
Using help	**Использование справки**	Отображает инструкцию пользования справочной системой.
Files	**Файлы**	Позволяет конфигурировать меню HELP. С помощью этой команды файлы справки могут быть добавлены в справочную систему.
About	**О программе**	Отображает информацию об IDE. См. раздел 6.13.3.

Некоторые из разделов справки (или даже все) могут быть не установлены по причине экономии места на диске. В этом случае см. документацию в меню операционной системы **Free Pascal – Documentations**.

6.5. Редактирование текста

В этом разделе описаны основные принципы редактирования текста в редакторе исходного кода. IDE работает подобно многим другим текстовым редакторам, поэтому в данном разделе в основном описаны отличительные особенности IDE от других редакторов.

6.5.1. Режим вставки

Обычно IDE работает в режиме вставки. Это означает, что печатаемый текст будет вставляться перед текстом, который находится после курсора. В режиме замещения любой текст, находящийся после курсора, будет заменяться печатаемым текстом.

Если IDE находится в режиме вставки, то курсор имеет вид мигающей линии. Если IDE находится в режиме замещения, то курсор имеет вид мигающего прямоугольника, высота которого равна высоте строки. Переключение между режимами выполняется клавишей INSERT или комбинацией клавиш CTRL-V.

6.5.2. Блоки

Выделение текста в IDE выполняется таким же образом, как в Turbo Pascal IDE. Это незначительно отличается от выделения текста в Windows.

Текст может быть выделен тремя способами:

1. С помощью мыши, удерживая нажатой левую кнопку, перетаскивать мышь, выделяя требуемый участок текста.
2. С помощью клавиатуры отметить начало выделения, нажав комбинацию клавиш CTRL-K-B, затем переместить курсор в конец выделения и нажать комбинацию клавиш CTRL-K-K.
3. С помощью клавиатуры нажать и удерживать клавишу SHIFT, перемещаться по тексту с помощью клавиш со стрелками.

Имеются также специальные команды выделения:

1. Текущую строку можно выделить командой CTRL-K-L
2. Текущее слово можно выделить командой CTRL-K-T

В Free Pascal IDE выделенный текст остаётся выделенным даже после перемещения курсора с помощью клавиатуры, поэтому термин «блок» более подходит для выделения, и далее мы будем называть выделение текста термином «блок».

Для работы с блоком могут быть использованы несколько команд:

- Перемещение блока к курсору (CTRL-K-V)
- Копирование блока в место нахождения курсора (CTRL-K-C)
- Удаление блока (CTRL-K-Y)
- Запись блока в файл (CTRL-K-W)
- Чтение содержимого файла в блок (CTRL-K-R). Если этот блок уже есть, то он не заменяется этой командой. Файл вставляется в текущую позицию курсора, а затем вставленный текст выделяется.
- Смещение блока вправо (CTRL-K-I)
- Смещение блока влево (CTRL-K-U)
- Печать содержимого блока (CTRL-K-P)

При поиске и замене поиск может быть ограничен содержимым блока.

6.5.3. Настройки закладок

IDE предоставляет функции, позволяющие вам устанавливать закладки в текущей позиции курсора. Затем можно быстро вернуться к этой позиции из любого места редактора с помощью комбинации клавиш.

В исходном файле можно установить до 9 закладок. Закладка устанавливается комбинацией клавиш CTRL-K-ЦИФРА (здесь цифра – это цифровая клавиша от 1 до 9). Для перехода к предварительно установленной закладке нажмите CTRL-Q-ЦИФРА.

ПРИМЕЧАНИЕ
В настоящий момент закладки не сохраняются при выходе из IDE. Это может измениться в будущих версиях.

6.5.4. Переход к строке

Имеется возможность перейти непосредственно к указанной строке исходного кода. Для этого откройте окно перехода к строке командой меню `SEARCH – GOTO LINE NUMBER`.

В появившемся окне нужно ввести номер строки и нажать клавишу `ENTER` или щёлкнуть кнопку `OK`. Окно перехода к строке показано на рис. 6.4.

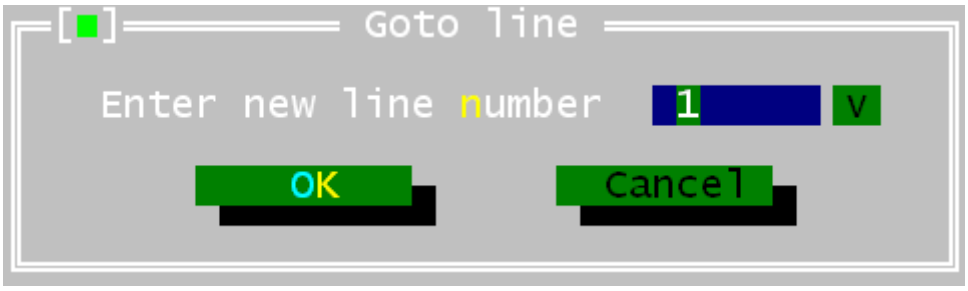


Рис. 6.4. Окно перехода к строке.

6.5.5. Подсветка синтаксиса

IDE способна выполнять подсветку синтаксиса, то есть раскрашивать различными цветами элементы языка Pascal. Когда текст вводится в редакторе исходного кода, IDE пытается распознать элементы языка и установить для них соответствующий цвет.

Подсветка синтаксиса может быть настроена по желанию пользователя, используя команду меню `OPTIONS – ENVIRONMENT – COLORS` (в некоторых версиях компилятора эта команда отсутствует). В окне настроек цветов должна быть выбрана группа `Syntax` (Синтаксис). В списке элементов этой группы отображаются различные синтаксические элементы, которые могут быть раскрашены в свой цвет:

Команда	Перевод	Описание
Whitespace	**Свободное место**	Пустое пространство между словами. Учтите, что для свободного места за текстовыми блоками будет использоваться цвет фона.
Comments	**Комментарии**	Комментарии всех стилей в Free Pascal.
Reserved words	**Ключевые слова**	Все зарезервированные слова Free Pascal (см. также документ «Справочная информация»).
Strings	**Строки**	Выражения со строковыми константами.
Numbers	**Числа**	Десятичные числа.
Hex numbers	**16-ричные числа**	Числа в шестнадцатиричном представлении.
Assembler	**Ассемблер**	Любые ассемблерные блоки.
Symbols	**Символы**	Распознаваемые символы (переменные, типы).
Directives	**Директивы**	Директивы компилятора.
Tabs	**Табуляция**	Символы табуляции в исходном коде могут иметь цвет, отличный от цвета свободного места.

Редактор использует настройки по умолчанию. Но экспериментально вы можете подобрать для себя наиболее удобные настройки цветов. Хорошая цветовая схема помогает быстрее найти ошибки в исходном коде, так как результатом ошибки будет неправильная подсветка синтаксиса.

6.5.6. Завершение кода

Завершение кода означает, что редактор будет пытаться угадать текст, который вы печатаете. Он делает это путём проверки части текста (слова), которую вы уже напечатали. Как только фрагмент слова, уже напечатанный вами, можно будет использовать для идентификации ключевого слова из списка ключевых слов, то ключевое слово полностью будет отображено в небольшом подсвеченном прямоугольнике, сразу под печатаемым текстом. Нажатие клавиши ENTER завершит ввод текста, то есть автоматически вставит это слово в текст, и вам не придётся печатать это слово полностью.

Эта функция не является аналогом завершения кода в Delphi или Lazarus, когда происходит заполнение аргументов функции или выбор методов объекта.

Функция завершения кода может быть настроена в соответствующем окне, вызвать которое можно командой меню OPTIONS – ENVIRONMENT – CODECOMPLETE. Список ключевых слов, ввод которых может быть завершён, можно изменить вручную в этом окне. Окно завершения кода показано на рис. 6.5.

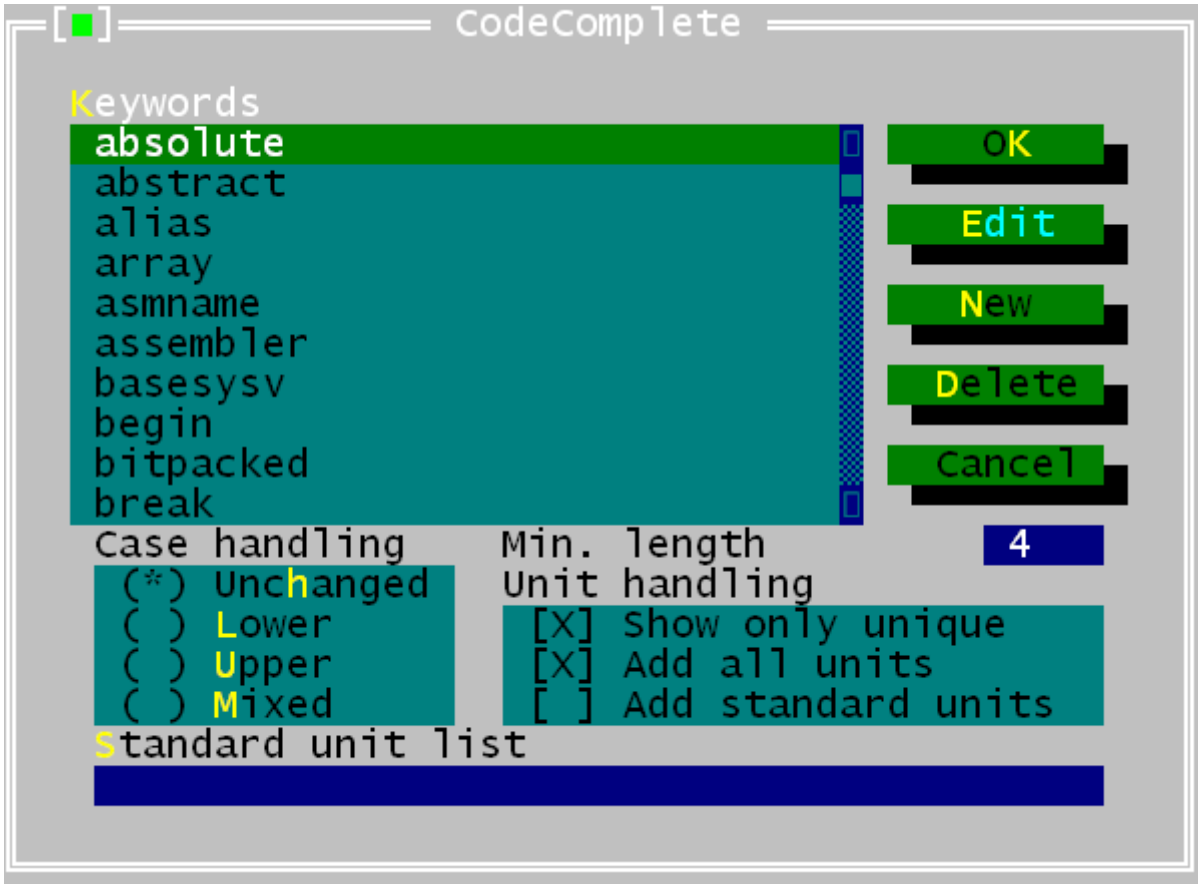


Рис. 6.5. Окно настройки завершения кода.

В этом окне в алфавитном порядке определен текущий список ключевых слов, которые доступны для завершения.

После сохранения все ключевые слова будут доступны при следующем старте IDE. Дублирование ключевых слов в списке не допускается. Если попытаться добавить в список уже существующее слово, то будет выдано сообщение об ошибке.

Ниже описаны кнопки окна настроек завершения кода.

Кнопка	Перевод	Описание
OK	**OK**	Сохраняет все изменения и закрывает окно.
Edit	**Редактировать**	Вызывает окно, в котором можно изменить выбранное ключевое слово.
New	**Новый**	Вызывает окно, в котором можно ввести новое ключевое слово и добавить его в список.
Delete	**Удалить**	Удаляет выделенное ключевое слово из списка.
Cancel	**Отмена**	Отменяет все изменения и закрывает окно.

6.5.7. Шаблоны кода

Шаблоны кода позволяют вставлять большие куски кода за один раз. Каждый шаблон кода идентифицируется уникальным именем. Это имя можно использовать для вставки связанного с ним шаблона в текст.

Например, имя **ifthen** связано со следующим куском кода:

```
If | Then
begin
end
```

Вставить шаблон кода в текст можно так: напечатать его имя, а затем нажать комбинацию клавиш CTRL-J, если курсор находится справа от имени шаблона.

Если курсор находится перед именем шаблона, то будет открыто окно для выбора шаблона (в некоторых версиях эта функция работает некорректно и программа зависает, также программа зависает, если напечатано несуществующее имя шаблона – примечание переводчика).

Вертикальная черта (|) в шаблоне кода означает, что при вставке шаблона в текст на это место будет помещён курсор. В тексте вертикальной черты не будет. В описанном выше примере курсор будет установлен между словами **if** и **then**, где можно будет сразу напечатать выражение.

Шаблон кода можно добавить или отредактировать в окне шаблонов кода (рис. 6.6). Окно шаблонов кода можно вызвать через меню OPTIONS – ENVIRONMENT – CODETEMPLATES.

Верхний список в окне шаблонов отображает имена всех известных шаблонов. В нижней части окна отображается текст шаблона, который выбран в данный момент в списке имён. Ниже описаны кнопки окна шаблонов:

Кнопка	Перевод	Описание
OK	**OK**	Сохраняет все изменения и закрывает окно.
Edit	**Редактировать**	Вызывает окно, в котором можно изменить выбранный шаблон.
New	**Новый**	Вызывает окно, в котором можно ввести новый шаблон и добавить его в список.
Delete	**Удалить**	Удаляет выделенный шаблон из списка.
Cancel	**Отмена**	Отменяет все изменения и закрывает окно.

После сохранения все шаблоны будут доступны при следующем старте IDE (имеется ввиду сохранение командой меню OPTIONS – SAVE).

Дублирование имён шаблонов в списке не допускается. Если попытаться добавить в список уже существующее имя, то будет выдано сообщение об ошибке.

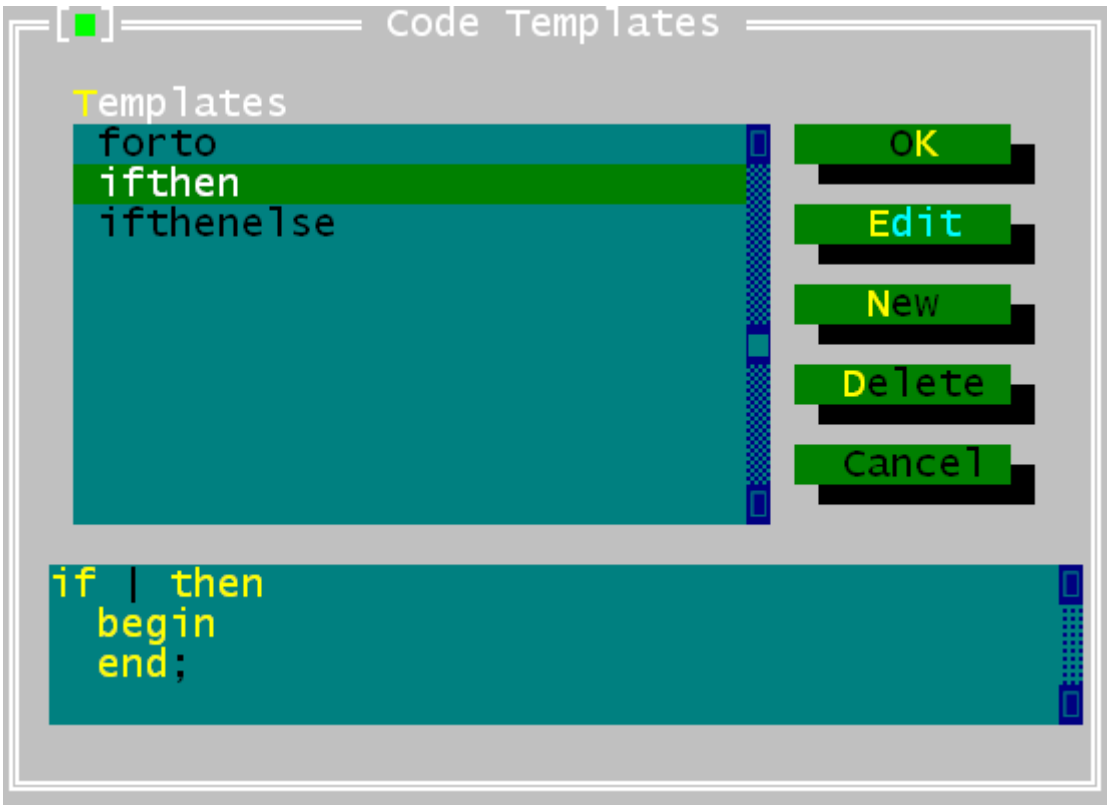


Рис. 6.6. Окно шаблонов кода.

6.6. Поиск и замещение

IDE имеет функцию поиска текста в активном окне редактора.

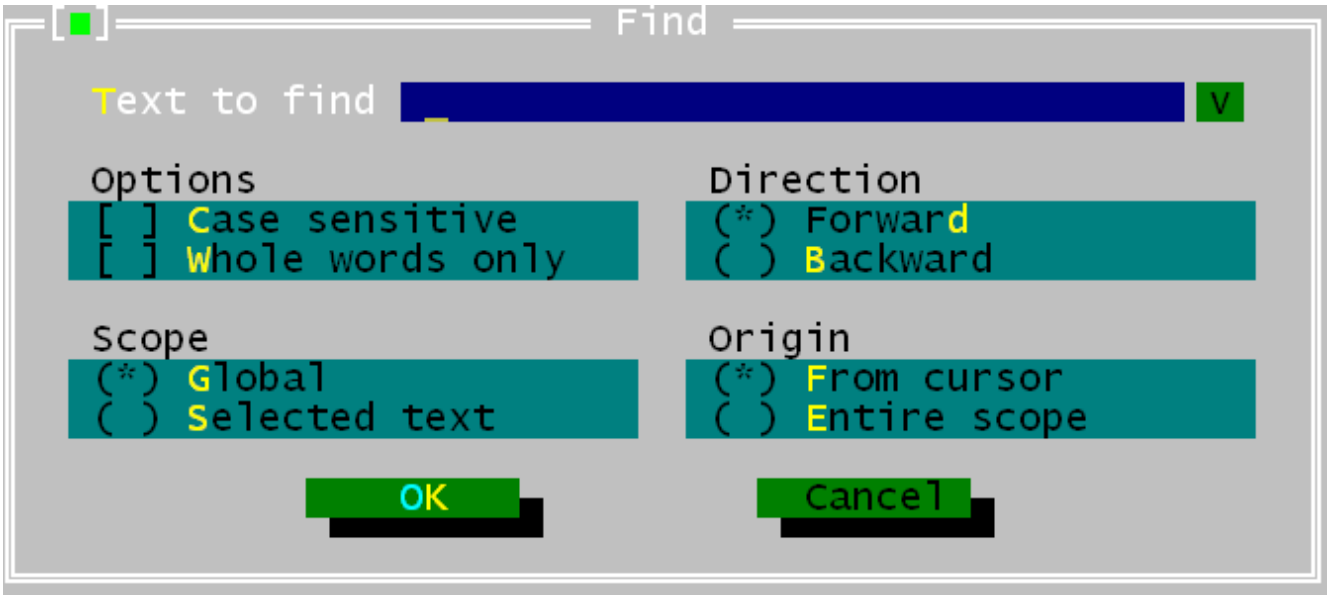


Рис. 6.7. Окно поиска Free Pascal IDE.

Поиск текста можно выполнить одним из следующих способов:

1. Выбрать в меню `SEARCH – FIND`
2. Нажать `CTRL-Q-F`

После этого появится диалоговое окно (рис. 6.7), где можно ввести следующие параметры:

Параметр	Перевод	Описание
Text to find	**Текст для поиска**	Текст, который нужно найти. Если курсор был установлен на какое-либо слово, то данное слово автоматически подставляется в это поле.
Case sensitive	**Учитывать регистр**	Если установлен, то поиск текста выполняется с учётом регистра символов.
Whole words only	**Только слово целиком**	Если установлен, то выполняется поиск целого слова, а не его части.
Direction	**Направление**	Поиск выполняется в одном из двух направлений: вперёд по тексту (Forward) или назад (Backward).
Scope	**Диапазон**	Диапазон поиска. Может быть двух видов: во всём тексте (Global) или только в выделенном фрагменте текста (Selected text).
Origin	**Начало поиска**	Определяет начало поиска. Может принимать одно из двух значений: от текущей позиции курсора (From cursor) или от начала границы диапазона поиска (Entire scope).

После закрытия окна выполняется поиск указанного текста с заданными параметрами.

Поиск можно повторить (используя те же параметры) одним из двух способов:

1. Выбрать в меню `SEARCH – SEARCH AGAIN`
2. Нажать `CTRL-L`

Также имеется функция замены найденного текста другим текстом, которая похожа на поиск текста:

1. Выбрать меню `SEARCH – REPLACE`
2. Нажать `CTRL-Q-A`

Появится окно, похожее на окно поиска (рис. 6.8).

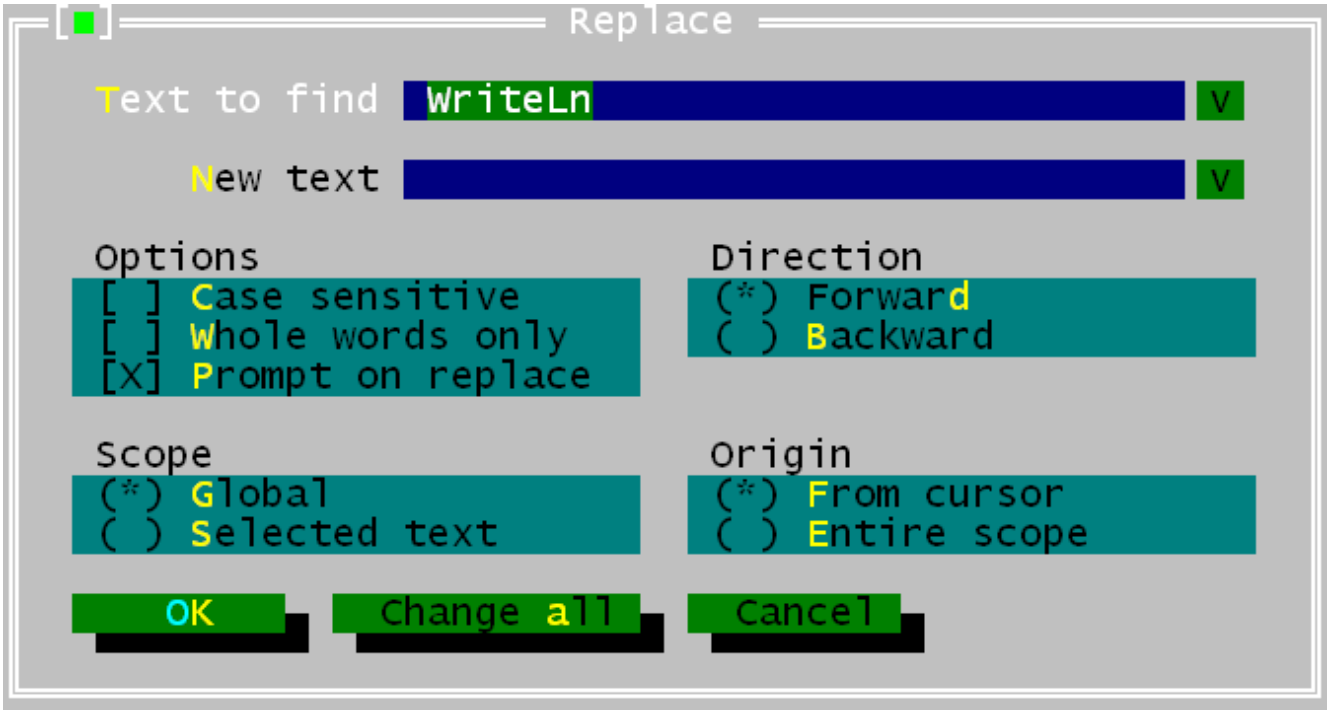


Рис. 6.8. Окно поиска и замены.

В этом окне, кроме параметров, имеющихся в окне поиска, добавлены следующие опции:

Параметр	Перевод	Описание
New text	**Новый текст**	Текст, который будет вставлен вместо найденного текста.
Prompt on replace	**Запрос на замену**	Перед заменой IDE будет запрашивать у пользователя подтверждение.

Если окно закрыто кнопкой `OK`, то будет заменён только следующий найденный текст. Если окно закрыто кнопкой «`Change all`» (заменить всё), то весь найденный текст будет заменён.

6.7. Обозреватель идентификаторов

Обозреватель идентификаторов позволяет выполнить поиск всех имеющихся идентификаторов. Идентификаторами могут быть переменные, типы, процедуры или константы, которые имеются в программе или модуле.

Для включения обозревателя идентификаторов, программа или модуль должны быть откомпилированы с информацией обозревателя. Эта опция устанавливается в настройках компилятора.

IDE позволяет просматривать следующие типы идентификаторов:

Символ	Перевод	Описание
Procedure	**Процедура**	Позволяет быстро перейти к объявлению или телу процедуры.
Objects	**Объекты**	Быстрый просмотр объекта.
Modules	**Модули**	Просмотр модуля.
Globals	**Глобально**	Просмотр любых глобальных символов.
Arbitrary symbol	**Произвольный символ**	Просмотр произвольного символа.

Во всех случаях сначала должен быть выбран идентификатор для просмотра. После этого появится окно обозревателя. В окне обозревателя отображаются все размещения, где перечислены идентификаторы. Чтобы быстро перейти в нужное место, нужно выбрать размещение идентификатора и нажать пробел. Строка, содержащая идентификатор, будет подсвечена.

Если окно с исходным файлом, где расположен идентификатор, не открыто, то будет открыто новое окно, в которое загрузится исходный текст. Затем окно можно закрыть обычным способом.

Поведение обозревателя можно настроить в окне настроек обозревателя идентификаторов, которое вызывается из меню `OPTIONS – BROWSER`. Это окно показано на рис. 6.9. Здесь можно настроить следующие параметры:

Symbols (Символы)		
Параметр	**Перевод**	**Описание**
Labels	**Метки**	Отображать метки.
Constants	**Константы**	Отображать константы.
Types	**Типы**	Отображать типы.
Variables	**Переменные**	Отображать переменные.
Procedures	**Процедуры**	Отображать процедуры.
Inherited	**Потомки**	Отображать потомков.

Sub-browsing (Подчинённый обозреватель). Определяет, каким образом обозреватель должен отображать элементы составных идентификаторов, таких как записи или классы.		
Параметр	**Перевод**	**Описание**
New browser	**Новое окно**	Элементы отображаются в новом окне обозревателя.
Replace current	**Заменять текущие**	Содержимое текущего окна заменяется элементами выбранного комплексного идентификатора.

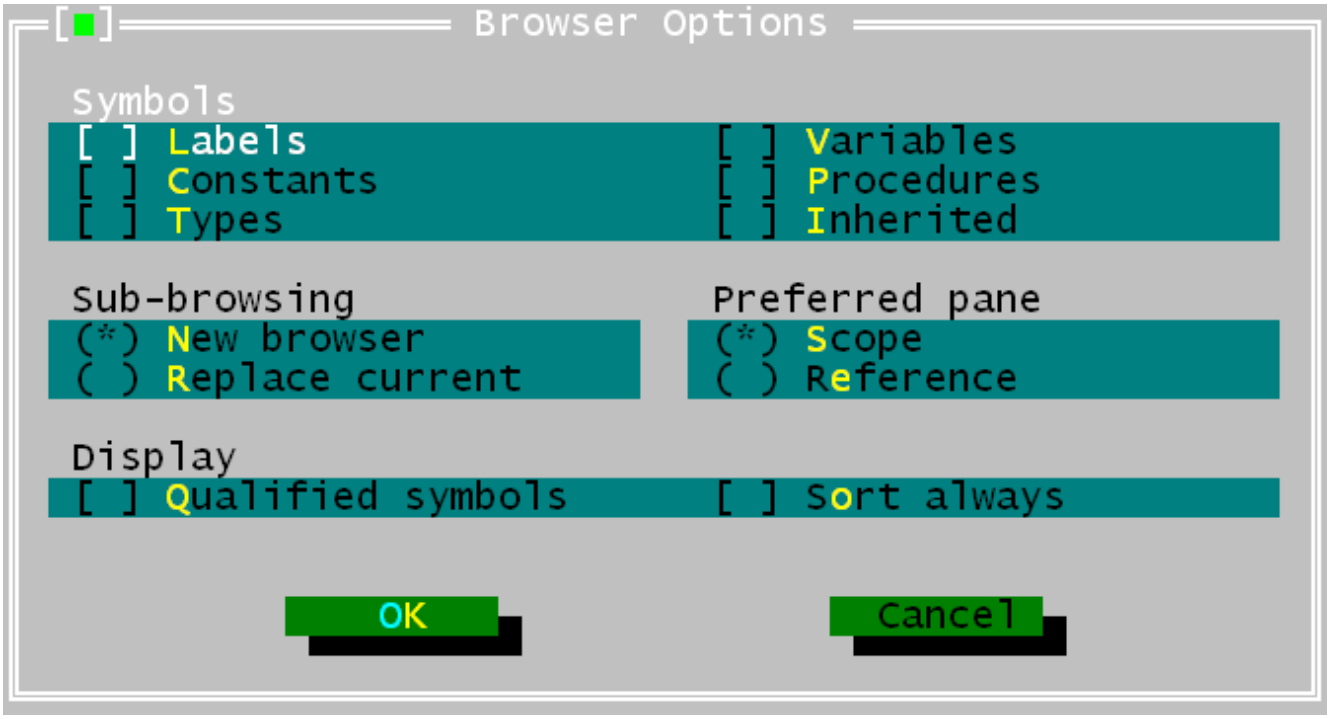


Рис. 6.9. Окно настроек обозревателя идентификаторов.

Preferred pane (Предпочтительная область). Определяет, как отображать обозреватель при открытии.		
Параметр	**Перевод**	**Описание**
Scope	**Область**	
Reference	**Ссылка**	

Display (Экран). Определяет, как обозреватель должен отображать идентификаторы.		
Параметр	**Перевод**	**Описание**
Qualified symbols	**Квалифицировать символы**	
Sort always	**Сортировать всегда**	Сортировать символы в окне обозревателя.

6.8. Запуск программ

Откомпилированная программа может быть запущена на выполнение из IDE. Это можно сделать следующими способами:

1. Выбрать меню `RUN – RUN`
2. Нажать `CTRL-F9`

Если в программу необходимо передать параметры, то это можно сделать через меню `RUN – PARAMETERS`. Окно ввода параметров показано на рис. 6.10.

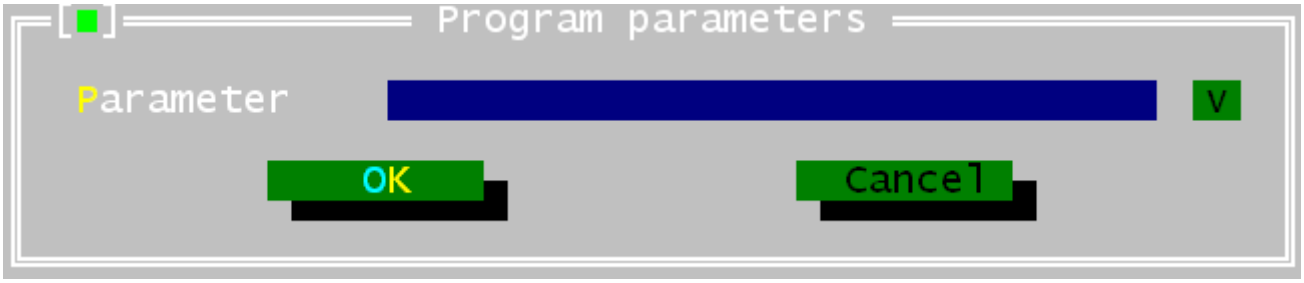


Рис. 6.10. Окно ввода параметров программы.

Запущенная программа будет выполняться до тех пор, пока

1. не выполнен нормальный выход из программы,
2. не случится ошибка,
3. не достигнута точка останова или
4. программа будет сброшена пользователем.

Последняя альтернатива возможна только в том случае, если программа откомпилирована с отладочной информацией.

Кроме того, можно установить где-нибудь в исходном коде курсор, и выполнить программу до этого места. Такой запуск программы можно осуществить следующими способами:

1. Выбрать в меню `RUN – GOTO CURSOR`
2. Нажать `F4`

Опять же это возможно только в том случае, если программа откомпилирована с отладочной информацией.

Можно также выполнять программу строку за строкой. Нажатие клавиши `F8` выполняет следующую строку программы. Если программа ещё не запущена, то будет выполнен запуск программы. Повторные нажатия клавиши `F8` будут выполнять программу строка за строкой (шаг за шагом), а IDE будет отображать выполняемую строку в редакторе исходного кода. Если в какой-либо строке находится вызов подпрограммы, то нажатие клавиши `F8` будет выполнять подпрограмму целиком, то есть без захода в тело подпрограммы (подпрограмма выполняется перед возвратом в IDE). Если необходимо выполнить код подпрограммы в пошаговом режиме, то вместо клавиши `F8` нужно нажимать клавишу `F7`. Использование клавиши `F7` позволяет выполнять подпрограммы строка за строкой.

Если выполнение подпрограммы начато в пошаговом режиме, то командой меню `RUN – UNTIL RETURN` можно выполнить программу до конца текущей подпрограммы, то есть не придётся нажимать клавишу `F7`, до тех пор, пока не закончится выполнение подпрограммы.

Принудительно завершить программу, не дожидаясь её окончания, можно следующими способами:

1. Выбрать в меню `RUN – PROGRAM RESET`
2. Нажать `CTRL-F2`

Программа будет закрыта (работает не во всех случаях – примечание переводчика).

6.9. Отладка программ

Для того чтобы иметь возможность отладки, программа должна компилироваться с отладочной информацией. Компиляция программы с отладочной информацией позволяет также:

1. Выполнять программу в пошаговом режиме
2. Использовать точки останова
3. Проверять содержимое переменных или распределение памяти во время выполнения программы

6.9.1. Использование точек останова

Точки останова используются в программе для тех случаев, когда необходимо остановить выполнение программы в определённом месте. Когда программа достигает точки останова, она останавливается и передаёт управление в IDE. Затем можно продолжить выполнение программы с места останова.

Чтобы установить точку останова на текущей строке кода, выберите в меню `DEBUG – BREAKPOINT` или нажмите `CTRL-F8`. Список текущих точек останова можно получить через команду меню `DEBUG – BREAKPOINT LIST`. Окно списка показано на рис. 6.11.

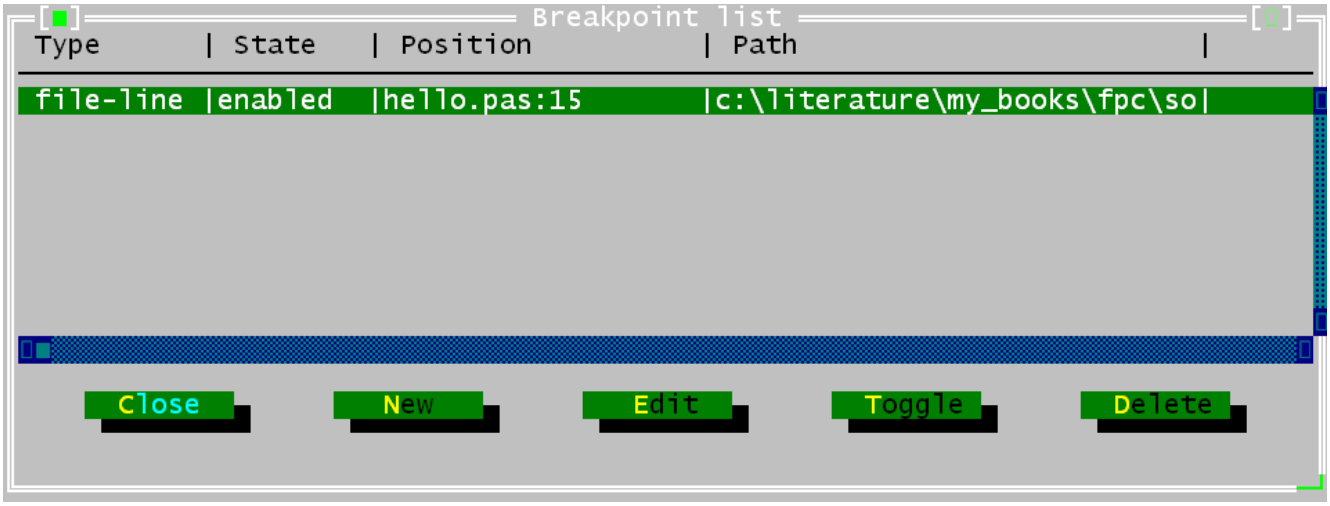


Рис. 6.11. Окно списка точек останова.

Элементы окна списка точек останова описаны ниже.

Элемент	Перевод	Описание
New	**Новая**	Отображает диалоговое окно свойств точки останова, где можно ввести параметры для новой точки останова.
Edit	**Редактировать**	Отображает диалоговое окно свойств точки останова, где можно изменить параметры выбранной точки останова.
Delete	**Удалить**	Удаляет выбранную точку останова.
Toggle	**Переключатель**	Включает или выключает точку останова, то есть изменяет состояние (графа State на рис. 6.11) на противоположное.
Close	**Закрыть**	Закрывает окно.

Окно параметров точки останова показано на рис. 6.12. Здесь можно установить следующие параметры:

Параметр	Перевод	Описание
Type	**Тип**	Тип точки останова. Существуют следующие типы:
Function (Функция)		Программа будет остановлена при обнаружении функции с указанным именем.
file-line (Строка файла)		Программа будет остановлена при обнаружении в файле с указанным именем и номером строки.
watch (выражение)		Можно ввести выражение, и программа будет остановлена, как только это выражение будет изменено.
awatch (доступ к выражению)		Можно ввести выражение, которое ссылается на место в памяти, и программа будет остановлена, как только будет выполнен доступ к этому месту в памяти.
address (адрес)		Остановка программы при получении адреса.
rwath (чтение выражения)		Можно ввести выражение, которое ссылается на место в памяти, и программа будет остановлена, как только будет выполнено чтение этого участка памяти.
Name	**Имя**	Имя функции или файла.
Conditions	**Условия**	Здесь можно ввести выражение, которое должно быть ИСТИННО, чтобы программа остановилась. Вводимое выражение должно быть правильным GDB-выражением.
Line	**Строка**	Номер строки в файле, на которой программа должна остановиться.
Ignore count	**Количество пропусков**	Это значение определяет, сколько раз программа проигнорирует точку останова, прежде чем остановиться.

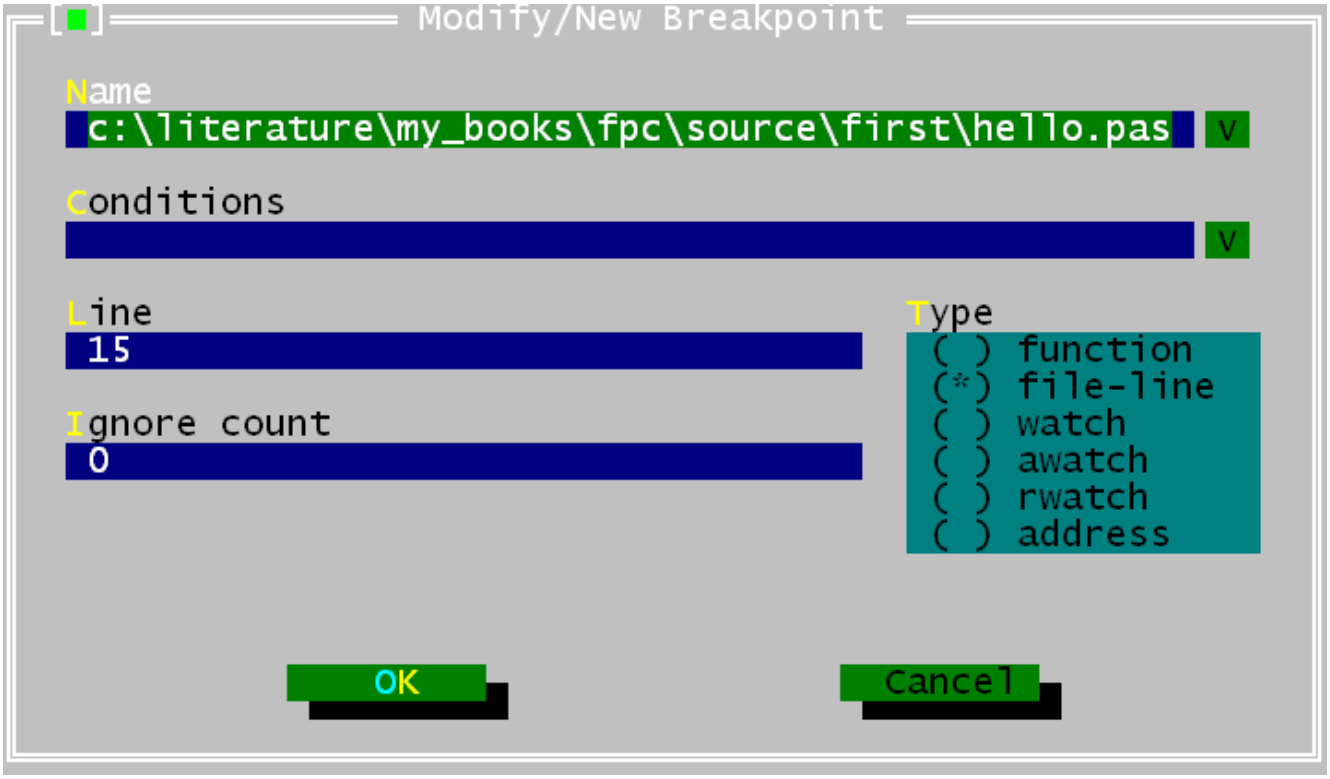


Рис. 6.12. Окно параметров точки останова.

ПРИМЕЧАНИЯ

1. Так как для отладки IDE использует GDB, то все выражения необходимо вводить в верхнем регистре.
2. Выражения, которые ссылаются на место в памяти, должны быть не более 16 байтов на Linux или go32v2 для процессоров Intel, так как отладочные регистры процессоров Intel используются для мониторинга памяти.
3. Отслеживание местоположения памяти не работает на Win32 без применения специальных «патчей».

6.9.2. Отслеживание выражений

Если программа откомпилирована с отладочной информацией, то можно использовать функцию отслеживания выражений, которые могут быть вычислены IDE и отображены в отдельном окне. Когда выполнение программы останавливается (например, на точке останова), то все наблюдаемые выражения вычисляются, и отображаются значения этих выражений.

Создать новое наблюдаемое выражение можно через меню `DEBUG – ADD WATCH` или комбинацией клавиш `CTRL-F7`. При этом появится окно редактирования наблюдаемых выражений (рис. 6.13), где можно ввести новое выражение.

В этом же окне отображаются любые предыдущие и текущее значения выражения.

ПРИМЕЧАНИЕ
Так как IDE использует GDB для отладки, необходимо вводить все выражения в верхнем регистре в FreeBSD.

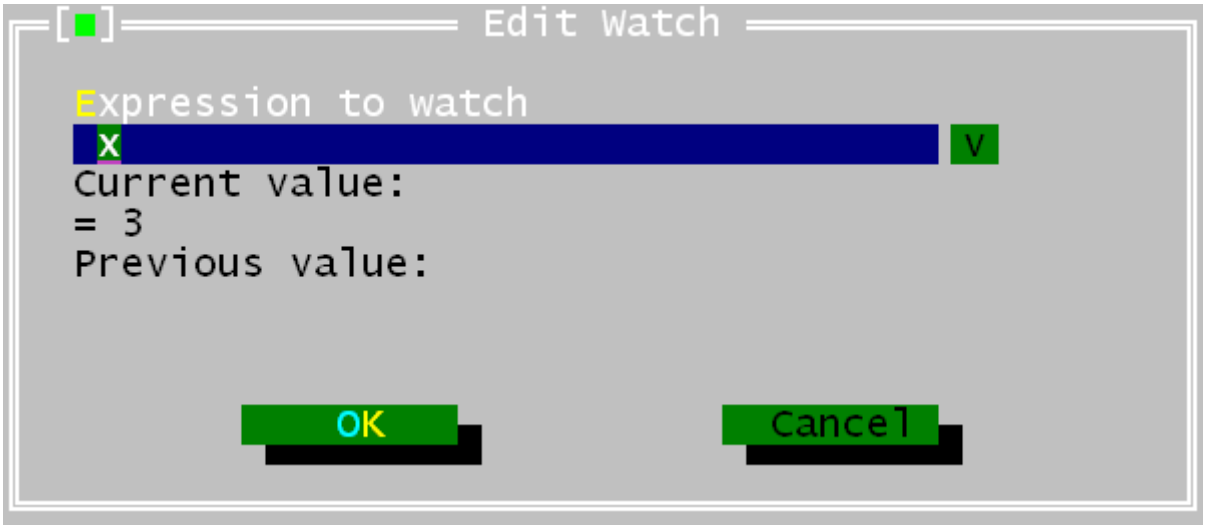


Рис. 6.13. Окно редактирования наблюдаемого выражения.

Список выражений и их значений можно увидеть в окне наблюдаемых выражений (рис. 6.14). Это окно вызывается командой меню `DEBUG – WATCHES`.

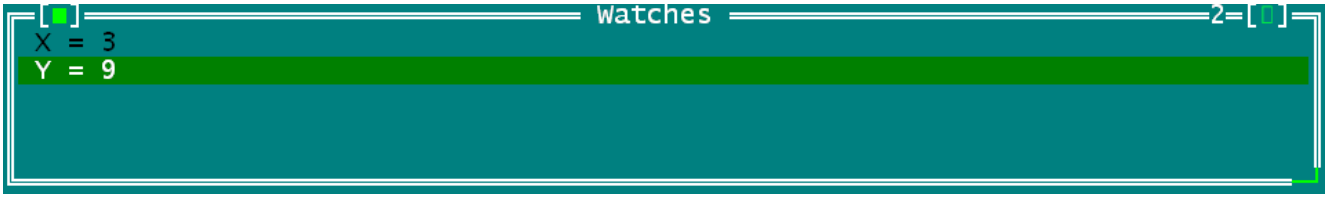


Рис. 6.14. Список наблюдаемых выражений.

Нажатие клавиши `ENTER` или пробела вызовет окно параметров выражения, которое выделено в данный момент в списке выражений (на рис. 6.14 выделена переменная `Y`).

Список выражений обновляется каждый раз, когда IDE получает управление, если программа находится в режиме отладки.

6.9.3. Стек вызовов

Стек вызовов помогает просматривать ход выполнения программы. Он отображает список процедур, которые вызываются в данный момент. Процедуры отображаются в обратном порядке. Окно стека вызовов можно отобразить с помощью меню `DEBUG – CALL STACK`. Здесь отображаются адреса или имена всех активных в текущий момент подпрограмм с их именами файлов и адресами. Если в подпрограмму передаются параметры, то они также отображаются. Окно стека вызовов показано на рис. 6.15.

ПРИМЕЧАНИЕ ПЕРЕВОДЧИКА
Как и в случае с окном наблюдаемых выражений, сначала нужно перевести программу в режим отладки, и только потом вызывать окно наблюдаемых выражений или окно стека вызовов, иначе при пуске программы могут появляться сообщения об ошибках.

Рис. 6.15. Окно стека вызовов.

Если в окне стека вызовов нажать пробел, то связанная элементом стека строка будет подсвечена в окне редактора исходного кода.

6.9.4. Окно GDB

Окно GDB предоставляет возможность прямого управления отладчиком GDB. Здесь команды отладчика могут быть напечатаны таким же образом, как и непосредственно в GDB. Ответ GDB будет также отображаться в этом окне.

Более подробную информацию по использованию отладчика GDB можно найти в разделе «10.2. Использование gdb для отладки вашей программы», но последние данные о GDB лучше искать в руководстве по GDB непосредственно на сайте Free Software Foundation. Окно отладчика GDB показано на рис. 6.16.

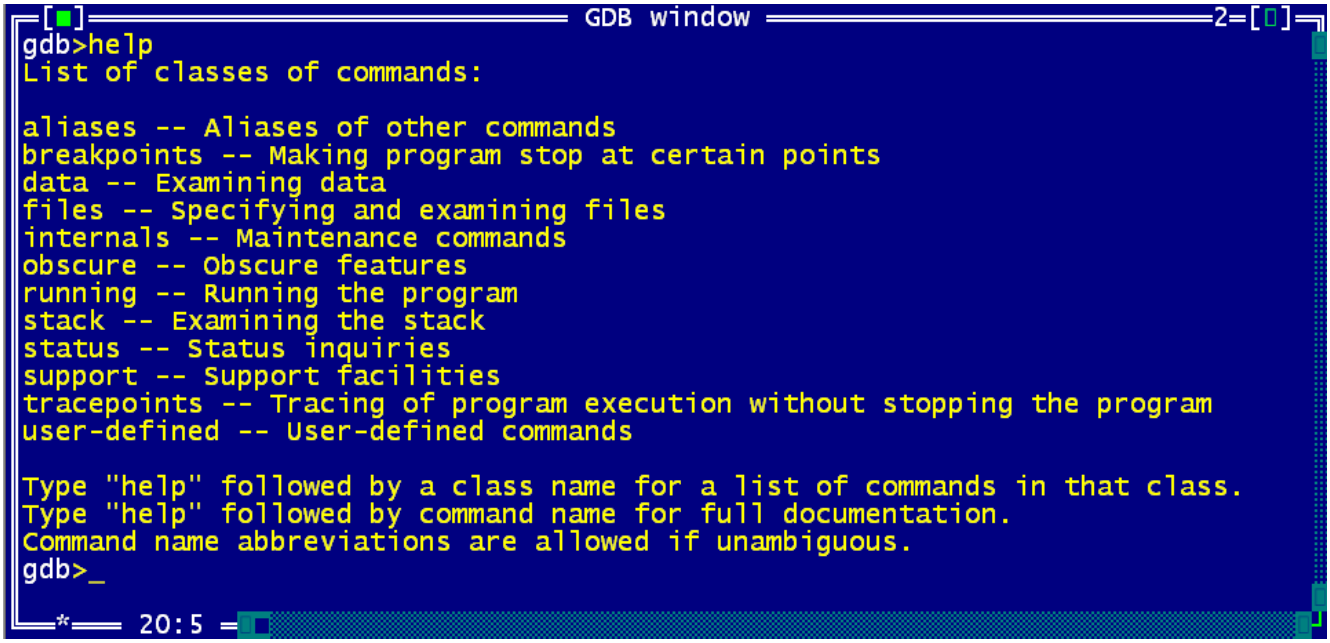


Рис. 6.16. Окно отладчика GDB.

6.10. Использование инструментов

Меню TOOLS предоставляет лёгкий доступ к внешним инструментам. Здесь также имеется три предопределённых инструмента программиста: ASCII-таблица, программа **grep** и калькулятор. Доступ к внешним инструментам также можно получить через это меню.

6.10.1. Окно сообщений

Результат работы внешних инструментов перенаправляется в IDE и отображается в окне сообщений. Окно сообщений (рис. 6.17) отображается автоматически, когда запущен внешний инструмент. Окно сообщений также можно вызвать вручную, выбрав в меню TOOLS – MESSAGES или нажав клавишу F11.

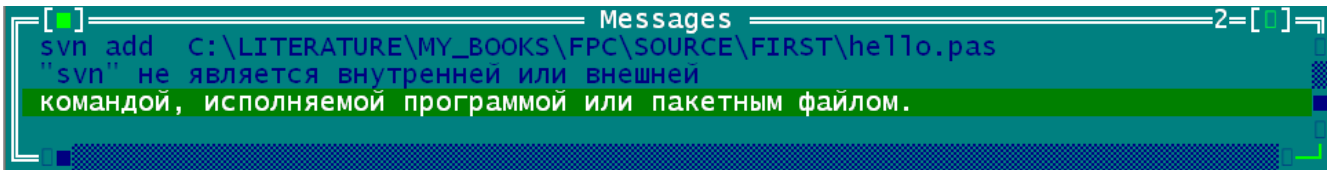


Рис. 6.17. Окно сообщений.

Если в выходных данных инструмента содержатся имена файлов и номера строк, то окно сообщений можно использовать для навигации по исходному коду так же как и при работе с окном стека вызовов:

1. Нажатие клавиши ENTER или двойной щелчок по нужной строке в окне сообщений приведёт к быстрому переходу в указанный участок исходного кода и закрытию окна сообщений.
2. Нажатие пробела также приведёт к переходу в указанный участок исходного кода, но окно сообщений не потеряет фокус и останется на экране. Это позволит выбрать другое сообщение с помощью клавиш управления курсором и перейти к нему.

Алгоритмы, которые извлекают имена файлов и номера строк из выходных данных инструмента довольно совершенны, но в некоторых случаях они могут не работать (рекомендации по улучшению этих алгоритмов всегда приветствуются).

6.10.2. Grep

Один внешний инструмент, имеющийся в меню TOLLS, уже предопределён: это утилита **grep**, которую можно вызвать командой меню TOOLS – GREP или комбинацией клавиш SHIFT-F2. Эта утилита выполняет поиск указанного текста в файлах и возвращает строки, содержащие этот текст. Искомая строка может быть регулярным выражением. Чтобы данная команда меню работала, в системе должна быть установлена программа **grep**, так как она не поставляется в комплекте с Free Pascal.

Окно сообщений можно использовать для комбинированного с **grep** поиска специальных участков текста.

Утилита **grep** поддерживает регулярные выражения. Регулярное выражение – это строка со специальными символами, которые описывают целый класс выражения. Командная строка DOS или LINUX имеет ограниченную поддержку регулярных выражений: ввод

```
ls *.pas (или dir *.pas)
```

позволяет получить список всех файлов Паскаля в директории. Здесь ***.pas** – это нечто похожее на регулярное выражение. Оно использует маску для описания целого класса строк: строки, которые заканчиваются на ***.pas**. Регулярные выражения часто бывают более сложными, например, [A-Z][0-9]+ описывает все строки, которые начинаются с прописной буквы латинского алфавита, за которой следует одна или более цифр.

Описание регулярных выражений выходит за рамки данного документа. Пользователи LINUX могут получить более подробную информацию об использовании утилиты **grep**, набрав в командной строке **man grep**.

6.10.3. Таблица ASCII-символов

Меню инструментов также предоставляет таблицу ASCII-символов (`TOOLS – ASCII TABLE`). Эту таблицу (рис. 6.18) можно использовать для просмотра кодов ASCII, а также для вставки символа из таблицы в активное окно редактора.

Рис. 6.18. Таблица ASCII-символов.

Чтобы посмотреть код символа, переместите курсор на этот символ в таблице с помощью клавиш со стрелками или щёлкните по символу кнопкой мыши. Код выбранного символа отображается в нижней части таблицы в десятичном и шестнадцатеричном представлениях.

Вставить символ в редактор можно следующими способами:

1. Двойным щелчком на нужном символе
2. Нажатием клавиши `ENTER`
3. Нажатием комбинации `CTRL-ENTER` (в некоторых версиях первые два способа не работают – примечание переводчика)

Использование таблицы особенно удобно для вставки символов псевдографики в строковые константы.

Окно таблицы ASCII-символов остаётся активным до тех пор, пока не будет активировано другое окно, поэтому за один раз можно вставить множество символов.

6.10.4. Калькулятор

Калькулятор позволяет быстро выполнять вычисления без выхода из IDE. Это простой калькулятор, который не учитывает приоритет операторов, и не поддерживает операторы, заключённые в скобки.

Результат вычислений можно вставить в текст активного окна с помощью комбинации клавиш `CTRL-ENTER`. Окно калькулятора показано на рис. 6.19.

Калькулятор поддерживает все основные математические операции, такие как сложение, вычитание, деление и умножение. Список операций приведён в таблице 6.1.

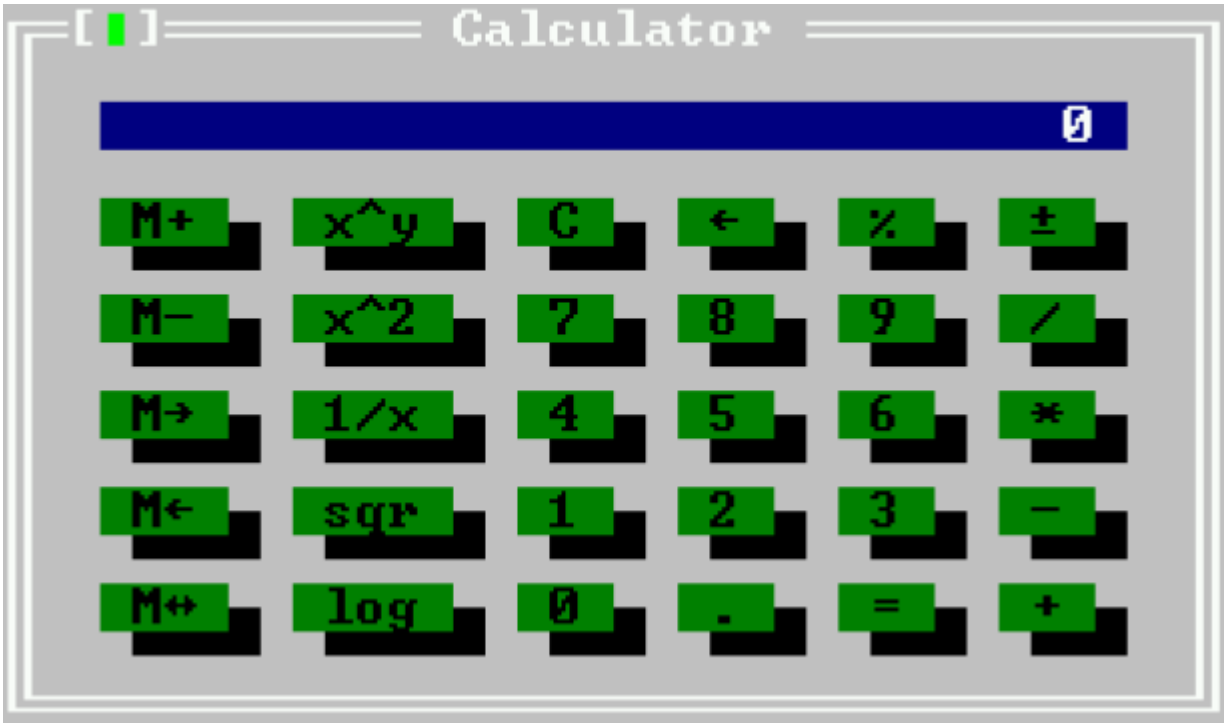


Рис. 6.19. Калькулятор Free Pascal.

Таблица 6.1. Основные математические операции.

Операция	Кнопка	Клавиша
Сложение двух чисел	+	+
Вычитание одного числа из другого	-	-
Умножение двух чисел	*	*
Деление	/	/
Удаление последней введённой цифры	<-	Backspace
Очистка дисплея	C	C
Смена знака	±	
Вычисления с процентами	%	%
Получение результата вычислений	**=**	**ENTER**

Но также возможны более сложные операции, такие как возведение в степень и логарифмирование. Эти операции перечислены в таблице 6.2.

Таблица 6.2. Сложные математические операции.

Операция	Кнопка	Клавиша
Возведение в степень	x^y	
Вычисление обратной функции	1/x	
Вычисление квадратного корня	sqr	
Вычисление натурального логарифма	log	
Возведение в квадрат	x^2	
Прибавить число с дисплея к числу в памяти	M+	
Вычесть число на дисплее из числа в памяти	M-	
Переместить содержимое памяти на дисплей	M->	
Переместить содержимое дисплея в память	M<-	
Поменять местами содержимое дисплея и памяти	M<->	

6.10.5. Добавление новых инструментов

Меню TOOLS может быть расширено добавлением любых внешних программ, ориентированных на командную строку. Выходные данные этих программ будут перехвачены и отображены в окне сообщений.

Добавление инструмента в меню TOOLS можно выполнить командой меню OPTIONS – TOOLS. Окно добавления инструмента показано на рис. 6.20.

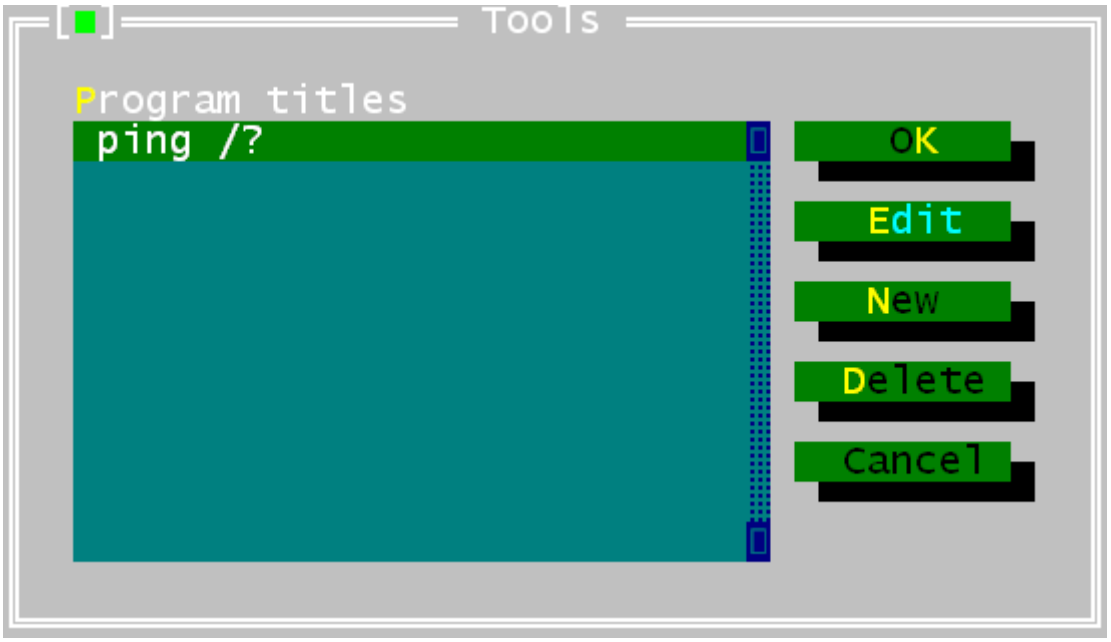


Рис. 6.20. Окно управления инструментами.

В этом окне можно выполнить следующие действия:

Элемент	Перевод	Описание
New	**Новый**	Отображает диалоговое окно параметров инструмента, где можно ввести новый инструмент.
Edit	**Редактировать**	Отображает диалоговое окно параметров выбранного инструмента, где изменить его параметры.
Delete	**Удалить**	Удаляет выбранный инструмент из списка.
Cancel	**Отмена**	Отменяет все изменения и закрывает окно.
OK		Сохраняет все изменения и закрывает окно.

Настройки инструментов записываются в конфигурационный файл рабочего стола. За исключением случая, когда включено авто-сохранение конфигурационного файла, настройки должны сохраняться вручную командой OPTIONS – SAVE.

6.10.6. Мета параметры

При вводе командной строки для инструмента, можно использовать мета параметры. Мета параметры – это переменные, которые замещаются указанными значениями перед передачей в командную строку. Например, мета параметр $EDNAME заменяет полное имя файла, который открыт в активном окне редактора.

Параметр	Описание
$CAP	Перехватывает выходные данные инструмента.
$CAP_MSG()	Перехватывает выходные данные инструмента и помещает их в окно сообщений.
$CAP_EDIT()	Перехватывает выходные данные инструмента и помещает их в отдельное окно редактора.
$COL	Заменяет столбец от курсора в активном окне редактора. В случае неактивного окна или активного диалогового окна заменяет на 0.
$CONFIG	Заменяет полное имя файла текущего конфигурационного файла.
$DIR()	Если аргументом является полное имя файла, то в этом аргументе заменяет полное имя к каталогу, включая последний разделитель, например, `$DIR('d:\data\myfile.pas') вернёт d:\data\`
$DRIVE()	Если аргументом является полное имя файла, то в этом аргументе заменяет букву диска, например, `$DIR('d:\data\myfile.pas') вернёт d:`
$EDNAME	Заменяет полное имя файла, который открыт в активном окне редактора.
$EXENAME	Заменяет имя исполняемого файла, которой должен создаваться при использовании команды **make** (например, для заданного первичного файла или из активного окна редактора).
$EXT()	Если аргументом является имя файла, то в этом аргументе заменяет расширение файла, например, `$EXT('d:\data\myfile.pas') вернёт .pas`
$LINE	Заменяет номер строки, на которой установлен курсор в активном окне редактора. Если нет активных или редактируемых окон, то вернёт 0.
$NAME()	Если аргументом является имя файла, то в этом аргументе заменяет имя файла (включая точку), например, `$NAME('d:\data\myfile.pas') вернёт myfile.`
$NAMEEXT()	Если аргументом является полное имя файла, то в этом аргументе заменяет имя файла с расширением, например, `$NAMEEXT('d:\data\myfile.pas') вернёт myfile.pas`
$NOSWAP	Ничего не делает, предоставляется только для совместимости с Turbo Pascal.
$PROMPT()	Отображает на экране диалоговое окно, где можно редактировать все аргументы, которые идут после этого. Аргументы, которые находятся до ключевого слова $PROMPT, не предоставляются для редактирования. Параметр $PROMPT может также принимать в качестве аргумента имя файла. Если имя файла передано в качестве аргумента, то $PROMPT() загрузит описание диалога из указанного файла. Например: `$PROMPT(cvsco.tdf)` Будет анализировать файл **cvsco.tdf**, создавать диалоговое окно в соответствии с данными этого файла и отображать это окно на экране. После закрытия окна, информация, введённая пользователем, применяется для создания инструмента командной строки. Подробности см. в разделе 6.10.7.
$SAVE	Перед выполнением команды активное окно редактора сохраняется, даже если не было выполнено изменений.
$SAVE_ALL	Перед выполнением команды все окна сохраняются без запроса на подтверждение.
$SAVE_CUR	Перед выполнением команды активное окно редактора сохраняется без запроса на подтверждение, если были выполнены изменения в тексте.
$SAVE_PROMPT	Перед выполнением команды появляется окно с запросом на подтверждение сохранения для всех не сохранённых файлов.
$WRITEMSG()	Записывать подробную выходную информацию инструмента в файл, имя которого передаётся в качестве параметра.

6.10.7. Создание диалогового окна командной строки

Когда определяется инструмент, имеется возможность отобразить для пользователя диалоговое окно, запрашивающее дополнительные аргументы, используя макрокоманду `$PROMPT(ИмяФайла)`. Дистрибутив Free Pascal содержит несколько готовых диалоговых окон, таких как окно инструмента **grep**. Файлы этих окон находятся в каталоге установки программы и имеют расширение **.tdf**.

В этом разделе разъясняется формат файла описания диалогового окна. Формат этого файла имеет сходство с **.INI**-файлами Windows, где каждый раздел в файле описывает какой-либо элемент (или управление) в окне. Кнопки **OK** и **Cancel** будут автоматически добавлены в нижнюю часть окна, поэтому нет необходимости указывать их при определении диалогового окна.

В файле имеется специальный раздел **Main**, который описывает общий размер окна и то, каким образом выходной результат окна будет помещён в командную строку.

ПРИМЕЧАНИЕ
При использовании ключевых слов, содержащих строковые значения, эти значения должны заключаться в двойные кавычки, например:

```
Title="Dialog title"
```

Раздел **Main** должен содержать следующие ключевые слова:

Ключевое слово	Перевод	Описание
Title	**Заголовок**	Заголовок окна. Отображается в рамке заголовка окна. Строка должна заключаться в двойные кавычки.
Size	**Размер**	Размер окна. Задаётся в формате (Столбцы, Строки), например: `Size=(59,9)` означает, что окно имеет 59 символов в ширину и 9 в высоту. В этот размер не входит рамка окна.
CommandLine	**Командная строка**	Определяет, каким образом командная строка будет передана в программу, основываясь на данных, введённых в диалоговом окне. Напечатанный здесь текст будет передан после замещения некоторых элементов управления их значениями. Элемент управления – это имя какого-либо элемента в окне, заключённое между символами процента (%). Имя элемента управления будет заменяться текстом, который связан с этим элементом. Рассмотрим пример: `CommandLine="-n %l% %v% %i% %w% %searchstr% %filemask%"` Здесь значения, связанные с элементами управления с именами **l, v, i, w** и **searchstr** и **filemask** будут вставлены в командную строку.

Default	**По умолчанию**	Имя элемента управления, который является элементом по умолчанию, то есть элемент, который имеет фокус при открытии окна. В следующем примере показан правильный раздел **main**:
		
		`[Main]`
		`Title="GNU Grep"`
		`Size=(56,9)`
		`CommandLine="-n %l% %v% %i% %w% %searchstr% %filemask%"`
		`Default="searchstr"`
		
		После раздела main требуется определить раздел для каждого элементу управления, который отображается в диалоговом окне. Каждый раздел имеет имя элемента, который описывается в разделе, например:
		
		`[CaseSensitive]`
		`Type=CheckBox`
		`Name="~C~ase sensitive"`
		`Origin=(2,6)`
		`Size=(25,1)`
		`Default=On`
		`On="-i"`
		
		Каждый раздел элемента должен иметь необходимый минимум ключевых слов, связанных с ним, которые описаны ниже.

Описание ключевых слов раздела элемента управления:

Ключевое слово	Перевод	Описание
Type	**Тип**	Тип элемента управления. Возможны следующие значения:
Label (Надпись)		Обычная текстовая надпись, которая будет отображаться в окне. Какой-либо другой элемент управления может быть связан с этой надписью таким образом, что фокус будет переходить на этот элемент управления, если пользователь нажмёт клавишу с буквой, которая подсвечена в этой надписи.
InputLine (Ввод текста)		Редактируемое поле для ввода данных, где можно ввести текст.
CheckBox (Флажок)		Флажок, который может принимать два состояния (установлен или сброшен).
Origin	**Начальная координата**	Определяет, где должен размещаться элемент управления. Начальная координата указывается в формате (Левый, Верхний), а левый верхний угол окна имеет координаты (1, 1) (не учитывая рамку окна).
Size	**Размер**	Определяет размер элемента управления в формате (Столбцы, Строки).

Каждый элемент управления имеет некоторые специальные ключевые слова, связанные с этим элементом. Эти ключевые слова будут описаны ниже.

Элемент «Надпись» (Type=Label) имеет следующие специфические ключевые слова:

Ключевое слово	Перевод	Описание
Text	**Текст**	Текст, отображаемый в надписи. Если одна из букв текста должна быть подсвечена, для того, чтобы имелась возможность использовать её как «горячую клавишу», то она должна быть заключена в символы тильды (~). Например, в тексте `Text="~T~ext to find"` будет подсвечена буква **T**.
Link	**Ссылка**	Здесь можно указать имя элемента управления окна. Если имя указано, то нажатие комбинации клавиш ALT с буквой, которая подсвечена в надписи, переведёт фокус на элемент, указанный в данном параметре.

Элемент «Надпись» не применяется для передачи текста в командную строку. Этот элемент используется только для информирования и навигации. В следующем примере описан раздел с элементом «Надпись»:

```
[label2]
Type=Label
Origin=(2,3)
Size=(22,1)
Text="File ~m~ask"
Link="filemask"
```

Элемент «Ввод текста» (Type=InputLine) позволяет вводить любой текст. Текст из этого элемента будет передан в командную строку, если в этом есть необходимость. Для этого элемента имеются следующие специфические ключевые слова:

Ключевое слово	Перевод	Описание
Value	**Значение**	Можно указать стандартное значение (текст), которое будет автоматически выводиться в этот элемент при открытии окна (значение по умолчанию).

В следующем примере описан раздел с элементом «Ввод текста»:

```
[filemask]
Type=InputLine
Origin=(2,4)
Size=(22,1)
Value="*.pas *.pp *.inc"
```

Элемент «Флажок» (Type=CheckBox) может принимать два состояния (установлен или сброшен). Значение, связанное с каждым из этих состояний, может быть передано в командную строку. Для этого элемента имеются следующие специфические ключевые слова:

Ключевое слово	Перевод	Описание
Name	**Имя**	Текст, который отображается рядом с флажком. Если в тексте есть подсвеченная буква, то можно установить или сбросить состояние флажка комбинацией клавиш ALT-БУКВА.
Default	**По умолчанию**	Определяет значение по умолчанию (сброшен или установлен) при открытии окна.
On	**Установлен**	Текст, связанный с этим флажком, если он установлен.
Off	**Сброшен**	Текст, связанный с этим флажком, если он сброшен.

В следующем примере описан раздел с элементом «Флажок»:

```
[i]
Type=CheckBox
Name="~C~ase sensitive"
Origin=(2,6)
Size=(25,1)
Default=On
On="-i"
```

В этом примере, если флажок установлен, то значение **–i** будет добавлено в командную строку инструмента. Если флажок сброшен, то никакое значение не будет добавлено.

6.11. Управление проектом и опции компилятора

Управление проектом в Паскале более простое, чем в C. Компилятор знает, какие требуются исходные файлы, модули и т.п. Поэтому среда разработки Free Pascal не нуждается в полноценном менеджере проектов, подобно некоторым средствам разработки на C. Тем не менее имеются некоторые настройки IDE, которые применяются в проекте.

6.11.1. Первичный файл

Без первичного файла IDE компилирует/запускает исходный файл активного окна, когда выполняются команды меню `Run`. Если первичный файл указан, то всегда выполнятся компиляция/запуск этого исходного файла, даже если активно окно с другим исходным файлом. Первичный файл можно определить, вызвав диалоговое окно выбора файла командой меню `COMPILE – PRIMARY FILE…` Только команда меню `COMPILE – COMPILE` будет компилировать исходный файл активного окна независимо от того, определён первичный файл или нет. Это актуально для больших проектов, которые уже отредактированы, и требуется только проверка синтаксиса текущего исходного файла.

Команда меню `COMPILE – CLEAR PRIMARY FILE` устанавливает настройки IDE по умолчанию, то есть команды `COMPILE` и `RUN` будут действовать на исходный файл активного окна.

6.11.2. Окно каталогов

В окне каталогов можно указать директории, где компилятор будет искать модули, библиотеки и объектные файлы. Здесь также можно указать каталог, где будут сохраняться выходные файлы (`EXE` и `PPU`). Допускается водить несколько каталогов (кроме каталога для выходных файлов), разделяя их точкой с запятой. Окно каталогов показано на рис. 6.21.

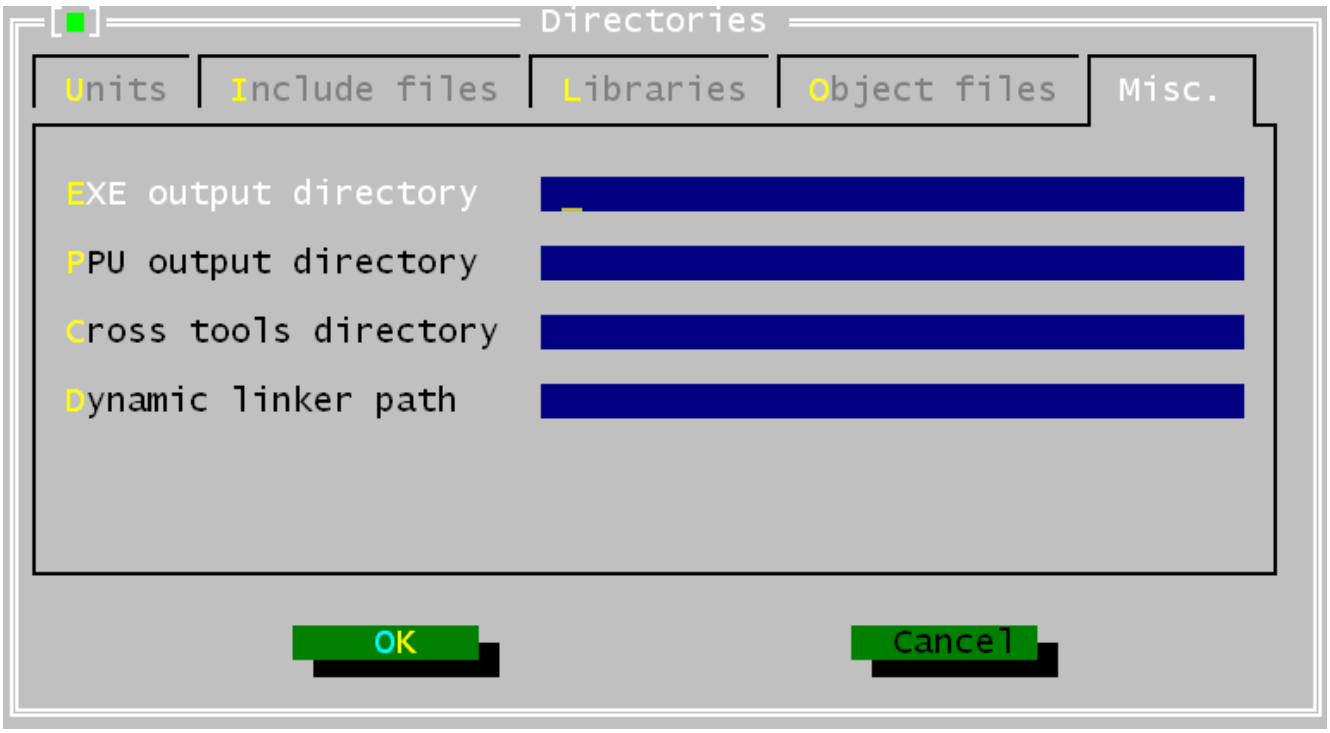


Рис. 6.21. Окно каталогов.

Окно каталогов можно вызвать командой меню `OPTIONS - DIRECTORIES`. В более свежих версиях в этом случае будет вызвано окно, показанное на рис. 6.21, где имеется несколько вкладок для разных каталогов. Можно указать следующие каталоги:

Каталог	Перевод	Описание
EXE & PPU directories	**Каталоги для EXE и PPU**	Определяет каталог, где будут сохраняться исполняемые файлы (EXE) и откомпилированные модули (PPU). См. также опцию командной строки –FE, раздел «5.1.3. Параметры, касающиеся файлов и каталогов».
Object directories	**Каталоги объектных файлов**	Определяет каталог, где компилятор будет искать внешние объектные файлы. См. также опцию командной строки –Fo, раздел «5.1.3. Параметры, касающиеся файлов и каталогов».
Library directories	**Каталоги библиотек**	Определяет каталог, где компилятор (точнее, компоновщик) будет искать внешние библиотеки. См. также опцию командной строки –Fl, раздел «5.1.3. Параметры, касающиеся файлов и каталогов».
Include directories	**Каталоги подключаемых файлов**	Определяет каталог, где компилятор будет искать подключаемые файлы, подключаемые директивой {$i}. См. также опцию командной строки –Fi или –I, раздел «5.1.3. Параметры, касающиеся файлов и каталогов».
Unit directories	**Каталоги модулей**	Определяет каталог, где компилятор будет искать откомпилированные модули. Компилятор всегда сначала просматривает текущий каталог, а также некоторые стандартные директории. См. также опцию командной строки –Fu, раздел «5.1.3. Параметры, касающиеся файлов и каталогов».

6.11.3. Целевая операционная система

Команда меню `COMPILE - TARGET` позволяет определить целевую операционную систему для каждого компилируемого исходного файла. Изменение целевой ОС не влияет на параметры или директории компилятора. Настройки, которые можно здесь установить, связаны с опцией `-T` командной строки (см. раздел «5.1.4. Параметры, контролирующие результат компиляции»). Простое окно выбора целевой системы показано на рис. 6.22. В реальности в этом окне будут отображаться только те ОС, которые поддерживаются вашей IDE (их может быть как больше, так и меньше).

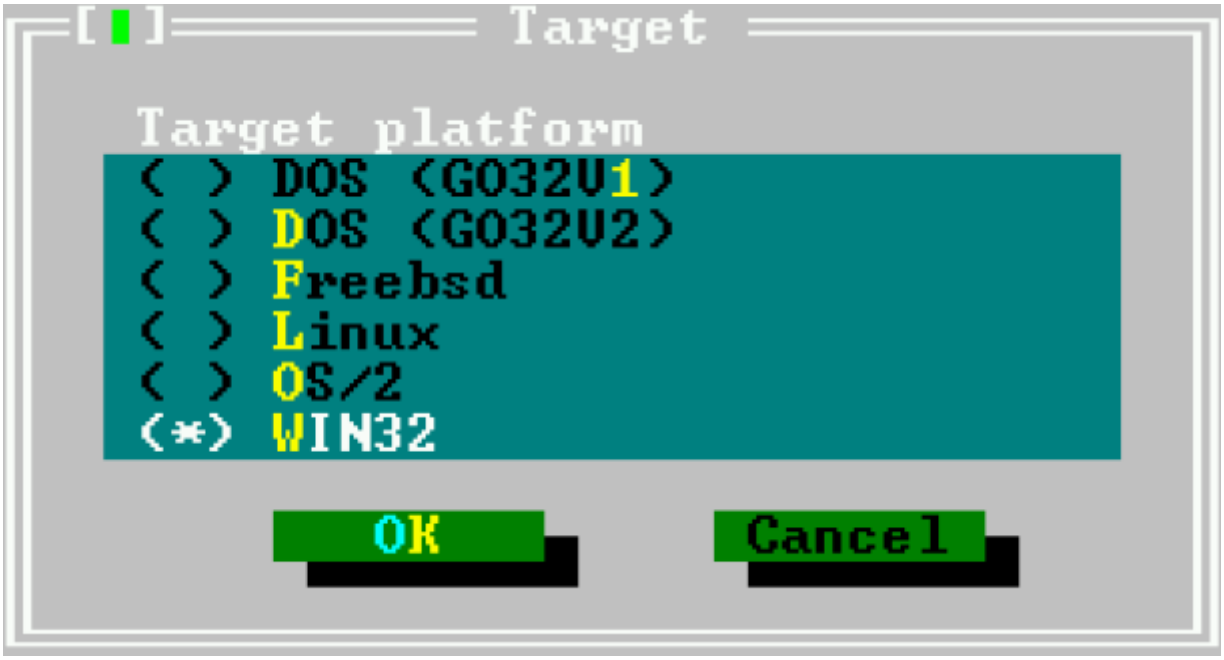


Рис. 6.22. Окно выбора целевой операционной системы.

Следующие целевые операционные системы могут быть установлены (этот список определяется платформой, для которой скомпилирована IDE):

ОС	Описание
DOS (go32v1)	Этот переключатель не будет отображаться, когда эта система перестанет поддерживаться.
DOS (go32v2)	Компиляция для DOS, используя версию 2 расширителя Go32.
FreeBSD	Компиляция для FreeBSD.
Linux	Компиляция для LINUX.
OS/2	Компиляция для OS/2 (используя расширитель EMX).
Windows	Компиляция для Windows.

Целевая операционная система, выбранная в текущий момент, отображается в меню `COMPILE` напротив команды `TARGET`. В исходном состоянии установлена операционная система, для которой была откомпилирована IDE.

6.11.4. Опции компилятора

Меню `OPTIONS – COMPILER` позволяет устанавливать параметры, которые влияют на поведение компилятора. При выборе этого меню появляется окно с несколькими вкладками. Всего имеется шесть вкладок:

Вкладка	Перевод	Описание
Syntax	**Синтаксис**	Здесь можно определить параметры, которые влияют на различные представления синтаксиса в исходном коде. См. также опцию командной строки –S, раздел «5.1.5. Параметры для исходных кодов (опции языка)».
Code generation	**Генерация кода**	Эти параметры управляют генерацией кода. См. также опцию командной строки –C или –X.
Verbose	**Информация**	Эти параметры устанавливает генерацию подробных сообщений компилятора. Сообщения компилятора отображаются в окне сообщений, которое можно вызвать клавишей F12.
Browser	**Обозреватель**	Параметры, относящиеся к генерации информации обозревателя. Информация обозревателя необходима для генерации символов обозревателя для работы.
Assembler	**Ассемблер**	Параметры, относящиеся к чтению ассемблерных блоков (-R в командной строке) и генерации ассемблера (-A в командной строке).
Processor	**Процессор**	Здесь можно выбрать целевой процессор.

На каждой странице (вкладке) имеются два блока для ввода данных: первый для определения состояний и второй – для дополнительных аргументов компилятора. Символы и аргументы должны разделяться точкой с запятой.

Вкладка СИНТАКСИС показана на рис. 6.23.

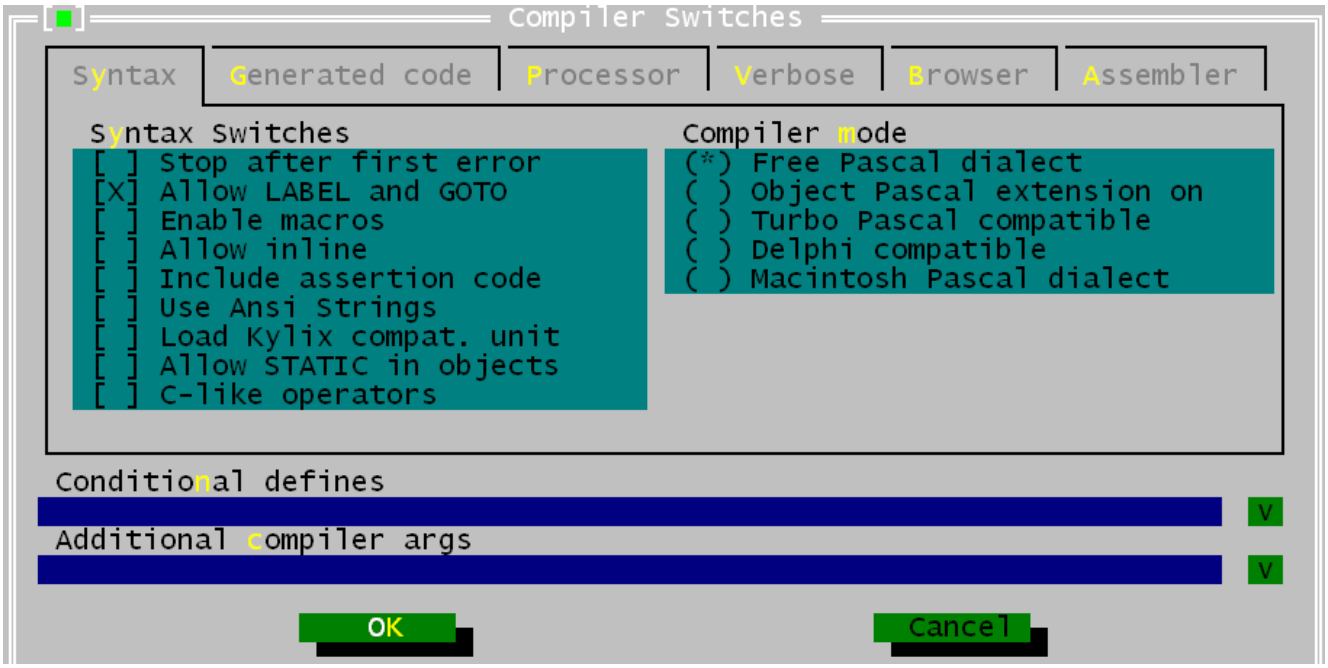


Рис. 6.23. Окно параметров компилятора. Вкладка СИНТАКСИС.

Вкладка ГЕНЕРАЦИЯ КОДА показана на рис. 6.24.

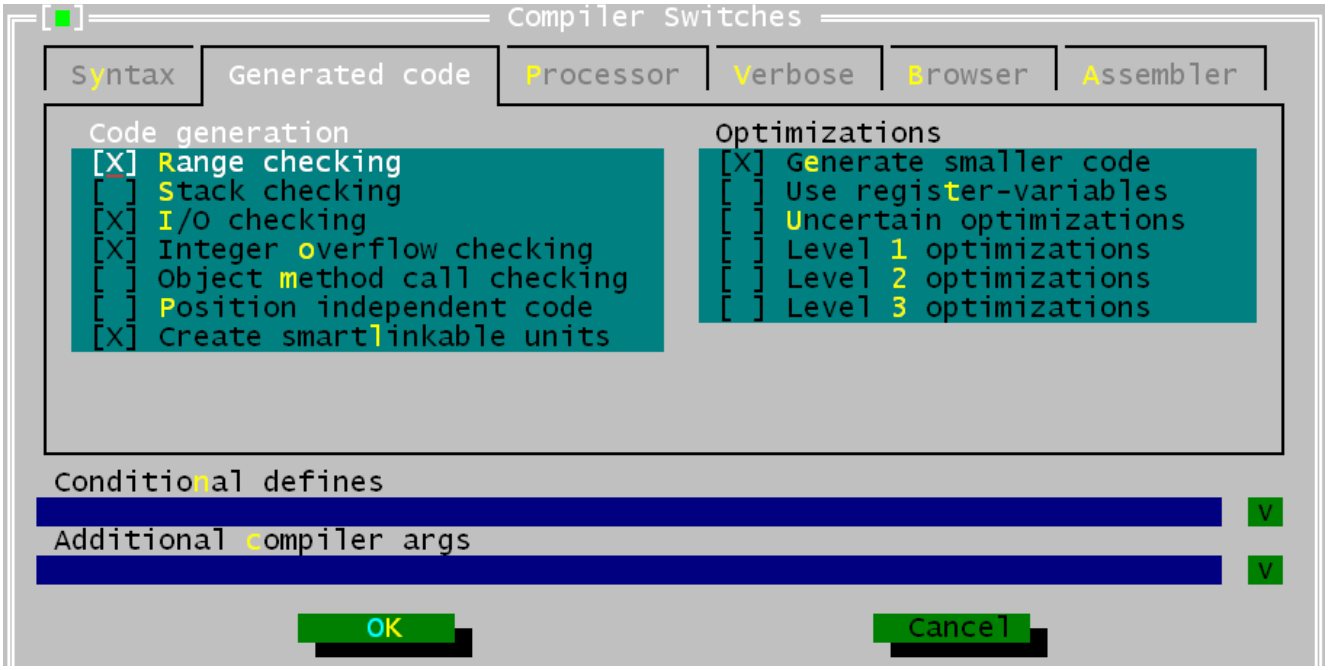


Рис. 6.24. Окно параметров компилятора. Вкладка ГЕНЕРАЦИЯ КОДА.

На вкладке СИНТАКСИС можно установить следующие опции:

Опция	Перевод	Описание
Stop after first error	**Остановиться после первой ошибки**	Если опция установлена, то компилятор останавливается после обнаружения первой ошибки. Обычно компиляция выполняется пока не случится фатальная ошибка. См. также опцию командной строки –Se, раздел «5.1.5. Параметры для исходных кодов (опции языка)».
Allow label and goto	**Разрешать метки и переходы**	Если опция установлена, то компилятор разрешает использование меток и оператора **goto**. См. также опцию командной строки –Sg, раздел «5.1.5. Параметры для исходных кодов (опции языка)».
Enable macros	**Включить макросы**	Разрешить использовать макросы. См. также опцию командной строки –Sm, раздел «5.1.5. Параметры для исходных кодов (опции языка)».
Allow inline	**Разрешить функции**	Разрешить использовать встроенные функции. См. также опцию командной строки –Sc, раздел «5.1.5. Параметры для исходных кодов (опции языка)» (Возможно, это опечатка – примечание переводчика).
Include assertion code	**Включать условное кодирование**	Включать в код оператор **Assert**.
Load Kylix compat. unit	**Загружать Kylix модуль**	Загружать совместимый с **Kylix** модуль.
Allow STATIC in objects	**Разрешить STATIC в объектах**	Разрешить модификатор **Static** для методов объекта. См. также опцию командной строки –St, раздел «5.1.5. Параметры для исходных кодов (опции языка)».
C-like operators	**Си-подобные операторы**	Разрешить использовать некоторые операторы стиля Си, такие как +=, -= и т.п. См. также опцию командной строки –Sc, раздел 5.1.5.
Compiler mode	**Режим компилятора**	Выбор подходящего режима компилятора. Может принимать следующие значения:
Free Pascal Dialect (Диалект Free Pascal)		По у молчанию используется этот режим (FPC).
Object Pascal extensions on (Включить расширения Object Pascal)		Включить использование классов и исключений. См. также опцию командной строки –Sd, раздел 5.1.5.
Turbo Pascal compatible (совместимость с Турбо Паскаль)		Пытаться использовать максимально возможную совместимость с Турбо Паскаль. См. также опцию командной строки –So, раздел 5.1.5.
Delphi compatible (Совместимость с Делфи)		Пытаться использовать максимально возможную совместимость с Делфи. См. также опцию командной строки –Sd, раздел 5.1.5».
Macintosh Pascal Dialect (Диалект Macintosh Pascal)		Пытаться использовать максимально возможную совместимость с Макинтош.

На вкладке ГЕНЕРАЦИЯ КОДА можно установить следующие опции:

Опция	Перевод	Описание
Run-time checks	**Проверки времени выполнения**	Проверки кода во время генерации. Если какая-либо из этих проверок закончится неудачно, то будет сгенерирована ошибка времени выполнения. Следующие проверки могут быть выполнены:
Range checking (Проверка диапазона)		Проверять операции с перечисляемыми типами и множествами. См. также опцию командной строки –Cr, раздел 5.1.4.
Stack checking (Проверка стека)		Проверять, хватает ли размера стека. См. также опцию командной строки –Cs, раздел 5.1.4.
I/O checking (Проверка ввода-вывода)		Проверять результат операций ввода-вывода. См. также опцию командной строки –Ci, раздел 5.1.4.
Integer overflow checking (Проверка целочисленного переполнения)		Проверять результат целочисленных операций. См. также опцию командной строки –Co, раздел 5.1.4.
Object method call checking (Проверка вызовов методов объекта)		Проверять правильность указателя метода перед его вызовом.
Position independent code (Независимый код)		Генерировать PIC код.
Create smartlinkable units (Создавать «умные» модули)		Генерировать модули в режиме «умная компоновка».
Optimizations	**Оптимизации**	Оптимизации, используемые при компиляции.
Generate faster code (Генерировать быстрый код)		Аналогично опции командной строки –OG.
Generate smaller code (Генерировать меньший код)		Аналогично опции командной строки –Og.

Более подробно об этих переключателях см. в разделе «5.1.4. Параметры, контролирующие результат компиляции».

Вкладка ПРОЦЕССОР показана на рис. 6.25.

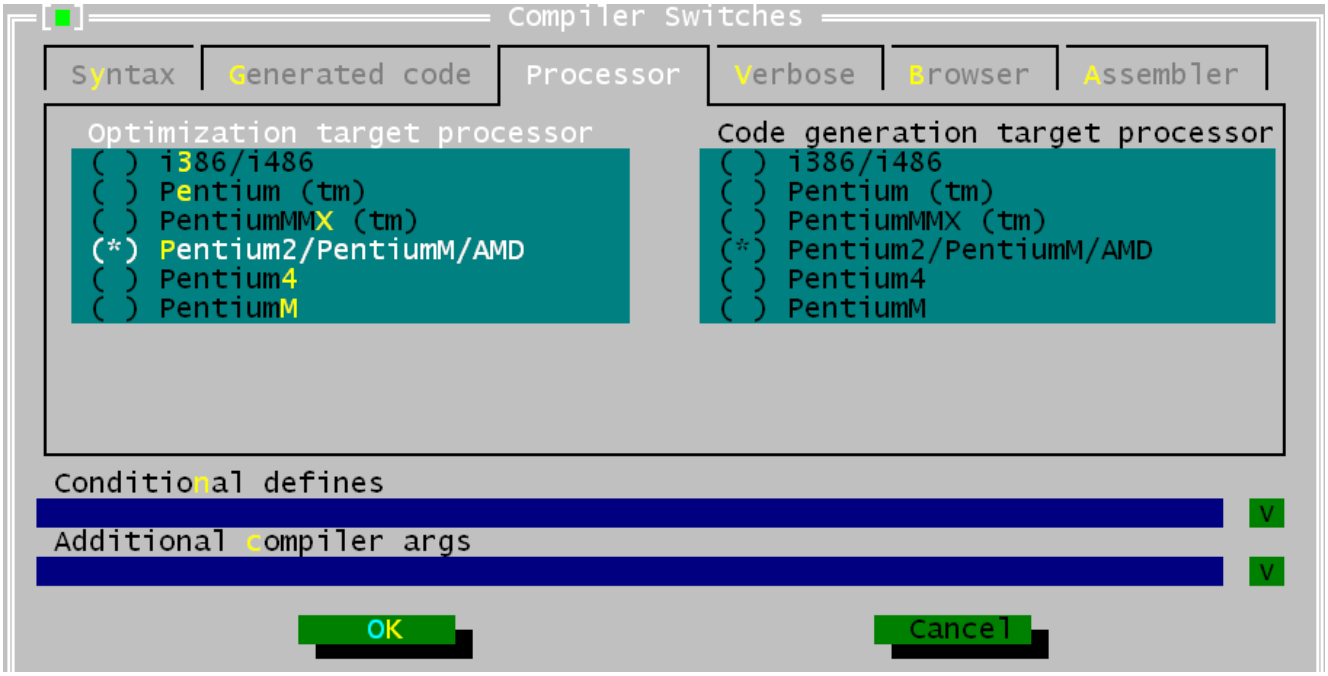


Рис. 6.25. Окно параметров компилятора. Вкладка ПРОЦЕССОР.

На вкладке ПРОЦЕССОР можно установить целевой процессор. Компилятор может использовать различные оптимизации для различных процессоров.

Вкладка ИНФОРМАЦИЯ показана на рис. 6.26.

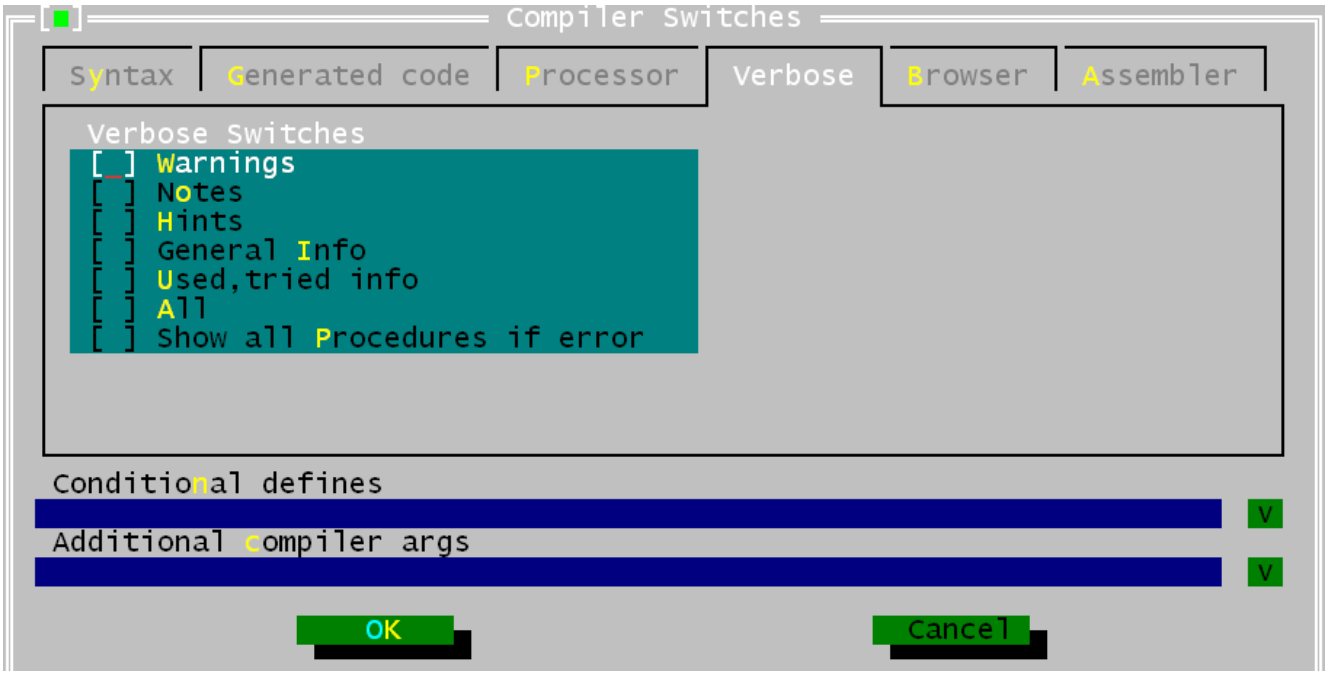


Рис. 6.26. Окно параметров компилятора. Вкладка ИНФОРМАЦИЯ.

На этой вкладке можно установить следующие параметры (для командной строки это параметр `-v`, см. раздел «5.1.2. Параметры обратной связи»):

Опция	Перевод	Описание
Warnings	**Предупреждения**	Генерировать предупреждения. Соответствует параметру `-vw` в командной строке.
Notes	**Замечания**	Генерировать замечания. Соответствует параметру `-vn` в командной строке.
Hints	**Подсказки**	Генерировать подсказки. Соответствует параметру `-vh` в командной строке.
General info	**Общая информация**	Генерировать общую информацию. Соответствует параметру `-vi` в командной строке.
Used, tried info	**Использование и проверка**	Генерировать информацию об использовании и проверке файлов. Соответствует параметру `-vut` в командной строке.
All	**Всё**	Генерировать всю возможную информацию. Соответствует параметру `-va` в командной строке.
Show all procedures if error	**Показать процедуру в случае ошибки**	Если при использовании загружаемой процедуры произойдёт ошибка, то показывать все процедуры. Соответствует параметру `-vb` в командной строке.

Вкладка ОБОЗРЕВАТЕЛЬ показана на рис. 6.27.

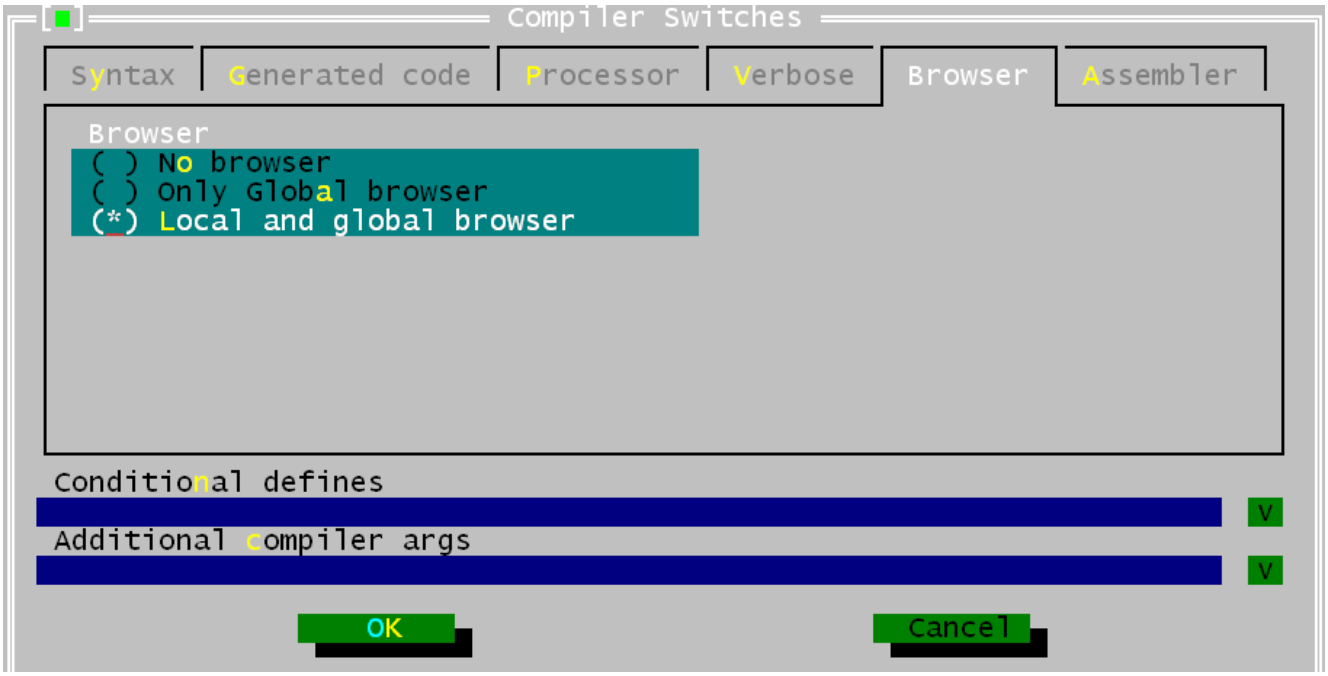


Рис. 6.27. Окно параметров компилятора. Вкладка ОБОЗРЕВАТЕЛЬ.

На этой вкладке можно установить параметры обозревателя:

Опция	Перевод	Описание
No browser	**Нет обозревателя**	По умолчанию. Не отображать информацию, генерируемую компилятором.
Only global browser	**Только глобальный обозреватель**	Информация генерируется только для глобальных идентификаторов, то есть для идентификаторов, определённых НЕ в процедуре или функции. Соответствует параметру `-b` в командной строке.
Local and global browser	**Локальный и глобальный обозреватель**	Информация генерируется для всех идентификаторов, то есть в том числе и для тех, которые определены в процедуре или функции. Соответствует параметру `-bl` в командной строке.

ПРИМЕЧАНИЕ
Если информация для обозревателя не генерируется, то идентификаторы обозревателя в IDE не будут работать.

Вкладка АССЕМБЛЕР показана на рис. 6.28. В реальности эта вкладка может быть другой (зависит от целевого процессора, для которого была откомпилирована IDE).

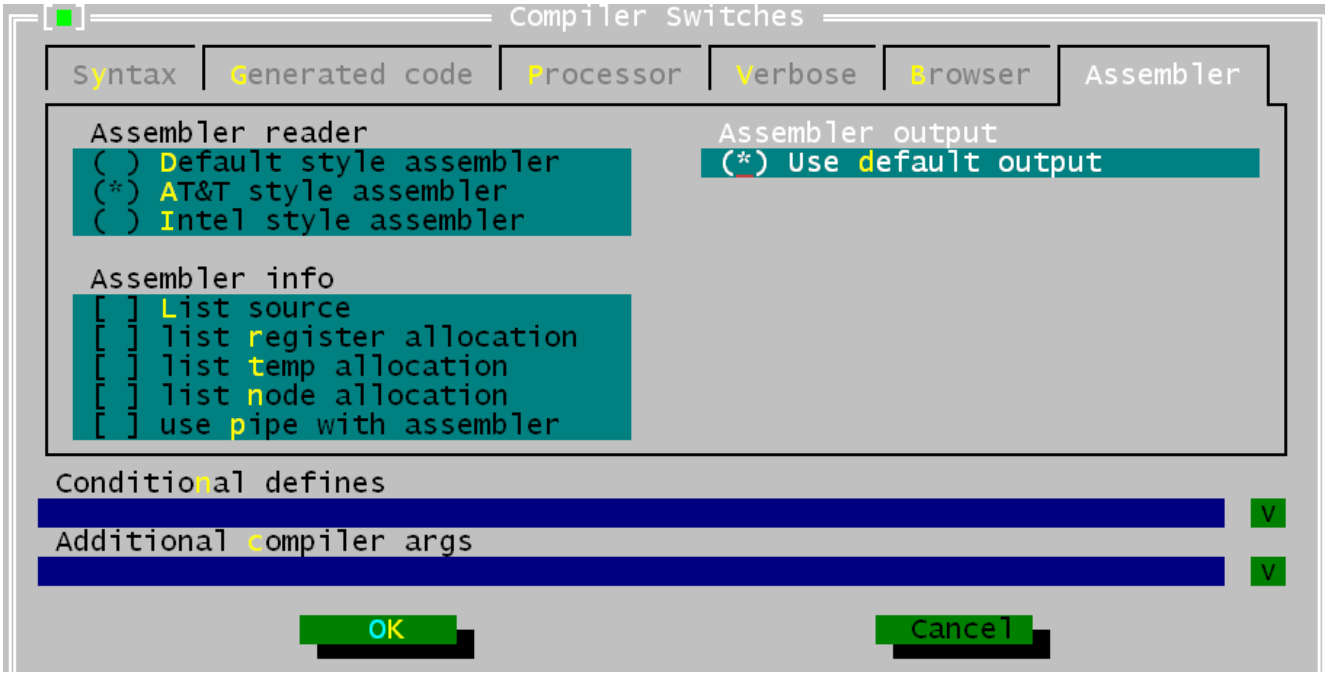


Рис. 6.28. Окно параметров компилятора. Вкладка АССЕМБЛЕР.

На вкладке АССЕМБЛЕР можно установить следующие параметры:

Опция	Перевод	Описание
Assembler reader	Позволяет установить стиль для ассемблерных блоков в исходном коде. Эти опции глобальные, но локальный стиль ассемблера можно изменить директивой компилятора.	
AT&T assembler		Стиль ассемблера AT&T. Соответствует параметру `-Ratt` в командной строке.
Intel style assembler		Стиль ассемблера Intel. Соответствует параметру `-Rintel` в командной строке.
Assembler info	**Информация**	Этот параметр определяет, какая информация будет записываться в файлы ассемблера в виде комментариев:
List source (Список исходников)		Строки исходного кода будут записываться в файлы ассемблера вместе генерируемым ассемблером (соответствует параметру `-al` в командной строке).
List register allocation		Помещение/извлечение информации встроенного регистра компилятора будет записываться в файл ассемблера (соответствует параметру `-ar`).
List temp allocation		Помещение/извлечение информации временного регистра будет записываться в файл ассемблера (соответствует параметру `-at` в командной строке).
List node allocation		Помещение/извлечение информации с замечаниями будет записываться в файл ассемблера (соответствует параметру `-an` в командной строке).
Use pipe with assembler		Использовать канал на Unix-системе, если код ассемблера передаётся во внешний ассемблер.
Последние три параметра являются особенно полезными для отладки самого компилятора, но редко используются в обычном режиме.		
Assembler output	**Выход ассемблера**	Этот параметр указывает компилятору, как нужно генерировать ассемблер.
Use default output		Определяется целевой платформой.
Use GNU as		Использовать ассемблер **GNU as** (соответствует параметру `-Aas`).
Use NASM coff		Использовать ассемблер **NASM coff** (go32v2, параметр `-Anasmcoff`).
Use NASM elf		Использовать ассемблер **NASM elf** (LINUX, соответствует параметру `-Anasmelf`).
Use NASM obj		Использовать ассемблер **NASM obj** (соответствует параметру `-Anasmobj`).
Use MASM		Использовать ассемблер **MASM** (соответствует параметру `-Amasm`).
Use TASM		Использовать ассемблер **TASM** (соответствует параметру `-Atasm`).
Use coff		Записывать бинарные coff-файлы, напрямую используя встроенный ассемблер (go32v2, в командной строке параметр `-Acoff`).
Use pecoff		Записывать бинарные pecoff-файлы, напрямую используя встроенный ассемблер (Win32).

6.11.5. Опции компоновщика

Меню OPTIONS – LINKER позволяет устанавливать параметры компоновщика. Здесь можно определить, как будут скомпонованы библиотеки и модули, а также способ вызова компоновщика. Окно параметров компоновщика показано на рис. 6.29.

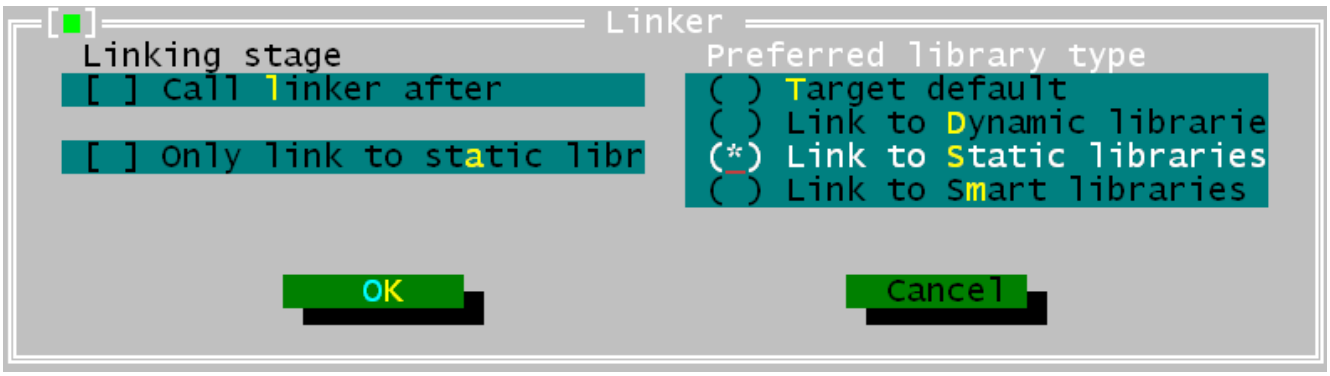


Рис. 6.29. Окно параметров компоновщика.

Следующие параметры можно изменить:

Параметр	Перевод	Описание
Call linker after	**Вызывать компоновщик после**	Эта опция устанавливается, когда используется сценарий вызова компоновщика. Соответствует параметру –s командной строки (см. раздел «5.1.4. Параметры, контролирующие результат компиляции»).
Only link to static library	**Компоновать только статические библиотеки**	Использовать только статические библиотеки.
Preferred library type	**Предпочитаемый тип библиотеки**	С помощью этой опции можно установить тип компонуемой библиотеки:
Target default (По умолчанию)		Определяется настройками целевой платформы.
Dynamic libraries (Динамические библиотеки)		Пытаться компоновать модули в динамические библиотеки (параметр –XD командной строки).
Static libraries (Статические библиотеки)		Пытаться компоновать модули в статические библиотеки (параметр –XS командной строки).
Smart libraries («Умные» библиотеки)		Пытаться выполнить «умную компоновку» модулей в библиотеки (параметр –XX командной строки).

6.11.6. Размер памяти

В окне настроек размера памяти (вызывается из меню OPTIONS – MEMORY SIZES) можно установить размер областей памяти для проекта. Окно настроек размера памяти показано на рис. 6.30.

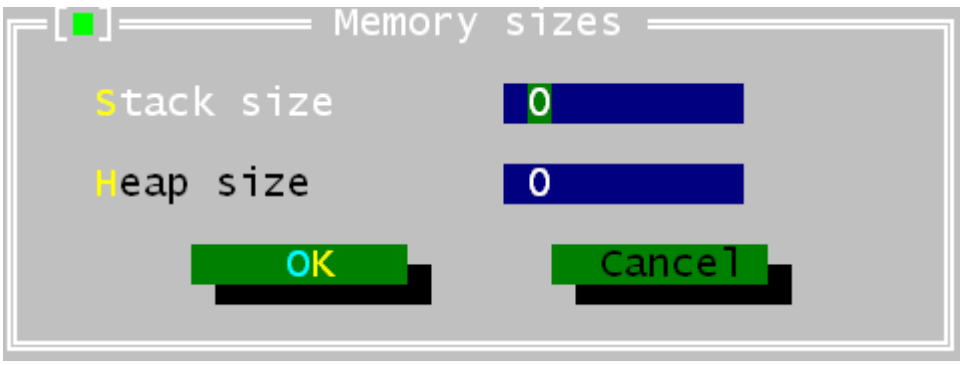


Рис. 6.30. Окно настроек размера памяти.

Следующие параметры можно изменить:

Параметр	Перевод	Описание
Stack size	**Размер стека**	Размер стека в байтах (параметр `-Cs` командной строки). Этот размер может игнорироваться некоторыми системами.
Heap size	**Размер «кучи»**	Размер кучи в байтах (параметр `-Ch` командной строки). Учтите, что размер кучи увеличивается динамически, насколько это позволяет операционная система.

6.11.7. Опции отладки

В окне параметров отладки (вызывается из меню `OPTIONS – MEMORY SIZES`) можно установить некоторые параметры включения отладочной информации в бинарный файл. Кроме того, здесь можно добавить дополнительные опции компилятора. Окно параметров отладки показано на рис. 6.31.

Следующие параметры можно изменить:

Параметр	Перевод	Описание
Debugging information	**Отладочная информация**	Указывает компилятору, какую отладочную информацию нужно включать в исполняемый файл. Можно выбрать следующее:
Strip all debug symbols from executable (удалять все отладочные символы из EXE-файла)		Будут удаляться все отладочные символы и информация из исполняемого файла (параметр `-Xs` командной строки).
Skip debug information generation (Пропускать генерацию отладочной информации)		Не генерировать никакую отладочную информацию.
Generate debug symbol information (Генерировать символьную отладочную информацию)		Включать отладочную информацию в исполняемый файл (параметр `-g` командной строки). Учтите, что отладочная информация для модулей в библиотеке реального времени (RTL) не будет включена, пока не будет доступна версия RTL, откомпилированная с отладочной информацией. Только модули данного проекта будут содержать отладочную информацию.
Generate also backtrace line information (Генерировать информацию отслеживания строк)		Будет выполняться компиляция с отладочной информацией и дополнительно в бинарный файл будет включен модуль **lineinfo**, так что в случае ошибки будут сохраняться имена файлов и номера строк вызываемых процедур (параметр `-gl` командной строки).
Generate valgrind compatible debug info (Генерировать совместимую с Valgrind отладочную информацию)		Генерировать отладочную информацию, которая может быть прочитана программой **Valgrind** (инструмент для проверки памяти).
Profiling switches	**Профилирующие переключатели**	Указывает компилятору, надо ли включать код профиля в бинарный файл.
No profile information (Нет профильной информации)		Не имеет никакого эффекта, установлен по умолчанию
Generate Profile code for gprof (Генерировать профильный код для gprof)		Если установлен, то профилирующий код включается в бинарный файл (параметр `-p` командной строки).
Use another TTY for Debuggee	**Использовать другую программу для отладки**	Будет выполняться попытка перенаправить выход программы, которая содержит отладочную информацию, в другое окно (терминал). Имя файла программы должно быть указано здесь.

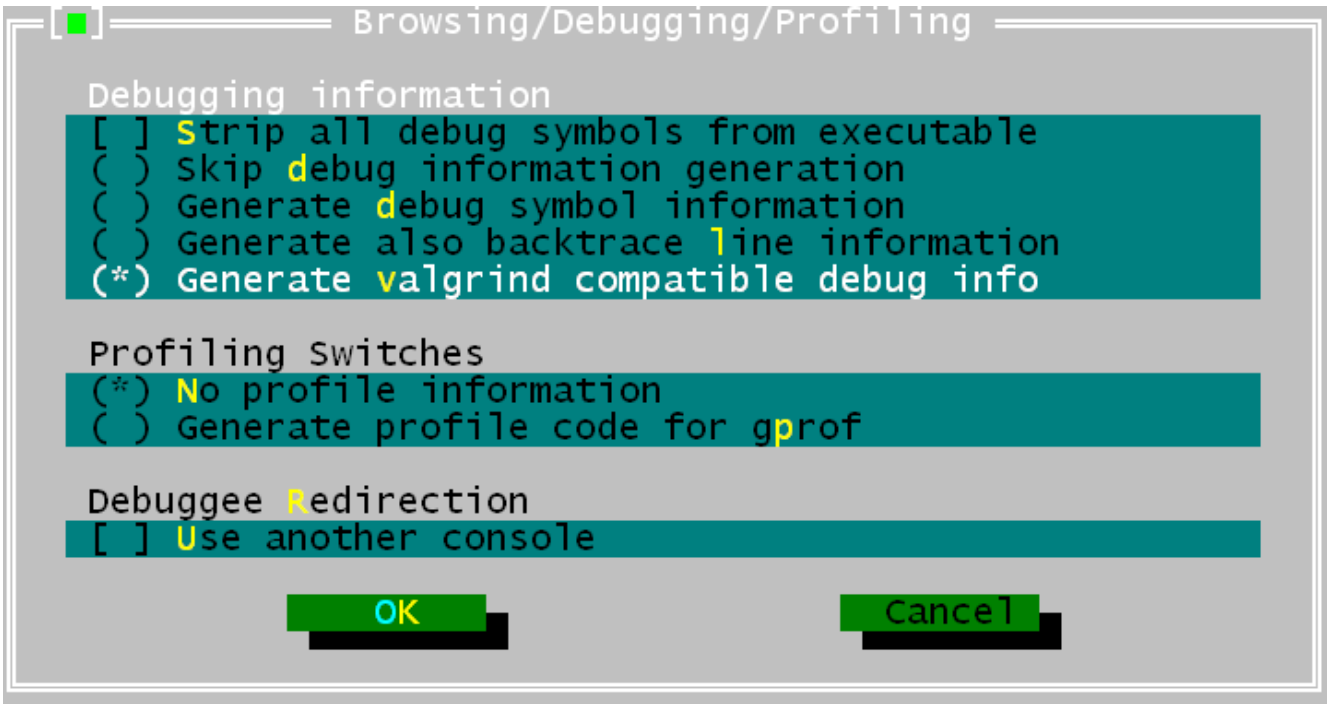


Рис. 6.31. Окно параметров отладки.

6.11.8. Переключатели режимов

IDE позволяет сохранять установленные настройки компилятора под общим именем. Предоставляются три имени, под которыми могут быть сохранены настройки:

Имя	Перевод	Описание
Normal	**Норма**	Нормальная (быстрая компиляция).
Debug	**Отладка**	Используется для отладки. Предназначена для установки множества отладочных функций. Также полезна для установки определённых состояний, которые, например, позволяют включать какой-либо отладочный код.
Release	**Дистрибутив**	Для компиляции программы, которая предназначена для распространения. Отладочная информация отключена, исполняемый файл будет меньшего размера, оптимизации должны использоваться.

При выборе одного из этих режимов будут загружены опции компилятора, которые были сохранены последний раз, когда данный режим был активен. То есть это своего рода один из способов установки или сброса опций.

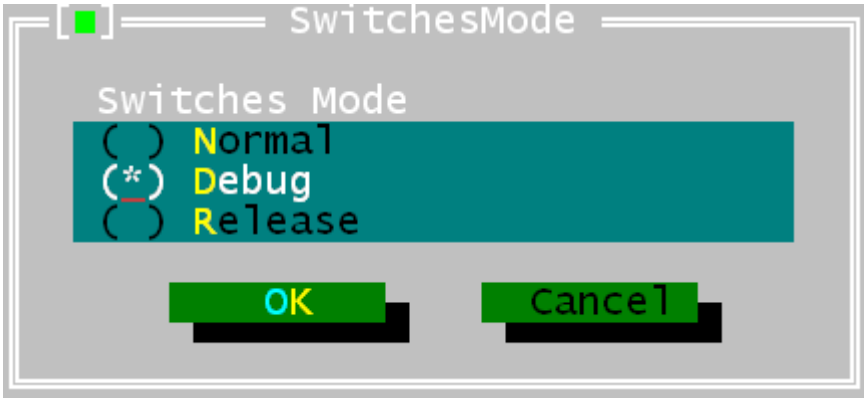


Рис. 6.32. Окно выбора режима.

Когда вы сохраняете настройки, убедитесь, что выбран нужный вам режим компилятора, иначе может получиться так, что опции компилятора режима отладки будут применены в режиме дистрибутива. Окно выбора режима показано на рис. 6.32.

6.12. Настройки IDE

Кроме большого количества описанных выше параметров IDE можно изменять цвета, разрешение экрана и другие настройки конфигурации IDE. Получить доступ к этим настройкам можно через подменю ENVIRONMENT меню OPTIONS.

6.12.1. Предпочтения

Диалоговое окно ПРЕДПОЧТЕНИЯ (рис. 6.33) вызывается командой меню OPTIONS – ENVIRONMENT – PREFERENCES.

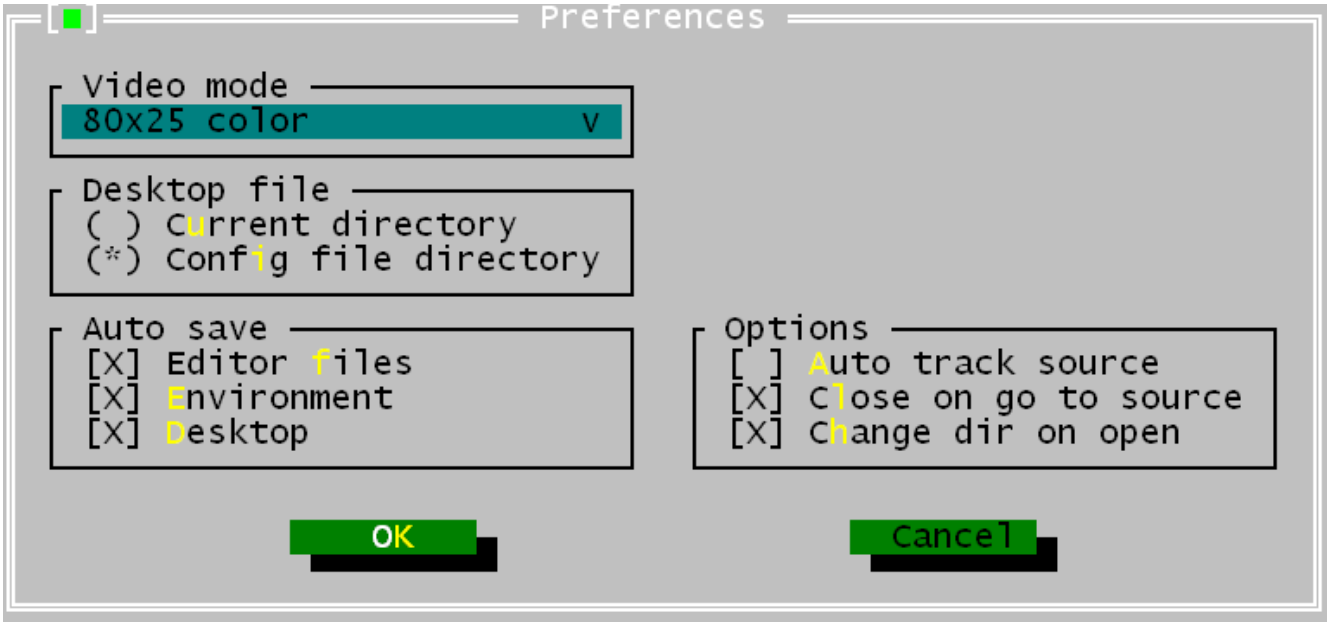


Рис. 6.33. Окно ПРЕДПОЧТЕНИЯ.

Имя	Перевод	Описание
Video mode	**Видеорежим**	В выпадающем списке, который находится в верхней части окна, можно выбрать видеорежим. Список доступных видеорежимов определяется операционной системой, на которой работает IDE.

ПРИМЕЧАНИЕ:

1. Видеорежим можно выбрать, нажав пробел, или щёлкнув по строке списка. При раскрытом выпадающем списке и открытом диалоговом окне новый видеорежим не будет применён.
2. Для DOS-версии IDE учтите следующее: если используются режимы VESA, то скорость обновления дисплея может быть очень низкой. На старых видеокартах (1998 г. и ранее) по возможности используйте драйвер UniVBE от SciTech (можно загрузить с http://www.informatik.fh-muenchen.de/ifw98223/vbehz.htm).

Имя	Перевод	Описание
Desktop file	**Файл рабочего стола**	Здесь можно указать место сохранения файла рабочего стола: в текущей директории (Current Directory) или в директории, в которой был найден конфигурационный файл (Config file Directory).
Auto save	**Авто-сохранение**	Здесь можно указать, какие файлы будут автоматически сохраняться при запуске программы или при выходе из IDE. Значения могут быть следующие:
Editor files (Файлы редактора)		Содержимое всех открытых окон редактора будет сохранено.
Environment (Окружение)		Текущие настройки окружения будут сохранены.
Desktop (Рабочий стол)		В файле рабочего стола будут сохранены все настройки рабочего стола (открытые окна, списки истории, точки останова и т.п.).
Options	**Опции**	
Auto track source		Автоматически отслеживать исходный код.
Close on go to source (Закрывать при переходе в исходник)		Если установлен, то окно сообщений закрывается, когда выполняется переход к строке исходного кода.
Change dir on open (Изменять директорию при открытии)		Когда файл открыт, то каталог, в котором находится файл, становится текущим каталогом.

6.12.2. Рабочий стол

Диалог `РАБОЧИЙ СТОЛ` позволяет указать элементы рабочего стола, какие необходимо сохранять между сеансами работы. То есть эти элементы сохраняются при выходе из IDE, и восстанавливаются при следующем запуске. Сохранение выполняется в файле **fp.dsk**. Окно `РАБОЧИЙ СТОЛ` вызывается командой меню `OPTIONS – ENVIRONMENT – DESKTOP` и показано на рис. 6.34.

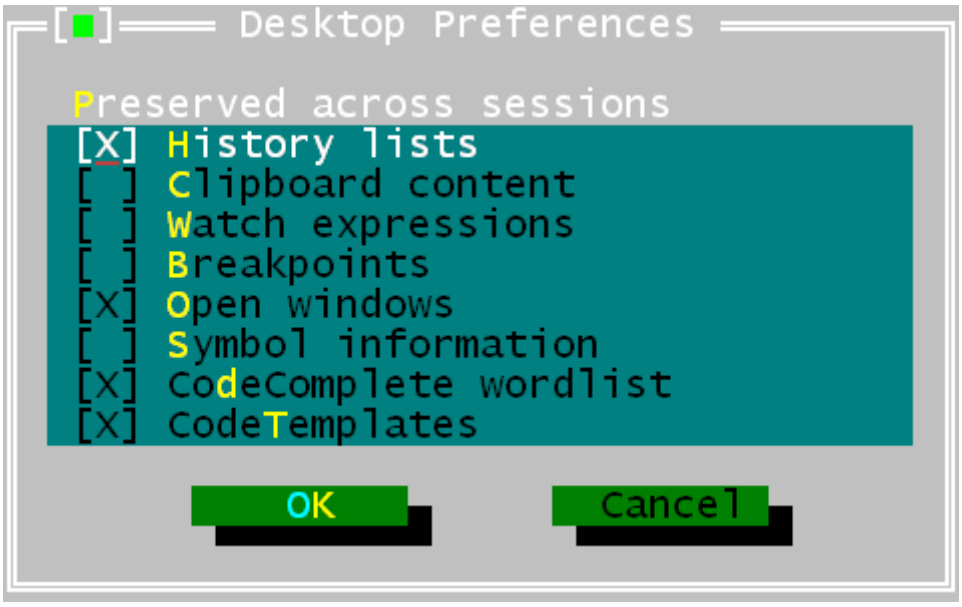


Рис. 6.34. Окно РАБОЧИЙ СТОЛ.

Элементы, которые могут быть сохранены при завершении сеанса и восстановлены при следующем запуске, перечислены ниже.

Элемент	Перевод	Описание
History list	**Список истории**	Большинство полей ввода данных имеют список истории, где хранятся ранее вводимые в эти поля значения. Эти значения можно затем выбрать из списка истории. Если эта опция установлена (по умолчанию), то история сохраняется в файле рабочего стола.
Clipboard content	**Содержимое буфера обмена**	Если опция установлена, то содержимое буфера обмена также сохраняется на диск. По умолчанию НЕ установлена.
Watch expressions	**Наблюдаемые выражения**	Если опция установлена, то наблюдаемые выражения сохраняются в файле рабочего стола. По умолчанию НЕ установлена.
Breakpoints	**Точки останова**	Если опция установлена, то все точки останова с их свойствами сохраняются в файле рабочего стола. По умолчанию НЕ установлена.
Open windows	**Открытые окна**	Если опция установлена (по умолчанию), то список открытых в редакторе файлов сохраняется в файле рабочего стола и при следующем запуске IDE все окна будут восстановлены.
Symbol information	**Символьная информация**	Если опция установлена, то информация для символьного обозревателя сохраняется в файле рабочего стола. По умолчанию НЕ установлена.
CodeComplete wordlist	**Список слов завершения кода**	Если опция установлена (по умолчанию), то список слов, используемых для завершения кода, сохраняется в файле рабочего стола.
CodeTemplates	**Шаблоны кода**	Если опция установлена (по умолчанию), то определённые шаблоны кода сохраняются в файле рабочего стола.

ПРИМЕЧАНИЕ:

Формат файла рабочего стола отличается для разных версий. Поэтому, если вы установили новую версию, может потребоваться удаление файла **fp.dsk** из всех мест поиска (то есть из всех каталогов, где IDE ищет этот файл).

6.12.3. Редактор

Диалоговое окно настроек редактора показано на рис. 6.35.

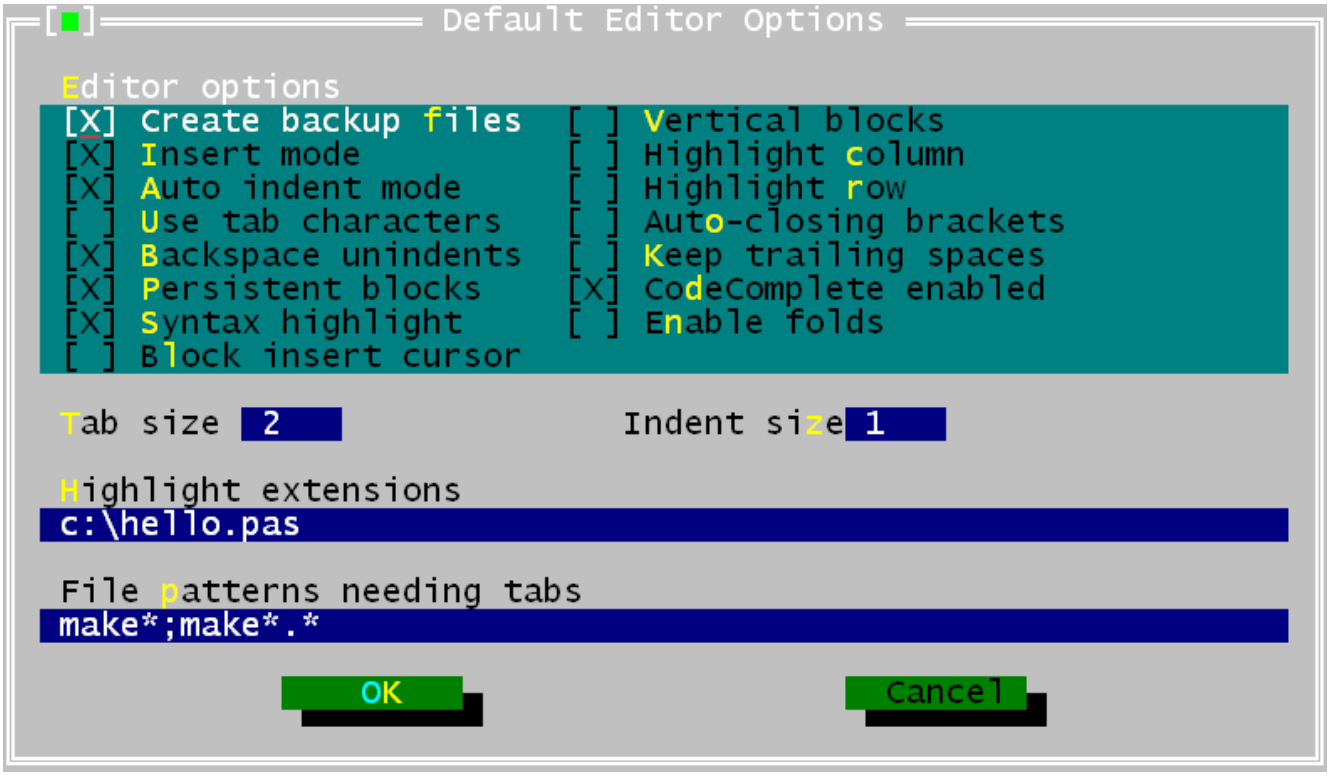


Рис. 6.35. Окно РЕДАКТОР.

В окне настроек редактора можно настроить поведение окон редактора. Учтите, что некоторые опции имеют эффект только на вновь открытых окнах, а на уже открытых окнах не сказываются (например, вертикальные блоки, подсветка строк/столбцов).

Следующие настройки могут быть установлены:

Опция	Перевод	Описание
Create backup files	**Создавать резервные копии файлов**	Определяет, должен ли редактор сохранять резервные копии и восстанавливать файлы из резервных копий. По умолчанию включена.
Insert mode	**Режим вставки**	Стартовать с режимом вставки.
Auto indent mode	**Режим авто отступа**	Включает режим «умного» отступа. Это означает, что при нажатии клавиши ENTER, курсор будет перемещён на ту же вертикальную позицию, с которой начинается текущая строка. По умолчанию включена.
Use tab characters	**Использовать символы табуляции**	Использовать символ табуляции при нажатии клавиши TAB. Обычно при нажатии клавиши TAB вместо символа табуляции вставляются пробелы. Если эта опция установлена, то будут вставляться символы табуляции. По умолчанию отключена.
Backspace unindents	**Удалять авто отступы**	Нажатие клавиши BackSpace будет удалять автоотступы, если они имеются в начале текущей строки. Вместо удалённого автоотступа будет помещён первый символ строки. По умолчанию опция включена.
Persistent blocks	**Фиксированные блоки**	Если выполнено выделение, то при перемещении курсора выделение не разрушается, то есть выделенный блок остаётся выделенным (перемещение курсора мышью снимает выделение). По умолчанию опция включена.
Syntax highlight	**Подсветка синтаксиса**	Использовать подсветку синтаксиса для типов файлов, которые перечислены в списке расширений подсвечиваемых файлов (**Highlight extensions** – см. ниже). По умолчанию опция включена.
Block insert cursor	**Курсор в режиме вставки**	Установить вид курсора как блок вместо символа подчёркивания. По умолчанию курсор имеет вид прямоугольника (блока) в режиме вставки, и отображается как символ подчёркивания в обычном режиме. Установка этой опции, которая по умолчанию выключена, меняется местами вид курсора.
Vertical blocks	**Вертикальные блоки**	При выделении блока, который содержит несколько строк, выделение не будет распространяться до конца строки в пределах этого блока. То есть будет выделяться только прямоугольник, границы которого определяются положением курсора. По умолчанию опция выключена.
Highlight column	**Подсветка колонки**	Если опция включена, то колонка, где находится курсор, будет подсвечиваться (то есть все символы в этой колонке будут окрашены цветом подсветки, по умолчанию - белым). По умолчанию опция выключена.
Highlight row	**Подсветка строки**	Если опция включена, то текущая строка (то есть строка, где установлен курсор) будет выделена цветом. По умолчанию опция выключена.
Auto closing brackets	**Автоматически закрывать скобки**	При включенной опции, если напечатать символ скобки («(»,«{» или «[»), то закрывающая скобка («)»,«}» или «]») будет вставлена автоматически. По умолчанию опция выключена.
Keep trailing spaces	**Сохранять пробелы в конце**	При сохранении файла все пробелы в конце строк удаляются. Эта опция отключает данную функцию, то есть если опция включена, то все пробелы в конце строк будут сохраняться в файл. По умолчанию опция выключена.
Codecomplete enabled	**Включить завершение кода**	Опция включает функцию завершения кода. По умолчанию опция включена.
Enable folds	**Включить сворачивание**	Включает сворачивание кода. По умолчанию опция выключена.
Tab size	**Размер табуляции**	Количество пробелов, которые вставляются в текст при нажатии клавиши TAB.
Indent size	**Размер отступа**	Количество пробелов, на которое перемещается блок текста при вызове функции отступа. По умолчанию равно 2.
Highlight extensions	**Расширения для подсветки**	Если подсветка синтаксиса включена, то в редакторе будут подсвечиваться тексты файлов, список расширений которых перечислен в этой строке. Файлы должны разделяться точкой с запятой (;). По умолчанию указаны файлы ***.pas;*.pp;*.inc**.
File patterns needing tabs	**Шаблоны файлов, которым требуется табуляция**	Для некоторых типов файлов требуется использовать реальные символы табуляции, а не пробелы. В этом поле можно ввести список таких файлов. Для этих файлов всегда будут использоваться символы табуляции. По умолчанию это файлы **make*;make*.***.

ПРИМЕЧАНИЕ
Введённые изменения не будут применены в текущем окне редактора. Чтобы изменения вступили в силу, нужно закрыть и снова открыть окно. При этом все изменения будут потеряны, если перед закрытием программы не будет выполнено сохранение опций.

6.12.4. Клавиатура и мышь

Диалоговое окно настроек клавиатуры и мыши вызывается через меню `OPTIONS – ENVIRONMENT – KEYBOARD & MOUSE`, показано на рис. 6.36. Здесь можно установить настройки клавиатуры и мыши, такие как чувствительность мыши.

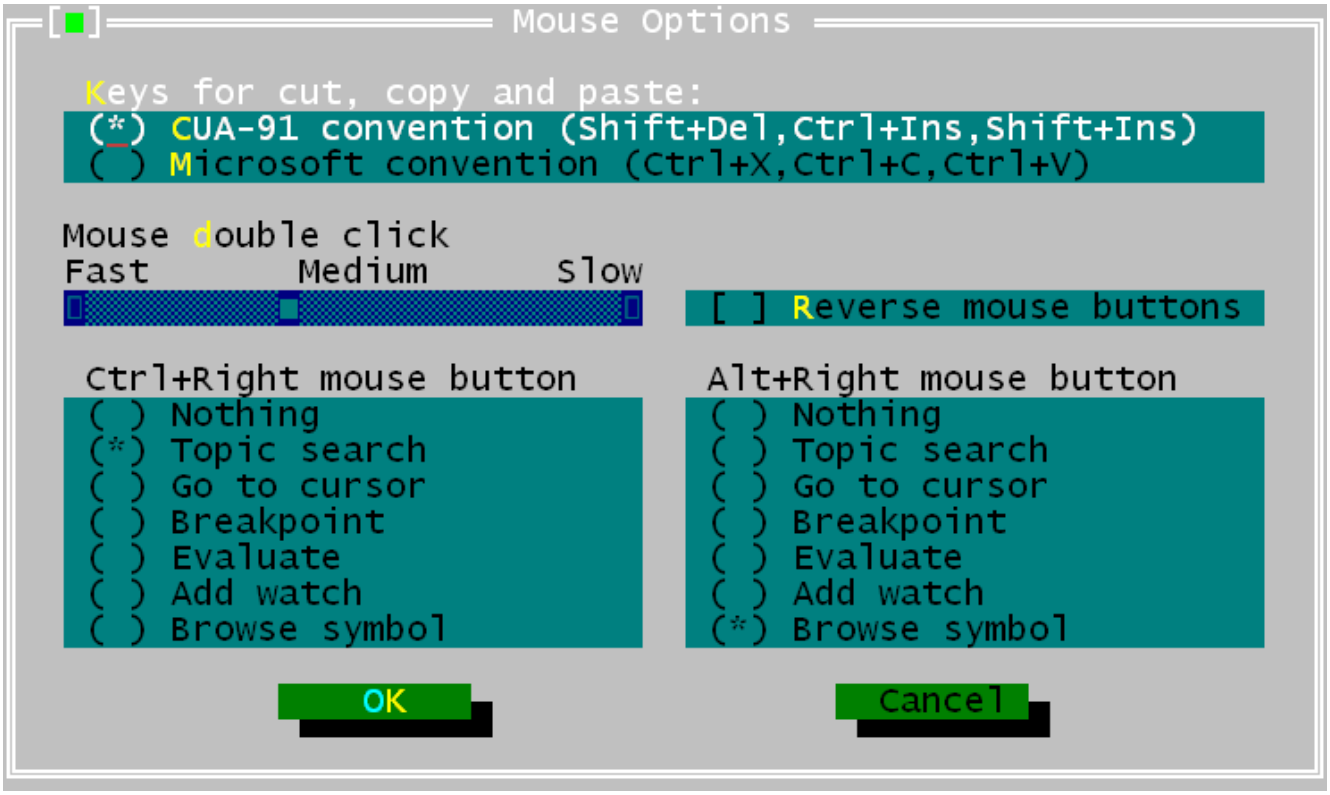


Рис. 6.36. Окно настроек клавиатуры и мыши.

Следующие настройки могут быть установлены:

Опция	Перевод	Описание
Key for copy, cut and paste	**Клавиши для копирования, вырезания и вставки**	Устанавливает клавиши для работы с буфером обмена: • Соглашения CUA-91 (SHIFT+DEL, CTRL+INS, SHIFT+INS) • Соглашения Microsoft (CTRL+X, CTRL+C, CTRL+V)
Mouse double click	**Двойной щелчок мыши**	Этот ползунок можно использовать для регулировки скорости двойного щелчка мышью. FAST (быстро) – время между двумя щелчками будет очень небольшим. SLOW (медленно) – время между двумя щелчками достаточно большое.
Reverse mouse buttons	**Реверс кнопок мыши**	Поведение левой и правой кнопок мыши можно поменять местами, установив этот флажок. Это может оказаться полезным для левшей.
CTRL+RIGHT mouse buttons	**CTRL+правая кнопка мыши**	Назначает действие для правой кнопки мыши при удерживаемой клавише CTRL.
ALT+RIGHT mouse buttons	**ALT+правая кнопка мыши**	Назначает действие для правой кнопки мыши при удерживаемой клавише ALT.

Следующие действия могут быть назначены для правой кнопки мыши при удерживаемых клавишах CTRL или ALT:

Действие	Перевод	Описание
Nothing	**Ничего**	Никакие действия не связаны с этими событиями.
Topic search	**Поиск раздела**	Выполняется поиск ключевого слова, на котором установлен курсор мыши, в разделе справки.
Go to cursor	**Перейти к курсору**	Программа выполняется до строки, где расположен курсор мыши.
Breakpoint	**Точка останова**	Устанавливает точку остановка в позиции курсора мыши.
Evaluate	**Вычисление**	Определяет значение переменной, на которой установлен курсор мыши.
Add watch	**Добавить наблюдаемое выражение**	Добавляет переменную, на которой находится курсор мыши, в список наблюдаемых выражений.
Browse symbol	**Отобразить символ**	Символ, на котором находится курсор мыши, отображается в обозревателе.

6.13. Справочная система

Больше информации о работе с IDE или о различных вызовах RTL, разъяснения относительно синтаксиса языка Паскаль и т.п. можно найти в справочной системе. Справочная система активируется клавишей F1.

6.13.1. Навигация по справочной системе

Справочная система содержит ссылки на справочные материалы. Эти ссылки связаны с другими разделами справки. Ссылки отмечены цветом, отличным от цвета текста. Гиперссылки могут быть активированы одним из следующих способов:

- Щелчком по ссылке кнопкой мыши
- Используя клавиши TAB и SHIFT-TAB для перемещения по ссылкам и нажатием клавиши ENTER для перехода к разделу, связанному со ссылкой

Нажатие комбинации SHIFT-F1 отображает содержание справочной системы на экране. Для возврата к предыдущему разделу нужно нажать ALT-F1. Это также работает, если окно справки не отображается на экране (в этом случае окно будет активировано).

6.13.2. Работа с файлами справки

IDE содержит справочную систему, которая может отображать справочные файлы следующих форматов:

Тип файла	Описание
TPH	Формат справочного файла Turbo Pascal.
INF	Формат справочного файла OS/2.
NG	Формат Norton Guide Help.
HTML	HTML-файлы.

В будущих версиях могут быть добавлены новые форматы файлов. Однако и указанные выше типы охватывают широкий спектр справочных файлов.

ПРИМЕЧАНИЕ
Относительно поддержки файлов HTML следует учесть, что:

1. Обозреватель HTML-файлов справочной системы имеет ограничения и может просматривать только базовые HTML-файлы (исключая просмотр изображений), так как он предназначен только для просмотра файлов справки Free Pascal.
2. Если обозреватель находит в файле справки графический файл, то он пытается найти файл с таким же именем, но с расширением **.ans**. Если такой файл будет найден, то он будет интерпретирован как псевдографический ANSI-файл, и будет использоваться для отображения картинки с помощью текстовых символов. Все диалоговые окна IDE отображаются по такому принципу.

Через меню `HELP – FILES` можно вызвать окно (рис. 6.37), где можно добавить справочные файлы, или удалить их из списка.

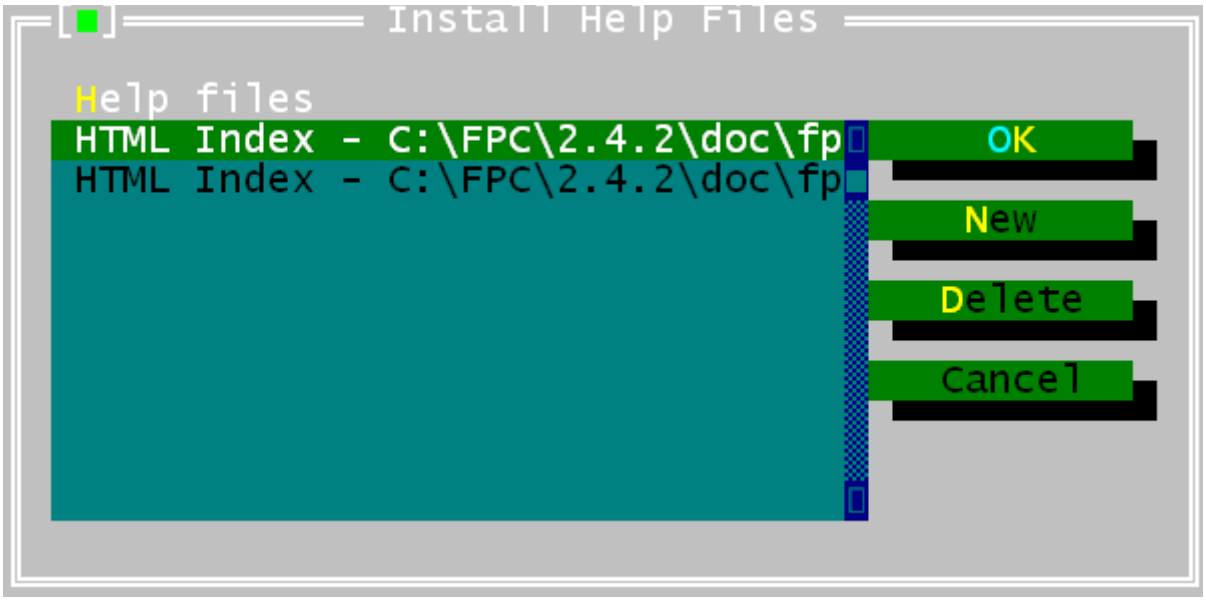


Рис. 6.37. Окно выбора файла справки.

Диалоговое окно содержит список файлов, которые будут отображаться в таблице содержания окна справочной системы. Каждый элемент в списке содержит заголовок и путь к файлу. В этом окне можно выполнить следующие действия:

Действие	Перевод	Описание
New	**Новый**	Добавляет новый файл. Вызывает окно, где можно ввести путь к справочному файлу. Если добавляется файл HTML, то диалоговое окно будет отображать заголовок этого файла. Этот заголовок затем будет включён в содержание справки.
Delete	**Удалить**	Удаляет выделенный файл из справочной системы. Файл НЕ удаляется с жёсткого диска, а только из справочной системы.
Cancel	**Отмена**	Отменяет все изменения и закрывает окно.
OK	**OK**	Сохраняет изменения и закрывает окно.

Документация Free Pascal в HTML-формате может быть добавлена в справочную систему. Эта документация затем может просматриваться из IDE. Если Free Pascal установлен с помощью инсталлятора, то инсталлятор должен автоматически добавить документацию в список справочных файлов, если документация была установлена.

6.13.3. Окно О ПРОГРАММЕ

Окно «О программе» вызывается через меню `HELP – ABOUT`. В этом окне отображается такая информация, как номер версии IDE, дата сборки, версии компилятора и отладчика и т.п. Если вы будете связываться с нами по вопросам ошибок в программе, то используйте информацию этого окна для идентификации версий IDE.

Здесь также отображается информация о правообладателях.

6.14. Горячие клавиши

Большая часть сочетаний клавиш, используемых IDE совместима с WordStar и должна быть знакома пользователям Turbo Pascal. Ниже перечислены следующие таблицы:

1. В таблице 6.4 описаны некоторые сочетания клавиш для управления IDE и Справочной системой.
2. В таблице 6.5 описаны сочетания клавиш для компиляции, запуска и отладки программ.
3. В таблице 6.6 описаны клавиши для навигации.
4. В таблице 6.7 описаны клавиши для работы с редактором.
5. В таблице 6.8 описаны все команды для работы с блоками.
6. В таблице 6.9 описаны команды выбора/замены.
7. В таблице 6.10 описаны некоторые основные сочетания клавиш, которые не вошли в вышеперечисленные категории.

Таблица 6.4. Основные команды.

Команда	Клавиши	Альтернатива
Вызов справки	F1	
Переход к предыдущему разделу справки	ALT – F1	
Поиск слова от позиции курсора в справке	CTRL – F1	
Вызов индекса справки	SHIFT – F1	
Закрыть активное окно	ALT – F3	
Развернуть/свернуть окно	F5	
Переместить/развернуть окно	CTRL – F5	
Перейти в следующее окно	F6	
Перейти в предыдущее окно	SHIFT – F6	
Меню	F10	
Контекстное меню	ALT – F10	
Список окон	ALT – 0	
Активировать другое окно	ALT – <Цифра>	
Вызвать утилиту grep	SHIFT – F2	
Выход из IDE	ALT – X	

Таблица 6.5. Компилятор.

Команда	Клавиши	Альтернатива
Сбросить отладчик/программу	CTRL – F2	
Отобразить стек	CTRL – F3	
Запустить как дальний вызов с позиции курсора	F4	
Перейти на пользовательский экран	ALT – F5	
Отслеживать информацию	F7	
Добавить объект для наблюдения	CTRL – F7	
Пошаговое выполнение	F8	
Установить точку останова в текущую строку	CTRL – F8	
Создать	F9	
Выполнить	CTRL – F9	
Компилировать активный исходный файл	ALT – F9	
Вызвать окно сообщений	F11	
Вызвать окно сообщений компилятора	F12	

Таблица 6.6. Навигация по тексту.

Команда	Клавиши	Альтернатива
На один символ влево	Стрелка влево	CTRL – S
На один символ вправо	Стрелка вправо	CTRL – D
На одну строку вверх	Стрелка вверх	CTRL – E
На одну строку вниз	Стрелка вниз	CTRL – X
На одно слово влево	CTRL – Стрелка влево	CTRL – A
На одно слово вправо	CTRL – Стрелка вправо	CTRL – F
Прокрутка на одну строку вверх	CTRL – W	
Прокрутка на одну строку вниз	CTRL – Z	
На одну страницу вверх	Page Up	
На одну страницу вниз	Page Down	
В начало строки	Home	CTRL – Q – S
В конец строки	End	CTRL – Q – D
К первой строке окна	CTRL – Home	CTRL – Q – E
К последней строке окна	CTRL – End	CTRL – Q – X
К первой строке файла	CTRL – Page Up	CTRL – Q – R
К последней строке файла	CTRL – Page Down	CTRL – Q – C
К предыдущей позиции курсора	CTRL – Q – P	
Найти соответствующий разделитель блока	CTRL – Q – [	
Найти последний соответствующий разделитель блока	CTRL – Q –]	

Таблица 6.7. Редактирование.

Команда	Клавиши	Альтернатива
Удалить символ	DEL	CTRL – G
Удалить символ слева	BACKSPACE	CTRL – H
Удалить строку	CTRL – Y	
Удалить конец строки от курсора	CTRL – Q – Y	
Удалить слово	CTRL – T	
Вставить строку	CTRL – N	
Переключиться в режим вставки	INSERT	CTRL – V

Таблица 6.8. Команды для работы с блоками.

Команда	Клавиши	Альтернатива
Перейти к началу выделенного текста	CTRL – Q – B	
Перейти в конец выделенного текста	CTRL – Q – K	
Выделить текущую строку	CTRL – K – L	
Распечатать выделенный текст	CTRL – K – P	
Выделить текущее слово	CTRL – K – T	
Удалить выделенный текст	CTRL – DEL	CTRL – K – Y
Копировать выделенный текст до позиции курсора	CTRL – K – C	
Переместить выделенный текст к позиции курсора	CTRL – K – V	
Копировать выделенный текст в буфер обмена	CTRL – INSERT	
Переместить (вырезать) выделенный текст в буфер обмена	SHIFT – DEL	
Сдвинуть вправо выделенный блок на одну колонку	CTRL – K – I	
Сдвинуть влево выделенный блок на одну колонку	CTRL – K – U	
Вставить текст из буфера обмена	SHIFT – INSERT	
Вставить файл	CTRL – K – R	
Записать выделенный текст в файл	CTRL – K – W	
Перевести в верхний регистр выделенный блок	CTRL – K – N	
Перевести в нижний регистр выделенный блок	CTRL – K – O	
Перевести в верхний регистр слово	CTRL – K – E	
Перевести в нижний регистр слово	CTRL – K – F	

Таблица 6.9. Команды выбора/замены.

Команда	Клавиши	Альтернатива
Отметить начало выделяемого текста	CTRL – K – B	
Отметить конец выделяемого текста	CTRL – K – K	
Снять выделение	CTRL – K – Y	
Расширить выделение на один символ влево	SHIFT – Стрелка влево	
Расширить выделение на один символ вправо	SHIFT – Стрелка вправо	
Расширить выделение до начала строки	SHIFT – HOME	
Расширить выделение до конца строки	SHIFT – END	
Расширить выделение на одну строку вверх по той же колонке	SHIFT – Стрелка вверх	
Расширить выделение на одну строку вниз по той же колонке	SHIFT – Стрелка вниз	
Расширить выделение на одно слово влево	CTRL – SHIFT – Стрелка влево	
Расширить выделение на одно слово вправо	CTRL – SHIFT – Стрелка вправо	
Расширить выделение на одну страницу вверх	SHIFT – Page Up	
Расширить выделение на одну страницу вниз	SHIFT – Page Down	
Расширить выделение до начала файла	CTRL – SHIFT – HOME	CTRL – SHIFT – Page Up
Расширить выделение до конца файла	CTRL – SHIFT – END	CTRL – SHIFT – Page Down

Таблица 6.10. Прочие команды.

Команда	Клавиши	Альтернатива
Сохранить файл	F2	
Открыть файл	F3	
Поиск	CTRL – Q – F	
Повторить поиск	CTRL – L	
Поиск и замена	CTRL – Q – A	
Установить метку	CTRL – K – N (где N от 0 до 9)	
Перейти к метке	CTRL – Q – N (где N от 0 до 9)	
Отмена	ALT – BACKSPACE	

7. ПЕРЕНОС И СОВМЕСТИМОСТЬ КОДА

7.1. Режимы компилятора Free Pascal

Команда Free Pascal старается создать компилятор, который мог бы компилировать код, максимально совместимый с кодом для Turbo Pascal, Delphi или компиляторов Паскаля Mac: это должно сделать перенос кода, написанного для этих компиляторов, максимально лёгким.

В то же время разработчики Free Pascal внедрили несколько расширений в язык Object Pascal. Для согласования этих отличий, а также для того, чтобы люди, работающие с Turbo Pascal и Delphi были уверены, что смогут легко перенести свой код, компилятор Free Pascal имеет концепцию «режимов компиляции». Выбор того или иного режима компиляции выполняется путём установки/сброса соответствующих функциональных переключателей. Это позволяет определить совместимый режим, в котором будут поддерживаться только те функции, которые присущи оригинальному компилятору. На данный момент поддерживаются 5 режимов:

Режим	Описание
FPC	Это оригинальный режим компилятора Free Pascal. Здесь поддерживаются все языковые конструкции, исключая классы, интерфейсы и исключения. Объекты поддерживаются в этом режиме. Этот режим установлен по умолчанию.
OBJFPC	Этот режим аналогичен режиму FPC, но здесь поддерживаются ещё и классы, интерфейсы и исключения.
TP	Режим совместимости с Turbo Pascal. В этом режиме компилятор пытается имитировать поведение компилятора Turbo Pascal насколько это возможно. Очевидно, может компилироваться только 32-х или 64-битный код.
DELPHI	Режим совместимости с Delphi. В этом режиме компилятор пытается имитировать поведение компилятора Delphi насколько это возможно. Все функции Delphi 7 работают. Функции Delphi 7 для .NET технологии не работают.
MACPAS	Режим совместимости с MacPascal. В этом режиме компилятор пытается выполнять все функции, которые поддерживаются в Mac Pascal. В редких случаях выполняется попытка компилировать универсальный интерфейс.

Режим компиляции можно установить для каждого модуля, то есть каждый модуль может иметь свой собственный режим компиляции. Это означает, что в одной программе можно комбинировать модули с разными режимами компиляции. Режим можно установить одним из следующих способов:

1. В командной строке с помощью параметра **–M**.
2. В исходном файле, с помощью директивы `{$MODE }`

В обоих случаях имя режима передаётся в качестве аргумента. Если в исходном коде модуля или программы не указан режим, то используется режим, переданный через командную строку. Если в исходном коде режим указан, то он используется независимо от того, какой режим был указан в командной строке.

Пример компиляции модуля с параметром `–M` приведён ниже:

```
fpc -MOBJFPC myunit
```

тот же результат получим с помощью директивы **MODE**:

```
{$MODE OBJFPC}
Unit myunit;
```

Директива **MODE** должна быть всегда помещена перед оператором **uses** в модуле или программе, так как установки режима могут влиять результат загрузки дополнительных модулей после того, как первый модуль будет загружен.

Учтите, то директива **MODE** – это глобальная директива, то есть она распространяется на весь модуль. Только одна директива может быть определена в модуле.

Режим не оказывает влияния на доступность модулей. Все доступные модули могут быть использованы, независимо от режима, который использует программа или модуль.

7.2. Turbo Pascal

Free Pascal был специально разработан максимально похожим на Turbo Pascal. Конечно, с определёнными отличиями (ограничениями). Некоторые из этих отличий связаны с тем, что Turbo Pascal был разработан для 16-битной архитектуры, в то время как Free Pascal – для 32-битной/64-битной. Другие отличия являются результатом того, что Free Pascal предназначен для работы с большим количеством операционных систем.

В общих чертах можно сказать, что если вы храните ваши исходные коды в ANSI Pascal, то вы не будете иметь проблем с переносом кода в Free Pascal с Turbo Pascal или даже Delphi. Для большей совместимости конструкции Turbo Pascal поддерживаются, особенно если вы используете переключатели **–Mtp** или –**MObjfpc**.

В следующих разделах представлен список конструкций Turbo Pascal и Delphi, которые не поддерживаются в Free Pascal, а также пути, которые используются Free Pascal для развития Turbo Pascal.

7.2.1. Вещи, которые не работают

Здесь перечислен список элементов, которые определены/предоставляются Turbo Pascal, но не поддерживаются Free Pascal. Если возможно, то указаны причины этого.

1. Дубликаты меток позволяются в Turbo Pascal, но не в Free Pascal. Это ошибка Turbo Pascal, поэтому поддерживать её нет смысла.
2. В Turbo Pascal список параметров предварительно объявленных функций и процедур не обязательно должен соответствовать точно. В Free Pascal это обязательно, так как может влиять на работу механизма перегружаемых функций Free Pascal. Однако с помощью опции **–M** (см. раздел «5.1.5. Параметры для исходных кодов (опции языка)») можно преодолеть это ограничение.
3. В Turbo Pascal переменные MEM, MEMW, MEML и PORT для работы с памятью и доступа к портам недоступны в системном модуле. Это делает программу зависимой от операционной системы. Под ДОС в модуле GO32 находятся конструкции MEM. Под LINUX модуль **ports** предоставляет такие конструкции для переменной **Ports**.
4. Turbo Pascal позволяет создавать процедуры и переменные с именами, которые запрещается использовать в Free Pascal, так как они являются ключевыми. То есть эти слова зарезервированы в Free Pascal и Delphi, но не являются зарезервированными в Turbo Pascal, например PROTECTED, PUBLIC, PUBLISHED, TRY, FINALLY, EXCEPT, RAISE. Для решения этой проблемы используйте параметр **–Mtp**, если требуется откомпилировать код Turbo Pascal, содержащий эти слова. См. также приложение B (список зарезервированных слов).
5. Зарезервированные слова Turbo Pascal FAR и NEAR игнорируются. Это связано с тем, что данные слова использовались из-за ограничений 16-битной среды окружения, а Free Pascal – это 32/64-битный компилятор.
6. Директива INTERRUPT будет работать в Free Pascal только для целевой платформы DOS. Другие операционные системы не позволяют обрабатывать прерывания в пользовательских программах.
7. По умолчанию Free Pascal использует синтаксис ассемблера AT&T. В основном из-за того, что Free Pascal использует GNU as. Однако доступны и другие формы ассемблера. Подробности см. в «Руководстве программиста».

8. Turbo Vision для Turbo Pascal имеет аналог для Free Pascal, который называется Free Vision и на 100% совместим с Turbo Vision.
9. Модуль **overlay** от Turbo Pascal недоступен. Это также связано с тем, что Free Pascal 32/64-битный компилятор, поэтому размер программы не должен вызывать проблем.
10. Параметры командной строки компилятора отличаются.
11. Опции и директивы компилятора по большей части совместимы, но их несколько больше.
12. Бинарные файлы модулей несовместимы. Это значит, что в проекте Free Pascal вы не сможете использовать файлы откомпилированных в Turbo Pascal модулей с расширением .TPU.
13. В Free Pascal структура **TextRec** (для внутреннего описания файлов) не является бинарно-совместимой с TP или Delphi.
14. В Free Pascal множества по умолчанию являются 4-байтными. Это значит, что некоторые множества, которые возможны в Turbo Pascal, не допустимы в Free Pascal. Однако эту проблему можно решить установкой размера множества (см. «Руководство программиста»).
15. Файл открыт только для вывода (используя **fmOutput**), если он открыт процедурой **Rewrite**. Чтобы иметь возможность чтения файла, он должен быть открыт процедурой **Reset**.
16. Деструкторы Turbo Pascal позволяют иметь параметры. Это не выполняется в Free Pascal: по умолчанию деструкторы в Free Pascal не имеют параметров. Это ограничение может быть снято опцией **–So**.
17. Turbo Pascal допускает более одного деструктора на объект. В Free Pascal может быть только один деструктор. Это ограничение может быть также снято опцией **–So**.
18. Порядок вычисления выражений не всегда одинаков в Turbo Pascal и в Free Pascal. В следующем выражении:

    ```
    a := g(2) + f(3);
    ```

    не гарантируется, что `g(2)` будет вычислено раньше, чем `f(3)`.
19. В Free Pascal необходимо использовать оператор адреса @ при определении процедурных переменных.

7.2.2. Вещи, которые являются дополнительными

Здесь приведён список элементов, которые возможны в Free Pascal, но не существуют в Turbo Pascal или Delphi.

1. Функции Free Pascal могут возвращать сложные типы, такие как записи и массивы.
2. В Free Pascal можно использовать функцию, возвращающую значение, внутри самой функции, как переменную. Например:

    ```
    function a : longint;
      begin
       a := 12;
       while a > 4 do
         begin
           {...}
         end;
      end;
    ```

    Описанный выше пример будет работать в TP, но компилятор будет подразумевать, что `a > 4` – это рекурсивный вызов. Если вы и на самом деле хотите выполнить рекурсивный вызов, вы должны добавить скобки () после имени функции:

```
function a : longint;
  begin
   a := 12;
   {Это рекурсивный вызов}
   while a() > 4 do
     begin
       {...}
     end;
  end;
```

3. В Free Pascal частично поддерживаются языковые конструкции Delphi (см. «Руководство программиста»).
4. В Free Pascal процедура `Exit` позволяет вернуть значение для функции:

```
function a : longint;
  begin
   a := 12;
   {Это рекурсивный вызов}
   while a() > 4 do
     begin
       exit(a*67); {При выходе функция вернёт значение a*67 }
     end;
  end;
```

5. Free Pascal поддерживает перегружаемые функции. Это означает, что вы можете определить множество функций с одинаковыми именами, но различными аргументами. Например:

```
procedure DoSomething (a : longint);
  begin
    {...}
  end;

procedure DoSomething (a : real);
  begin
    {...}
  end;
```

Затем вы можете вызвать процедуру `DoSomething` с параметрами типа `longint` или `real`. Эта особенность подразумевает, что функция, объявленная первой, должна быть всегда определена в заголовке полным именем, например:

```
procedure x (v : longint); forward;
  {...}

procedure x; {Это перегрузит ранее объявленную процедуру x}
  begin
    {...}
  end;
```

Такая конструкция вызовет генерацию ошибки компилятором, потому что компилятор не найдёт объявление процедуры **`procedure`** `x (v : longint)`. Вместо такого варианта вы должны объявлять вашу процедуру `x` следующим образом:

```
procedure x (v : longint);
  {Это правильное определение предварительно объявленной x}
  begin
    {...}
  end;
```

Опция командной строки **–So** (см. раздел «5.1.5. Параметры для исходных кодов (опции языка)») отключает перегрузку функций. Если вы используете эту опцию, то описанные выше примеры будут компилироваться как в Turbo Pascal.

6. Перегрузка операторов. Free Pascal допускает перегрузку операторов, то есть вы можете определить оператор «+» для матриц.
7. В файловых системах FAT16 и FAT32 поддерживаются длинные имена файлов.

7.2.3. Режим совместимости с Turbo Pascal

Если вы компилируете программу с опцией **–Mtp**, то компилятор будет пытаться имитировать работу компилятора Turbo Pascal следующими способами:

- Назначая процедурные переменные, не требуя наличия оператора @. Одно из различий между компиляторами Turbo Pascal и Free Pascal заключается в том, что Free Pascal требует указывать символ адреса при связывании значения с процедурной переменной. В режиме совместимости с Turbo Pascal это не требуется.
- Перегрузка процедур отключена. Если перегрузка процедур отключена, то нет необходимости повторять список параметров в заголовке функции.
- Предварительно объявленная процедура не требует полного списка параметров, если они уже определены. В соответствии с правилами перегрузки функций в Free Pascal вы должны всегда указывать полный список параметров функции при её объявлении, даже если она была объявлена как функция с ранним связыванием, используя слово `Forward`. В режиме совместимости с Турбо Паскаль перегрузка не функционирует, поэтому вы можете пропустить список параметров:

```
Procedure a (L : Longint); Forward;
  ...

Procedure a; {Нет необходимости повторять (L : Longint)}
  begin
    ...
  end;
```

- Вызовы рекурсивных процедур обрабатываются различно. В следующем примере:

```
Function expr : Longint;
  begin
    ...
    Expr:=L;
    Writeln(Expr);
    ...
  end;
```

в режиме совместимости Турбо Паскаль функция будет вызвана рекурсивно, когда будет выполняться оператор `Writeln`. В Free Pascal результатом функции будет вывод значения на экран. Чтобы рекурсивно вызвать функцию в Free Pascal, исходный код должен быть таким:

```
Function expr : Longint;
  begin
    ...
    Expr:=L;
    Writeln(Expr());
    ...
  end;
```

- Любой текст после завершающего оператора `End.` будет игнорироваться. В нормальном режиме этот текст также выполняется.

- Вы не можете назначать процедурные переменные нетипизированным указателям. Следующий пример неправильный:

```
a: Procedure;
b: Pointer;
begin
  b := a; // Будет сгенерирована ошибка.
```

- Оператор @ нужно печатать, если применяются процедурные переменные.
- Не допускаются вложенные комментарии.

ПРИМЕЧАНИЕ
Функции `MemAvail` и `MaxAvail` больше недоступны в Free Pascal версии 2.0 и выше. Причина этой несовместимости следующая: на современных операционных системах вопрос «Доступной свободной памяти» не актуален для приложений по следующим причинам:

1. Процессор запрашивает у ОС, сколько памяти нужно приложению. Другие приложения могут разместиться в любом месте памяти.
2. Понятие «свободная память» включает в себя не только свободную оперативную память, но и место на диске (файл подкачки, размер которого можно изменить), а также память, распределённую под другие приложения, но не используемую ими, которую можно временно занять.

По этой причине программы, которые используют функции `MemAvail` и `MaxAvail`, должны быть переписаны таким образом, чтобы не использовать эти функции. Сделать это можно тремя способами:

1. Использовать исключения для отслеживания ситуаций нехватки памяти.
2. Установить глобальную переменную `ReturnNilIfGrowHeapFails` в значение TRUE и проверять после каждого выделения памяти, что указатель не равен `Nil`.
3. Не заботиться о выделении памяти и просто объявить функцию `MaxAvail`, которая всегда будет возвращать `High(LongInt)` или другую подобную константу.

7.2.4. Пояснения по длинным именам файлов по DOS

Длинные имена файлов поддерживаются, начиная с Windows 95. Компиляция для целевой платформы Windows подразумевает, что длинные имена файлов поддерживаются всеми функциями, которые так или иначе имеют доступ к именам файлов или дисков.

Кроме этого, Free Pascal поддерживает использование длинных имён файлов в системном модуле и модуле DOS, также выполняющихся для go32v2. Системный модуль содержит логическую переменную `LFNSupport`. Если значение этой переменной равно TRUE, то все функции системного модуля и модуля DOS будут использовать длинные имена файлов. Если они доступны. Это должно быть так на Windows 95/98, но не на Windows NT/2000. Системный модуль проверяет это, вызывая функцию DOS `71A0h` и проверяя, поддерживаются ли длинные имена на диске `C:`.

Возможно отключить поддержку длинных имён файлов. Для этого нужно переменной `LFNSupport` присвоить значение FALSE. Но в общем случае рекомендуется компилировать программы, которые работают с длинными именами файлов, что присуще ОС Windows.

7.3. Перенос кода Delphi

Перенос кода Delphi должен быть абсолютно безболезненным. В режиме `Delphi` компилятор пытается имитировать поведение компилятора Delphi, насколько это возможно. Этот режим можно включить, используя опцию командной строки `-Mdelphi`, или вставив в исходный код следующий текст перед словами `unit` или `program`:

```
{$IFDEF FPC}
{$MODE DELPHI}
{$ENDIF FPC}
```

Это гарантирует, что код будет компилироваться в режимах Delphi и FPC.

И всё же есть несколько вещей, которые работать не будут. В режиме совместимости Delphi компилятор работает подобно компилятору Delphi 7. Новые конструкции в более поздних версиях Delphi (особенно версии, которые работают с технологией .NET) не поддерживаются.

7.3.1. Отсутствующие языковые конструкции

По уровню языковой совместимости FPC максимально совместим с Delphi: он может компилировать FreeCLX, свободно распространяемую библиотеку Widget, которая поставлялась с Delphi 6/7 и Kylix.

На текущий момент отсутствуют только следующие языковые конструкции:

1. Методы `Dynamic` в реальности являются одинаковыми с `Virtual`.
2. `Const` для параметра процедуры применять нет необходимости, так как по сути это переменная или значение, помещённое по ссылке.
3. Пакеты не поддерживаются.

Имеется несколько конструкций встроенного ассемблера, которые не поддерживаются, потому что Free Pascal изначально разрабатывался как платформенно-независимый компилятор и маловероятно, что эти конструкции будут поддерживаться в будущем.

Учтите, что опция `-Mobjfpc` позволяет добиться большей совместимости с Delphi, но этот режим более строгий, чем режим Delphi. Основные отличия следующие:

1. Параметры и локальные переменные методов не могут иметь одинаковые имена со свойствами класса, в котором они определены.
2. Оператор адреса (@) необходимо использовать при объявлении процедурной переменной (или обработчиков событий).
3. `AnsiStrings` не включен по умолчанию.

7.3.2. Отсутствующие вызовы и API несовместимость

Delphi сильно привязана к Windows. Поэтому она содержит множество ориентированных на Windows API функций (таких как поиск и открытие файлов, загрузка библиотек).

Free Pascal изначально разрабатывался для использования на разных платформах, поэтому некоторые функции, очень специфические для Windows, в Free Pascal отсутствуют. Перечисленные ниже основные пункты должны быть хорошо осмыслены.

- По умолчанию Free Pascal генерирует консольные приложения. Это означает, что при разработке программ для Windows вы должны явно включить тип приложения в исходном коде:

```
{$APPTYPE GUI}
```

- Модуль `Windows` предоставляет доступ к большинству основных функций API Win32. Одинаковые функции могут иметь различный список параметров: вместо объявления параметра, переданного по ссылке (`var`), используется указатель (как в `C`). Для большинства случаев Free Pascal предоставляет перегруженные версии таких функций.
- `Widestrings`. Управление `Widestrings` не автоматизировано в Free Pascal, так как различные платформы имеют различные методы расшифровки `Widestrings` и `Multi-Byte Character Sets`. FPC поддерживает Widestrings, но не всегда использует такую же кодировку, что и Windows. Учтите, что для правильной работы с Widestrings, вам необходимо подключать модуль `cwstring` для платформ Unix/LINUX. Этот модуль инициализирует менеджер Widestrings с необходимыми функциями, которые используют библиотеку C для выполнения всех необходимых функций.
- Потоки. В настоящий момент Free Pascal не предоставляет управление потоками для всех платформ. В Unix потребуется связь с библиотеками `C`, если имеется необходимость получить управление потоками в приложении FPC. Это означает, что модуль `cthreads` должен быть подключен для управления потоками.
- Имеется множество примеров `SetLastOSError`. Это не поддерживается и не будет поддерживаться.
- Чувствительность к регистру имён файлов. Паскаль – это не чувствительный к регистру язык, поэтому используемые имена также должны быть не чувствительны к регистру символов. Free Pascal гарантирует нечувствительность к регистру путём поиска файлов с именами в нижнем регистре. Kylix этого не гарантирует, поэтому могут возникнуть проблемы при наличии двух файлов с одинаковыми именами в разных регистрах.
- RTTI не сохраняется таким же путём, как в Delphi. Формат максимально совместим, но может отличаться. Это не должно вызвать проблем, если используются API модуля `TypeInfo` и не выполняется попыток прямого доступа к RTTI.
- По умолчанию размер множеств отличается от размера, принятого в Delphi. Но установить размер можно директивой компилятора или опцией командной строки.
- Аналогично, размер перечисляемых типов по умолчанию отличается от размера, принятого в Delphi. Опять же, установить размер можно директивой компилятора или опцией командной строки.
- В общем случае не делается предположений о внутренней структуре сложных типов, таких как записи, объекты, классы и связанные с ними структуры. Например, макет таблицы VMT отличается, выравнивание полей в записи может отличаться и т.п.
- То же можно сказать про основные типы: на разных процессорах старший и младший байты слова или целого числа могут размещаться в разных местах как на процессоре Intel.
- Имена локальных переменных и аргументы методов не должны совпадать с именами свойств или полей класса: это плохая привычка, так как это может привести к конфузам, связанным с одинаковыми именами.

7.3.3. Режим совместимости с Delphi

Переключение в режим совместимости с Delphi имеет следующие эффекты:

1. Поддержка классов, исключений и потоков будет включена.
2. Модуль `objpas` будет загружаться первым. Этот модуль переопределяет некоторые основные типы: например, `Integer` будет 32-битным числом.
3. Оператор адреса (@) больше не нужно устанавливать для обработчиков событий (то есть назначать процедурные переменные или свойства).
4. Имена локальных переменных и параметры методов в классах могут совпадать с полями или свойствами класса.
5. Ключевое слово `String` подразумевает `AnsiString` по умолчанию.
6. Оператор перегрузки операций выключен.

7.3.4. Лучшие правила переноса

Если обнаружены отличия в вызовах Delphi/FPC, то лучше подумать, как сделать так, чтобы не вставлять в исходный код оператор IFDEF. Например, можно создать отдельный модуль, который будет использоваться только для совместимости с FPC.
Отсутствующие/несовместимые функции могут быть реализованы в этом модуле. Это сделает код более читабельным и лёгким для изучения.

Если вы найдёте отличие языковой конструкции, то контактируйте с командой Free Pascal для сообщения об ошибке.

7.4. Создание переносимого кода

Free Pascal разработан как кросс-платформенный компилятор. Это означает, что основные модули RTL используются на всех платформах, а поведение компилятора одинаково для всех платформ (насколько это возможно). Язык Object Pascal одинаков для всех платформ. Тем не менее, FPC поставляется с множеством модулей, которые не являются переносимыми, но предоставляют доступ ко всем возможностям, которые имеются в платформе. Следующие моменты необходимо обдумать при написании переносимого кода:

- Старайтесь избегать применения модулей, специфических для какой-либо платформы. Системный модуль, модули объектов и классов, а также модуль `SysUtils`, гарантируют работу на всех системах. Также как и модуль DOS, но в меньшей степени.
- Избегайте использовать прямой доступ к устройствам. Ограниченно доступ к устройствам возможен для большинства платформ при помощи модулей `Video`, `Mouse` и `Keyboard`.
- Не используйте трудные для расшифровки соглашения по именам файлов. Подробнее см. ниже.
- Учитывайте внутреннее представление типов. Разные процессоры хранят информацию по-разному.
- Если есть необходимость использовать какие-то специфические системные функции, то лучше выделить эти функции в отдельный модуль. Тогда перенос кода ограничится переопределением данного модуля для другой платформы.
- Не используйте ассемблер без особой необходимости, так как у каждого процессора свой набор команд. Некоторые команды не будут работать даже на разных процессорах одного семейства.
- Не думайте, что указатели и целые числа имеют одинаковый размер. Это справедливо для 32-битных процессоров Intel, но не обязательно для других процессоров. Тип `PtrInt` – это псевдоним для целочисленного типа, который имеет одинаковый размер с указателем. Функция `SizeInt` используется во всех случаях, где необходимо узнать размер.

Системный модуль содержит несколько констант, которые описывают доступ к файловой системе:

Константа	Описание
AllFilesMask	Маска файла, которая возвращает все файлы в каталоге. * - на Unix-подобных платформах, *.* - на DOS и Windows-подобных платформах.
LineEnding	Символ или строка, которые описывают маркер конца строки, используемый на текущей платформе. Обычно это один из #10, #13#10 или #13.
LFNSupport	Логическое значение, которое определяет, поддерживает ли система длинные имена файлов.
DirectorySeparator	Символ, который работает как разделитель между каталогами в имени файла.
DriveSeparator	Для систем, которые поддерживают буквы дисков, это символ, который используется для разделения буквы диска от пути к файлу.
PathSeparator	Символ, используемый для разделения элементов в списке (в особенности PATH).
maxExitCode	Максимальное значение для процесса exitcode.
MaxPathLen	Максимальная длина имени файла, включая путь к файлу.
FileNameCaseSensitive	Логическое значение, определяющее, нужно ли учитывать регистр при работе с файлами.
UnusedHandle	Значение, используемое для неиспользованного/повреждённого дескриптора файла.
StdInputHandle	Значение дескриптора файла, связанного со стандартным устройством ввода. Не всегда равно 0 (ноль).
StdOutputHandle	Значение дескриптора файла, связанного со стандартным устройством вЫвода. Не всегда равно 1.
StdErrorHandle	Значение дескриптора файла, связанного со стандартным устройством диагностики вЫвода. Не всегда равно 2.
CtrlZMarksEOF	Логическое значение, определяющее, нужно ли отмечать символом #26 конец файла (по соглашению на старых MS-DOS).

Для упрощения написания переносимого кода, файловые процедуры Free Pascal модулей `system` и `sysutils` воспринимают разделитель каталогов в пути к файлу одинаково для Windows и Unix, то есть / и \ являются эквивалентными. Это означает, что вы можете использовать символ / в Windows, и он будет трансформирован в обратный слеш (\). И наоборот, в Unix вы можете использовать \.

Этой функцией управляют две предопределённые переменные в системном модуле:

Константа	Описание
AllowDirectorySeparators	Определяет символы, которые используются в именах файлов и обрабатываются как разделители директорий. Они трансформируются в символ DirectorySeparator.
AllowDriveSeparators	Определяет символы, которые используются в именах файлов и обрабатываются как разделители дисков. Они трансформируются в символ DriveSeparator.

8. УТИЛИТЫ, ПОСТАВЛЯЕМЫЕ С FREE PASCAL

Кроме компилятора и библиотеки RTL, Free Pascal предоставляет несколько полезных программ и модулей. В данном разделе описаны эти программы и модули.

8.1. Демонстрационные программы и примеры

Комплект демонстрационных программ входит в состав дистрибутива Free Pascal. Эти программы не имеют никаких других целей, кроме демонстрации возможностей Free Pascal. Они размещены в каталоге **DEMO**.

Доступны примеры всех программ, приведённых в документации. Все примеры размещены в каталоге **examples**.

8.2. fpcmake

Fpcmake – это конструктор программ Free Pascal.

Эта программа читает конфигурационный файл **Makefile.fpc** и конвертирует его в соответствующий **Makefile** для чтения GNU make и для компиляции вашего проекта. Это похоже на работу **GNU autoconf** или **Imake** для создания X-проектов.

Fpcmake принимает имена файлов как параметры командной строки. Для каждого из этих файлов программа будет создавать **Makefile** в том же каталоге, где размещён файл, перезаписывая существующие файлы.

Если в программу не передаются параметры, то она только пытается прочитать файл **Makefile.fpc** в текущем каталоге и пытается создать **Makefile** исходя из данных файла **Makefile.fpc**. Любые существующие файлы с именем **Makefile** будут стёрты.

Формат конфигурационного файла **Makefile.fpc** и другие подробности описаны в приложениях к документу «Руководство программиста».

8.3. fpdoc – документирование модулей Паскаль

Fpdoc – это программа, которая генерирует полную документацию с перекрёстными ссылками для модуля. Она генерирует документацию для каждого идентификатора, найденного в модуле в разделе **interface**. Документация может быть нескольких форматов, таких как HTML, RTF, текстовый документ, гипертекстовая страница и LaTeX. В отличие от других инструментов документирования, документация может быть в отдельном файле (в XML формате), поэтому набор документов не будет хаотичным. Совместно с программой **makeskel** создаётся пустой XML-файл, добавляются элементы для новых идентификаторов.

Программы **Fpdoc** и **makeskel** описаны в документе «Документирование кода Free Pascal».

8.4. h2pas – конвертер заголовочных файлов C в модули Паскаля

h2pas позволяет конвертировать заголовочный файл C в модуль на языке Паскаль. Программа может обрабатывать конструкции языка C, которые найдены в заголовочном файле C, а затем транслирует его в дубликат на Паскале.

В разделе «8.4.2. Конструкции» подробно описаны возможности транслятора. Модуль с объявлениями на Паскале можно затем использовать для доступа к коду, написанному на C.

Выходной результат работы программы записывается в файл с таким же именем, что и оригинальный заголовочный файл C, который был использован как входной файл, но с расширение **.pp**. Выходной файл, создаваемый программой **h2pas**, можно настраивать разными способами с множеством опций.

8.4.1. Опции

Выходом программы **h2pas** можно управлять с помощью следующих опций:

Опция	Описание
-d	Использовать **external;** для всех объявленных процедур и функций.
-D	Использовать **external ИмяБиблиотеки Имя 'ИмяФункции'** для всех объявленных процедур и функций.
-e	Генерировать серию констант вместо перечисляемых типов для конструкции **enum** языка C.
-i	Создавать подключаемый файл вместо модуля (пропуская заголовок модуля).
-l ИмяБиблиотеки	Определяет имя библиотеки для объявления внешней функции.
-o ВыходнойФайл	Определяет имя выходного файла. По умолчанию это имя входного файла, но с расширением **.pp**.
-p	Использовать букву **P** в качестве первой буквы параметров типа **pointer** вместо символа ^.
-s	Очищать комментарии из входного файла. По умолчанию комментарии конвертируются в комментарии Паскаля, но могут быть перемещены, затем комментарии нужно будет проверить.
-t	Приписывать в начале имён типов букву **T** (используется в соответствии с соглашениями Borland, которые подразумевают, что все имена типов начинаются с буквы **T**).
-v	Заменять в функциях параметры-указатели на параметры-ссылки. Следует использовать с осторожностью, так как некоторые операции могут предполагать, что указатель равен **Nil**.
-w	Заголовочный файл является заголовочным файлом Win32 (добавляет поддержку некоторых специальных макросов).
-x	Обрабатывать SYS_TRAP заголовочных файлов PalmOS.

8.4.2. Конструкции

Программа распознаёт следующие объявления и операторы C:

Элемент	Описание
Объявления	Объявления будут заменяться на константы в Паскале, если они являются простыми объявлениями. Макросы заменяются (если это возможно) на функции. Однако аргументы всегда являются целыми числами, поэтому их нужно заменить вручную при необходимости. Простые выражения в объявлениях операторов распознаются в большинстве случаев, как арифметические операторы: сложение, вычитание, умножение, деление, логические операторы, операторы сравнения, операторы сдвига. Конструкции C (A ? B : C) также распознаются и транслируются в конструкции Паскаля с оператором **IF** (однако это может вызвать проблемы).
Операторы препроцессора	Условные команды препроцессора распознаются и транслируются в эквивалентные директива компилятора Паскаля. Специальные команды #ifdef __cplusplus Также распознаются и удаляются.

Элемент	Описание
Определение типов	Определение типа (typedef) заменяется в Паскале на соответствующий тип. Следующие основные типы распознаются: • char заменяется на char. • float заменяется на real (=double в Free Pascal). • int заменяется на longint. • long заменяется на longint. • long int заменяется на longint. • short заменяется на integer. • unsigned заменяется на cardinal. • unsigned char заменяется на byte. • unsigned int заменяется на cardinal. • unsigned long int заменяется на cardinal. • unsigned short заменяется на word. • void игнорируется. Эти типы также заменяются, если они появляются в аргументах функций и процедур.
Функции и процедуры	Функции и процедуры транслируются как есть. Параметр-указатель может быть заменён на параметр-ссылку (используя аргумент **var**) при помощи опции **–p** командной строки. Функции, которые имеют переменное количество аргументов, заменяются на функции с модификатором **cvar** (это используется для аргумента **array of const**).
Спецификаторы	Спецификатор **extern** распознаётся, однако он игнорируется. Спецификатор **packed** также распознаётся и заменяется на директиву PACKRECORDS. Спецификатор **const** также распознаётся, но игнорируется.
Модификаторы	Если указана опция **–w**, то распознаются следующие модификаторы: STDCALL CDECL CALLBACK PASCAL WINAPI APIENTRY WINGDIAPI как определено в дескрипторах win32. В добавок опция **-x** распознаёт модификатор SYS_TRAP.
Перечисления	Конструкции **enum** заменяются на перечисляемые типы. Имейте ввиду, что в C перечисляемые типы могут иметь значения, связанные с ними. Free Pascal также позволяет это делать в некоторой степени. Если вы знаете, что с перечислениями связаны значения, то лучше использовать опцию **–e**, которая заменяет перечисления на серии целочисленных констант.
Объединения	Объединения (unions) заменяются на записи.
Структуры	Структуры заменяются на записи Паскаля с C-упаковкой.

8.5. h2paspp – препроцессор для h2pas

h2paspp может быть использован как постой препроцессор для **h2pas**. Он удаляет некоторые конструкции, которые вызывают затруднения у **h2pas**. **h2paspp** читает один или более заголовочных файлов C и выполняет их предварительную обработку, записывая результат в файл с тем же именем, что и оригинал, как это было описано выше. Он не выполняет полную обработку всех символов C, но обрабатывает следующие директивы:

Элемент	Описание
#define Символ	Определяет новый **Символ**. Учтите, что макросы не поддерживаются.
#if Символ	Текст, следующий за этой директивой, будет включён, если **Символ** определён.
#ifdef Символ	Текст, следующий за этой директивой, будет включён, если **Символ** определён.
#ifndef Символ	Текст, следующий за этой директивой, будет включён, если **Символ** НЕ определён.
#include ИмяФайла	Эта директива удаляется, за исключением случая, когда используется опция **–l**, которая подключает файл **ИмяФайла** и записывает его в выходной файл.
#undef Символ	Этот **Символ** будет не определён.

8.5.1. Применение

h2paspp принимает один или более имён файлов в качестве параметров и обрабатывает эти файлы. Программа будет читать входные файлы и записывать результат в выходной файл с тем же именем, что и оригинал, если не включена опция **–o**. В случае включения этой опции, указывается имя выходного файла. Учтите, что может быть только один выходной файл.

8.5.2. Опции

Программа имеет небольшое количество опций, которые позволяют управлять её поведением:

Опция	Описание
-dСимвол	Определяет **Символ** перед стартом процесса.
-h	Выводить короткую справку.
-l	Включать подключаемые файлы вместо перетаскивания подключаемых операторов.
-oВыходнойФайл	Эта опция позволяет записывать результат в файл с именем **ВыходнойФайл**. Учтите, что здесь можно указать только одно имя файла.

8.6. Программа ppudump

ppudump – это программа, которая показывает содержимое модуля Free Pascal. Она поставляется с компилятором. Вы можете выполнить следующую команду:

```
ppudump [Опции] foo.ppu
```

чтобы посмотреть содержимое модуля **foo.ppu**. Вы можете указать несколько файлов в одной командной строке.

Опции можно использовать для изменения количества отображаемой информации. По умолчанию отображается вся доступная информация. Вы можете установить уровень количества выводимой информации, используя опцию **–Vxxx**. Здесь **xxx** – это комбинация следующих символов:

Символ	Описание
h	Отображать информацию заголовка.
i	Отображать информацию интерфейсной части.
m	Отображать информацию раздела **implementation**.
d	Отображать только объявления интерфейсной части.
s	Отображать только идентификаторы интерфейсной части.
b	Отображать информацию обозревателя.
a	Отображать всю информацию (по умолчанию, если опция **–V** не используется).

8.7. Программа ppumove

ppumove – это программа, которая создаёт общие или статические библиотеки из нескольких модулей. Она похожа на программу **tpumove**, которая поставляется с Турбо Паскаль. Программа поставляется вместе с компилятором. Пример использования:

```
ppumove [Опции] unit1.ppu unit2.ppu ... unitn.ppu
```

В выше описанном примере **Опции**, это:

Опция	Описание
-b	Генерировать пакетный файл, содержащий связи и архивы команд, которые должны быть выполнены. Имя этого файла будет **pmove.sh** на LINUX (и Unix-подобных ОС), и **pmove.bat** на WINDOWS и DOS.
-d xxx	Установить каталог для размещения выходных файлов, где **xxx** – имя каталога.
-e xxx	Установить расширение файлов перемещаемых модулей, где **xxx** – расширение. По умолчанию это расширение **.ppl**. Вы не должны указывать точку при установке расширения.
-o xxx	Установить имя выходного файла, то есть имя файла, содержащего все модули. Этот параметр является обязательным, если вы используете несколько файлов. На LINUX программа ppumove присоединит к началу имени файла префикс lib, если его там нет, и добавит расширение, соответствующее типу библиотеки.
-q	Работать в фоновом режиме.
-s	Создавать статическую библиотеку вместо динамической. По умолчанию создаётся динамическая библиотека на LINUX.
-w	Указать программе, что она работает по Windows NT. Это изменит компонующей и архивирующей программ на **ldw** и **arw** соответственно.
-h или -?	Показывать короткую справку.

Работа программы заключается в следующем: она берёт каждый файл модуля и модифицирует его таким образом, что компилятор будет знать, какой код из этого модуля добавить в библиотеку. Новые файлы модулей будут иметь расширение **.ppl**, которое можно изменить с помощью опции **–e**. Затем программа поместит все объектные файлы модулей в одну библиотеку, статическую или динамическую (это определяется опцией **–s**).

Имя библиотеки должно устанавливаться опцией **–o**. При необходимости к началу имени будет добавляться префикс **lib** (для LINUX). Расширение будет установлено как **.a** для статических библиотек, для общедоступных библиотек расширение будет **.so** для LINUX и **.dll** для Windows NT и OS/2.

Следующий пример:

```
./ppumove -o both -e ppl ppu.ppu timer.ppu
```

Будет генерировать такой выходной файл для LINUX:

```
PPU-Mover Version 2.1.1
Copyright (c) 1998-2007 by the Free Pascal Development Team
Processing ppu.ppu... Done.
Processing timer.ppu... Done.
Linking timer.o ppu.o
Done.
```

И это создаст следующие файлы:

Файл	Описание
libboth.so	Общедоступная библиотека, содержащая код из файлов **ppu.o** и **timer.o**. Под Windows NT этот файл будет называться **both.dll**.
timer.ppl	Файл модуля, который указывает компилятору Free Pascal, где искать код модуля **timer** в библиотеке.
ppu.ppl	Файл модуля, который указывает компилятору Free Pascal, где искать код модуля **ppu** в библиотеке.

После этого вы должны использовать или включать в дистрибутив вашей программы файлы **libboth.so**, **timer.ppl** и **ppu.ppl**.

8.8. ptop – программа изящного форматирования кода Паскаль

8.8.1. Программа ptop

ptop – это программа изящного форматирования исходного кода, написанная Питером Грогоно (Peter Grogono). Программа основана на старых версиях Ledgard, Hueras и Singer, модернизирована командой Free Pascal (объекты, потоки, способность к изменению конфигурации и т.п.).

Способность к изменению конфигурации и законченный дизайн являются преимуществом этой программы перед множеством других подобных программа для Турбо Паскаль, например, SIMTEL.

Программа достаточно проста в работе:

```
ptop "[-v] [-i Отступ] [-b РазмерБуфера ][-c ФайлОпций] ВходнойФайл ВыходнойФайл"
```

Здесь параметр **ВходнойФайл** – это файл на Паскале, который требуется преобразовать. Преобразованный файл будет записан в **ВыходнойФайл**. Если **ВыходнойФайл** существует, то он будет перезаписан.

Несколько опций позволяют настроить поведение программы:

Опция	Описание
-h	Записывать по мере возможности параметры и синтаксис командной строки.
-c ptop.cfg	Читать некоторые конфигурационные настройки из указанного файла вместо использования внутренних настроек по умолчанию. Конфигурационный файл не обязателен, программа может работать без него. См. также опцию **–g**.
-i Отступ	Устанавливает количество пробелов для отступов от BEGIN END и других блоков.
-b РазмерБуфера	Устанавливает размер буфера для потока (по умолчанию 255). 0 – значение неправильное, которое игнорируется.
-v	Подробная информация. Обычно только количество прочитанных/записанных строк выходного файла и некоторые сообщения об ошибках.
-c ptop.cfg	Записать настройки по умолчанию в конфигурационный файл **ptop.cfg**. Содержимое этого файла затем можно изменить по вашему усмотрению, а затем использовать с опцией **–c**.

8.8.2. Конфигурационный файл ptop

Создавать и распространять конфигурационный файл нет необходимости, так как вы можете изменить стандартный конфигурационный файл по вашему усмотрению. Конфигурационный файл никогда не загружается по умолчанию, так что если вы хотите использовать его, вы должны всегда указывать параметр **-c ptop.cfg**.

Структура конфигурационного файла – это простой набор блоков, повторённых несколько раз (20…30) для каждого ключевого слова Паскаль, известного программе **ptop** (см. настроенный по умолчанию конфигурационный файл или исходный код программы **ptopu.pp**, чтобы посмотреть ключевые слова, которые известны программе).

Основной блок конфигурационного файла содержит одну или две строки, описывающие, как программа должна реагировать на определённое ключевое слово. Первое вхождение строки без квадратных скобок имеет следующий формат:

```
КлючевоеСлово=Опция1,Опция2,Опция3,…
```

Если одна из опций – это «dindonkey» (см. ниже), то вторая строка – с квадратными скобками – должна быть:

```
[КлючевоеСлово]=ДругоеКлючевоеСлово1,ДругоеКлючевоеСлово2,…
```

Вы можете увидеть блок, содержащий два типа идентификаторов: ключевые слова (КлючевоеСлово и ДругоеКлючевоеСлово1…2) и опции (Опция1…3).

Ключевые слова – это встроенные в структуру языка Паскаль идентификаторы, типа BEGIN, END, CASE, IF, THEN, ELSE, IMPLEMENTATION. По умолчанию конфигурационный файл содержит список большинства из них.

Кроме реальных ключевых слов языка Паскаль используются различные кодовые слова для операторов и комментариев (см. таблицу 8.1).

Таблица 8.1. Ключевые слова для операторов.

Кодовое слово	Оператор
casevar	: в конструкции case (не то же самое, что двоеточие - **colon**).
becomes	:=
delphicomment	//
opencomment	{ или (*
closecomment	} или *)
semicolon	;
colon	:
equals	=
openparen	[
closeparen	]
period	.

Кодовые слова опций определяют действия, которые будут выполнены, когда будет найдено ключевое слово перед этим кодовым словом (см. таблицу 8.2).

Опция «dindonkey», описанная в таблице 8.2, требует некоторых пояснений. «dindonkey» - это сокращение от «DeINDent ON associated KEYword» (Структурировать на связанном ключевом слове). Если это кодовое слово имеется как опция для какого-либо ключевого слова в первой строке, то далее во второй строке требуется указать это ключевое слово в квадратных скобках. Тогда для всех ключевых слов, перечисленных во второй строке, будет удалён отступ в исходном коде, если они следуют ПОСЛЕ указанного в квадратных скобках ключевого слова. Например, строки

```
else=crbefore,dindonkey,inbytab,upper
[else]=if,then,else
```

Означают следующее:

- Ключевым словом для этого блока является **else**, так как находится в левой части строк.
- Опция **crbefore** указывает на то, что никакого другого кода в этой строке перед ключевым словом **else** быть не должно.
- Опция **dindonkey** указывает программе, что нужно удалять отступы, если синтаксический анализатор находит любые ключевые слова, перечисленные в строке, где **else** заключено в квадратные скобки.
- Опция inbytab означает, что нужно делать отступ с помощью табуляции.
- Опция upper указывает программе, что ключевое слово нужно переводить в верхний регистр (**else** или **Else** будет преобразовано в **ELSE**).

Попытайтесь поэкспериментировать с конфигурационным файлом, пока не найдёте наиболее приемлемый для себя вариант. Способность к изменению конфигурации и возможности программы **ptop** являются довольно мощными. Например, если вы любите писать все ключевые слова большими буквами, то с помощью этой программы вы можете преобразовать все такие слова в слова в верхнем регистре.

Программа работает в фоновом режиме, поэтому для проверки её работы нужно сгенерировать исходный файл и попытаться откомпилировать его, если программа не выдала сообщений об ошибках.

Таблица 8.2. Опции.

Опция	Что делает
crsupp	Запрещает CR (возврат каретки) перед ключевым словом.
crbefore	Вставляет CR перед ключевым словом (не используется с crsupp).
blinbefore	Пустая строка перед ключевым словом.
dindonkey	Отменяет отступ для связанного ключевого слова (см. выше).
dindent	Отменяет отступ (всегда).
spbef	Пробел перед ключевым словом.
spaft	Пробел после ключевого слова.
gobsym	Печатает символы, которые следуют за ключевым словом, но не влияют на структуру. Печатает, пока не встретится завершающий символ (завершающие символы являются закодированными на аппаратном уровне в pptop, нет необходимости их заменять).
inbytab	Отступ с помощью табуляции.
crafter	Вставляет CR после ключевого слова.
upper	Печатает ключевое слово в верхнем регистре.
lower	Печатает ключевое слово в нижнем регистре.
capital	Печатает в верхнем регистре первую букву слова, остальные буквы в нижнем регистре.

8.8.3. Модуль ptopu

Исходные коды программы **PtoP** размещены в двух файлах. Один – это модуль, содержащий объект, который выполняет все основные операции по работе с кодом, другой – это командный процессор, выполняющий методы этого объекта таким образом, что их можно использовать в командной строке. Это решение делает возможным включать объект в программу (например, в IDE), а затем использовать его для форматирования кода.

Объект находится в модуле **PtoPU**, и объявлен следующим образом:

```
TPrettyPrinter=Object(TObject)
  Indent    : Integer; { How many characters to indent ? }
  InS       : PStream;
  OutS      : PStream;
  DiagS     : PStream;
  CfgS      : PStream;
  Constructor Create;
  Function PrettyPrint : Boolean;
end;
```

Использовать этот объект очень просто. Процедура использования объекта заключается в следующем:

1. Создать объект, используя конструктор.
2. Установить поток **InS**. Это открывает поток, из которого будет читаться исходный код Паскаль. Это обязательный шаг.
3. Установить поток **OutS**. Это открывает поток, в который будет записывать структурированный исходный код Паскаль. Это обязательный шаг.
4. Установить поток **DiagS**. Вся диагностика будет записываться в этот поток. Этот шаг не является обязательным. Если вы его не выполните, то диагностика выполняться не будет.
5. Установить поток **CfgS**. Конфигурация читается из этого потока (см. предыдущий раздел, где подробно описано конфигурирование). Этот шаг не является обязательным. Если вы его не выполните, то будет использоваться конфигурация по умолчанию.
6. Установить переменную **Indent**. Это количество пробелов для отступа. Символы табуляции не используются в программе. Этот шаг не является обязательным. Эта переменная инициализируется числом 2.
7. Вызвать **PrettyPrint**. Это начнёт чтение исходных кодов из **InS** и запись результата в **OutS**. Функция возвращает TRUE, если не было ошибок, иначе возвращает FALSE.

Таким образом, минимальный набор операторов будет таким:

```
Procedure CleanUpCode;
var
  Ins,OutS : PBufStream;
  PPRinter : TPrettyPrinter;
begin
  Ins:=New(PBufStream,Init('ugly.pp',StopenRead,TheBufSize));
  OutS:=New(PBufStream,Init('beauty.pp',StCreate,TheBufSize));
  PPrinter.Create;
  PPrinter.Ins:=Ins;
  PPrinter.outS:=OutS;
  PPrinter.PrettyPrint;
end;
```

использование потоков позволяет очень быстро форматировать код и является очень полезным при редактировании.

8.9. Программа rstconv

Программа **rstconv** преобразует файлы ресурсов (если вы используете раздел ресурсов) в файлы **.po**, которые затем могут быть обработаны программой **GNU msgfmt**.

Использовать программу очень просто. Возможны следующие опции:

Опция	Описание
-i Файл	Использовать указанный **Файл** вместо стандартного файла ввода **stdin**.
-o Файл	Записывать результат в указанный **Файл**. Эта опция обязательна.
-f Формат	Указывает формат результата. В настоящий момент поддерживается только один формат результата: **po** для **GNU gettext .po**. Это формат по умолчанию.

Пример:

```
rstconv -i resdemo.rst -o resdemo.po
```

конвертирует файл **resdemo.rst** в файл **resdemo.po**.

Более подробную информацию о программе вы можете найти в «Руководстве программиста» в разделе, описывающем ресурсы.

8.10 Программа unitdiff

8.10.1. Краткий обзор

Эта программа показывает различия между интерфейсными разделами двух модулей.

```
unitdiff [--disable-arguments] [--disable-private] [--disable-protected]
[--help] [--lang=language] [--list] [--output=filename] [--sparse]
file1 file2
```

8.10.2. Описание и использование

Программа сканирует один или два исходных файла модулей Free Pascal и выводит список всех доступных идентификаторов (для одного файла) или находит отличия в списках идентификаторов двух модулей (для двух файлов).

Вы можете выполнить программу только с одним обязательным параметром (с одним именем файла). В этом случае программа просто выведет список идентификаторов.

Для общего случая программа должна вызываться с двумя параметрами:

```
unitdiff input1 input2
```

В этом случае программа определит различия между интерфейсами двух модулей, или выведет список идентификаторов обоих модулей. Результат по умолчанию выводится в стандартное устройство вывода.

8.10.3. Опции

Большинство опций не являются обязательными. Настройки по умолчанию подходят для большинства случаев.

Опция	Описание
–disable-arguments	Не проверять аргументы функций и процедур. По умолчанию проверяется.
–disable-private	Не проверять поля и методы класса в разделе **private**. По умолчанию проверяются.
–disable-protected	Не проверять поля и методы класса в разделе **protected**. По умолчанию проверяются.
–help	Вывести короткую справку и выйти.
–lang=Язык	Установить язык для выходного файла. Устанавливает строки, используемые для заголовков в различных частях файлов документации (по умолчанию Английский). На текущий момент доступны следующие языки: **de**: Немецкий. **fr**: Французский. **nl**: Dutch.
–list	Отображать только список доступных идентификаторов для модуля или модулей. Если только один модуль указан в командной строке, то эта опция устанавливается автоматически.
–output=ИмяФайла	Определить, куда помещать выходной результат. По умолчанию результат выводится в стандартное устройство вывода (на экран).
–sparse	Использовать краткий режим, в котором выводятся только имена идентификаторов. Не выводятся типы и описания типов. По умолчанию описания типов выводятся.

9. МОДУЛИ, КОТОРЫЕ ПОСТАВЛЯЮТСЯ С FREE PASCAL

Здесь перечислены модули, которые поставляются вместе с дистрибутивом Free Pascal. Так как они отличаются в зависимости от операционной системы, то здесь сначала описываются универсальные (общие) модули, а затем описываются модули, предназначенные для работы с конкретной операционной системой.

9.1. Стандартные модули

Следующие модули являются стандартными и применяются для всех платформ, поддерживаемых Free Pascal. Ниже приведены краткие описания этих модулей.

Модуль	Описание
charset	Модуль предоставляет таблицу наборов символов.
cmem	Модуль используется для замены менеджера памяти Free Pascal на менеджер памяти библиотеки C.
crt	Этот модуль похож на одноимённый модуль Турбо Паскаль. Он выполняет цветной вывод на консоль, перемещает текстовый курсор и читает клавиатуру.
dos	Этот модуль предоставляет основные процедуры для доступа к функциям операционной системы. Включая поиск файлов, доступ к переменным окружения, получение версии операционной системы, получение и установка системного времени. Учтите, что некоторые из процедур этого модуля дублируют процедуры модуля **sysutils**.
dynlibs	Предоставляет кросс-платформенный доступ к динамическим библиотекам.
getopts	Этот модуль предоставляет вам механизм обработки параметров командной строки **GNU getopts**.
graph	Этот модуль является устаревшим. Он предоставляет основные графические обработчики, процедуры для рисования линий на экране, отображения текста в графическом режиме и т.п. Предоставляет те же функции, что и одноимённый модуль Турбо Паскаль.
heaptrc	Модуль для отладки при использовании кучи. При выходе из программы, он определяет использованную память и дампы не освобождённых блоков памяти (если имеются).
keyboard	Предоставляет основные процедуры для обработки нажатий клавиш клавиатуры независимо от платформы и поддерживает запись драйверов пользователя.
macpas	Модуль выполняет несколько функций, доступных только в режиме MACPAS. Этот модуль не нужно подключать. Он автоматически подключается в режиме MACPAS.
math	Этот модуль содержит общие математические процедуры (тригонометрические функции, логарифмы и т.п.), а также множество комплексных операций (суммирование массивов, функции нормализации и т.п.).
matrix	Модуль предоставляет процедуры для манипуляции с матрицами.
mmx	Модуль предоставляет поддержку расширений MMX для вашего кода.
mouse	Предоставляет основные процедуры обработки команд мыши независимо от платформы, поддерживая пользовательские драйверы.
objects	Предоставляет основные объекты для стандартных объектов Турбо Паскаль. Также объекты файловых потоков и потоков памяти, как отсортированные, так и не отсортированные коллекции и строковые потоки.
objpas	Используется для совместимости с Делфи. Вы не должны никогда загружать этот модуль явно, он загружается автоматически в режиме Делфи.
printer	Этот модуль предоставляет доступ к основным операциям с принтером, используя стандартные процедуры ввода-вывода.
sockets	Предоставляет доступ к сокетам и программированию TCP/IP.
strings	Предоставляет основные процедуры обработки строк для типа **pchar**, совместимых с подобными процедурами стандартной библиотеки C.
system	Этот модуль доступен для всех поддерживаемых платформ. Кроме всего прочего он включает в себя основные файловые операции вода-вывода, процедуры для работы с памятью, все вспомогательные процедуры компилятора и процедуры работы с каталогами.
strutils	Множество часто используемых дополнительных процедур для обработки строк.
dateutils	Процедуры для работы с датой и временем.
sysutils	Является альтернативой одноименному модулю Делфи. Включает в себя процедуры доступа к файлам, процедуры для работы с датой и временем, поиска файлов, преобразований строк и дат.
typinfo	Предоставляет доступ к информации в реальном времени, аналогичен Делфи.
variants	Предоставляет основные функции для работы с вариантными типами.
video	Предоставляет основные процедуры для работы с экраном, независимо от платформы с поддержкой пользовательских драйверов.

9.2. Модули для DOS

Модуль	Описание
emu387	Модуль предоставляет поддержку для эмулятора сопроцессора.
go32	Модуль предоставляет доступ для совместимости с расширителем GO32 DOS.
ports	Содержит различные конструкции port[] для низкоуровневого ввода-вывода.

9.3. Модули для WINDOWS

Модуль	Описание
wincrt	Для работы со стандартными консольными и **GUI** окнами, в отличие от модуля **crt**, который работает только с консольными приложениями.
Windows	Этот модуль предоставляет доступ ко всем функциям **Win32 API**. Была предпринята попытка сделать этот модуль максимально совместимым с Делфи, чтобы перенос кода с Делфи на Free Pascal был как можно проще.
opengl	Предоставляет доступ к низкоуровневым функциям **Windows OpenGL**.
winmouse	Предоставляет доступ к мыши в Windows.
ole2	Предоставляет доступ к объектам **OLE**, совместимым с Windows.
winsock	Предоставляет доступ в Windows к сокетам **API Winsock**.
Jedi windows header translations	Эти модули содержат Jedi-переводы заголовков **Windows API** и поставляются с дистрибутивом Free Pascal. Имена этих модулей начинаются с префикса **jw**, затем следует имя соответствующей API.

9.4. Модули для LINUX и BSD-подобных систем

Модуль	Описание
baseunix	Базовые операции Unix, в основном поднабор спецификации POSIX. Использование этого модуля возможно с большинством Unix-систем.
clocale	Этот модуль инициализирует национальные установки в модуле **sysutils** с настройками, полученными из библиотеки C.
cthreads	Этот модуль должен быть указан первым или вторым в разделе uses вашей программы: он будет использовать потоки **Posix** для включения потоков в вашей программе **FPC**.
cwstring	Если используются процедуры для работы с **widestring**, то этот модуль должен быть указан первым в разделе uses вашей программы: он будет инициализировать менеджер **widestring** в системном модуле с процедурами, которые используют функции библиотеки C для обработки преобразований **widestring** и для других операций с **widestring**.
errors	Возвращает строку, описывающую код ошибки операционной системы.
Libc	Это интерфейс для **GLibc** на системе **linux I386**. Он не будет работать на других платформах и предоставляется в основном для совместимости с **Kylix**.
oldlinux	Этот модуль устаревший. Он предоставляет доступ к операционной системе LINUX. Он предоставляет большинство процедур для обработки файлового вода-вывода. Он выполняет большинство конструкций библиотеки C, которые вы найдёте на Unix-системах. Однако рекомендуется использовать модули **baseunix**, **unixtype** и **unix**, так как они менее машиннозависимы.
ports	Содержит различные конструкции **port[]** для низкоуровневого ввода-вывода. Они предоставляются только для совместимости и не рекомендуются для частого использования. Программы, использующие эти конструкции, должны быть запущены под пользователем **root** или **setuid root**, и не обеспечивают должную безопасность для вашей системы.
termio	Процедуры управления терминалами, которые совместимы с процедурами библиотеки C.
unix	Дополнительные операторы Unix.
unixtype	Все типы, используемые обычно на Unix-платформах.

9.5. Модули для OS/2

Модуль	Описание
doscalls	Интерфейс для библиотеки **doscalls.dll**.
dive	Интерфейс для библиотеки **dive.dll**.
emx	Предоставляет доступ к расширениям EMX.
pm*	Модули интерфейсов для функций (GUI) Менеджера Презентаций (PM).
viocalls	Интерфейс для библиотеки работы с экраном **viocalls.dll**.
moucalls	Интерфейс для библиотеки работы с мышью **moucalls.dll**.
kbdcalls	Интерфейс для библиотеки работы с клавиатурой **kbdcalls.dll**.
moncalls	Интерфейс для библиотеки работы с мониторингом **moncalls.dll**.
winsock	Предоставляет доступ к (эмулированным) сокетам WINDOWS API Winsock.
ports	Содержит различные конструкции **port[]** для низкоуровневого ввода-вывода.

9.6. Доступность модулей

Стандартные модули доступны для каждой из поддерживаемых систем.

10. ОТЛАДКА ВАШИХ ПРОГРАММ

Free Pascal поддерживает отладочную информацию для GNU-отладчика **gdb** или его Windows-двойника **Insight**, или **ddd** для LINUX. Отладчик может записывать два типа отладочной информации:

- **stabs** – отладочная информация старого формата.
- **dwarf** – отладочная информация нового офрмата.

Оба формата распознаются GDB.

Эти разделы кратко описывают, как использовать функцию отладки. Мы не пытаемся полностью описать GNU-отладчик. Больше информации о GNU-отладчике можно найти в руководстве пользователя GNU.

Free Pascal также поддерживает **gprof**, программу протоколирования GNU (см. раздел 10.4).

10.1. Компилирование программы с поддержкой отладчика

Первым делом вы должны убедиться, что компилятор работает с поддержкой отладчика. К сожалению, нет способа проверить это в режиме реального времени, кроме как попытаться компилировать программу с поддержкой отладчика.

Для компилирования программы с поддержкой отладчика, просто укажите опцию **–g** в командной строке, как показано ниже:

```
fpc -g hello.pp
```

Это включит отладочную информацию в исполняемый файл, сгенерированный из вашего исходного кода. Учтите, что это сильно увеличит размер вашей программы. Поэтому, если вы планируете распространение программы, то отладочную информацию лучше не включать.

Учтите, что показанный выше пример включает отладочную информацию **только для кода, который был сгенерирован** при компиляции файла **hello.pp**. Это означает, что если вы используете несколько модулей (например, системный модуль), которые компилируются без поддержки отладчика, то отладочная информация не будет доступна для этих модулей.

Имеются два решения этой проблемы:

1. Перекомпилировать все модули вручную с опцией **–g**.
2. Указать опцию **–B** при компиляции с поддержкой отладчика. Это перекомпилирует все модули и вставит отладочную информацию для каждого модуля.

Второе решение может иметь нежелательные эффекты. Может оказаться, что некоторые модули будут не найдены, или при компиляции не будут выполнены соответствующие условия и т.п.

Это всё, что требуется сделать для того, чтобы исполняемый файл содержал необходимую отладочную информацию, которая будет использоваться **GNU gdb**.

10.2. Использование gdb для отладки вашей программы

Чтобы использовать программу **gdb** для отладки, вы должны запустить отладчик и передать в него в качестве параметра полное имя программы:

```
gdb hello
```

или для DOS:

```
gdb hello.exe
```

Это выполнит запуск отладчика, который сразу же загрузит вашу программу в память, но выполнение ваше программы пока не начнётся. Вместо этого вы увидите примерно следующее сообщение:

```
GNU gdb 6.6.50.20070726-cvs
Copyright (C) 2007 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "x86_64-suse-linux".
(gdb)
```

Реальное сообщение будет несколько отличаться, в зависимости от вашей операционной системы и версии отладчика.

Для запуска вашей программы вы можете использовать команду **run**. Вы можете указать дополнительные параметры командной строки, которые будут переданы в вашу программу, например:

```
(gdb) run -Опция -ДругаяОпция НеобходимыйПараметр
```

Если ваша программа запустится без проблем, то **gdb** проинформирует вас об этом и вернёт код выхода вашей программы. Если код выхода равен нулю, то отобразится сообщение «Program exited normally» (Программа завершена нормально).

Если что-то пошло не так (ошибка сегментации и т.п.), то **gdb** остановит выполнение вашей программы и проинформирует вас об этом соответствующим сообщением. Затем вы можете использовать другие команды **gdb**, чтобы посмотреть что случилось. Как альтернативный вариант, вы можете указать **gdb** остановиться на конкретном месте программы с помощью команды **break**.

Ниже перечислен краткий список команд **gdb**, которые вы можете использовать при работе:

Команда	Описание
quit	Выход из отладчика.
kill	Остановить запущенную программу.
help	Выводит справку по командам **gdb**.
file	Загружает новую программу в отладчик.
directory	Добавляет новый каталог в список путей для поиска исходных файлов. ПРИМЕЧАНИЕ: Моя копия **gdb** требует '.' для явного добавления пути поиска, иначе она не находит исходные файлы.
list	Список исходных кодов программ объёмом в 10 строк. В качестве опции вы можете указать номер строки или имя функции.
break	Устанавливает точку останова и указывает строку или функцию.
awatch	Устанавливает точку наблюдения за выражением. Точка наблюдения останавливает выполнение вашей программы, однако можно значение выражения прочитать или изменить.

В приложении E приведён простой пример файла инициализации для **gdb**. Там получен хороший результат при отладке программ Free Pascal. Подробно см. руководство пользователя для **gdb** или функцию **help** в **gdb**.

И текстовая версия IDE, и Lazarus, используют GDB как выходной буфер для отладки. Это более предпочтительно, так как они избавляют пользователя от ввода множества параметров и облегчают работу с отладчиком.

10.3. Пояснения по работе с gdb

Имеются некоторые особенности Free Pascal, о которых вы должны помнить при работе с **gdb**. Ниже перечислены основные из них:

1. Free Pascal генерирует информацию для GDB символами в верхнем регистре. Это результат того, что Паскаль – не чувствительный к регистру язык. Поэтому при обращении к переменным или функциям, вы должны печатать их имена в верхнем регистре. Например, если вы хотите наблюдать за значением переменной **count**, вы должны напечатать:

   ```
   watch COUNT
   ```

   Или если вы хотите остановить программу точно на какой-то функции (например, на **MyFunction**), то вы должны напечатать:

   ```
   break MYFUNCTION
   ```

2. **gdb** не понимает множества.
3. **gdb** не понимает строки. Строки представлены в **gdb** как записи с множеством полей или как массив символов, содержащий строку. Вы можете также использовать следующую функцию для печати строк:

   ```
   define pst
   set $pos=&$arg0
   set $strlen = {byte}$pos
   print {char}&$arg0.st@($strlen+1)
   end

   document pst
     Print out a Pascal string
   End
   ```

   Если вы вставите её в файл **gdb.ini**, вы сможете просмотреть строку с помощью этой функции. Это пример файла **gdb.ini** есть в приложении E.

4. Объекты трудно обрабатывать, потому что **gdb** ориентирована в первую очередь на C и C++. Обходным путём является представление методов объектов как функций с дополнительным параметром `this` (в нижнем регистре!). Имя такой функции – это объединение имени объекта и имени функции, разделённые двумя символами подчёркивания. Например, метод **TPoint.Draw** должен быть преобразован в **TPOINT__DRAW** и для остановки программы на этой функции нужно использовать:

   ```
   break TPOINT__DRAW
   ```

5. Глобальные перегружаемые функции вводят **gdb** в заблуждение, так как они имеют одинаковые имена. Поэтому вы не можете определить точку останова на перегружаемой функции, если только вы не знаете номер строки, в которой находится нужная вам функция. В этом случае вы можете установить точку останова на этой строке.

10.4. Поддержка для gprof, профайлера GNU

Вы можете компилировать ваши программы с поддержкой профайлера (подпрограмма протоколирования, позволяющая оценить время выполнения отдельных функций). Для этого вам требуется запускать компилятор с опцией **–pg**. Компилятор будет вставлять необходимую для профайлера информацию.

Затем вы можете запустить вашу программу обычным способом:

```
ВашаПрограмма
```

Где **ВашаПрограмма** – это ваш исполняемый файл.

Когда ваша программа завершит работу, то будет сгенерирован файл с именем **gmon.out**. Далее вы можете запустить профайлер для просмотра результата. Может оказаться полезным перенаправить результат в файл, так как информации может оказаться немало:

```
gprof yourexe > profile.log
```

Подсказка:
Вы можете использовать опцию **–flat** для уменьшения количества информации в выходном результате программы **gprof**. В этом случае в результате будет только информация о временных интервалах.

Подробную информацию о работе с профайлером **GNU gprof** см. в руководстве по использованию.

10.5 Обнаружение утечек памяти кучи

Free Pascal Имеет механизм для обнаружения утечек памяти кучи. Это встроенный модуль для управления памятью, который анализирует выделение/освобождение памяти и печатает отчёт об использовании памяти после выхода из программы.

Модуль, который выполняет эти функции, называется **heaptrc**. Если вы хотите использовать его, то вы должны подключить его первым модулем в разделе **uses** вашей программы. Альтернативный вариант – запускать компилятор с опцией **–gh**, и он подключит этот модуль автоматически.

После выполнения программы вы получите отчёт, подобный приведённому ниже:

```
Marked memory at 0040FA50 invalid
Wrong size : 128 allocated 64 freed
  0x00408708
  0x0040CB49
  0x0040C481
Call trace for block 0x0040FA50 size 128
  0x0040CB3D
  0x0040C481
```

Выходной результат работы модуля **heaptrc** можно настраивать с помощью нескольких переменных. Это можно делать также с помощью переменных среды окружения. Больше информации вы можете найти в документации по модулям.

10.6. Номера строк при отслеживании ошибок в реальном времени

Обычно, когда происходит ошибка во время выполнения программы, вы получаете список адресов, которые представляют отслеживание вызовов стека, то есть адреса всех процедур, которые были запущены в момент возникновения ошибки.

Этот список не очень информативный, поэтому существует модуль, который генерирует имена файлов и номера строк вызываемых процедур, использующих адреса стека. Этот модуль называется **lineinfo**.

Вы можете использовать этот модуль, указав опцию **–gl** при запуске компилятора. Модуль будет автоматически подключен. Также можно использовать этот модуль, явно указав его имя в разделе **uses**, но вы должны убедиться, что ваша программа компилируется с отладочной информацией.

Пример программы:

```
program testline;

procedure generateerror255;

  begin
    runerror(255);
  end;

procedure generateanerror;

  begin
    generateerror255;
  end;

begin
  generateanerror;
end.
```

Если откомпилировать эту программу с опцией **–gl**, то получим следующий выходной результат:

```
Runtime error 255 at 0x0040BDE5
  0x0040BDE5 GENERATEERROR255, line 6 of testline.pp
  0x0040BDF0 GENERATEANERROR, line 13 of testline.pp
  0x0040BE0C main, line 17 of testline.pp
  0x0040B7B1
```

Это более понятно, чем обычное сообщение. Убедитесь, что все модули вашей программы компилируются с отладочной информацией, иначе имя файла и номер строки могут быть не найдены.

10.7. Комбинирование heaptrc и lineinfo

Если вы комбинируете применение модулей **heaptrc** и **lineinfo**, то выходной результат модуля **heaptrc** будет содержать имена файлов и имена строк процедур, которые отслеживались в стеке. В таких случаях выходной результат будет примерно следующим:

```
Marked memory at 00410DA0 invalid
Wrong size : 128 allocated 64 freed
  0x004094B8
  0x0040D8F9 main, line 25 of heapex.pp
  0x0040D231
Call trace for block 0x00410DA0 size 128
  0x0040D8ED main, line 23 of heapex.pp
  0x0040D231
```

Если встречаются строки без имени файла или номера строки, то это означает, что в данный модуль не была включена отладочная информация.

ПРИЛОЖЕНИЕ A. Алфавитный список опций командной строки.

Ниже приведён список всех опций командной строки, как он генерируется компилятором:

```
Free Pascal Compiler version 2.5.1 [2010/07/11] for x86_64
Copyright (c) 1993-2009 by Florian Klaempfl
/usr/local/lib/fpc/2.5.1/ppcx64 [options] <inputfile> [options]
Поместите + после логического переключателя опции для включения опции, и - для
выключения опции
-a    Компилятор не удаляет сгенерированный файл ассемблера
      -al   Включать строки исходного кода в файл ассемблера
      -an   Включать примечания в файл ассемблера
      -ap   Использовать каналы вместо создания временных файлов ассемблера
      -ar   Включать информацию о выделении/освобождении регистров в файл
            ассемблера
      -at   Включать информацию о временном выделении/освобождении в файл
            ассемблера
-A<x> Выходной формат:
      -Adefault   Использовать ассемблер по умолчанию
      -Aas        Использовать ассемблер GNU AS
-b    Генерировать информацию обозревателя
      -bl   Генерировать информацию о локальных идентификаторах
-B    Включать в сборку все модули
-C<x> Опции генерации кода:
      -Ca<x>      Выбрать ABI, см. возможные значения для fpc -i
      -Cb         Генерировать код с обратным порядком байтов
      -Cc<x>      Установить по умолчанию соглашение о вызовах в <x>
      -CD         Создавать также динамическую библиотеку (не поддерживается)
      -Ce         Компиляция с эмуляцией плавающей точки
      -Cf<x>      Выбрать набор инструкций fpu для использования, возможные
                  значения см. для fpc -i
      -CF<x>      Точность минимальной константы с плавающей точкой (по умолчанию,
                  32, 64)
      -Cg         Генерировать код PIC
      -Ch<n>      Куча в <n> байтов (в пределах 1023...67107840)
      -Ci         Проверка ввода-вывода
      -Cn         Пропускать стадию компоновки
      -Co         Проверять переполнение целочисленных операций
      -CO         Проверять возможность переполнения для целочисленных операций
      -Cp<x>      Выбрать набор инструкций, см. значения для fpc -i
      -CP<x>=<y>  Настройки компоновки
            -CPPACKSET=<y>    Здесь <y> устанавливает распределение: 0, 1 или
                              DEFAULT или NORMAL, 2, 4 и 8
      -Cr         Проверять диапазон
      -CR         Проверять правильность вызова метода объекта
      -Cs<n>      Установить проверку размера <n> стека
      -Ct         Проверка стека (только для тестирования, см. документацию)
      -CX         Создавать библиотеку «умной компоновки»
-d<x> Определить идентификатор <x>
-D    Генерировать DEF-файл
      -Dd<x>      Установить дескриптор для <x>
      -Dv<x>      Установить версию DLL для <x>
-e<x> Установить путь для исполняемого файла
-E    То же, что и -Cn
-fPIC То же, что и -Cg
-F<x> Установиь имена файлов и пути:
      -Fa<x>[,y]  (для программы) загрузить модули <x> и [y] перед разбором
                  раздела uses
      -Fc<x>      Установить входную кодовую страницу для <x>
      -FC<x>      Установить RC бинарное имя компилятора для <x>
      -Fd         Отключить встроенный Кеш каталога компилятора
      -FD<x>      Установить каталог для поиска утилит компилятора
```

```
      -Fe<x>      Перенаправить выход ошибок в <x>
      -Ff<x>      Добавить <x> в структуру пути (только Darwin)
      -FE<x>      Установит путь для выхода exe/unit в <x>
      -Fi<x>      Добавить <x> в путь подключения
      -Fl<x>      Добавить <x> в путь библиотек
      -FL<x>      Использовать <x> как динамический компоновщик
      -Fm<x>      Загрузить таблицу преобразований Юникод из <x>.txt директорию
                  компилятора
      -Fo<x>      Добавить <x> в путь объектов
      -Fr<x>      Загрузить файл сообщений об ошибках <x>
      -FR<x>      Установить компоновщик ресурсов (.res) в <x>
      -Fu<x>      Добавить <x> в пут модулей
      -FU<x>      Установить выходной путь модулей в <x>, отменяет -FE
      -FW<x>      Записать полностью сгенерированную программу в <x>
      -Fw<x>      Загрузить ранее записанную программу из <x>
-g    Генерация отладочной информации (по умолчанию форматируется для целевой
      платформы)
      -gc   Генерировать проверку для указателей
      -gh   Использовать модуль heaptrace (для отладки памяти)
      -gl   Использовать модуль lineinfo (отображение информации)
      -go<x>      Установить опции отладочной информации
            -godwarfsets      Включить DWARF отладочную информацию для типа
                              «множества» (не работает с gdb < 6.5)
            -gostabsabsincludes     Записывать абсолютные/полные пути подключаемых
                                    файлов в Stabs
            -godwarfmethodclassprefix Префикс имён методов в DWARF с именем класса
      -gp   Сохранять регистр в именах идентификаторов
      -gs   Генерировать отладочную информацию Stabs
      -gt   Отбрасывать локальные переменные
      -gv   Генерировать программы с отслеживанием Valgrind
      -gw   Генерировать отладочную информацию DWARFv2 (тоже, что и -gw2)
      -gw2  Генерировать отладочную информацию DWARFv2
      -gw3  Генерировать отладочную информацию DWARFv3
-i    Информация
      -iD   Вернуть дату компилятора
      -iV   Вернуть короткую версию компилятора
      -iW   Вернуть полную версию компилятора
      -iSO  Вернуть операционную систему компилятора
      -iSP  Вернуть процессор компилятора
      -iTO  Вернуть целевую ОС
      -iTP  Вернуть целевой процессор
-I<x>       Добавить <x> для пути подключения
-k<x>       Поместить <x> в компоновщик
-l    Записать логотип
-M<x> Установить режим языка для <x>
      -Mfpc       Диалект Free Pascal (по умолчанию)
      -Mobjfpc    FPC-режим с поддержкой Object Pascal
      -Mdelphi    Режим совместимости с Delphi 7
      -Mtp        Режим совместимости с TP/BP 7.0
      -Mmacpas    Режим совместимости с Macintosh Pascal
-n    Не читать конфигурационный файл по умолчанию
-N<x> Узловая иерархическая оптимизация
      -Nu   Разворачивать циклы
-o<x> Изменить имя создаваемого исполняемого файла на <x>
-O<x> Оптимизации:
      -O-   Отключить оптимизации
      -O1   Уровень оптимизации 1 (быстрый и дружественный к отладчику)
      -O2   Уровень оптимизации 2 (-O1 + быстрая оптимизация)
      -O3   Уровень оптимизации 3 (-O2 + медленные оптимизации)
      -Oa<x>=<y> Установить выравнивание
      -Oo[NO]<x> Включить или отключить оптимизации, см. значения в fpc -i
      -Op<x> Установить целевой процессор для оптимизации, см. значения в fpc -i
      -OW<x> Генерировать обратную связь для оптимизации <x>
      -Ow<x> Выполнить программу с оптимизацией <x>, см. значения в fpc -i
```

```
      -Os   Оптимизировать сначала размер, затем скорость
-pg   Генерировать код для профайлера gprof
-R<x> Стиль чтения ассемблера:
      -Rdefault   Использовать ассемблер по умолчанию для целевого процессора
-S<x> Опции синтаксиса:
      -S2   То же, что и -Mobjfpc
      -Sc   Поддерживать операторы, подобные C (*=,+=,/= и -=)
      -Sa   Включить формальные утверждения
      -Sd   То же, что и -Mdelphi
      -Se<x> Параметры ошибок. <x> - комбинация из следующих:
            <n>   : Останавливать компилятор после <n> ошибок (по умолчанию 1)
            w     : Останавливать компилятор также после предупреждений
            n     : Останавливать компилятор после замечаний
            h     : Останавливать компилятор после подсказок
      -Sg   Включить LABEL и GOTO (по умолчанию в -Mtp и -Mdelphi)
      -Sh   Использовать по умолчанию ansistrings вместо shortstrings
      -Si   Включить inlining процедур и функций, объявленных как "inline"
      -Sk   Загружать модуль fpcylix
      -SI<x> Установить стиль интерфейса в <x>
            -SIcom COM-совместимый интерфейс (по умолчанию)
            -SIcorba CORBA-совместимый интерфейс
      -Sm   Поддерживать макросы, подобные C (global)
      -So   То же, что и -Mtp
      -Ss   Имя конструктора должно быть инициализировано (деструктор должен быть
            выполнен)
      -Sx   Включить ключевые слова исключений (по умолчанию в режимах
            Delphi/ObjFPC)
-s    Не вызывать ассемблер и компоновщик
      -sh   Генерировать сценарий для компоновщика для данной системы
      -st   Генерировать сценарий для компоновщика для целевой системы
      -sr   Пропускать фазу распределения регистров (используется с -alr)
-T<x> Целевая операционная система:
      -Tlinux Linux
-u<x> Неопределённый символ <x>
-U    Параметры модулей:
      -Un   Не проверять совпадения имени модуля с именем файла
      -Ur   Генерировать выходные файлы модулей (никогда автоматически не
            перекомпилировать)
      -Us   Компилировать системный модуль
-v<x> Подробности. <x>  - комбинация из следующих букв:
      e : Показывать ошибки (по умолчанию)
      0 : Не показывать ничего (кроме ошибок)
      w : Показывать предупреждения
      u : Показывать информацию модулей
      n : Показывать примечания
      t : Показывать проверенные/используемые файлы
      h : Показывать подсказки
      c : Показывать условия
      i : Показывать основную информацию
      d : Показывать отладочную информацию
      l : Показывать номера строк
      r : Режим совместимости Rhide/GCC
      s : Показывать временные отметки
      q : Показывать номера сообщений
      a : Показывать всё
      x : Информацию исполняемого файла (только Win32)
      b : Записывать в файл имена сообщений
      p : Записывать дерево анализа в tree.log с полными путями
      v : Записывать в fpcdebug.txt отладочную информацию
      m<x>,<y> : Не показывать сообщения с номером <x> и <y>
-We Использовать внешние ресурсы (Darwin)
```

```
-X Параметры исполняемого файла:
      -Xc   Передавать --shared/-dynamic для компоновщика (BeOS, Darwin, FreeBSD,
            Linux)
      -Xd   Не использовать стандартный путь для поиска библиотек (необходимо для
            кросс-платформенности)
      -Xe   Использовать внешний компоновщик
      -Xg   Создавать отладочную информацию в отдельном файле и добавлять раздел
            со ссылками на неё в исполняемый файл
      -XD   Пытаться компоновать модули динамически (определяется
            FPC_LINK_DYNAMIC)
      -Xi   Использовать внутренний компоновщик
      -Xm   Генерировать карту ссылок
      -XM<x> Установить имя главной процедуры программы 'main' (по умолчанию
            'main')
      -XP<x> Присоединять спереди имён префикс <x>
      -Xr<x> Установить путь компоновщика в <x> (необходимо для кросс-
            платформенной компиляции, см. t)
      -XR<x> Присоединять спереди <x> для всех путей поиска компоновщика (BeOS,
            Darwin, FreeBSD, Linux)
      -Xs   Обрезать все идентификаторы для исполняемого файла
      -XS   Пытаться компоновать модули статически (по умолчанию, определяется
            FPC_LINK_STATIC)
      -Xt   Компоновать статические библиотеки (-static передаётся в компоновщик)
      -XX   Пытаться выполнять «умную компоновку» модулей (определяется
            FPC_LINK_SMART)
-?    Отобразить эту справку
-h    Отобразить эту справку без ожидания
```

Если вы не согласны с переводом, то можете посмотреть этот список непосредственно при работе с компилятором))).

ПРИЛОЖЕНИЕ B. Алфавитный список зарезервированных слов.

absolute
abstract
and
array
as
asm
assembler
begin
break
case
cdecl
class
const
constructor
continue
cppclass
deprecated
destructor
div
do
downto
else
end
except
exit
export
exports
external
experimental
fail
false
far
file
finally
for
forward
function
goto
if
implementation
in
index
inherited
initialization
inline
interface
interrupt
is
label
library
mod
name
near
nil
not
object
of
on
operator
or
otherwise
packed
popstack
private
procedure
program
property
protected
public
raise
record
reintroduce
repeat
self
set
shl
shr
stdcall
string
then
to
true
try
type
unimplemented
unit
until
uses
var
virtual
while
with
xor

ПРИЛОЖЕНИЕ C. Сообщения компилятора.

Это приложение содержит список всех сообщений компилятора. Этот список сообщений сгенерирован из исходных кодов самого компилятора, и должен быть достаточно полным. Только ошибки ассемблера не включены в этот список.

Пояснения по вопросам управления сообщениями можно найти в разделе «5.1.2. Параметры обратной связи».

C1. Основные сообщения компилятора

В этом разделе описаны сообщения, которые не связаны с фатальными ошибками, но несут полезную информацию. Количество (полноту) этих сообщений можно определить различными опциями установки подробностей с помощью переключателя **–v**.

Сообщение	Описание
Compiler: Сообщение	Если используется опция **–vt**, то эта строка говорит вам, что компилятор используется.
Compiler OS: Сообщение	Если используется опция **–vd**, то в этой строке выводится исходная операционная система.
Info: Target OS: Сообщение	Если используется опция **–vd**, то в этой строке выводится целевая операционная система.
Using executable path: Сообщение	Если используется опция **–vt**, то в этой строке выводится, где компилятор ищет свои бинарные файлы.
Using unit path: Сообщение	Если используется опция **–vt**, то в этой строке выводится, где компилятор ищет откомпилированные модули. Вы можете установить этот путь опцией **–Fu**.
Using include path: Сообщение	Если используется опция **–vt**, то в этой строке выводится, где компилятор ищет подключаемые файлы (файлы, указанные в {$I xxx}). Вы можете установить этот путь опцией **–Fi**.
Using library path: Сообщение	Если используется опция **–vt**, то в этой строке выводится, где компилятор ищет библиотеки. Вы можете установить этот путь опцией **–Fl**.
Using object path: Сообщение	Если используется опция **–vt**, то в этой строке выводится, где компилятор ищет объектные файлы (файлы, указанные в {$L xxx}).. Вы можете установить этот путь опцией **–Fo**.
Info: Сообщение1 lines compiled, Сообщение2 sec Сообщение3	Если используется опция **–vi**, то в этой строке выводится отчёт о количестве откомпилированных строк (Сообщение1) и времени, потраченном на компиляцию (реальное время, не программное время).
Fatal: No memory left	Компилятору не хватает памяти для компиляции вашей программы. Имеется несколько способов решения этой проблемы: • Если вы используете опцию «сборки», то попытайтесь откомпилировать некоторые модули вручную. • Если вы компилируете большую программу, то разделите её на модули и компилируйте их отдельно. • Если первые два способа не помогли, то перекомпилируйте компилятор с большей кучей (вы можете использовать опцию **–Ch** для этого, см. раздел «5.1.4. Параметры, контролирующие результат компиляции»).
Info: Writing Resource String Table file: Сообщение	Это сообщение отображается, сели компилятор записывает файл Resource String Table, содержащий все строковые ресурсы для программы.
Error: Writing Resource String Table file: Сообщение	Это сообщение отображается, если компилятор обнаружил ошибку при записи файла Resource String Table.
Info: Fatal:	Префикс для фатальной ошибки.
Info: Error:	Префикс для ошибок.
Info: Warning:	Префикс для предупреждений.
Info: Note:	Префикс для примечаний.
Info: Hint:	Префикс для подсказок.

Сообщение	Описание
Error: Path "Сообщение" does not exist	Путь, который выводится в Сообщении, не существует.
Fatal: Compilation aborted	Компиляция была прервана.
bytes code	Размер сгенерированного исполняемого кода, в байтах.
bytes data	Размер сгенерированных данных программы, в байтах.
Info: Сообщение warning(s) issued	Общее количество предупреждений, выданных во время компиляции.
Info: Сообщение hint(s) issued	Общее количество подсказок, выданных во время компиляции.
Info: Сообщение note(s) issued	Общее количество замечаний, выданных во время компиляции.

C2. Сообщения сканера

В этом разделе описаны сообщения, которые генерируются сканером. Сканер выполняет работу по проверке лексической структуры файлов Паскаля, например, пытается найти зарезервированные слова, строки и т.п. Он также обрабатывает директивы и условия компиляции.

Сообщение	Описание
Fatal: Unexpected end of file	Это сообщение обычно появляется в одном из следующих случаев: • Исходный файл закончился, но оператор **end.** не был найден. • Подключаемый файл закончился на середине. • Комментарий не был закрыт
Fatal: String exceeds line	Это означает, что строка не была закрыта кавычкой ‘, и поэтому заняла несколько строк исходного кода.
Fatal: illegal character "Сообщение1" (Сообщение2)	Неправильный символ был обнаружен во входном файле.
Fatal: Syntax error, "Символ1" expected but "Символ2" found	Это означает, что компилятор ожидал появления Символа1, но обнаружил Символ2. Это может случиться практически в любом месте (ошибка в языке Паскаль).
Start reading includefile Сообщение1	Вы используете опцию **–vt** и компилятор говорит вам, что началось чтение подключаемого файла.
Warning: Comment level Сообщение found	Если используется опция **–vw**, то компилятор предупреждает вас, если находит вложенные комментарии. Вложенные комментарии не допускаются в Турбо Паскаль и в Делфи, и могут вызвать ошибку в исходном коде.
Note: Ignored compiler switch "Сообщение1"	Если используется опция **–vn**, то компилятор предупреждает вас, если игнорирует этот переключатель.
Warning: Illegal compiler switch "Сообщение1"	Вы подключили директиву компилятора (например, {$...}), которую он не может распознать.
Warning: Misplaced global compiler switch	Опция компилятора неуместна, и должна быть размещена в начале модуля или программы.
Error: Illegal char constant	Это сообщение появится, если вы укажете символ с помощью ASCII-кода, например, #96, но число будет слишком большое и лежать за пределами таблицы ASCII.
Fatal: Can’t open file "Сообщение1"	Free Pascal не может найти исходный файл программы или модуля, указанный вами в командной строке.
Fatal: Can’t open include file "Сообщение1"	Free Pascal не может найти исходный файл, указанный вами в скобках {$include..}.
Error: Illegal record alignment specifier "Сообщение1"	Вы указали {$PACKRECORDS n} или {$ALIGN n} с неправильным значением n. Для $PACKRECORDS правильные значения – это 1, 2, 4, 8, 16, 32, C, NORMAL, DEFAULT, а для $ALIGN – это значения 1, 2, 4, 8, 16, 32, ON, OFF. В режиме MacPas $ALIGN также поддерживает MAC68K, POWER и RESET.
Error: Illegal enum minimum-size specifier "arg1"	Вы указали {$PACKENUM n} с неправильным значением n. Только 1,2,4, NORMAL или DEFAULT являются правильными.
Error: $ENDIF expected for Сообщ1 Сообщ2 defined in Сообщ3 line Сообщ4	Ваши операторы условной компиляции не сбалансированы.

Сообщение	Описание
Error: Syntax error while parsing a conditional compiling expression	Это ошибка в выражении следующих директив компилятора: {$if ..}, {$ifc } или {$setc }.
Error: Evaluating a conditional compiling expression	Это ошибка в выражении следующих директив компилятора: {$if ..}, ifcorsetc.
Warning: Macro contents are limited to 255 characters in length	Содержимое макроса не может быть более 255 символов.
Error: ENDIF without IF(N)DEF	Ваши операторы {$IFDEF ..} and {$ENDIF} не сбалансированы.
Fatal: User defined: Сообщение1	Произошла ошибка, определённая пользователем. См. также «Руководство программиста».
Error: User defined: Сообщение1	Произошла ошибка, определённая пользователем. См. также «Руководство программиста».
Warning: User defined: Сообщение1	Предупреждение, определённое пользователем. См. также «Руководство программиста».
Note: User defined: Сообщение1	Замечание, определённое пользователем. См. также «Руководство программиста».
Hint: User defined: Сообщение1	Подсказка, определённая пользователем. См. также «Руководство программиста».
Info: User defined: Сообщение1	Информация, определённая пользователем. См. также «Руководство программиста».
Error: Keyword redefined as macro has no effect	Вы не можете переопределить ключевые слова в макросе.
Fatal: Macro buffer overflow while reading or expanding a macro	Ваш макрос или результат его работы слишком большие для компилятора.
Warning: Expanding of macros exceeds a depth of 16.	При извлечении уровень вложенности вашего макроса превысил 16. Компилятор не будет дальше разворачивать макрос, потому что это может означать, что используется рекурсия.
Warning: compiler switches aren't supported in // styled comments	Переключатели компилятора установлены нормальном стиле комментариев Паскаль, и не поддерживают стиль C++ (//).
Handling switch "Сообщение1"	Если вы включили отладочную информацию (-vd), то компилятор сообщает вам, если он оценивает условия операторов компиляции.
ENDIF Сообщение1 found	Если вы включили условные сообщения (-vc), то компилятор выдаёт сообщение, если находит условные операторы.
IFDEF Сообщение1 found, arg2	Если вы включили условные сообщения (-vc), то компилятор выдаёт сообщение, если находит условные операторы.
IFOPT Сообщение1 found, Сообщение2	Если вы включили условные сообщения (-vc), то компилятор выдаёт сообщение, если находит условные операторы.
IF Сообщение1 found, Сообщение2	Если вы включили условные сообщения (-vc), то компилятор выдаёт сообщение, если находит условные операторы.
IFNDEF Сообщение1 found, Сообщение2	Если вы включили условные сообщения (-vc), то компилятор выдаёт сообщение, если находит условные операторы.
ELSE Сообщение1 found, Сообщение2	Если вы включили условные сообщения (-vc), то компилятор выдаёт сообщение, если находит условные операторы.
Skipping until...	Если вы включили условные сообщения (-vc), то компилятор выдаёт сообщение, если находит условные операторы, и какие части он пропускает или компилирует.
Info: Press <return> to continue	Если используется опция –vi, то компилятор останавливает компиляцию, если обнаруживает директиву {$STOP} и ожидает, пока не будет нажата клавиша ENTER.
Warning: Unsupported switch "Сообщение"	Если включены предупреждения (-vw), то компилятор предупреждает вас о неподдерживаемых переключателях. Это означает, что данные переключатели используются в Делфи или Турбо Паскаль, но не используются в Free Pascal.
Warning: Illegal compiler directive "Сообщение"	Если включены предупреждения (-vw), то компилятор предупреждает вас о нераспознанных переключателях. Список распознаваемых переключателей см. в руководстве программиста.
Back in Сообщение	Если вы используете переключатель –vt, то компилятор говорит вам, когда он заканчивает читать подключаемый файл.
Warning: Unsupported application type: "Сообщение"	Вы получите это предупреждение, если вы указали неизвестный тип приложения с директивой {$APPTYPE}.
Warning: APPTYPE is not supported by the target OS	Директива {$APPTYPE} поддерживается только определёнными операционными системами.

Сообщение	Описание
Warning: DESCRIPTION is not supported by the target OS	Директива {$DESCRIPTION} не поддерживается целевой операционной системой.
Note: VERSION is not supported by target OS	Директива {$VERSION} не поддерживается целевой операционной системой.
Note: VERSION only for exes or DLLs	Директива {$VERSION} используется только для исполняемых файлов или исходных кодов DLL.
Warning: Wrong format for VERSION directive "Сообщение"	Неправильный формат директивы {$VERSION}. Директива должна иметь формат ПолнаяВерсия.СокращённаяВерсия, где ПолнаяВерсия и СокращённаяВерсия – это слова.
Error: Illegal assembler style specified "Сообщение"	Если вы указали режим ассемблера директивой {$ASMMODE xxx}, а компилятор не распознал указанный вами режим.
Warning: ASM reader switch is not possible inside asm statement, "Сообщение" will be effective only for next	Невозможно переключиться с одного режима ассемблера на другой в пределах одного ассемблерного блока. Новый режим ассемблера будет использоваться только для следующего ассемблерного блока.
Error: Wrong switch toggle, use ON/OFF or +/-	Вы должны использовать ON или OFF или + или – для переключателя.
Error: Resource files are not supported for this target	Целевая платформа, для которой выполняется компиляция, не поддерживает файлы ресурсов.
Warning: Include environment "Сообщение" not found in environment	Подключаемая переменная окружения не была найдена в среде окружения. Она будет заменена пустой строкой.
Error: Illegal value for FPU register limit	Правильные значения для этой директивы 0…8 и NORMAL/DEFAULT.
Warning: Only one resource file is supported for this target	Целевая платформа, для которой выполняется компиляция, поддерживает только один файл ресурсов. Первый найденный файл ресурсов будет использоваться, остальные игнорируются.
Warning: Macro support has been turned off	Объявление макроса было найдено, но поддержка макросов отключена, поэтому макрос будет игнорироваться. Включение поддержки макросов при компиляции выполняется опцией –Sm в командной строке или директивой {$MACRO ON} в исходном коде.
Error: Illegal interface type specified. Valids are COM, CORBA or DEFAULT.	Указанный тип интерфейса не поддерживается.
Warning: APPID is only supported for PalmOS	Директива {$APPID} поддерживается только для целевой платформы PalmOS.
Warning: APPNAME is only supported for PalmOS	Директива {$APPNAME} поддерживается только для целевой платформы PalmOS.
Error: Constant strings can't be longer than 255 chars	Одна строковая константа должна содержать не более 255 символов. Попытайтесь разделить строку на несколько более мелких частей, а затем соединить их с помощью оператора +.
Fatal: Including include files exceeds a depth of 16.	Если используются подключаемые файлы, то уровень вложенности должен быть не более 16. Компилятор не будет дальше разворачивать файлы, потому что это может означать, что используется рекурсия.
Fatal: Too many levels of PUSH	Максимально допускается 20 уровней при использовании инструкции PUSH. Эта ошибка возможна только в режиме MacPas.
Error: A POP without a preceding PUSH	Эта ошибка возможна только в режиме MacPas.
Error: Macro or compile time variable "Сообщение" does not have any value	Выражение условной компиляции нельзя вычислить.
Error: Wrong switch toggle, use ON/OFF/DEFAULT or +/-/*	Вы должны использовать ON или OFF или DEFAULT или + или – или * для данного переключателя.
Error: Mode switch "Сообщение" not allowed here	Переключатель режима уже был использован, или, в случае с опцией –Mmacpas, переключатель режима обнаружен после UNIT.
Error: Compile time variable or macro "Сообщение" is not defined.	Выражение условной компиляции нельзя вычислить. Только в режиме MacPas.
Error: UTF-8 code greater than 65535 found	Free Pascal обрабатывает строки UTF-8 как widestrings, то есть код символа ограничен 65535.
Error: Malformed UTF-8 string	Данная строка не является правильной строкой UTF-8.
UTF-8 signature found, using UTF-8 encoding	Компилятор нашёл закодированную сигнатуру UTF-8 ($ef, $bb, $bf) в начале файла, поэтому будет интерпретировать его как файл UTF-8.
Error: Compile time expression: Wanted Сообщ1 but got Сообщ2 at Сообщ3	Во время компиляции обнаружено неправильное выражение при проверке типов.

Сообщение	Описание
Note: APPTYPE is not supported by the target OS	Директива {$APPTYPE} поддерживается только определённой операционной системой.
Error: Illegal optimization specified "Сообщение"	Вы указали оптимизацию директивой {$OPTIMIZATION xxx}, но компилятор не распознаёт указанную вами оптимизацию.
Warning: SETPEFLAGS is not supported by the target OS	Директива {$SETPEFLAGS} не поддерживается целевой операционной системой.
Warning: IMAGEBASE is not supported by the target OS	Директива {$IMAGEBASE} не поддерживается целевой операционной системой.
Warning: MINSTACKSIZE is not supported by the target OS	Директива {$ MINSTACKSIZE} не поддерживается целевой операционной системой.
Warning: MAXSTACKSIZE is not supported by the target OS	Директива {$ MAXSTACKSIZE} не поддерживается целевой операционной системой.
Error: Illegal state for $WARN directive	Только ON и OFF можно использовать как значения с директивой компилятора {$WARN}.
Error: Illegal set packing value	Только 0, 1, 2, 4, 8, DEFAULT и NORMAL распознаются как параметры упаковки.
Warning: PIC directive or switch ignored	Некоторые целевые ОС, такие как WINDOWS, не поддерживают и не нуждаются в PIC, поэтому директива PIC игнорируется.
Warning: The switch "Сообщение" is not supported by the currently selected target	Некоторые переключатели компилятора, такие как $E, не поддерживаются на всех целевых ОС.
Warning: Framework-related options are only supported for Darwin/Mac OS X	Фреймворк является неизвестной концепцией, или не поддерживается FPC, или операционная система отличается от Darwin/Mac OS X.
Error: Illegal minimal floating point constant precision "Сообщение"	Правильная минимальная точность для плавающей точки может быть константой по умолчанию, 32 и 64, что означает соответственно минимум (обычно 32 бит), 32 бита и 64 бита точности.
Warning: Overriding name of "main" procedure multiple times, was previously set to "Сообщение"	Для главной процедуры указано более одного имени. Только последнее имя будет использоваться.

C3. Сообщения синтаксического анализатора

В этом разделе описаны сообщения синтаксического анализатора. Синтаксический анализатор выполняет работу по проверке семантики вашего языка, то есть проверяет правильность языковых конструкций Паскаля.

Сообщение	Описание
Error: Parser - Syntax Error	Ошибка Турбо Паскаль была обнаружена. Это обычно происходит, если неправильный символ найден в исходном файле.
Error: INTERRUPT procedure can't be nested	Процедура INTERRUPT должна быть глобальной.
Warning: Procedure type "Сообщение" ignored	Указанная процедурная директива игнорируется программами FPC.
Error: Not all declarations of "Сообщение" are declared with OVERLOAD	Если вы хотите использовать директиву OVERLOAD, то все объявления должны объявляться с OVERLOAD.
Error: Duplicate exported function name "Сообщение"	Экспортируемые имена функций в указанной DLL не должны совпадать.
Error: Invalid index for exported function	Индекс функции DLL должен быть в диапазоне 1..$FFFF.
Warning: Relocatable DLL or executable arg1 debug info does not work, disabled	В данный момент невозможно включить отладочную информацию в перемещаемую DLL.
Warning: To allow debugging for win32 code you need to disable relocation with -WN option	Попытка включить отладочную информацию в перемещаемую DLL или EXE-файл. Используёте –WN, если хотите отлаживать исполняемые файлы Win32.
Error: Constructor name must be INIT	Вы объявили конструктор объекта, имя которого не инициализировано, а переключатель -Ss установлен. См. опцию -Ss (раздел «5.1.5. Параметры для исходных кодов (опции языка)»).
Error: Destructor name must be DONE	Вы объявили деструктор объекта, имя которого не завершено, а переключатель -Ss установлен. См. опцию -Ss (раздел «5.1.5. Параметры для исходных кодов (опции языка)»).

Сообщение	Описание
Error: Procedure type INLINE not supported	Вы пытаетесь компилировать программу с стилем C++, но забыли указать опцию -Si (-Si, см. раздел «5.1.5. Параметры для исходных кодов (опции языка)»). По умолчанию компилятор не поддерживает стиль C++.
Warning: Constructor should be public	Конструктор должен быть в разделе **public** объявления объекта (класса).
Warning: Destructor should be public	Деструктор должен быть в разделе **public** объявления объекта (класса).
Note: Class should have one destructor only	Вы должны объявлять только один деструктор для класса.
Error: Local class definitions are not allowed	Классы должны быть объявлены глобально. Они не могут быть объявлены внутри процедуры или функции.

Fatal: Anonymous class definitions are not allowed	Было обнаружено неправильное объявление объекта (класса), например, объект или класс без методов, не являются рпоизводными от другого объекта. Пример:
	
	`Type o = object`
	`  a : longint;`
	`end;`
	
	вызовет ошибку.

Note: The object "Сообщение" has no VMT	Это замечание означает, что объявленный объект не имеет таблицы виртуальных методов.
Error: Illegal parameter list	Вы вызываете функцию с параметрами, которые отличаются от параметров, указанных при объявлении функции.
Error: Wrong number of parameters specified for call to "Сообщение"	Это ошибка в списке параметров функции или процедуры – неправильное количество параметров.
Error: overloaded identifier "Сообщение" isn't a function	Компилятор обнаружил идентификатор с именем, которое совпадает с именем перегружаемой функции, но этот идентификатор не является функцией, которая может быть перегружаемой.
Error: overloaded functions have the same parameter list	Вы объявили перегружаемые функции, но с одинаковым списком параметров. Перегружаемая функция должна иметь не менее 1 отличия в параметрах от объявленной функции.
Error: function header doesn't match the previous declaration "Сообщение"	Вы объявили функцию с одинаковыми параметрами, но тип результата отличается.
Error: function header "Сообщ1" doesn't match forward : var name changes Сообщ2 => Сообщ3	Вы объявили функцию в интерфейсной части, или с директивой forward, но с различным списком параметров.

Note: Values in enumeration types have to be ascending	Free Pascal допускает перечисления как в C. Проверьте следующие два объявления:
	
	`type a = (A_A,A_B,A_E:=6,A_UAS:=200);`
	`type a = (A_A,A_B,A_E:=6,A_UAS:=4);`
	
	Второе объявление вызовет ошибку, потому что значение A_UAS должно быть больше, чем A_E, то есть не меньше 7.

Error: With cannot be used for variables in a different segment	Локальная переменная записывается в стек, но это невозможно, если переменная принадлежит другому сегменту.
Error: function nesting > 31	Вы можете объявлять вложенные функции только до 31 уровня в глубину.
Error: range check error while evaluating constants	Константы находятся за пределами допустимого диапазона.
Warning: range check error while evaluating constants	Константы находятся за пределами допустимого диапазона.
Error: duplicate case label	Вы указали одну и ту же метку 2 раза в конструкции **case**.
Error: Upper bound of case range is less than lower bound	Верхняя граница метки в конструкции **case** меньше, чем нижняя граница, а это неправильно.
Error: typed constants of classes or interfaces are not allowed	Вы не можете объявить константу типа класса или объекта.
Error: functions variables of overloaded functions are not allowed	Вы пытаетесь назначить перегружаемую функцию процедурной переменной. Это не допускается.
Error: string length must be a value from 1 to 255	Длина shortstring в Паскале ограничена 255 символами. Вы пытаетесь объявить строку длиной меньше 1 или больше 255 символов.

Сообщение	Описание
Warning: use extended syntax of NEW and DISPOSE for instances of objects	Если вы имеете указатель на объектный тип, то оператор **new(a)** не будет инициализировать объект (то есть конструктор не вызывается), даже если имеется свободное место для выделения памяти. Вы должны использовать оператор **new(a,init)**. Он выделит память и вызовет конструктор объекта.
Warning: use of NEW or DISPOSE for untyped pointers is meaningless	Использование NEW или DISPOSE для нетипизированных указателей не имеет смысла.
Error: use of NEW or DISPOSE is not possible for untyped pointers	Вы не можете использовать **new(p)** или **dispose(p)**, если **p** – это нетипизированный указатель, потому что для него нельзя определить размер. Это принято для совместимости с режимами TP и DELPHI, но компилятор предупредит вас, если найдёт такую конструкцию.
Error: class identifier expected	Это сообщение появляется, если компилятор обнаружил объявление процедуры, которая содержит точку, то есть метод объекта или класса, но тип перед точкой не известен компилятору.
Error: type identifier not allowed here	Вы не можете использовать тип внутри выражения.
Error: method identifier expected	Идентификатор не является методом. Это сообщение появляется, если компилятор обнаружил объявление процедуры, которая содержит точку, то есть метод объекта или класса, но имя процедуры не является процедурой данного типа.
Error: function header doesn't match any method of this class "Сообщение"	Идентификатор не является методом. Это сообщение появляется, если компилятор обнаружил объявление процедуры, которая содержит точку, то есть метод объекта или класса, но имя процедуры не является процедурой данного типа.
procedure/function Сообщение	Если используется переключатель **-vd**, то компилятор сообщает вам, когда он начинает выполнять процедуру или функцию.
Error: Illegal floating point constant	Компилятор ожидал увидеть выражение с плавающей точкой, но получил нечто другое.
Error: FAIL can be used in constructors only	Вы использовали ключевое слово **fail** вне метода конструктора.
Error: Destructors can't have parameters	Вы объявили деструктор со списком параметров. Методы деструктора не могут иметь параметры.

Error: Only class methods can be referred with class references	Эта ошибка возникает в ситуации, подобной следующей:
	
	`Type :`
	`Tclass = Class of Tobject;`
	`Var C : TClass;`
	`begin`
	`...`
	`C.free`
	
	Free – это не метод класса и, следовательно, не может быть вызван со ссылкой на класс.

Error: Only class methods can be accessed in class methods	Это связано похоже на предыдущую ошибку. Вы не можете вызвать метод объекта из внутреннего метода класса. Следующий код вызовет такую ошибку:
	
	`class procedure tobject.x;`
	`begin`
	`free`
	
	Так как **free** – это нормальный метод класса, он не может быть вызван из метода класса.

Error: Constant and CASE types do not match	Одна из меток конструкции CASE имеет тип, отличный от типа переменной переключателя CASE.
Error: The symbol can't be exported from a library	Вы можете экспортировать процедуры и функции, только когда вы пишите библиотеку. Вы не можете экспортировать переменные и константы.
Warning: An inherited method is hidden by "Сообщение"	Метод, который объявлен как **virtual** в классе-предке, должен быть перезаписан в классе-потомке с директивой **override**. Если вы не укажете директиву **override**, вы скроете метод предка.
Error: There is no method in an ancestor class to be overridden: "Сообщение"	Вы пытаетесь переписать директивой **override** виртуальный метод класса-предка, который не существует.
Error: No member is provided to access property	Вы не указали директиву **read** для свойства.

Сообщение	Описание
Warning: Stored property directive is not yet implemented	Это сообщение больше не используется, так как записанная директива была выполнена.

Error: Illegal symbol for property access	Это ошибка директив read или write для свойства типа массива. Если вы объявляете свойство-массив, вы можете получить доступ к нему только с помощью процедуры или функции. Следующий код вызовет такую ошибку:
	
	`tmyobject = class`
	`i : integer;`
	`property x [i : integer]: integer read I write i;`

Error: Cannot access a protected field of an object here	Поля, которые объявлены в разделе **protected** объекта или класса, не могут быть доступны из другого модуля.
Error: Cannot access a private field of an object here	Поля, которые объявлены в разделе **private** объекта или класса, не могут быть доступны из другого модуля.
Error: Overridden methods must have the same return type: "Сообщение2" is overriden by "Сообщение1" which has another return	Если вы объявили перегружаемые методы в определении класса, они должны иметь одинаковый тип возвращаемого результата.
Error: EXPORT declared functions can't be nested	Вы не можете объявить функцию или процедуру внутри функции или процедуры, которая была объявлена как экспортная процедура.
Error: Methods can't be EXPORTed	Вы не можете объявить процедуру, которая является методом для экспортируемого объекта.
Error: Call by var for arg no. Сообщ1 has to match exactly: Got "Сообщ2" expected "Сообщ3"	Если вызывающая функция объявлена с параметрами **var**, то переменные в вызываемой функции должны иметь точно такой же тип. Они не преобразуются автоматически.
Error: Class isn't a parent class of the current class	Когда вызываются наследуемые методы, вы пытаетесь вызвать метод не связанный с классом. Вы можете только вызвать метод класса-предка.
Error: SELF is only allowed in methods	Вы пытаетесь использовать параметр **self** вне метода объекта. Только методы могут получать параметры **self**.
Error: Methods can be only in other methods called direct with type identifier of the class	Конструкция, подобная следующей: **НекийТип.НекийМетод** допускается только для методов.
Error: Illegal use of ':'	Вы использовали формат : (двоеточие) 2 раза в выражении, которое не предусматривает такой формат.
Error: range check error in set constructor or duplicate set element	Объявление множества содержит ошибку. Какой-то элемент находится вне диапазона или имеется два одинаковых элемента.
Error: Pointer to object expected	Вы указали неправильный тип в операторе **new**. Расширенный синтаксис **new** требует параметра типа объект.
Error: Expression must be constructor call	Когда используете расширенный синтаксис **new**, вы должны указать метод конструктора объекта, когда пытаетесь его создать. Процедура, которую вы указали, не является конструктором.
Error: Expression must be destructor call	Когда используете расширенный синтаксис **dispose**, вы должны указать метод деструктора объекта, когда пытаетесь его разрушить. Процедура, которую вы указали, не является деструктором.
Error: Illegal order of record elements	При объявлении записи-константы вы указали поля в неправильном порядке.
Error: Expression type must be class or record type	Для оператора **with** необходим аргумент, который имеет тип записи или класса. Вы используете выражение, которое не относится ни к одному из данных типов.
Error: Procedures can't return a value	В Free Pascal вы можете указать возвращаемое значение для функции, когда используете оператор **exit**. Эта ошибка случается, если вы пытаетесь сделать то же самое с процедурой. Процедура не может возвращать значение.
Error: constructors and destructors must be methods	Вы объявили процедуру как конструктор или деструктор, когда процедура не является методом класса.
Error: Operator is not overloaded	Вы пытаетесь использовать перегружаемый оператор, когда он не является перегружаемым для этого типа.
Error: Impossible to overload assignment for equal types	Вы не можете перегрузить аргумент для типов, которые компилятор рассматривает как эквивалентные.
Error: Impossible operator overload	Комбинация оператор/аргумент/возвращаемое значение является несовместимой.
Error: Re-raise isn't possible there	Вы пытаетесь вызвать исключение там, где это недопустимо. Вы можете вызвать исключение только в блоке **except**.

Сообщение	Описание
Error: The extended syntax of new or dispose isn't allowed for a class	Вы не можете создать экземпляр класса с расширенным синтаксисом **new**. Для этого должен использоваться конструктор. По тем же соображениям вы не можете использовать вызов **dispose** для уничтожения экземпляра класса. Для этого нужно использовать деструктор.
Error: Procedure overloading is switched off	Если используется переключатель **–So**, то перегрузка процедур отключена. Turbo Pascal не поддерживает перегрузку функций.
Error: It is not possible to overload this operator. Related overloadable operators (if any) are: Сообщение	Вы пытаетесь перегрузить оператор, который не может быть перегружен. Следующие операторы не могут быть перегружаемыми: `+, -, *, /, =, >, <, <=, >=, is, as, in, **, :=`
Error: Comparative operator must return a boolean value	Если перегружающий оператор – это **=**, то функция должна возвращать логическое значение.
Error: Only virtual methods can be abstract	Вы объявили метод как абстрактный, когда он не объявлен как виртуальный.
Fatal: Use of unsupported feature!	Вы пытаетесь форсировать компилятор, когда он к этому не готов.
Error: The mix of different kind of objects (class, object, interface, etc) isn't allowed	Вы не можете порождать объекты, классы и интерфейсы (**objects**, **classes**, **cppclasses** и **interfaces**) друг от друга. Например, класс не может быть предком объекта и наоборот.
Warning: Unknown procedure directive had to be ignored: "Сообщение"	Вы указали неизвестную процедурную директиву.

Error: absolute can only be associated to one variable	Вы не можете указать более одной переменной перед директивой **absolute**. Таким образом, следующая конструкция вызовет эту ошибку:
	
	`Var Z : Longint;`
	`X,Y : Longint absolute Z;`

Error: absolute can only be associated with a var or const	Адрес директивы absolute может только указывать на переменную или константу. Таким образом, следующий код вызовет эту ошибку:
	
	`Procedure X;`
	`var p : longint absolute x;`

Error: Only one variable can be initialized	Вы не можете указать более одной переменной с начальным значением в режиме Делфи.
Error: Abstract methods shouldn't have any definition (with function body)	Абстрактные методы можно только объявлять. Вы не можете их выполнить. Они должны быть перегружены в классе-потомке.
Error: This overloaded function can't be local (must be exported)	Вы определили перегружаемую функцию в разделе **implementation** модуля, но она не ссылается на объявление в разделе **interface**.
Warning: Virtual methods are used without a constructor in "Сообщение"	Если вы определили объекты или классы, содержащие виртуальные методы, то вам необходимо иметь конструктор и деструктор для их инициализации. Компилятор нашёл объект или класс с виртуальными методами, которые не имеют пары конструктор/деструктор.
Macro defined: Сообщение	Если используется **–vc**, то компилятор сообщает вам, если он определяет макрос.
Macro undefined: Сообщение	Если используется **–vc**, то компилятор сообщает вам, если он не определяет макрос.
Macro Сообщение1 set to Сообщение2	Если используется **–vc**, то компилятор сообщает вам, когда макрос получает значения.
Info: Compiling Сообщение	Если вы включили информационные сообщения (**-vi**), компилятор сообщает вам, что модули перекомпилированы.
Parsing interface of unit Сообщение	Сообщение о том, что начато чтения интерфейсной части модуля.
Parsing implementation of Сообщение	Сообщение о том, что начато чтения исполняемой части модуля, библиотеки или программы.
Compiling Сообщение for the second time	Если вы запросили отладочную информацию (**-vd**), компилятор сообщает вам, что он повторно перекомпилировал модули.
Error: No property found to override	Вы хотите перегрузить свойство существующего класса-предка, но там такого свойства нет.
Error: Only one default property is allowed	Вы указали свойство как Default, но класс уже имеет свойство по умолчанию, а класс может иметь только одно такое свойство.
Error: The default property must be an array property	Только массив свойств класса может быть сделан свойствами по умолчанию.

Сообщение	Описание
Error: Virtual constructors are only supported in class object model	Вы не можете иметь виртуальные конструкторы в объектах. Это допускается только для классов.
Error: No default property available	Вы пытаетесь получить доступ к свойству класса по умолчанию, но этот класс (или один из его предков) не имеет свойства по умолчанию.
Error: The class can't have a published section, use the {$M+} switch	Если вы хотите определить в классе раздел published, вы должны использовать переключатель **{$M+}**, который включает генерацию информации.

Error: Forward declaration of class "Сообщение" must be resolved here to use the class as ancestor	Чтобы иметь возможность использовать объект как объект-предок, его нужно сначала определить. Эта ошибка случается в следующей ситуации:
	
	`Type ParentClas = Class;`
	`ChildClass = Class(ParentClass)`
	`...`
	`end;`
	
	где **ParentClass** объявлен, но не определён.

Error: Local operators not supported	Вы не можете перегружать локальные операторы, то есть внутри определения процедуры или функции.
Error: Procedure directive "Сообщение" not allowed in interface section	Эта процедурная директива не допускается в разделе **interface** модуля. Вы можете использовать её только в разделе **implementation**.
Error: Procedure directive "Сообщение" not allowed in implementation section	Эта процедурная директива не допускается в разделе **implementation** модуля. Вы можете использовать её только в разделе **interface**.
Error: Procedure directive "Сообщение" not allowed in procvar declaration	Эта процедурная директива не может быть частью объявления процедурного типа или типа функции.
Error: Function is already declared Public/Forward "Сообщение"	Вы получите эту ошибку, если функция дважды объявлена как forward. Или если она имеется в разделе interface, а затем имеется объявление forward в разделе implementation.
Error: Can't use both EXPORT and EXTERNAL	Эти две процедурные директивы являются взаимно исключающими.
Warning: "arg1" not yet supported inside inline procedure/function	Встроенные процедуры не поддерживают это объявление.
Warning: Inlining disabled	Встроенные процедуры отключены.
Info: Writing Browser log Сообщение	Если информационные сообщения включены, то компилятор предупреждает вас, когда записывает лог обозревателя (генерируемый переключателем **{$Y+ }**).
Hint: may be pointer dereference is missing	Компилятор предполагает, что указатель нужно разыменовать.
Fatal: Selected assembler reader not supported	Выбранный ассемблер (с **{$ASMMODE xxx}** не поддерживается. Компилятор может работать с или без поддержки редкого ассемблера.
Error: Procedure directive "Сообщение" has conflicts with other directives	Вы указали процедурную директиву, которая конфликтует с другой директивой. Например, **cdecl** и **pascal** являются взаимно исключающими, то есть несовместимыми.
Error: Calling convention doesn't match forward	Эта ошибка случается, когда вы объявляете функцию или процедуру, например, с **cdecl**, но пропускаете эту директиву в разделе **implementation**, или наоборот. Соглашение о вызовах – это часть объявления функции, которая должна быть повторена в определении функции.
Error: The default value of a property must be constant	Значение по умолчанию объявленного свойства должно быть известно во время компиляции. Значение, которое вы указали, будет известно только во время выполнения. Эта ошибка случается, например, если вы указали имя переменной как значение по умолчанию.
Error: Symbol can't be published, can be only a class	Только переменные типа класса могут быть в разделе **published** класса, если они не объявлены как свойства.
Error: This kind of property can't be published	Свойства в разделе **published** не могут быть массивом. Они должны быть перемещены в раздел **public**. Свойства в разделе **published** должны быть порядкового, вещественного, строкового типа или множеством.
Error: An import name is required	Некоторые целевые платформы требуют имени для импортируемой процедуры или спецификатора **cdecl**.
Error: Division by zero	Было обнаружено деление на ноль.
Error: Invalid floating point operation	Операция с двумя значениями вещественного типа вызвала переполнение или деление на ноль.
Error: Upper bound of range is less than lower bound	Верхняя граница объявленного массива меньше, чем нижняя граница, а это недопустимо.

Сообщение	Описание
Warning: string "Сообщение1" is longer than "Сообщение2"	Размер строковой константы больше, чем размер, указанный вами при определении строкового типа.
Error: string length is larger than array of char length	Размер строковой константы больше, чем размер, указанный вами при определении **Array[x..y] of char**.
Error: Illegal expression after message directive	Free Pascal поддерживает только целочисленные или строковые значения в качестве констант для сообщений.
Error: Message handlers can take only one call by ref. parameter	Метод, объявленный с директивой **message** как обработчик сообщения, может принять только один параметр, который должен быть передан по ссылке. Параметры, передаваемые по ссылке, используют директиву **var**.
Error: Duplicate message label: "Сообщение"	Метка для сообщения используется дважды в одном объекте/классе.
Error: Self can only be an explicit parameter in methods which are message handlers	Параметр **Self** может передаваться ТОЛЬКО в метод, который объявлен как обработчик сообщения.
Error: Threadvars can be only static or global	Потоковые переменные должны быть статическими или глобальными. Вы не можете объявить локальный поток для процедуры. Локальные переменные являются всегда локальными в потоке, потому что каждый поток свой собственный стек и локальные переменные записываются в стек.
Fatal: Direct assembler not supported for binary output format	Вы не можете использовать ассемблер напрямую, если используете бинарный писатель. Вы берите другой выходной формат или используйте другой ассемблер.
Warning: Don't load OBJPAS unit manually, use n{n$mode objfpcn} or n{n$mode delphin} instead	Вы пытаетесь загрузить модуль **ObjPas** вручную из раздела uses. Это плохая идея. Используйте директивы **{$MODE OBJFPC}** или **{$mode delphi}**, которые загружают этот модуль автоматически.
Error: OVERRIDE can't be used in objects	Override не поддерживается для объектов, используйте virtual вместо этого для перегрузки метода объекта-предка.
Error: Data types which require initialization/finalization can't be used in variant records	Некоторые типы данных (например, **ansistring**) требуют кода инициализации/финализации, который полностью генерируется компилятором. Такие типы данных нельзя использовать в вариантной части записи.
Error: Resourcestrings can be only static or global	Строка ресурсов не может быть объявлена локально, только глобально или с директивой **static**.
Error: Exit with argument can't be used here	Выход из блока с аргументом для возвращаемого значения не может здесь использоваться. Эта ошибка может случиться, например, в блоке **try..except** или **try..finally**.
Error: The type of the storage symbol must be boolean	Если вы указали символ **storage** в объявлении свойства, то оно должно быть логического типа.
Error: This symbol isn't allowed as storage symbol	Вы не можете использовать этот типа символа как спецификатор **storage** в объявлении свойства. Вы можете использовать только методы результатом логического типа, логическими полями класса или логическими константами.
Error: Only classes which are compiled in $M+ mode can be published	Поле типа класса в разделе **published** класса может быть только классом, который был скомпилирован в **{$M+}** или который является потомком такого класса. Обычно такой класс должен быть потомком от **TPersistent**.

Error: Procedure directive expected	Эта ошибка случается, если вы имеете директиву **{$Calling}** без указанного условия вызова. Это также случается, если процедура объявлена в блоке констант и вы используете точку с запятой (;) после объявления процедуры, которое должна следовать за процедурной директивой. Правильное объявление:
	
	`const`
	`p : procedure;stdcall=nil;`
	`p : procedure stdcall=nil;`

Error: The value for a property index must be of an ordinal type	Значение, которое вы используете для индекса свойства, должно быть порядкового типа.
Error: Procedure name too short to be exported	Длина имени процедуры или функции должна быть не менее 2 символов. Это потому, что имеется ошибка в **dlltool**, который неправильно распознаёт файлы **.def** с именами длиной в 1 символ.
Error: No DEFFILE entry can be generated for unit global vars	Нет вхождений **DEFFILE**, которые могут быть сгенерированы для глобальных переменных модуля.
Error: Compile without -WD option	Вам нужно компилировать этот файл без опции –WD в командной строке.
Fatal: You need ObjFpc (-S2) or Delphi (-Sd) mode to compile this module	Вам нужно использовать **{$MODE OBJFPC}** или **{$MODE DELPHI}** для компиляции этого файла. Или использовать соответствующую опцию командной строки, любую из -objfpc или -MDelphi.
Error: Can't export with index under Сообщение	Экспортирование функций или процедур с указанным индексом не поддерживается данной целевой платформой.

Сообщение	Описание
Error: Exporting of variables is not supported under Сообщение	Экспортирование переменных не поддерживается данной целевой платформой.
Error: Improper GUID syntax	Код GUID имеет неправильный синтаксис. Он должен иметь формат: `{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}` Где каждый X – шестнадцатиричное число.
Warning: Procedure named "Сообщ1" not found that is suitable for implementing the Сообщ2.Сообщ3	Компилятор не может найти подходящую процедуру, которая выполняет данный метод интерфейса. Процедура с таким именем найдена, но параметры не совпадают.

Error: interface identifier expected	Это сообщение появляется, если компилятор сканирует объявление класса, который содержит **interface** функции, подобно следующему:
	
	`type`
	`TMyObject = class(TObject, IDispatch)`
	`function IUnknown.QueryInterface=MyQueryInterface;`
	`....`
	
	а **interface** не перечислен в списке потомков.

Error: Type "Сообщение" can't be used as array index type	Типы, подобные **qword** или **int64**, не допускается использовать в индексах массивов.
Error: Con- and destructors aren't allowed in interfaces	Объявления конструктора и деструктора не допускаются в интерфейсах. В большинстве случаев метод QueryInterface интерфейса IUnknown можно использовать для создания нового интерфейса.
Error: Access specifiers can't be used in INTERFACES	Спецификаторы доступности **public**, **private**, **protected** и **published** нельзя использовать в интерфейсах, потому что все методы интерфейса должны быть доступны (**public**).
Error: An interface can't contain fields	Объявления полей не допускаются в интерфейсах. Интерфейс может содержать только методы и свойства с методом чтения/записи и спецификатором доступа.
Error: Can't declare local procedure as EXTERNAL	Объявление локальных процедур как внешних невозможно. Локальные процедуры получают скрытые параметры, которые делают вероятность ошибки очень высокой.
Warning: Some fields coming before "Сообщение" weren't initialized	В режиме **Delphi** не все поля записи-константы могут быть инициализированы, компилятор предупреждает вас, когда обнаруживает такую ситуацию.
Error: Some fields coming before "Сообщение" weren't initialized	Во всех режимах синтаксиса, кроме режима **Delphi**, вы не можете инициализировать некоторые поля в середине записи-константы.
Warning: Some fields coming after "Сообщение" weren't initialized	Вы можете инициализировать некоторые поля в конце записи-константы (компилятор их инициализирует со значением 0 автоматически). Это может вызвать труднонаходимые проблемы.
Error: VarArgs directive (or '...' in MacPas) without CDecl/CPPDecl/MWPascal and External	Директива **varargs** (или "..." параметр **varargs** в режиме MacPas) может использоваться только с процедурами и функциями, которые объявлены как **external** и одним из **cdecl**, **cppdecl** или **mwpascal**. Эта функциональность поддерживается только для совместимости интерфейса функций C, подобных **printf**.
Error: Self must be a normal (call-by-value) parameter	Вы не можете объявить **Self** как параметр **const** или **var**, он должен быть всега параметром, передаваемым по значению.
Error: Interface "Сообщение" has no interface identification	Если вы хотите назначить интерфейс для константы, то интерфейс должен иметь значение GUID.
Error: Unknown class field or method identifier "Сообщение"	Свойства должны ссылаться на поле или метод в этом же классе.
Warning: Overriding calling convention "Сообщение1" with "Сообщение2"	Имеются две директивы в объявлении процедуры, которые определяют соглашение о вызовах. Только последняя директива будет использоваться.
Error: Typed constants of the type "procedure of object" can only be initialized with NIL	Вы не можете связать адрес метода с типизированной константой, которая имеет тип «процедура объекта», потому что такая константа требует два адреса: для метода (который известен во время компиляции) и для объекта или класса (который неизвестен во время компиляции).

Error: Default value can only be assigned to one parameter	Невозможно указать значение по умолчанию для нескольких параметров. Следующий пример неправильный:
	
	`Procedure MyProcedure (A,B : Integer = 0);`
	
	Вместо этого вы должны написать так:
	
	`Procedure MyProcedure (A : Integer = 0; B : Integer = 0);`

Сообщение	Описание
Error: Default parameter required for "Сообщение"	Указанный параметр требует значения по умолчанию.
Warning: Use of unsupported feature!	Вы пытаетесь заставить компилятор выполнить какое-либо действие, которое он пока не может выполнить.
Hint: C arrays are passed by reference	Какой-то массив, переданный в функцию C, является переданным по ссылке.
Error: C array of const must be the last argument	Вы не можете добавить какой-либо другой параметр после массива констант для функции **cdecl**, так как размер стека для этого параметра неизвестен.
Hint: Type "Сообщение" redefinition	Это сообщение указывает на то, что ранее объявленный тип был переопределён как-то иначе. Это может быть (а может и не быть) потенциально ошибкой исходного кода.
Warning: cdecl'ared functions have no high parameter	Функции, объявленные с модификатором **cdecl**, не имеют дополнительного явного параметра.
Warning: cdecl'ared functions do not support open strings	**Openstring** не поддерживается для функций, которые имеют модификатор **cdecl**.
Error: Cannot initialize variables declared as threadvar	Переменные, объявленные как **threadvar**, не могут быть инициализированы с значением по умолчанию. Такие переменные всегда обнуляются при старте нового потока.
Error: Message directive is only allowed in Classes	Директива **message** поддерживается только для типов **Class**.
Error: Procedure or Function expected	Метод класса может быть указан только для процедур и функций.
Warning: Calling convention directive ignored: "Сообщение"	Некоторые соглашения о вызовах поддерживаются только определёнными процессорами. Например, большинство не i386 портов поддерживаются только стандартным соглашением о вызовах ABI.
Error: REINTRODUCE can't be used in objects	**reintroduce** не поддерживается для объектов.
Error: Each argument must have its own location	Если размещение для аргументов указано явно, как того требуют некоторые соглашения системных вызовов (syscall), то каждый аргумент должен иметь своё собственное размещение. Мысли, подобные этим `procedure p(i,j : longint 'r1');` недопустимы.
Error: Each argument must have an explicit location	Если для одного аргумента указано явно размещение, то все аргументы процедуры должны иметь явно указанное размещение.
Error: Unknown argument location	Размещение, указанное для аргумента, не распознаётся компилятором.
Error: 32 Bit-Integer or pointer variable expected	**libbase** для MorphOS/AmigaOS можно получить только как переменную **longint**, **dword** или любой указатель.

Error: Goto statements aren't allowed between different procedures	Не допускается использовать оператор **goto**, ссылающийся на метку, которая находится за пределами текущей процедуры. Следующий пример показывает эту проблему:
	
	`...`
	`procedure p1;`
	`  label`
	`    l1;`
	`   procedure p2;`
	`   begin`
	`     goto l1; // Такой переход не допускается`
	`   end;`
	`begin`
	`  p2`
	`  l1:`
	`end;`
	`...`

Fatal: Procedure too complex, it requires too many registers	Тело вашей процедуры слишком большое для компилятора. Вы должны разделить процедуру на несколько меньших процедур.
Error: Illegal expression	Это может случиться во многих ситуациях. Обычно происходит при попытке вычислить выражение с константами.
Error: Invalid integer expression	Вы написали выражение, которое не является целочисленным, а компилятор ожидал целочисленный результат.

Сообщение	Описание
Error: Illegal qualifier	Один из следующих вариантов: • Вы пытаетесь получить доступ к полю переменной, которая не является записью • Вы индексируете переменную, которая не является массивом. • Вы разыменовываете переменную, которая не является указателем.
Error: High range limit < low range limit	Вы объявили поддиапазон, где верхний предел меньше, чем нижний предел диапазона.
Error: Exit's parameter must be the name of the procedure it is used in	Не локальный выход не допускается. Эта ошибка случается только в режиме **MacPas**.
Error: Illegal assignment to for-loop variable "Сообщение"	Тип переменной цикла **for** должен быть порядковым. Переменные цикла не могут быть строками или вещественными числами. Вы также не можете назначать значения переменным цикла внутри этого цикла (кроме режимов Delphi и TP). Используйте циклы **while** или **repeat** вместо цикла **for**, если вам необходимо снять вышеописанные ограничения.
Error: Can't declare local variable as EXTERNAL	Объявление локальных переменных как **external** не допускается. Только глобальные переменные могут быть связаны с внешними переменными.
Error: Procedure is already declared EXTERNAL	Процедура уже объявлена с директивой EXTERNAL в объявлении **interface** или **forward**.
Warning: Implicit uses of Variants unit	Тип **Variant** используется в каком-либо модуле без подключения модуля **Variants**. Компилятор требует явно указывать модуль Variants в разделе **uses**. Для удаления этого сообщения модуль Variants должен быть добавлен в список модулей в разделе **uses**.
Error: Class and static methods can't be used in INTERFACES	Спецификатор **class** и директива **static** не могут использоваться в интерфейсах, потому что все методы интерфейса должны быть общедоступными.
Error: Overflow in arithmetic operation	Операция с двумя целочисленными значениями вызвала переполнение.
Error: Protected or private expected	**strict** можно использовать только совместно с **protected** или **private**.
Error: SLICE can't be used outside of parameter list	**slice** можно использовать только для аргументов, принимающих параметр открытого массива.
Error: A DISPINTERFACE can't have a parent class	DISPINTERFACE – это специальный тип интерфейса, который не может иметь класса-предка.
Error: A DISPINTERFACE needs a guid	DISPINTERFACE всегда требует идентификации интерфейса (GUID).
Warning: Overridden methods must have a related return type. This code may crash, it depends on a Delphi parser bug	Если вы объявили перегружаемые методы в определении класса, они должны иметь одинаковый тип возвращаемого результата. Некоторые версии **Delphi** позволяют изменять тип возвращаемого результата методов интерфейса, и даже заменять процедуры на функции, но в результате код может оказаться неработоспособен на используемых типах и вызываемых методах.
Error: Dispatch IDs must be ordinal constants	Ключевое слово **dispid** должно следовать за константой порядкового типа (индекс **dispid**).
Error: The range of the array is too large	Несмотря на правильно указанный размер, количество элементов в массиве не может быть более чем **high(ptrint)**. Кроме того, тип, определяющий диапазон массива, должен быть поддиапазоном **ptrint**.
Error: The address cannot be taken of bit packed array elements and record fields	Если вы объявили массив или запись как **packed** в режиме Mac Pascal (или как **packed** в любом режиме с **{$bitpacking on}**), то он будет запакован до битового уровня. Это означает, что невозможно будет получить индивидуальный адрес элемента массива или поля записи. Исключение из этого правила возможно только в том случае, если размер упаковки кратен 8.
Error: Dynamic arrays cannot be packed	Только стандартные (и возможно в будущем – открытые) массивы могут быть упакованы.
Error: Bit packed array elements and record fields cannot be used as loop variables	Если вы объявили массив или запись как **packed** в режиме Mac Pascal (или как **packed** в любом режиме с **{$bitpacking on}**), то он будет запакован до битового уровня. Это означает, что для работы с таким массивом нельзя будет использовать циклы.
Error: VAR and TYPE are allowed only in generics	Использование VAR и TYPE для объявлений новых типов допускается только внутри **generics**.
Error: This type can't be a generic	Только классы, объекты, интерфейсы и записи могут использоваться как **generic**.

Сообщение	Описание
Warning: Don't load LINEINFO unit manually, Use the -gl compiler switch instead	Не используйте модуль **lineinfo** напрямую. Используйте переключатель **-gl**, который правильно и автоматически добавляет этот модуль для чтения выбранного типа отладочной информации. Модуль, который требуется для использования отладочной информации соответствующего типа, используется, когда компилируется бинарный файл.
Error: No function result type specified for function "Сообщение"	Когда вы объявляете функцию первый раз, вы должны объявлять её в полной форме, то есть со всеми параметрами и типом результата.
Error: Specialization is only supported for generic types	Типы, не являющиеся generics, не могут быть специализированы.
Error: Generics can't be used as parameters when specializing generics	Если специализация generic, то только не-generic типы могут использоваться как параметры.
Error: Constants of objects containing a VMT aren't allowed	Если объект требует VMT, потому что он содержит конструктор или виртуальные методы, то не допускается создавать в нём константы. В режимах TP и Delphi это разрешено для совместимости.
Error: Taking the address of labels defined outside the current scope isn't allowed	Не допускается брать адрес метки за пределами текущей процедуры.
Error: Cannot initialize variables declared as external	Переменные, объявленные как **external**, не могут быть инициализированы значением по умолчанию.
Error: Illegal function result type	Некоторые типы, например, файловые, не могут использоваться как результат, возвращаемый функцией.
Error: No common type possible between "Сообщ1" and "Сообщ2"	Для выполнения операций с целыми числами компилятор преобразовал оба операнда в их общий тип, который оказался неправильным. Для определения общего типа операндов, компилятор берёт **минимум** из минимальных значений обоих типов, а затем **максимум** из максимальных значений обоих типов. Общий тип получается **минимум**..**максимум**.
Error: Generics without specialization cannot be used as a type for a variable	Generics должен быть всегда специализирован перед использованием как тип переменной.
Warning: Register list is ignored for pure assembler routines	Если используется чисто ассемблерная процедура, то список с изменёнными регистрами игнорируется.
Error: Implements property must have class or interface type	Свойство, которое принадлежит интерфейсу, должно быть типа класса или интерфейса.
Error: Implements-property must implement interface of correct type, found "Сообщ1" expected "Сообщ2"	Свойство, которое принадлежит интерфейсу, на самом деле принадлежит другому интерфейсу.
Error: Implements-property must have read specifier	Свойство, которое принадлежит интерфейсу, должно иметь, по меньшей мере, спецификатор **read**.
Error: Implements-property must not have write-specifier	Свойство, которое принадлежит интерфейсу, может не иметь спецификатор **write**.
Error: Implements-property must not have stored-specifier	Свойство, которое принадлежит интерфейсу, может не иметь спецификатора **stored**.
Error: Implements-property used on unimplemented interface: "Сообщение"	Интерфейс, которому принадлежит свойство, не принадлежит классу.
Error: Floating point not supported for this target	Компилятор проанализировал выражение с плавающей точкой, но оно не поддерживается.
Error: Class "Сообщ1" does not implement interface "Сообщ2"	Делегированный интерфейс не принадлежит классу, данному в разделе **implements**.
Error: Type used by implements must be an interface	Ключевое слово **implements** должно применяться с типом **interface**.
Error: Variables cannot be exported with a different name on this target, add the name to the declaration using the "export"	На большинстве целевых платформ невозможно изменять имена, под которыми переменные экспортируются внутри операторов **exports** библиотеки. В таком случае вам нужно указать экспортируемое имя в точке, где объявлена переменная, используя **export** и директиву **alias**.
Error: Weak external symbols are not supported for the current target	Символ "**weak external**" – это символ, который может существовать, а может и нет (при **static** или **dynamic**) во время компоновки. Эта концепция может быть недоступна (или ещё не работать) на текущей платформа или процессоре.
Error: Forward type definition does not match	Классы и интерфейсы с объявлением forward должны иметь одинаковые типы с implemented. Интерфейс forward не может быть изменён в классе.
Note: Virtual method "Сообщ1" has a lower visibility (Сообщ2) than parent class Сообщ3 (Сообщ4)	Виртуальный метод перегружает метод, который объявлен с более высокой видимостью. Это может дать непредвиденные результаты.

Сообщение	Описание
Error: Fields cannot appear after a method or property definition, start a new visibility section first	Однажды определив метод или свойство в классе или объекте, вы не можете определить какие-либо поля впоследствии без начального раздела видимости (такого как **public**, **private** и т.п.).

Error: Parameters cannot contain local type definitions. Use a separate type definition in a type block.	В Паскале нельзя передавать определения типов в качестве параметров, даже если они семантически эквивалентны простым типам. Переменные или параметры должны быть одинакового типа, если они ссылаются на определения одинаковых типов. В результате это не позволяет объявлять новые типы внутри списка параметров, потому что тогда возможны ссылки на то же определение типа в заголовке процедуры в разделах **interface** и **implementation** модуля (оба заголовка процедуры определены с разными типами). Помните, что выражения, такие как “file of byte” или “string[50]” также должны определяться как новый тип. Например, вы не можете объявить процедуру:
	
	`Procedure MyProc(st : string[50]);`
	
	Это приведёт к данной ошибке. Правильно будет так:
	
	`Type TMyStr = string[50];`
	`Procedure MyProc(st : TMyStr);`

Error: ABSTRACT and SEALED conflict	ABSTRACT и SEALED не могут использоваться вместе в одном объявлении.
Error: Cannot create a descendant of the sealed class "Сообщение"	Неизвестный метод этого класса не может быть унаследован другим классом.
Error: SEALED class cannot have an ABSTRACT method	Неизвестный метод этого класса не может быть унаследован. Следовательно, ни один класс не способен перегрузить абстрактный метод в неизвестном классе.
Error: Only virtual methods can be final	Вы объявили метод как финальный, когда он не объявлен как виртуальный.
Error: Final method cannot be overridden: "Сообщение"	Вы пытаетесь перегрузить виртуальный метод родительского класса, который не существует.
Error: Invalid enumerator identifier: "Сообщение"	Только идентификаторы перечислений "MoveNext" и "Current" поддерживаются.
Error: Enumerator identifier required	Идентификатор "MoveNext" или "Current" должен сопровождать модификатор нумератора.
Error: Enumerator MoveNext pattern method is not valid.	Метод должен быть функцией с результатом типа **Boolean**, а соответствующий нумератор "MoveNext" должен быть функцией с результатом типа **Boolean** и не требовать аргументов.
Error: Enumerator Current pattern property is not valid. Property must have a getter.	Нумератор "Current" соответствующего свойства должен быть больше.
Error: Only one enumerator MoveNext method is allowed per class/object	Класс или объект может иметь только один нумератор **MoveNext**.
Error: Only one enumerator Current property is allowed per class/object	Класс или объект может иметь только один нумератор **Current**.
Error: For in loop cannot be used for the type "Сообщение"	**For** в цикле может использоваться не для всех типов. Например, он не может использоваться для нумераторов.

C4. Ошибки проверки типов

В этом разделе описаны все ошибки, которые могут случиться при проверке типов.

Сообщение	Описание
Error: Type mismatch	Это может произойти во многих случаях: • Назначенная вами переменная отличается от типа, который используется в выражении • Вы вызываете функцию или процедуру с параметрами, которые несовместимы с параметрами в объявлении функции или процедуры

Сообщение	Описание

Error: Incompatible types: got "Сообщ1" expected "Сообщ2"	Невозможно преобразование между двумя типами. Ещё одна причина – типы объявлены в разных объявлениях:
	
	`Var A1 : Array[1..10] Of Integer;`
	`  A2 : Array[1..10] Of Integer;`
	`Begin`
	`  A1:=A2; {Этот оператор также даёт такую ошибку, потому`
	`  что выполняется строгая проверка типов Паскаль}`
	`End.`

Error: Type mismatch between "Сообщ1" and "Сообщ2"	Типы не являются эквивалентными.
Error: Type identifier expected	Идентификатор не является типом, или вы забыли указать идентификатор **type**.
Error: Variable identifier expected	Это случается, если вы помещаете константу в процедуру (такую как **Inc** или **Dec**), в то время как процедура требует переменной. Для таких процедур в качестве параметров можно помещать только переменные.
Error: Integer expression expected, but got "Сообщение"	Компилятор ожидает выражения типа **integer**, но получает другой тип.
Error: Boolean expression expected, but got "Сообщение"	ТВыражение должно быть типа **boolean**. Оно должно возвращать **True** или **False**.
Error: Ordinal expression expected	Выражение должно быть порядкового типа, то есть максимум типа **Longint**. Эта ошибка случается, например, если вы указали второй параметр процедуры Inc или Dec, который не соответствует порядковому типу.
Error: pointer type expected, but got "Сообщение"	Переменная или выражения не являются указателем. Это случается, если вы помещаете переменную, которая не является указателем, в New или Dispose.
Error: class type expected, but got "Сообщение"	Переменная или выражение не являются типом **class**. Это обычно случается, если 1. Родительский класс в объявлении класса не является классом 2. Обработчик исключения (On) ссодержит идентификатор типа, который не является классом.
Error: Can't evaluate constant expression	Эта ошибка может случиться, если границы объявленного вами массива не обозначены порядковыми константами.
Error: Set elements are not compatible	Вы пытаетесь выполнить операцию с двумя множествами, в то время как типы элементов этих множеств не являются одинаковыми. Базовые типы множеств должны быть одинаковыми при объединении.
Error: Operation not implemented for sets	Некоторые бинарные операторы не определены для множеств. Это операторы: div, mod, **, >= и <=. Последние два могут быть определены для множеств в будущих версиях.
Warning: Automatic type conversion from floating type to COMP which is an integer type	Обнаружено явное преобразование типов из **real** в **comp**. s encountered. Поскольку **comp** – это 64-битное целое число, то это может вызвать ошибку.
Hint: use DIV instead to get an integer result	Если подсказки включены, то целочисленное деление с оператором '/' приведёт к этому сообщению, потому что результатом будет вещественный тип.
Error: string types doesn't match, because of $V+ mode	Если выполняется компиляция в режиме {$V+}, то строка, передаваемая вами в качестве параметра, должна быть точно такого же типа, как параметр процедуры.
Error: succ or pred on enums with assignments not possible	Если вы объявили перечисляемый тип в стиле C, например, так: Tenum = (a,b,e:=5); То вы не сможете использовать функции Succ или Pred с этим перечислением.
Error: Can't read or write variables of this type	Вы пытаетесь прочитать или записать переменную из файла или в файл текстового типа, который не поддерживает тип переменной. Только целочисленные типы, вещественные, pchars и strings можно читать из файла или записывать в текстовый файл. Логические переменные можно только записывать в текстовый файл.
Error: Can't use readln or writeln on typed file	**readln** и **writeln** можно использовать только с текстовыми файлами.
Error: Can't use read or write on untyped file.	**read** и **write** допускаются только для текстовых или типизированных файлов.

Сообщение	Описание
Error: Type conflict between set elements	Это означает, что не менее одного элемента множества имеют неправильный тип.
Warning: lo/hi(dword/qword) returns the upper/lower word/dword	Free Pascal поддерживает перегруженную версию **lo/hi** для **longint/dword/int64/qword**, которые возвращают наименьшее/наибольшее (результат типа слово/двойное слово) значение аргумента. Turbo Pascal позволяет использовать 16-битные **lo/hi**, которые возвращают биты 0..7 для **lo** и биты 8..15 для **hi**. Если вы хотите получить поведение, аналогичное Turbo Pascal, вы должны использовать приведение типов к **word** или **integer**.
Error: Integer or real expression expected	Первый аргумент для **str** должен быть типа **real** или **integer**.
Error: Wrong type "Сообщение" in array constructor	Вы пытаетесь использовать тип в конструкторе массива, который недопустим.
Error: Incompatible type for arg no. Сообщ1: Got "Сообщ2", expected "Сообщ3"	Вы пытаетесь передать неправильный тип в указанный параметр.
Error: Method (variable) and Procedure (variable) are not compatible	Вы не можете связать метод с процедурной переменной или процедуру с указателем на метод.
Error: Illegal constant passed to internal math function	Аргумент-константа, переданный в функцию **ln** или **sqrt** выходит за пределы диапазона для этой функции.
Error: Can't take the address of constant expressions	Невозможно получить адрес выражения-константы, потому что оно не записывается в память. Вы можете попробовать сделать типизированную константу. Эта ошибка может также появиться, если вы пытаетесь поместить свойство в параметр **var**.
Error: Argument can't be assigned to	Только выражение, которое может быть в левой части присваивания, может быть передано как вызов по ссылке аргумента. Примечание: Свойства могут использоваться в левой части присваивания, тем не менее, они не могут использоваться как аргументы.
Error: Can't assign local procedure/function to procedure variable	Не допускается присваивать локальные процедуры/функции процедурным переменным, потому что соглашение о вызовах локальных процедур/функций отличаются. Вы можете только присвоить локальную процедуру/функцию пустому указателю.
Error: Can't assign values to an address	Не допускается присваивать значение адресу переменной, константы, процедуры или функции. Вы можете попытаться выполнить компиляцию с опцией **-So**, если идентификатор является процедурной переменной.
Error: Can't assign values to const variable	Не допускается присваивать значение переменной, которая объявлена как константа. Обычно параметр объявляется как константа. Чтобы иметь возможность изменять значение, передавайте параметр по значению или параметр по ссылке (используя **var**).
Error: Array type required	Если вы хотите получить доступ к переменной, используя индекс '[<x>]', то тип должен быть массивом. В режиме FPC указатель также допускается.

Error: interface type expected, but got "Сообщение"	Компилятор ожидал для нумератора имя типа интерфейса, но получил нечто другое. Следующий код приведёт к этой ошибке:
	
	`Type`
	`TMyStream = Class(TStream,Integer)`

Hint: Mixing signed expressions and longwords gives a 64bit result	Если вы делите (или вычисляете модуль) выражения со знаком с типом **longword** (или наоборот), или если вы имеете переполнение и/или включена проверка диапазона и используется арифметическое выражение (+, -, *, div, mod), в котором оба числа со знаком и появляется **longwords**, то всё это вычисляется как 64-битная арифметическая операция, которая медленнее, чем обычная 32-битная. Вы можете избежать этого при помощи преобразования типа одного из операндов в подходящий для результата и другого операнда.
Warning: Mixing signed expressions and cardinals here may cause a range check error	Если вы используете бинарный оператор (and, or, xor) и один из операндов - это **longword**, в то время как другой – это выражение со знаком, то, если проверка диапазона включена, вы можете получить ошибку проверки диапазона, потому что в этом случае оба операнда преобразуются в longword перед выполнением операции. Вы можете избежать этого при помощи преобразования типа одного из операндов в подходящий для результата и другого операнда.
Error: Typecast has different size (Сообщ1 -> Сообщ2) in assignment	Преобразование типа при отличающихся размерах не допускается, когда переменная используется в присваивании.

Сообщение	Описание
Error: enums with assignments can't be used as array index	Если вы объявили перечисляемый тип, который имеет C-подобные присваивания, как показано ниже: `Tenum = (a,b,e:=5);` Вы не можете использовать его как индекс массива.
Error: Class or Object types "Сообщ1" and "Сообщ2" are not related	Выборка из одного класса в другой, в то время как класс/объект не являются связанными. Вероятно, это ошибка ввода.
Warning: Class types "arg1" and "arg2" are not related	Выборка из одного класса в другой, в то время как класс/объект не являются связанными. Вероятно, это ошибка ввода.
Error: Class or interface type expected, but got "arg1"	Компилятор ожидал имя класса или интерфейса, но получил другой тип или идентификатор.
Error: Type "Сообщение" is not completely defined	Эта ошибка случается, если тип не завершён, например, тип **pointer**, который указывает на неопределённый тип.
Warning: String literal has more characters than short string length	Размер строки-константы, которая связана с **shortstring**, больше максимального размера для **shortstring** (255 символов).
Warning: Comparison is always false due to range of values	Это сравнение беззнакового значения и константы со знаком, которая меньше нуля. По причине преобразования оператор всегда будет возвращать FALSE. Выполните явное преобразование константы в правильный диапазон, чтобы избежать этой проблемы.
Warning: Comparison is always true due to range of values	Это сравнение беззнакового значения и константы со знаком, которая меньше нуля. По причине преобразования оператор всегда будет возвращать TRUE. Выполните явное преобразование константы в правильный диапазон, чтобы избежать этой проблемы.
Warning: Constructing a class "Сообщ1" with abstract method "Сообщ2"	Например, создаваемый класс содержит неисполняемые абстрактные методы. Имеется вероятность, что случится ошибка времени исполнения 211 в коде, если эта процедура будет когда-либо вызвана. Все абстрактные методы должны быть перегружаемыми.
Hint: The left operand of the IN operator should be byte sized	Левый операнд в операторе **IN** не является порядковым или перечислением, который помещается в 8 бит. Это может привести к ошибке проверки диапазона. На текущий момент оператор **in** поддерживает левый оператор только в пределах байта. В случае с перечислениями, размер элемента перечисления может изменяться опциями **{$PACKENUM}** или **{$Zn}**.
Warning: Type size mismatch, possible loss of data / range check error	Это случается, когда меньшему типу присваивается значение, большее, чем исходный тип. Это означает, что может произойти ошибка проверки диапазона или уменьшение значения.
Hint: Type size mismatch, possible loss of data / range check error	Это случается, когда меньшему типу присваивается значение, большее, чем исходный тип. Это означает, что может произойти ошибка проверки диапазона или уменьшение значения.
Error: The address of an abstract method can't be taken	Не найдено тело абстрактного метода, поэтому адрес абстрактного метода не может быть назначен.
Error: Assignments to formal parameters and open arrays are not possible	Вы пытаетесь присвоить значение формальному параметру (нетипизированный var, const или out), или открытому массиву.
Error: Constant Expression expected	Компилятор ожидал выражение-константу, но получил выражение-переменную.
Error: Operation "Сообщ1" not supported for types "Сообщ2" and "Сообщ3"	Операция не допускается для указанных типов.
Error: Illegal type conversion: "Сообщ1" to "Сообщ2"	Когда выполняете преобразование типов, вы должны понимать, что размеры переменной и типа назначения одинаковы.
Hint: Conversion between ordinals and pointers is not portable	Если вы преобразуете тип **pointer** в **longint** (или наоборот), то код не будет компилироваться на машинах, использующих 64-разрядную адресацию.
Warning: Conversion between ordinals and pointers is not portable	Если вы преобразуете тип **pointer** в порядковый тип с другим размером (или наоборот), то могут возникнуть проблемы. Это предупреждение помогает в поиске 32-битного специального кода, где **cardinal**/**longint** используются для преобразования указателей в порядковые типы. Решением проблемы является использование вместо этого типов **ptrint**/**ptruint**.
Error: Can't determine which overloaded function to call	Вы вызываете перегруженную функцию с параметром, который не связан с каким-либо объявленным списком параметров, например, когда вы имеете объявленную функцию с параметрами **word** и **longint**, а затем вызываете её с параметром типа **integer**.
Error: Illegal counter variable	Переменная для цикла **for** должна быть порядкового типа. Переменные циклов не могут быть вещественными числами или строками.

Сообщение	Описание
Warning: Converting constant real value to double for C variable argument, add explicit typecast to prevent this.	В C значения вещественных констант по умолчанию имеют тип **double**. Из этих соображений, когда вы передаёте вещественную константу в функцию C в качестве параметра, компилятор FPC по умолчанию преобразует её в тип **double**. Если вы хотите контролировать этот процесс, добавьте для константы явное преобразование в нужный тип.
Error: Class or COM interface type expected, but got "Сообщение"	Некоторые операторы, такие как AS, применяются только для классов или COM-интерфейсов.
Error: Constant packed arrays are not yet supported	Вы не можете объявить битовый (упакованный) массив как типизированную константу.
Error: Incompatible type for arg no. Сообщ1: Got "Сообщ2" expected "(Bit)Packed Array"	Компилятор ожидает битовый (упакованный) массив как указанный параметр.
Error: Incompatible type for Сообщение no. Сообщ1: Got "Сообщ2" expected ""(not packed) Array"	Компилятор ожидает регулярный (то есть НЕ упакованный) массив как указанный параметр.
Error: Elements of packed arrays cannot be of a type which need to be initialised	Поддержка упакованных массивов, которым необходима инициализация (таких как **ansistrings**, или записей, содержащих **ansistrings**), пока не реализована.
Error: Constant packed records and objects are not yet supported	Вы не можете объявить битовый (упакованный) массив как типизированную константу в данное время.
Warning: Arithmetic "Сообщение" on untyped pointer is unportable to {$T+}, suggest typecast	Сложение/вычитание из нетипизированных указателей может работать по разному в {$T+}. Используёте преобразование типов для типизированных указателей.
Error: Can't take address of a subroutine marked as local	Нельзя получить адрес подпрограммы, помеченной как локальная.
Error: Can't export subroutine marked as local from a unit	Подпрограмма, помеченная как локальная, не может быть экспортирована из модуля.
Error: Type is not automatable: "Сообщение"	Только byte, integer, longint, smallint, currency, single, double, ansistring, widestring, tdatetime, variant, olevariant, wordbool и все интерфейсы являются automatable.
Hint: Converting the operands to "Сообщение" before doing the add could prevent overflow errors.	Сложение двух типов может вызвать ошибку переполнения. Обычно вы конвертируете результат в больший тип. Вы должны предотвращать такие ошибки, преобразуя операнды в этот тип перед сложением.
Hint: Converting the operands to "Сообщение" before doing the subtract could prevent overflow errors.	Вычитание между двумя типами может вызвать ошибку переполнения. Обычно вы конвертируете результат в больший тип. Вы должны предотвращать такие ошибки, преобразуя операнды в этот тип перед вычитанием.
Hint: Converting the operands to "Сообщение" before doing the multiply could prevent overflow errors.	Умножение между двумя типами может вызвать ошибку переполнения. Обычно вы конвертируете результат в больший тип. Вы должны предотвращать такие ошибки, преобразуя операнды в этот тип перед умножением.
Warning: Converting pointers to signed integers may result in wrong comparison results and range errors, use an unsigned	Виртуальное адресное пространство на виртуальных машинах располагается от $00000000 до $ffffffff. Многие операционные системы позволяют выделять память с адресами выше $80000000. Например, как WINDOWS, так и LINUX, допускают использование указателей в диапазоне от $0000000 до $bfffffff. Если вы преобразуете типы со знаком, это может вызвать ошибки переполнения и проверки диапазона, но также $80000000 < $7fffffff. Это может вызвать случайную ошибку в коде, подобно этому: "if p>q".
Error: Interface type Сообщение has no valid GUID	Если применяется оператор **as** для интерфейса или класса, то интерфейс (то есть правый операнд оператора **as**) должен иметь правильный GUID.
Error: Invalid selector name	Селектор **Objective-C** не может быть пустым, он должен быть правильным идентификатором или одинарным двоеточием, а если он содержит менее одного двоеточия, он также должен быть завершён.
Error: Expected Objective-C method, but got Сообщение	Селектор может быть создан только для методов Objective-C, не для любых других процедур/функций/методов.
Error: Expected Objective-C method or constant method name	Селектор может быть создан только для методов Objective-C, при задании имени используются строковые константы или идентификатор метода Objective-C, который является видимым из текущей области видимости.
Error: No type info available for this type	Информация о типах не генерируется для некоторых типов, таких как перечисления с пропусками в их диапазоне значений (включая перечисления, нижняя граница которых отлична от нуля).

C5. Символьная обработка

В этом разделе описаны все сообщения, которые могут случиться при обработке символов, то сеть все, что происходит при обработке имён процедур и переменных.

Сообщение	Описание
Error: Identifier not found "Сообщение"	Компилятору неизвестен этот символ. Обычно появляется, когда вы делаете орфографическую ошибку в имени переменной или процедуры, или если вы забыли объявить переменную.
Fatal: Internal Error in SymTableStack()	Внутренняя ошибка произошла в компиляторе. Если вы обнаружите такую ошибку, пожалуйста, контактируйте с разработчиками и попытайтесь предоставить точное описание обстоятельств, при которых произошла эта ошибка.
Error: Duplicate identifier "Сообщение"	Идентификатор уже был объявлен в текущей области видимости.
Hint: Identifier already defined in Сообщ1 at line Сообщ2	Идентификатор уже был объявлен в предыдущей области видимости.
Error: Unknown identifier "Сообщение"	Обнаружен необъявленный идентификатор, или он используется за пределами области видимости, где он был объявлен.
Error: Forward declaration not solved "Сообщение"	Это может произойти в двух случаях: • Вы объявили функцию в интерфейсной части, или с директивой **forward**, но не определили её в исполняемой части. • Вы ссылаетесь на тип, который не объявлен в текущем блоке **type**.
Error: Error in type definition	Это ошибка в определении нового типа массива. Один из разделителей диапазона в массиве объявлен неправильно. Например, **Array [1..1.25]** вызовет эту ошибку, так как в индексах массивов могут быть только порядковые типы, а 1.25 – это вещественный тип.
Error: Forward type not resolved "Сообщение"	Символ был предварительно определён, но его объявление не было найдено.
Error: Only static variables can be used in static methods or outside methods	Статический метод объекта может быть доступен только статическим переменным.
Fatal: record or class type expected	Переменная или выражение не являются типом **record** или **class**.
Error: Instances of classes or objects with an abstract method are not allowed	Вы пытаетесь создать экземпляр класса, который имеет не перегружаемые абстрактные методы.
Warning: Label not defined "Сообщение"	Метка была объявлена, но не определена.
Error: Label used but not defined "Сообщение"	Метка была объявлена и использована, но не определена.
Error: Illegal label declaration	Эта ошибка не должна никогда появляться. Это случается, если метка определена вне процедуры или функции.
Error: GOTO and LABEL are not supported (use switch -Sg)	Вы должны использовать опцию **-Sg** для компиляции программы, которая имеет метки и операторы **goto**. По умолчанию **label** (метка) и **goto** не поддерживаются.
Error: Label not found	Было обнаружено **goto Метка**, но метка не была объявлена.
Error: identifier isn't a label	Идентификатор, указанный после **goto**, не является меткой.
Error: label already defined	Вы определили метку дважды. Вы можете определить метку только один раз.
Error: illegal type declaration of set elements	Объявление множества содержит неправильное определение типа.
Error: Forward class definition not resolved "Сообщение"	Вы объявили класс, но не определили его.
Hint: Unit "Сообщ1" not used in Сообщ2	Модуль, на который есть ссылка в разделе **uses**, не используется.
Hint: Parameter "Сообщение" not used	Идентификатор был объявлен (локально или глобально), но не был использован (локально или глобально).
Note: Local variable "Сообщение" not used	Вы объявили, но не использовали переменную в процедуре или функции.
Hint: Value parameter "Сообщение" is assigned but never used	Идентификатор был объявлен (локально или глобально) и ему было присвоено значение, но после этого он не используется (локально или глобально).
Note: Local variable "Сообщение" is assigned but never used	Переменная в процедуре или функции объявлена и ей присвоено значение, но после этого она не используется.
Hint: Local Сообщ1 "Сообщ2" is not used	Локальный идентификатор не используется.

Сообщение	Описание
Note: Private field "Сообщ1.Сообщ2" is never used	Указанное приватное поле определено, но никогда не используется в коде.
Note: Private field " Сообщ1.Сообщ2" is assigned but never used	Указанное приватное поле определено и ему присвоено значение, но оно никогда не читается.
Note: Private method " Сообщ1.Сообщ2" never used	Указанный приватный метод определен, но никогда не используется в коде.
Error: Set type expected	Переменная или выражение не являются множеством. Это случается при использовании оператора **in**.
Warning: Function result does not seem to be set	Вы можете получить это предупреждение, если компилятор думает, что функция возвращает значение, которое не является множеством. Это не отображается для ассемблерных процедур или процедур, которые содержат ассемблерные блоки.
Warning: Type "Сообщение" is not aligned correctly in current record for C	Массивы C, размер которых не кратен 4, будут выровнены для C-структур.
Error: Unknown record field identifier "Сообщение"	Поле не существует в определении записи/объекта.
Warning: Local variable "Сообщение" does not seem to be initialized	Это сообщение отображается, если компилятор думает, что локальная переменная будет использована (то есть появится в правой части выражения) до того, как она была инициализирована (то есть до того, как она появилась в левой части оператора присваивания).
Warning: Variable "Сообщение" does not seem to be initialized	Это сообщение отображается, если компилятор думает, что переменная будет использована (то есть появится в правой части выражения) до того, как она была инициализирована (то есть до того, как она появилась в левой части оператора присваивания).
Error: identifier idents no member "Сообщение"	Эта ошибка генерируется, если с идентификатором записи, поля или метода выполняется присваивание, в то время как он не определён.
Hint: Found declaration: Сообщение	Вы получите это сообщение, если используете опцию **-vh**. В случае перегрузки процедуры, которая не была найдена. В сообщении будут перечислены все кандидаты перегружаемых процедур со списком их параметров.
Error: Data element too large	Вы получите это сообщение, если вы объявили элемент с данными, размер которых превышает принятый лимит (2 Гб для процессоров 80386+/68020+).
Error: No matching implementation for interface method "Сообщение" found	Найден метод, который не соответствует методу интерфейса. Проверьте типы аргументов и тип результата методов.
Warning: Symbol "Сообщение" is deprecated	Это означает, что используется идентификатор (переменная, процедура и т.п.), который объявлен **deprecated**. Идентификаторы **deprecated** могут уже не быть доступными в новых версиях модуля/библиотеки. Избегайте использовать эти идентификаторы, насколько это возможно.
Warning: Symbol "Сообщение" is not portable	Это означает, что используется идентификатор (переменная, процедура и т.п.), который объявлен **platform**. Значение этого идентификатора зависит от платформы и не должно использоваться, если исходный код должен быть переносимым.
Warning: Symbol "Сообщение" is not implemented	Это означает, что используется идентификатор (переменная, процедура и т.п.), который объявлен **unimplemented**. Этот идентификатора определён, но уже не выполняется на указанной платформе.
Error: Can't create unique type from this type	Только простые типы, такие как порядковые, вещественные и строки поддерживаются, если тип переопределён с помощью **type НовыйТип = type СтарыйТип;**.
Hint: Local variable "Сообщение" does not seem to be initialized	Это сообщение появляется, если компилятор думает, что локальная переменная будет использована (то есть появится в правой части выражения), в то время как она ещё не инициализирована (то есть не появлялась в левой части присваивания).
Hint: Variable "Сообщение" does not seem to be initialized	Это сообщение появляется, если компилятор думает, что переменная будет использована (то есть появится в правой части выражения), в то время как она ещё не инициализирована (то есть не появлялась в левой части присваивания).
Warning: Function result variable does not seem to initialized	Это сообщение появляется, если компилятор думает, что переменная результата функции будет использована (то есть появится в правой части выражения) перед её инициализацией (то есть перед тем, как она появится в левой части присваивания).
Hint: Function result variable does not seem to be initialized	Это сообщение появляется, если компилятор думает, что переменная результата функции будет использована (то есть появится в правой части выражения) перед её инициализацией (то есть перед тем, как она появится в левой части присваивания).

Сообщение	Описание
Warning: Variable "Сообщение" read but nowhere assigned	Вы прочитали значение переменной, но нигде не присвоили ей значение.
Hint: Found abstract method: Сообщение	Когда выводится предупреждение о создании класса/объекта с абстрактными методами, вы получаете эту подсказку, которая поможет вам найти неправильный метод.
Warning: Symbol "Сообщение" is experimental	Это означает, что используется идентификатор (переменная, процедура и т.п.), который объявлен как **experimental**. Экспериментальные идентификаторы могут быть удалены или изменят семантику в будущих версиях. Избегайте использовать эти идентификаторы, насколько это возможно.
Warning: Forward declaration "Сообщение" not resolved, assumed external	Это сообщение появляется, если вы объявили функцию в интерфейсе модуля в режиме **macpas**, не выполнили её.
Warning: Symbol "Сообщение" is belongs to a library	Это означает, что используется идентификатор (переменная, процедура и т.п.), который объявлен как **library**. Такие идентификаторы могут быть недоступны в других библиотеках.
Warning: Symbol "Сообщ1" is deprecated: "Сообщ2"	Это означает, что используется идентификатор (переменная, процедура и т.п.), который объявлен как **deprecated**. Такие идентификаторы могут быть недоступны в будущих версиях модуля/библиотеки. Избегайте использовать эти идентификаторы, насколько это возможно.
Error: Cannot find an enumerator for the type "Сообщение"	Это означает, что компилятор не может найти соответствующий нумератор для использования в цикле **for**. Для создания нумератора вам необходимо определить оператор нумератора или добавить метод **public** или **published GetEnumerator** для класса или объекта.
Error: Cannot find a "MoveNext" method in enumerator "Сообщение"	Это означает, что компилятор не может найти метод **public MoveNext** с возвращаемым типом **Boolean** в классе нумератора или объекте.
Error: Cannot find a "Current" property in enumerator "Сообщение"	Это означает, что компилятор не может найти свойство **public Current** с возвращаемым типом в классе нумератора или объекте.

C6. Сообщения генератора кода

В этом разделе описаны все сообщения, которые могут появиться, если генератор кода обнаружит ошибку кодирования.

Сообщение	Описание
Error: Parameter list size exceeds 65535 bytes	Для процессора I386 ограничен список параметров 65535 байтами (тому причиной инструкция RET).
Error: File types must be var parameters	Вы не можете указать файлы как значения параметров, то есть они всегда должны быть объявлены как параметры **var**.

Error: The use of a far pointer isn't allowed there	Free Pascal не поддерживает указатели **far**, поэтому вы не можете взять адрес выражения, которое имеет связь **far**. Конструкция **mem** имеет связь **far**, поэтому следующий код вызовет эту ошибку:
	
	`var p : pointer;`
	`...`
	`p:=@mem[a000:000];`

Error: EXPORT declared functions can't be called	Больше не используется.
Warning: Possible illegal call of constructor or destructor	Компилятор обнаружил, что конструктор или деструктор вызываются внутри метода. Это вероятно приведёт к проблемам, так как конструкторы/деструкторы требуют входных параметров.
Note: Inefficient code	Ваш оператор вызывает сомнения у компилятора.

Warning: unreachable code	Вы указали конструкцию, которая никогда не будет выполнена. Например:
	
	`while false do`
	`  begin`
	`    {.. Код ...}`
	`  end;`

Сообщение	Описание
Error: Abstract methods can't be called directly	Вы не можете вызвать абстрактный метод напрямую. Вместо этого вы должны перегружаемый дочерний метод, потому что абстрактный метод не выполняется.
Register Сообщ1 weight Сообщ2 Сообщ3	Отладочное сообщение. Отображается, если компилятор ожидает переменную для сохранения в регистрах.
Stack frame is omitted	Некоторые процедуры/функции не нуждаются в полном стековом фрейме, и им пренебрегают. Это сообщение появляется, если используется опция **–vd**.
Error: Object or class methods can't be inline.	Вы не можете иметь предопределённые методы объекта.
Error: Procvar calls cannot be inline.	Процедура с процедурной переменной не может быть предопределённой.
Error: No code for inline procedure stored	Компилятор не может записать код для предопределённой процедуры.
Error: Element zero of an ansi/wide- or longstring can't be accessed, use (set)length instead	Вы должны использовать **setlength** для установки длины для типов ansi/wide/longstring и **length**, чтобы получить длину такого строкового типа.
Error: Constructors or destructors cannot be called inside a 'with' clause	Внутри предложения **with** вы не можете вызвать конструктор или деструктор для объекта, который находится внутри этого предложения **with**.
Error: Cannot call message handler methods directly	Метод обработчика сообщений не может быть вызван напрямую, если он содержит явно аргумент **Self**.

Error: Jump in or outside of an exception block	Не допускается выполнять переход в или из блока обработки исключения, подобно этому: **try..finally..end**;. Например, следующий код приведёт к такой ошибке:
	`label 1;`
	`...`
	`try`
	`  if not(final) then`
	`  goto 1; // Эта строка вызовет ошибку`
	`finally`
	`  ...`
	`end;`
	`1:`
	`...`

Error: Control flow statements aren't allowed in a finally block	Не допускается использовать операторы управления потоком, такие как **break**, **continue** и **exit** внутри оператора **finally**. Следующий пример показывает проблему:
	`...`
	`try`
	`  p;`
	`finally`
	`...`
	`exit; // Этот exit недопустим`
	`end;`
	`...`
	Если в процедуре **p** произойдёт исключение, то будет выполнен блок **finally**. Если выполнение подойдёт к **exit**, то будет непонятно, что делать – выходить из процедуры или искать другой обработчик исключения.

Warning: Parameters size exceeds limit for certain cpu's	Это означает, что вы объявили больше чем 64К параметров, что не поддерживается данным целевым процессором.
Warning: Local variable size exceed limit for certain cpu's	Это означает, что вы объявили больше чем 32К локальных переменных, что не поддерживается данным целевым процессором.
Error: Local variables size exceeds supported limit	Это означает, что вы объявили больше чем 32К локальных переменных, что не поддерживается этим процессором.
Error: BREAK not allowed	Вы пытаетесь использовать **break** вне тела цикла.
Error: CONTINUE not allowed	Вы пытаетесь использовать **continue** вне тела цикла.
Fatal: Unknown compilerproc "Сообщение". Check if you use the correct run time library.	Компилятор ожидает, что библиотека времени исполнения содержит некие подпрограммы. Если вы видите эту ошибку и вы не изменяли код библиотеки времени исполнения, то весьма вероятно, что библиотека времени исполнения не согласуется с компилятором. Если вы изменяли библиотеку времени исполнения, то это означает, что вы удалили подпрограмму, которая требуется компилятору.

Сообщение	Описание
Fatal: Cannot find system type "Сообщение". Check if you use the correct run time library.	Компилятор ожидает, что библиотека времени исполнения содержит некие определения типов. Если вы видите эту ошибку и вы не изменяли код библиотеки времени исполнения, то весьма вероятно, что библиотека времени исполнения не согласуется с компилятором. Если вы изменяли библиотеку времени исполнения, то это означает, что вы удалили определение типа, который требуется компилятору.
Hint: Inherited call to abstract method ignored	Это сообщение появляется только в режиме **Delphi**, если вы вызываете абстрактный метод родительского класса через наследование (через **inherited**;). Такой вызов игнорируется.
Error: Goto label "Сообщение" not defined or optimized away	Метка, использованная в определении **goto**, не определена или находится в коде слишком далеко.

C7. Ошибки на стадии сборки/компоновки

В этом разделе представлен список ошибок, которые могут случиться, когда компилятор обрабатывает параметры командной строки или конфигурационные файлы.

Сообщение	Описание
Warning: Source operating system redefined	Исходная операционная система переопределена.
Info: Assembling (pipe) Сообщение	Сборка использует канал для внешнего ассемблера.
Error: Can't create assembler file: Сообщение	Указанный файл не может быть создан. Убедитесь, что у вас достаточно прав для создания этого файла.
Error: Can't create object file: Сообщение	Указанный файл не может быть создан. Убедитесь, что у вас достаточно прав для создания этого файла.
Error: Can't create archive file: Сообщение	Указанный файл не может быть создан. Убедитесь, что у вас достаточно прав для создания этого файла.
Error: Assembler Сообщение not found, switching to external assembling	Программа ассемблера не найдена. Компилятор создаст сценарий, который может быть использован для ассемблирования и компоновки программы.
Using assembler: Сообщение	Информационно сообщение, которое уведомляет о том, какой ассемблер будет использоваться.
Error: Error while assembling exitcode Сообщение	Произошла ошибка во время ассемблирования файла при использовании внешнего ассемблера. Подробную информацию об этой ошибке см. в документации на внешний ассемблер.
Error: Can't call the assembler, error Сообщение switching to external assembling	Ошибка произошла при вызове внешнего ассемблера. Компилятор создаст сценарий, который может быть использован для ассемблирования и компоновки программы.
Info: Assembling Сообщение	Информационное сообщение о том, что файл находится в процессе сборки.
Info: Assembling with smartlinking Сообщение	Информационное сообщение о том, что файл находится в процессе сборки с использованием «умной компоновки».
Warning: Object arg1 not found, Linking may fail !	Один из объектных файлов отсутствует, и компоновка, вероятно, завершилась неудачно. Проверьте пути.
Warning: Library arg1 not found, Linking may fail !	Один из библиотечных файлов отсутствует, и компоновка, вероятно, завершилась неудачно. Проверьте пути.
Error: Error while linking	Генерация ошибки во время компоновки.
Error: Can't call the linker, switching to external linking	Произошла ошибка при вызове внешнего компоновщика. Компилятор создаст сценарий, который может быть использован для ассемблирования и компоновки программы.
Info: Linking Сообщение	Информационное сообщение о том, что программа или библиотека находится в процессе компоновки.
Error: Util Сообщение not found, switching to external linking	Внешний инструмент не найден. Компилятор создаст сценарий, который может быть использован для ассемблирования и компоновки или выполнения заключительной обработки программы.
Using util Сообщение	Информационное сообщение о том, что внешняя программа (обычно постпроцессор) находится в процессе выполнения.
Error: Creation of Executables not supported	Создание исполняемых программ не поддерживается для этой платформы, поэтому она не выполняется в компиляторе.
Error: Creation of Dynamic/Shared Libraries not supported	Создание динамически загружаемых библиотек не поддерживается для этой платформы, поэтому она не выполняется в компиляторе.

Сообщение	Описание
Info: Closing script Сообщение	Информационное сообщение показывает, когда запись сценария внешней сборки и компоновки завершена.
Error: resource compiler "Сообщение" not found, switching to external mode	Внешний ресурс компилятора не был найден. Компилятор создаст сценарий, который может быть использован для ассемблирования, компиляции ресурсов и компоновки или выполнения заключительной обработки программы.
Info: Compiling resource Сообщение	Информационное сообщение о том, что выполняется компиляция ресурсов.
unit Сообщение can't be statically linked, switching to smart linking	Была запрошена статическая компоновка, но был использован модуль, который не допускает статической компоновки.
unit Сообщение can't be smart linked, switching to static linking	Была запрошена «умная» компоновка, но был использован модуль, который не допускает «умной» компоновки.
unit Сообщение can't be shared linked, switching to static linking	Была запрошена общая компоновка, но был использован модуль, который не допускает такой компоновки.
Error: unit Сообщение can't be smart or static linked	Была запрошена «умная» или статическая компоновка, но был использован модуль, который не допускает какой-либо из этих компоновок.
Error: unit Сообщение can't be shared or static linked	Была запрошена общая или статическая компоновка, но был использован модуль, который не допускает какой-либо из этих компоновок.
Calling resource compiler "Сообщ1" with "Сообщ2" as command line	Информационное сообщение о том, что командная строка использована для ресурсов компилятора.
Error: Error while compiling resources	Ресурс компилятора или конвертер вернул ошибку.
Error: Can't call the resource compiler "Сообщение", switching to external mode	Ошибка произошла при вызове ресурса компилятора. Компилятор создаст сценарий, который может быть использован для ассемблирования, компиляции ресурсов и компоновки или выполнения заключительной обработки программы.
Error: Can't open resource file "Сообщение"	Файл ресурсов не был открыт, произошла ошибка.
Error: Can't write resource file "Сообщение"	Файл ресурсов не был записан, произошла ошибка.

C8. Информационные сообщения программы

В этом разделе представлен список сообщений, которые компилятор генерирует, когда исполняемая программа создаётся, а только затем используется внутренний компоновщик.

Сообщение	Описание
Fatal: Can't post process executable Сообщение	Фатальная ошибка, когда компилятор неспособен выполнить заключительную обработку программы.
Fatal: Can't open executable Сообщение	Фатальная ошибка, когда компилятор не может открыть файл для выполнения.
Size of Code: Сообщение bytes	Информационное сообщение о размере созданного кода.
Size of initialized data: Сообщение bytes	Информационное сообщение о размере инициализированных данных.
Size of uninitialized data: Сообщение bytes	Информационное сообщение о размере освобождённых данных.
Stack space reserved: Сообщение bytes	Информационное сообщение о размере стека, который компилятор зарезервировал для программы.
Stack space committed: Сообщение bytes	Информационное сообщение о размере стека, который компилятор использовал для программы.

C9. Сообщения компоновщика

В этом разделе представлен список сообщений, которые генерирует внутренний компоновщик.

Сообщение	Описание
Fatal: Executable image size is too big for Сообщение target.	Фатальная ошибка, когда в результате компиляции получается слишком большая программа.
Warning: Object file "Сообщ1" contains 32-bit absolute relocation to symbol "Сообщ2".	Предупреждение о том, что 64-разрядный объектный файл содержит 32-разрядное рспределение памяти. В таком случае программа может быть загружена только в адресное пространство, ограниченное 4 ГБ.

C10. Сообщения загрузки модулей

В этом разделе представлен список сообщений, которые могут произойти, когда компилятор загружает модули с диска в память. Многие из этих сообщений являются чисто информационными.

Сообщение	Описание
Unitsearch: Сообщение	Если вы используете опцию **-vt**, то компилятор говорит вам, где он пытается искать файлы модулей.
PPU Loading Сообщение	Если используется опция **-vt**, то компилятор говорит вам, что он загружает модули.
PPU Name: Сообщение	Если используется флаг **-vu**, то отображаются имена модулей.
PPU Flags: Сообщение	Если используется флаг **-vu**, то отображаются флаги модулей.
PPU Crc: Сообщение	Если используется флаг **-vu**, то отображается проверка CRC (контрольных сумм) модулей.
PPU Time: Сообщение	Если используется флаг **-vu**, то отображается время компиляции модуля.
PPU File too short	Файл **ppu** слишком короткий, не все объявления представлены.
PPU Invalid Header (no PPU at the begin)	Файл модуля содержит как первые три байта ASCII-коды символов PPU.
PPU Invalid Version Сообщение	Этот файл модуля был откомпилирован с другой версией компилятора и не может быть прочитан.
PPU is compiled for another processor	Этот файл модуля был откомпилирован для другого типа процессора и не может быть прочитан.
PPU is compiled for an other target	Этот файл модуля был откомпилирован для другой целевой платформы и не может быть прочитан.
PPU Source: Сообщение	Если используется флаг **-vu**, то отображается имя исходного файла модуля.
Writing Сообщение	Если используется флаг **-vu**, то компилятор будет сообщать вам, куда он записывает файл модуля.
Fatal: Can't Write PPU-File	Произошла ошибка при записи файла модуля.
Fatal: Error reading PPU-File	Это означает, что файл был повреждён и содержит неправильную информацию. Необходима перекомпиляция.
Fatal: unexpected end of PPU-File	Неожиданный конец файла. Это означает, что файл PPU повреждён.
Fatal: Invalid PPU-File entry: Сообщение	Модуль, который пытается прочитать компилятор, повреждён или создан с более новой версией компилятора.
Fatal: PPU Dbx count problem	Несовместимая отладочная информация в модуле.
Error: Illegal unit name: Сообщение	Имя модуля не соответствует имени файла.
Fatal: Too much units	Free Pascal имеет ограничение до 1024 модулей в программе. Вы можете изменить это ограничение с помощью константы **maxunits** в файле **fmodule.pas** компилятора, а затем перекомпилировать компилятор.
Fatal: Circular unit reference between Сообщ1 and Сообщ2	Два модуля используют друг друга в интерфейсной части. Это допускается только в разделе **implementation**. Как минимум один из двух модулей должен подключаться к программе в разделе **implementation**.
Fatal: Can't compile unit Сообщение, no sources available	Был найден модуль, который требуется компилировать, но его исходные коды недоступны.
Fatal: Can't find unit Сообщ1 used by Сообщ2	Вы пытаетесь использовать модуль, для которого не найден файл PPU. Проверьте ваш конфигурационный файл на наличие правильных путей для этого модуля.
Warning: Unit Сообщ1 was not found but Сообщ2 exists	Это сообщение об ошибке больше не используется.
Fatal: Unit Сообщ1 searched but Сообщ2 found	В DOS обрезание имени файла модуля PPU до 8 символов может вызвать проблемы, если имя модуля более 8 символов.

Сообщение	Описание
Warning: Compiling the system unit requires the -Us switch	Если перекомпилируется системный модуль (это необходимо в некоторых специфических случаях), то должен быть указан переключатель **–Us**.
Fatal: There were Сообщение errors compiling module, stopping	Если компилятор обнаруживает фатальную ошибку или слишком много ошибок в модуле, то он останавливается с этим сообщением.
Load from Сообщ1 (Сообщ2) unit Сообщ3	Если вы используете флаг **–vu**, то в этом сообщении отображается, какой модуль загружается из какого модуля.
Recompiling Сообщ1, checksum changed for Сообщ2	Модуль перекомпилирован, потому что контрольные суммы указывают на то, что модуль был изменён.
Recompiling Сообщение, source found only	Если вы используете флаг **–vu**, то это сообщение говорит вам, почему текущий модуль перекомпилирован.
Recompiling unit, static lib is older than ppufile	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, если статическая библиотека модуля старше, чем сам файл модуля.
Recompiling unit, shared lib is older than ppufile	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, если общая библиотека модуля старше, чем сам файл модуля.
Recompiling unit, obj and asm are older than ppufile	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, если ассемблер или объектный файл модуля старше, чем сам файл модуля.
Recompiling unit, obj is older than asm	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, если ассемблерный файл модуля старше, чем объектный файл модуля.
Parsing interface of Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он начал синтаксический анализ интерфейсной части модуля.
Parsing implementation of Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он начал синтаксический анализ раздела **implementation** модуля.
Second load for unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он начал перекомпиляцию модуля повторно. Это может случиться при компиляции взаимосвязанных модулей.
PPU Check file Сообщ1 time Сообщ2	Если вы используете флаг **–vu**, то компилятор отображает имя файла, дату и время файла, когда произошла перекомпиляция.
Warning: Can't recompile unit Сообщение, but found modifed include files	В модуле были найдены изменения, связанные с подключаемыми файлами, но некоторые исходные коды не найдены. Перекомпиляция невозможна.
File Сообщ1 is newer than PPU file Сообщ2	Был найден изменённый исходный файл для модуля компилятора.
Trying to use a unit which was compiled with a different FPU mode	Попытка компиляции кода, когда модули используют не одинаковый формат плавающей точки. Для всех модулей эмуляция FPU должна быть либо включена, либо выключена.
Loading interface units from Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он начал загружать модули, объявленные в интерфейсной части модуля.
Loading implementation units from Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он начал загружать модули, объявленные в разделе **implementation** модуля.
Interface CRC changed for unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что контрольные суммы, вычисленные для интерфейсной части модуля, были изменены после синтаксического анализа раздела **implementation**.
Implementation CRC changed for unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что вычисленные контрольные суммы были изменены после синтаксического анализа раздела **implementation**.
Finished compiling unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он завершил компиляцию модуля.
Add dependency of Сообщ1 to Сообщ2	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он добавил связь между двумя модулями.
No reload, is caller: Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он не будет перегружать модуль, потому что это модуль, который хочет загрузить этот модуль.
No reload, already in second compile: Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он не будет перегружать модуль, потому что он уже выполняет повторную перекомпиляцию.

Сообщение	Описание
Flag for reload: Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он не будет перегружать модуль.
Forced reloading	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он перегружает модуль, потому что это было затребовано.
Previous state of Сообщ1: Сообщ2	Если вы используете флаг **–vu**, то компилятор показывает предыдущее состояние модуля.
Already compiling Сообщение, setting second compile	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он начал перекомпиляцию модуля второй раз. Это может случиться с взаимосвязанными модулями.
Loading unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он начал загрузку модуля.
Finished loading unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он закончил загрузку модуля.
Registering new unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он нашёл новый модуль и зарегистрировал его во внутреннем списке.
Re-resolving unit Сообщение	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он пересчитывает внутренние данные модуля.
Skipping re-resolving unit Сообщение, still loading used units	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он пропустил пересчет внутренних данных модуля, потому что эти данные не предназначены для пересчёта.
Unloading resource unit Сообщение (not needed)	Если вы используете флаг **–vu**, то компилятор выдаёт предупреждение, что он выгрузил модуль обработки ресурсов, так как не нашёл используемых ресурсов.
Error: Unit Сообщ1 was compiled using a different whole program optimization feedback input (Сообщ2, Сообщ3); recompile it	Если модуль был откомпилирован с использованием редко используемой программной оптимизации – whole program optimization (wpo) с файлом обратной связи (-FW<x> -OW<x>), то скомпилированная версия модуля является специфической для этой компиляции и не может быть использована в другом контексте. Это не рекомендуется делать, если вы можете использовать модуль в других программах или с другими выходными файлами wpo.

C11. Ошибки обработки командной строки

В этом разделе представлен список сообщений, которые могут произойти, когда компилятор находится в процессе обработки командной строки или конфигурационных файлов.

Сообщение	Описание
Warning: Only one source file supported, changing source file to compile from "Сообщ1" into "Сообщ2"	Вы можете указать только один исходный файл в командной строке. Если вы укажете несколько файлов, то компилироваться будет только последний, остальные будут игнорироваться. Это сообщение может означать, что вы забыли знак '-'.
Warning: DEF file can be created only for OS/2	Эту опцию можно использовать только тогда, когда вы компилируете для OS/2.
Error: nested response files are not supported	Вы не можете вложить зависимости файлов с опцией командной строки @file.
Fatal: No source file name in command line	Компилятор ожидал имя исходного файла в командной строке.
Note: No option inside Сообщение config file	Компилятор не нашёл никаких опций в этом конфигурационном файле.
Error: Illegal parameter: Сообщение	Вы указали неизвестную опцию.
Hint: -? writes help pages	Если получена неизвестная опция, это сообщение появится.
Fatal: Too many config files nested	Вы можете вложить не более 16 конфигурационных файлов.
Fatal: Unable to open file Сообщение	Файл не найден.
Reading further options from Сообщение	Отображается, если вы включили примечания, а компилятор переключился на другой файл опций.
Warning: Target is already set to: Сообщение	Отображается, если указано более одной опции **–T**.
Warning: Shared libs not supported on DOS platform, reverting to static	Если вы указали -CD для платформы DOS, это сообщение появится. Компилятор поддерживает только статические библиотеки под DOS.
Fatal: In options file Сообщ1 at line Сообщ2 too many nvar{n#IF(N)DEFs} encountered	Операторы #IF(N)DEF в файле опций не сбалансированы с операторами #ENDIF.
Fatal: In options file Сообщ1 at line Сообщ2 unexpected nvar{n#ENDIFs} encountered	Операторы #IF(N)DEF в файле опций не сбалансированы с операторами #ENDIF.

Сообщение	Описание
Fatal: Open conditional at the end of the options file	Операторы #IF(N)DEF в файле опций не сбалансированы с операторами #ENDIF.
Warning: Debug information generation is not supported by this executable **Hint: Try recompiling with -dGDB**	Возможно, что компилятор имеет исполняемый файл, который не поддерживает генерацию отладочной информации. Если вы используете такой исполняемый файл с опцией **-g**, то это сообщение будет появляться.
Warning: You are using the obsolete switch Сообщение	Это говорит о том, что вы используете опцию, которая не нужна или больше не поддерживается. Рекомендуется удалить эту опцию, чтобы избежать проблем в будущем, когда назначение опции может измениться.
Warning: You are using the obsolete switch Сообщ1, please use Сообщ2	Это говорит о том, что вы используете опцию, которая больше не поддерживается. Сейчас вы должны использовать вторую в сообщении опцию вместо первой. Рекомендуется заменить эту опцию, чтобы избежать проблем в будущем, когда назначение опции может измениться.
Note: Switching assembler to default source writing assembler	Предупреждение о том, что ассемблер был изменён, потому что вы использовали опцию **-a**, которая не может быть использована с бинарной записью ассемблера.
Warning: Assembler output selected "Сообщ1" is not compatible with "Сообщ2"	Выбранный выходной ассемблер "Сообщ1" не совместим с "Сообщ2".
Warning: "Сообщение" assembler use forced	Выбранный выходной ассемблер не может генерировать объектные файлы в правильном формате. По этой причине вместо выбранного использован ассемблер по умолчанию для данной целевой платформы.
Reading options from file Сообщение	Опции также читаются из этого файла.
Reading options from environment Сообщение	Опции также читаются из этой строки окружения.
Handling option "Сообщение"	Найдена дополнительная отладочная информация, которая будет обрабатываться.
***** press enter *****	Сообщение появляется, когда справочные материалы отображаются постранично. Если нажать клавишу ENTER, то появится следующая страница справки. Если вы нажмёте **q**, а затем ENTER, то выполнится выход из компилятора.
Hint: Start of reading config file Сообщение	Начат анализ конфигурационного файла.
Hint: End of reading config file Сообщение	Завершён анализ конфигурационного файла.
interpreting option "Сообщение"	Компилятор интерпретировал опцию.
interpreting firstpass option "Сообщение"	Компилятор интерпретировал опцию первый раз.
interpreting file option "Сообщение"	Компилятор интерпретировал опцию, которую он прочитал из конфигурационного файла.
Reading config file "Сообщение"	Компилятор начал читать конфигурационный файл.
found source file name "Сообщение"	Дополнительная информация об опциях. Отображается, если включена опция отладки.
Error: Unknown code page	Неизвестная кодовая страница для исходных файлов была запрошена. Компилятор поддерживает несколько встроенных кодовых страниц. Затребованная кодовая страница не входит в этот список. Вам придётся перекомпилировать компилятор с поддержкой нужной вам кодовой страницы.
Fatal: Config file Сообщение is a directory	Директории не могут использоваться как конфигурационные файлы.
Warning: Assembler output selected "Сообщение" cannot generate debug info, debugging disabled	Выбранный выходной ассемблер не может генерировать отладочную информацию, поэтому опции отладки отключены.
Warning: Use of ppc386.cfg is deprecated, please use fpc.cfg instead	Использование **ppc386.cfg** пока поддерживается для совместимости, однако, для мультиплатформных систем использовать его не рекомендуется. Пожалуйста, используйте вместо него файл **fpc.cfg**.
Fatal: In options file Сообщ1 at line Сообщ2 nvar{n#ELSE} directive without nvar{n#IF(N)DEF} found	Оператор #ELSE был найден в файле опций без соответствующего оператора #IF(N)DEF.
Fatal: Option "Сообщ1" is not, or not yet, supported on the current target platform	Не все опции поддерживаются или выполняются для всех целевых платформ. Это сообщение информирует о том, что выбранная опция несовместима с текущей выбранной целевой платформой.

Сообщение	Описание
Fatal: The feature "Сообщение" is not, or not yet, supported on the selected target platform	Не все функции поддерживаются или выполняются для всех целевых платформ. Это сообщение информирует о том, что выбранная функция несовместима с текущей выбранной целевой платформой.
Note: DWARF debug information cannot be used with smart linking on this target, switching to static linking	«Умная» компоновка в данный момент несовместима с отладочной информацией DWARF на большинстве платформ, поэтому в таких случаях «умная» компоновка отключается.
Warning: Option "Сообщение" is ignored for the current target platform.	Не все опции поддерживаются или выполняются для всех целевых платформ. Это сообщение информирует о том, что выбранная опция игнорируется для текущей выбранной целевой платформой.

C12. Сообщения программной оптимизации

В этом разделе представлен список ошибок, которые могут произойти, когда компилятор выполняет полную программную оптимизацию.

Сообщение	Описание
Fatal: Cannot open whole program optimization feedback file "Сообщение"	Компилятор не может открыть файл обратной связи с информацией программной оптимизации.
Processing whole program optimization information in wpo feedback file "Сообщение"	Компилятор начал процесс программной оптимизации с информацией, найденной в указанном файле.
Finished processing the whole program optimization information in wpo feedback file "Сообщение"	Компилятор завершил процесс программной оптимизации с информацией, найденной в указанном файле.
Error: Expected section header, but got "Сообщ2" at line Сообщ1 of wpo feedback file	Компилятор обнаружил заголовочный раздел в файле программной оптимизации (начиная с %), но не может найти её.
Warning: No handler registered for whole program optimization section "Сообщ2" at line Сообщ1 of wpo feedback file, ignoring	Компилятор не имеет обработчика для указанного раздела, поэтому этот раздел будет пропущен и выполнен переход к следующему разделу.
Found whole program optimization section "Сообщ1" with information about "Сообщ2"	Компилятор обнаружил раздел с информацией программной оптимизации, и согласно его обработчика раздел содержит информацию, пригодную для указанных целей.
Fatal: The selected whole program optimizations require a previously generated feedback file (use -Fw to specify)	Компилятору необходима информация, собранная во время предыдущей компиляции и запуска на выполнение выбранной программной оптимизации. Вы можете указать размещение файла обратной связи, содержащего эту информацию, используя опцию **-Fw**.
Error: No collected information necessary to perform "Сообщение" whole program optimization found	Вы указали компилятору файл с программной оптимизацией, но этот файл не содержит информацию, необходимую для выполнения указанной оптимизации. Вы можете перекомпилировать программу, используя опцию -OWxxx.
Fatal: Specify a whole program optimization feedback file to store the generated info in (using -FW)	Вы указали файл обратной связи, в который компилятор будет записывать программную оптимизацию, которая сгенерирована во время запуска компиляции. Это невозможно с использованием опции **-FW**.
Error: Not generating any whole program optimization information, yet a feedback file was specified (using -FW)	Компилятору была дана команда записывать программную оптимизацию в файл, указанный опцией **-FW**, но нет актуальной информации для файла. Классы генерируемой информации можно указать с помощью опции **-OWxxx**.
Error: Not performing any whole program optimizations, yet an input feedback file was specified (using -Fw)	Компилятору была дана команда выполнить все оптимизации (нет параметров **-Owxxx**), но тем не менее входной файл был указан (опцией **-Fwyyy**). Это может означать, что вы забыли указать параметр для **-Owxxx**, компилятор генерирует ошибку в таком случае.
Skipping whole program optimization section "Сообщение", because not needed by the requested optimizations	Файл программной оптимизации содержит раздел с информацией, которая не нужна для выбранной оптимизации.

Сообщение	Описание
Warning: Overriding previously read information for "Сообщ1" from feedback input file using information in section "Сообщ2"	Файл обратной связи содержит множество разделов, которые предоставляют одинаковые классы информации (то есть информация о каждом виртуальном методе может быть девиртуализирована). В этом случае используется информация из последнего найденного раздела. Включите отладку (**-vd**) чтобы посмотреть, какой класс информации предоставляет каждый раздел.
Error: Cannot extract symbol liveness information from program when stripping symbols, use -Xs-	На определённом идентификаторе не удалось извлечь символьную информацию из скомпонованной программы. Если символьная информация обрезана (опция **-Xs**), то это невозможно.
Error: Cannot extract symbol liveness information from program when when not linking	На определённом идентификаторе не удалось извлечь символьную информацию из скомпонованной программы. Если программа скомпонована не с этим компилятором, то это невозможно.
Fatal: Cannot find "Сообщ1" or "Сообщ2" to extract symbol liveness information from linked program	Определённый идентификатор нуждается во вспомогательной программе для извлечения символьной информации из скомпонованной программы. Обычно вспомогательная программа это **nm**, которая является частью утилит GNU.
Error: Error during reading symbol liveness information produced by "Сообщение"	Произошла ошибка во время чтения символьного файла, который был сгенерирован программой **nm** или **objdump**. Возможно, что идентификатор короче, чем ожидалось, или его формат был не понят.
Fatal: Error executing "Сообщ1" (exitcode: Сообщ2) to extract symbol information from linked program	Определённый идентификатор нуждается во вспомогательной программе для извлечения символьной информации из скомпонованной программы. Вспомогательная программа код ошибки, если она была запущена при компоновке программы.
Error: Collection of symbol liveness information can only help when using smart linking, use -CX -XX	Действующий идентификатор определён в области видимости в конце компоновки программы. Без «умной» компоновки/потери обрезанного кода, все идентификаторы всегда включены, несмотря на то, используются они или нет. Так что в этом случае все идентификаторы будут видны как действующие, что делает эту оптимизацию неэффективной.
Error: Cannot create specified whole program optimisation feedback file "Сообщение"	Компилятор не может создать файл, указанный в опции **-FW** для записи информации программной оптимизации.

C13. Ошибки ассемблера

В этом разделе представлен список ошибок, которые генерируются встроенным ассемблером. Они НЕ являются сообщениями самого ассемблера.

C13.1. Основные ошибки ассемблера

Сообщение	Описание
Divide by zero in asm evaluator	Эта фатальная ошибка случается, если ассемблерное выражение выполняет деление на ноль.
Evaluator stack overflow, Evaluator stack underflow	Эта фатальная ошибка случается, если ассемблерное выражение слишком большое, чтобы вычислить его. Попытайтесь уменьшить количество условий.
Invalid numeric format in asm evaluator	Эта фатальная ошибка случается, если не числовое значение обнаружено анализатором синтаксиса. В обычной ситуации эта ошибка никогда не случается.
Invalid Operator in asm evaluator	Эта фатальная ошибка случается, если математический оператор обнаружен анализатором синтаксиса. В обычной ситуации эта ошибка никогда не случается.
Unknown error in asm evaluator	Эта фатальная ошибка случается, если внутренняя ошибка обнаружена анализатором синтаксиса. В обычной ситуации эта ошибка никогда не случается.
Invalid numeric value	Это предупреждение появляется, если результат преобразования из восьмеричного, двоичного или шестнадцатеричного представления в десятичное выходит за пределы поддерживаемого диапазона.
Escape sequence ignored	Эта ошибка появляется, если в строке C обнаружена последовательность не ANSI C.

Сообщение	Описание
Asm syntax error - Prefix not found	Это случается при попытке использовать неправильный префикс команды.
Asm syntax error - Trying to add more than one prefix	Это случается при попытке добавить в инструкции более одного префикса.
Asm syntax error - Opcode not found	Вы пытаетесь использовать неподдерживаемый или неизвестный код операции.
Constant value out of bounds	Эта ошибка появляется, если анализатор синтаксиса определил, что используемое вами значение находится вне границ, либо используется с кодом операции, либо с объявлением константы.
Non-label pattern contains @	Это применяется только для ассемблеров стиля **m68k** и **Intel**. Сообщение появляется, если вы пытаетесь использовать идентификатор (НЕ метку) с префиксом @.
Internal error in Findtype()	Внутренняя ошибка в Findtype().
Internal Error in ConcatOpcode()	Внутренняя ошибка в ConcatOpcode().
Internal Errror converting binary	Внутренняя ошибка двоичного преобразования.
Internal Errror converting hexadecimal	Внутренняя ошибка шестнадцатеричного преобразования.
Internal Errror converting octal	Внутренняя ошибка восьмеричного преобразования.
Internal Error in BuildScaling()	Внутренняя ошибка в BuildScaling().
Internal Error in BuildConstant()	Внутренняя ошибка в BuildConstant().
internal error in BuildReference()	Внутренняя ошибка в BuildReference().
internal error in HandleExtend()	Внутренняя ошибка в HandleExtend().
Internal error in ConcatLabeledInstr()	Эта ошибка не должна никогда случиться. Если вы её увидели, то это новый баг в анализаторе ассемблера. Пожалуйста, контактируйте с разработчиками.
Opcode not in table, operands not checked	Это предупреждение появляется только при компиляции системного модуля или связанных с ним файлов. Не проверяйте его выполнение на операндах и кодах операций.
@CODE and @DATA not supported	Эти конструкции Turbo Pascal не поддерживаются.
SEG and OFFSET not supported	Эти конструкции Turbo Pascal не поддерживаются.
Modulo not supported	Операции с константами по модулю не поддерживаются.
Floating point binary representation ignored	Двоичное представление плавающей точки игнорируется.
Floating point hexadecimal representation ignored	Шестнадцатеричное представление плавающей точки игнорируется.
Floating point octal representation ignored	Восьмеричное представление плавающей точки игнорируется. Это предупреждения появляется, если константа с плавающей точкой объявлена в системе счисления, отличной от десятичной. Преобразование этих форматов невозможно. Вы должны использовать десятичное представление чисел с плавающей точкой.
Identifier supposed external	Это предупреждение появляется, если идентификатор не найден в таблице символов.
Functions with void return value can't return any value in asm code	Только процедуры с возвращаемым значением могут возвращать набор значений.
Error in binary constant	Ошибка в двоичной константе.
Error in octal constant	Ошибка в восьмеричной константе.
Error in hexadecimal constant	Ошибка в шестнадцатеричной константе.
Error in integer constant	Ошибка в целочисленной константе. Эта ошибка случается, если вы пытаетесь использовать выражение с константами, которое является неправильным или его значение за пределами диапазона.
Invalid labeled opcode	Неправильный отмеченный код операции.
Asm syntax error - error in reference	Ошибка синтаксиса.
Invalid Opcode	Неправильный код операции.
Invalid combination of opcode and operands	Неправильная комбинация кода операции или операндов.
Invalid size in reference	Неправильный размер в ссылке.
Invalid middle sized operand	Средний операнд неправильного размера.
Invalid three operand opcode	Неправильный третий операнд кода операции.
Assembler syntax error	Ошибка синтаксиса.
Invalid operand type	Вы пытаетесь использовать неправильную комбинацию кода операции или операндов. Проверьте синтаксис и если вы уверены, что всё правильно, обратитесь к разработчику.
Unknown identifier	Идентификатор, к которому вы пытались получить доступ, не существует, или находится за пределами текущей области видимости.
Trying to define an index register more than once	Попытка определить более одного индексного регистра.

Сообщение	Описание
Trying to define a segment register twice	Попытка определить сегментный регистр дважды.
Trying to define a base register twice	Вы пытаетесь определить регистр базы дважды.
Invalid field specifier	Поле записи или объекта, к которому вы пытаетесь получить доступ, не существует, или неправильное.
Invalid scaling factor	Неправильный масштабный коэффициент.
Invalid scaling value	Неправильное масштабное значение.
Scaling value only allowed with index	Допустимыми масштабными значениями являются 1,2,4 или 8.
Cannot use SELF outside a method	Вы пытаетесь получить доступ к идентификатору SELF объекта за пределами метода.
Invalid combination of prefix and opcode	Код операции не может иметь префикс в этой команде.
Invalid combination of override and opcode	Этот код операции не может быть перегружен в этой комбинации.
Too many operands on line	Более трёх операндов в одной инструкции на m68k или i386. Вероятно, вы пытаетесь использовать неправильный синтаксис для этого кода операции.
Duplicate local symbol	Вы пытаетесь переопределить локальный идентификатор, такой как локальная метка.
Unknown label identifer	Нет такой метки.
Undefined local symbol	Неизвестный локальный идентификатор.
local symbol not found inside asm statement	Эта метка не определена в текущей области видимости.
Assemble node syntax error	Ошибка синтаксиса вложений.
Not a directive or local symbol	Оператор ассемблера неправильный, или вы не используете директиву распознавания.

C13.2. Ошибки I386

Сообщение	Описание
repeat prefix and a segment override on <= i386 ...	Проблема с прерываниями и префиксами команд могут случиться и вернуть ложный результат на процессорах 386 и более младших.
Fwait can cause emulation problems with emu387	Это предупреждение появляется, если используется инструкция FWAIT. Это может вызвать проблемы на системах, которые используют эмулятор em387.dxe.
You need GNU as version >= 2.81 to compile this MMX code	Код ассемблера MMX может компилироваться только с использованием GAS v2.8.1 или выше.
NEAR ignored	NEAR игнорируется.
FAR ignored	NEAR и FAR игнорируются в ассемблерах Intel, но поддерживаются для совместимости с 16-разрядными модулями.
Invalid size for MOVSX/MOVZX	Неправильный размер для MOVSX/MOVZX.
16-bit base in 32-bit segment	16-разрядная база в 32-разрядном сегменте.
16-bit index in 32-bit segment	16-разрядная адресация не поддерживается. Вы должны использовать 32-разрядную адресацию.
Constant reference not allowed	Не допускаются попытки адресации постоянной памяти в защищённом режиме.
Segment overrides not supported	Стиль Intel (например: rep ds stosb) перегрузки сегмента не поддерживается анализатором ассемблера.
Expressions of the form [sreg:reg...] are currently not supported	Для доступа к памяти в разных сегментах вы должны использовать синтаксис **sreg:[reg...]** вместо **[sreg:reg...]**.
Size suffix and destination register do not match	В синтаксисе intel AT&T, вы используете регистр, размер которого не согласуется с размером указанного операнда.
Invalid assembler syntax. No ref with brackets	Неправильный синтаксис ассемблера. Ссылка не заключена в скобки.
Trying to use a negative index register	Попытка использовать отрицательный регистр индекса.
Local symbols not allowed as references	Не допускается использовать локальный идентификатор как ссылку.
Invalid operand in bracket expression	Неправильный операнд в выражении, заключённом в скобки.
Invalid symbol name:	Неправильное имя идентификатора.
Invalid Reference syntax	Неправильный синтаксис ссылки.
Invalid string as opcode operand:	Неправильная строка как операнд кода операции.
Null label references are not allowed	Ссылка на пустую метку не допускается.

Сообщение	Описание
Using a defined name as a local label	Использование определённого имени как локальной метки.
Invalid constant symbol	Неправильный идентификатор константы.
Invalid constant expression	Неправильное выражение с константой.
/ at beginning of line not allowed	Не допускается начинать строку с символа /.
NOR not supported	NOR не поддерживается.
Invalid floating point register name	Неправильное имя регистра плавающей точки.
Invalid floating point constant:	Неправильная константа с плавающей точкой.
Asm syntax error - Should start with bracket	Ошибка синтаксиса – должно начинаться со скобки.
Asm syntax error - register:	Ошибка синтаксиса – регистр:
Asm syntax error - in opcode operand	Ошибка синтаксиса – в операнде кода операции.
Invalid String expression	Неправильное строковое выражение.
Constant expression out of bounds	Выражение с константой за пределами диапазона.
Invalid or missing opcode	Неправильный или отсутствующий код операции.
Invalid real constant expression	Неправильное выражение с вещественной константой.
Parenthesis are not allowed	Круглые скобки являются недопустимыми.
Invalid Reference	Неправильная ссылка.
Cannot use __SELF outside a method	Нельзя использовать __SELF вне метода.
Cannot use __OLDEBP outside a nested procedure	Нельзя использовать __OLDEBP вне вложенной процедуры.
Invalid segment override expression	Неправильное выражение перегрузки сегмента.
Strings not allowed as constants	Строки не допускаются как константы.
Switching sections is not allowed in an assembler block	Раздел переключателей не допускается в ассемблерном блоке.
Invalid global definition	Неправильное глобальное определение.
Line separator expected	Ожидается разделитель строк.
Invalid local common definition	Неправильное общее локальное определение.
Invalid global common definition	Неправильное общее глобальное определение.
assembler code not returned to text	Код ассемблера не возвращает текст.
invalid opcode size	Неправильный размер кода операции.
Invalid character: <	Неправильный символ: <
Invalid character: >	Неправильный символ: >
Unsupported opcode	Неподдерживаемый код операции.
Invalid suffix for intel assembler	Неправильный суффикс для ассемблера intel.
Extended not supported in this mode	**Extended** не поддерживается в этом режиме.
Comp not supported in this mode	**Comp** не поддерживается в этом режиме.
Invalid Operand:	Неправильный операнд.
Override operator not supported	Перегруженный оператор не поддерживается.

C13.3. Ошибки m68k

Сообщение	Описание
Increment and Decrement mode not allowed together	Вы пытаетесь использовать режимы dec/inc вместе.
Invalid Register list in movem/fmovem	Список регистров неправильный. Обычно диапазон регистров должен быть разделён знаком -, а каждый отдельный регистр отделён слешем.
Invalid Register list for opcode 68020+ mode required to assemble	Неправильный список регистров для кода операции режима 68020+, требуемого для ассемблера.

ПРИЛОЖЕНИЕ D. Ошибки времени выполнения.

Приложения, генерируемые при помощи Free Pascal, могут генерировать ошибки времени выполнения программы, если в программе обнаружено какое-либо ненормальное состояние. Ниже приведён список возможных ошибок времени выполнения и информация о причинах их возникновения.

Код	Ошибка	Описание
1	**Invalid function number**	Неправильный номер функции. Была попытка выполнить неправильный вызов операционной системы.
2	**File not found**	Файл не найден. Появляется при попытке удалить, переименовать или открыть несуществующий файл.
3	**Path not found**	Путь не найден. Возникает при обработке директорий, если путь не существует или неправильный. Также возникает при попытке доступа к несуществующему файлу.
4	**Too many open files**	Слишком много открытых файлов. Количество файлов, открытых в данный момент вашим процессом, превысило максимально допустимое. Некоторые операционные системы ограничивают количество файлов, которые могут быть открыты одновременно, и эта ошибка может случиться, если это ограничение было превышено.
5	**File access denied**	Отказано в доступе к файлу. Эта ошибка может произойти по одной из следующих причин: • Попытка открыть для записи файл с атрибутом «только чтение», или который находится в каталоге с атрибутом «только чтение». • Файл в текущий момент блокирован или используется другим процессом. • Попытка создать новый файл или директорию, в то время как файл или директория с таким именем уже существует. • Попытка прочитать файл, который был открыт в режиме «только запись». • Попытка записи в файл, который был открыт в режиме «только чтение». • Попытка удаления файла или директории, в то время как это невозможно. • Недостаточно прав для доступа к файлу или директории.
6	**Invalid file handle**	Неправильный дескриптор файла. Если появилось это сообщение, то используемая вами файловая переменная является «мусором». Это говорит о том, что ваша память содержит недостоверные данные.
12	**Invalid file access code**	Неправильный код доступа к файлу. Появляется, если **reset** или **rewrite** вызываются с неправильным значением **FileMode**.
15	**Invalid drive number**	Неправильный номер диска. Номер, полученный для функций Getdir или ChDir, указывает на несуществующий диск.
16	**Cannot remove current directory**	Невозможно удалить текущую директорию. Случается при попытке удалить текущую активную директорию.
17	**Cannot rename across drives**	Невозможно переименовать на разных дисках. Вы не можете переименовать файл, так как он заканчивается на другом диске или разделе диска.
100	**Disk read error**	Ошибка чтения диска. Ошибка происходит при чтении с диска. Обычно случается, если вы пытаетесь прочитать самый конец файла.
101	**Disk write error**	Ошибка записи диска. Случается, если диск заполнен, а вы пытаетесь записать на него данные.
102	**File not assigned**	Файл не назначен. Это происходит при вызове **Reset**, **Rewrite**, **Append**, **Rename** и **Erase**, если вы вызвали их с параметром, которому не назначен файл.
103	**File not open**	Файл не открыт. Случается при вызове следующих функций: **Close**, **Read**, **Write**, **Seek**, **EOf**, **FilePos**, **FileSize**, **Flush**, **BlockRead** и **BlockWrite**, если файл не открыт.
104	**File not open for input**	Файл не открыт для ввода. Случается при вызове следующих функций: **Read**, **BlockRead**, **Eof**, **Eoln**, **SeekEof** или **SeekEoln**, если файл не открыт с помощью **Reset**.
105	**File not open for output**	Файл не открыт для вывода. Случается при записи, если текстовый файл не открыт с помощью **Rewrite**.
106	**Invalid numeric format**	Неправильный числовой формат. Случается, если НЕ числовое значение читается из текстового файла, а ожидается числовое значение.
150	**Disk is write-protected (Critical error)**	Диск защищён от записи (критическая ошибка).
151	**Bad drive request struct length (Critical error)**	Неисправный привод (критическая ошибка).

Код	Ошибка	Описание
152	**Drive not ready (Critical error)**	Диск не читается (критическая ошибка).
154	**CRC error in data (Critical error)**	Ошибка контрольных сумм в данных (критическая ошибка).
156	**Disk seek error (Critical error)**	Ошибка поиска по диску (критическая ошибка).
157	**Unknown media type (Critical error)**	Неизвестный тип медиа-привода (критическая ошибка).
158	**Sector Not Found (Critical error)**	Сектор не найден (критическая ошибка).
159	**Printer out of paper (Critical error)**	В принтере нет бумаги (критическая ошибка).
160	**Device write fault (Critical error)**	Ошибка записи на диск (критическая ошибка).
161	**Device read fault (Critical error)**	Ошибка чтения с диска (критическая ошибка).
162	**Hardware failure (Critical error)**	Неисправность аппаратного обеспечения (критическая ошибка).
200	**Division by zero**	Деление на ноль. Приложение пытается разделить число на ноль.
201	**Range check error**	Ошибка проверки диапазона. Если вы скомпилировали вашу программу с включенной проверкой диапазона, то вы можете получить эту ошибку в следующих случаях: 1. Попытка доступа к массиву с индексом, который выходит за пределы объявленного диапазона. 2. Попытка присвоить значение переменной, которое выходит за пределы её диапазона (например, перечисляемые типы).
202	**Stack overflow error**	Ошибка переполнения стека. Размер стека вырос за пределы максимально допустимого (в некоторых случаях уменьшение размера локальных переменных поможет избежать этой проблемы), или стек нарушен. Эта ошибка отображается только в том случае, если включена проверка стека.
203	**Heap overflow error**	Ошибка переполнения кучи. Размер кучи вырос за пределы границ. Это может случиться при попытке явного распределения памяти функциями **New**, **GetMem** или **ReallocMem**, или при создании экземпляра класса или объекта недостаточно памяти. Учтите, что по умолчанию Free Pascal предоставляет растущую кучу, то есть куча будет пытаться получить больше памяти, чем необходимо. Однако, если куча достигнет максимального размера, предоставленного операционной системой или аппаратным обеспечением, то случится эта ошибка.
204	**Invalid pointer operation**	Неправильная операция с указателем. Вы получите эту ошибку, если вызовете **Dispose** или **Freemem** с неправильным указателем (например, Nil).
205	**Floating point overflow**	Переполнение плавающей точки. Вы пытаетесь использовать или произвести слишком большое вещественное число.
206	**Floating point underflow**	Потеря значащих разрядов плавающей точки. Вы пытаетесь использовать или произвести слишком маленькое вещественное число.
207	**Invalid floating point operation**	Неправильная операция с плавающей точкой. Может случиться, если вы пытаетесь вычислить квадратный корень или логарифм отрицательного числа.
210	**Object not initialized**	Объект не инициализирован. Если компиляция выполнена с включенной проверкой диапазона, то программа вызовет эту ошибку, если вы вызвали виртуальный метод, не вызвав при этом конструктор объекта.
211	**Call to abstract method**	Вызов абстрактного метода. Ваша программа пытается выполнить абстрактный виртуальный метод. Абстрактные методы должны быть перегружаемыми, а вызываться должен перегруженный метод.
212	**Stream registration error**	Ошибка регистрации потока. Это случается, если неправильный тип зарегистрирован в модуле **objects**.
213	**Collection index out of range**	Индекс коллекции вне диапазона. Вы пытаетесь получить доступ к элементу коллекции с неправильным индексом (модуль **objects**).

Код	Ошибка	Описание
214	**Collection overflow error**	Ошибка переполнения коллекции. Коллекция достигла максимального размера, а вы пытаетесь добавить новый элемент (модуль **objects**).
215	**Arithmetic overflow error**	Ошибка арифметического переполнения. Эта ошибка появляется, если результат арифметической операции выходит за пределы поддерживаемого диапазона. В отличие от Turbo Pascal, эта ошибка случается только для 32-разрядного или 64-разрядного переполнения. Причиной тому является тот факт, что все числа преобразуются в 32-разрядные или 64-разрядные при выполнении арифметических операций.
216	**General Protection fault**	Общая ошибка защищённого режима. Приложение пытается получить доступ к недействительному пространству памяти. Это может быть вызвано несколькими проблемами: 1. Разыменование пустого указателя. 2. Попытка доступа к памяти, которая находится за пределами доступной области (например, вызов move с неправильной длиной).
217	**Unhandled exception occurred**	Произошло необработанное исключение. Произошло исключение, но его обработчик не представлен. Модуль sysutils устанавливает обработчик исключений по умолчанию, который отлавливает все исключения и выполняет безопасный выход.
219	**Invalid typecast**	Неправильное преобразование типов. Появляется, когда выполняется неудачное преобразование типов для класса с использованием оператора **as**. Эта ошибка также появляется, если объект или класс преобразованы в неправильный класс или объект и вызывается виртуальный метод этого класса или объекта. Последняя ошибка может быть обнаружена только в том случае, если включена опция компилятора **–CR**.
222	**Variant dispatch error**	Ошибка организации варианта. Нет метода организации (dispatch) для вызова из варианта.
223	**Variant array create**	Создание вариантного массива. При создании вариантного массива возникла проблема. Обычно случается из-за нехватки памяти.
224	**Variant is not an array**	Вариант не является массивом. Эта ошибка случается, когда выполняется попытка совершить операцию с вариантным массивом над вариантом, который не является массивом.
225	**Var Array Bounds check error**	Ошибка проверки диапазона массива. Ошибка случается, если индекс вариантного массива вне границ.
227	**Assertion failed error**	Не **AssertErrorProc** процедурная переменная была установлена.
229	**Safecall error check**	Ошибка проверки безопасного вызова. Эта ошибка случается, если возникает проблема безопасного вызова, и нет доступного обработчика процедуры.
231	**Exception stack corrupted**	Исключение нарушения стека. Ошибка случается, если объект исключения найден, но недоступен.
232	**Threads not supported**	Потоки не поддерживаются. Менеджер потоков использует отдельный драйвер на некоторых операционных системах (например, Unix-ах). Модуль с этим драйвером должен быть указан в разделе **uses** программы, желательно как первый модуль (на **unix** это модуль **cthreads**).

ПРИЛОЖЕНИЕ E. Простой файл gdb.ini.

Здесь представлен листинг простого файла **gdb.ini**, который даёт лучший результат, чем использование **gdb**. Под LINUX вы должны поместить этот файл как **.gdbinit** в вашу домашнюю директорию или в текущую директорию.

```
set print demangle off
set gnutarget auto
set verbose on
set complaints 1000
dir ./rtl/dosv2
set language c++
set print vtbl on
set print object on
set print sym on
set print pretty on
disp /i $eip

define pst
set $pos=&$arg0
set $strlen = {byte}$pos
print {char}&$arg0.st@($strlen+1)
end

document pst
  Print out a Pascal string
end
```

ПРИЛОЖЕНИЕ F. Опции и настройки.

В таблице F.1 сведены доступные логические директивы компилятора и связанные с ними опции командной строки. Другие директивы и связанные с ними опции см. в таблице F.2. Более подробно об опциях командной строки см. в разделе «5. КОНФИГУРАЦИЯ КОМПИЛЯТОРА». Более подробно о директивах см. в руководстве программиста.

Таблица F.1. Логические опции и директивы.

Короткая	Длинная	Опция	Пояснение
$A[+/-]	$ALIGN[ON/OFF]		Выравнивание данных.
$B[+/-]	$BOOLEVAL[ON/OFF]		Режим логической оценки.
$C[+/-]	$ASSERTIONS[ON/OFF]	-Sa	Включать формальное утверждение.
$D[+/-]	$DEBUGINFO[ON/OFF]	-g	Включать отладочную информацию.
$E[+/-]			Эмуляция сопроцессора.
$F[+/-]			Ближний и дальний вызов функций (игнорируется).
$G[+/-]			Генерировать код 80286 (игнорируется).
	$GOTO[ON/OFF]	-Sg	Поддерживать GOTO и Label.
	$HINTS[ON/OFF]	-vh	Показывать подсказки.
$H[+/-]	$LONGSTRINGS[ON/OFF]	-Sh	Использовать **ansistrings**.
$I[+/-]	$IOCHECKS[ON/OFF]	-Ci	Проверять результат операций ввода/вывода.
	$INLINE[ON/OFF]	-Si	Поддерживать стиль строки C++.
$L[+/-]	$LOCALSYMBOLS[ON/OFF]		Информация локальных идентификаторов.
$M[+/-]	$TYPEINFO[ON/OFF]		Генерировать RTTI для классов.
	$MMX[ON/OFF]		Поддерживать Intel MMX.
$N[+/-]			Поддерживать плавающую точку.
	$NOTES[ON/OFF]	-vn	Выдавать примечания.
$O[+/-]			Поддерживать перекрытие памяти (игнорируется).
$P[+/-]	$OPENSTRINGS[ON/OFF]		Поддерживать открытые строки.
$Q[+/-]	$OVERFLOWCHECKS[ON/OFF]	-Co	Проверять переполнение.
$R[+/-]	$RANGECHECKS[ON/OFF]	-Cr	Проверять диапазон.
$S[+/-]		-Ct	Проверять стек.
	$SMARTLINK[ON/OFF]	-CX	Использовать «умную компоновку».
	$STATIC[ON/OFF]	-St	Разрешить использовать **static**.
$T[+/-]	$TYPEDADDRESS[ON/OFF]		Печатать адреса.

Таблица F.2. Опции и директивы.

Короткая	Длинная	Опция	Пояснение
	$APPTYPE	-W	Тип приложения (Win32/OS2).
	$ASMMODE	-R	Режим ассемблера.
	$DEFINE	-d	Определение идентификатора.
	$DESCRIPTION		Установка описания программы.
	$ELSE		Переключатель условной компиляции.
	$ENDIF		Завершение условной компиляции.
	$FATAL		Отчёт о фатальной ошибке.
	$HINT		Выводить сообщение с подсказками.
$I файл	$INCLUDE		Подключить **файл** или буквенный текст.
	$IF		Начало условной компиляции.
	$IFDEF NAME		Начало условной компиляции.
	$IFNDEF		Начало условной компиляции.
	$IFOPT		Начало условной компиляции.
	$INCLUDEPATH	-Fi	Установка пути подключаемых файлов.
	$INFO		Выводить информационные сообщения.
$L файл	$LINK		Компоновать объектный **файл**.
	$LIBRARYPATH	-Fl	Установить путь библиотек.
	$LINKLIB имя		Библиотека компоновщика.
$M Мин, Макс	$MEMORY		Установить размер памяти.
	$MACRO	-Sm	Разрешить использование макросов.
	$MESSAGE		Выводить сообщения.
	$MODE		Установить режим совместимости.
	$NOTE		Выводить замечания.
	$OBJECTPATH	-Fo	Установить путь объектов.
	$OUTPUT	-A	Установить выходной формат.
	$PACKENUM		Размер типа перечислений.
	$PACKRECORDS		Выравнивание элементов записи.
	$SATURATION		Насыщенность (игнорируется).
	$STOP		Остановить компиляцию.
	$UNDEF	-u	Неопределённый идентификатор.

ПРИЛОЖЕНИЕ G. Получение последних исходных кодов или инсталляторов.

Free Pascal находится в непрерывной разработке. Время от времени создаются новые наборы инсталляторов. Они могут быть загружены с сайта Free Pascal. Загруженные данные обычно включают в себя исходные файлы, из которых создан дистрибутив.

В некоторых случаях имеет смысл никогда не устанавливать необходимые файлы, потому что можно загрузить новые исходные файлы и перекомпилировать их, что позволит устранить известные на текущий момент проблемы.

Учтите, что последние исходные коды могут компилироваться, но могут и не компилироваться: иногда они разбиты на части.

Есть три пути получения новой версии.

G1. Загрузка через Subversion

Все исходные коды Free Pascal поставляются в Subversion (свободная централизованная система управления версиями программного обеспечения) и могут быть загружены без регистрации с сервера Subversion. С подходящим клиентом Subversion, следующее местоположение можно использовать:

```
http://svn.freepascal.org/svn/fpc/trunk/
```

Репозиторий содержит последние исходные коды компилятора, RTL и пакеты.

Документацию и примеры из документации можно найти здесь:

```
http://svn.freepascal.org/svn/fpcdocs/trunk/
```

Все файлы, необходимые для создания дистрибутива, можно найти здесь:

```
http://svn.freepascal.org/svn/fpcbuild/trunk/
```

Этот репозиторий содержит внешние компоновщики для других 2 репозиториев и содержит все сценарии, демо-файлы и другие файлы, необходимые для создания нового дистрибутива Free Pascal.

Free Pascal сохраняет законченные подверсии, которые используются для создания новых дистрибутивов после изменения основной версии. Подверсии размещены здесь:

```
http://svn.freepascal.org/svn/fpc/branches/fpc_X_Y
```

где `X` и `Y` – это номер главной версии Free Pascal. Например, законченная версия 2.2.x Free Pascal доступна по адресу

```
http://svn.freepascal.org/svn/fpc/branches/fixes_2_2
```

Архив Subversion также можно найти на зеркальном сервере

```
svn2.freepascal.org.
```

G2. Загрузка zip-архива

Каждый день создаётся zip-архив, который содержит исходные файлы в том виде, в каком они находятся на текущий день. Эти архивы доступны здесь:

```
http://ftp.freepascal.org/develop.var
```

Загрузить исходные файлы подверсии можно здесь:

```
ftp://ftp.freepascal.org/pub/fpc/snapshot/trunk/source/fpc.zip
```

и законченные подверсии здесь:

```
ftp://ftp.freepascal.org/pub/fpc/snapshot/fixes/source/fpc.zip
```

Создание zip-файлов – это автоматический процесс. Эти файлы также создаются каждый день.

G3. Загрузка текущей копии

Некоторые члены команды разработчиков Free Pascal также поддерживают пригодные к установке срезы версий на текущий момент, которые сделаны из исходных файлов текущего дня. Эти исходные файлы не гарантируют правильной работы. Срез определённого дня может быть недоступен, или ответственный за него персонал не имел возможности его сделать. Эти срезы могут быть загружены с тех же страниц, что и архивы:

```
http://ftp.freepascal.org/develop.var
```

Срезы также делаются для разработчиков подверсий, как только имеется законченная подверсия. Они доступны здесь:

```
ftp://ftp.freepascal.org/pub/fpc/snapshot/trunk/
```

и

```
ftp://ftp.freepascal.org/pub/fpc/snapshot/fixes/
```

Сайт FTP имеет зеркало, иногда с зеркала загрузка идёт быстрее.

ДРУГИЕ КНИГИ

У меня есть и другие книги по программированию:

	**Основы программирования** Книга для тех, кто хочет стать программистом, но совершенно не знает с чего начать. Даже если вы полный чайник, эта книга научит вас создавать свои первые программы. Ссылка: http://info-master.su/books/op/
	**Основы C++** Эта книга уже посерьёзнее. Название говорит само за себя. Подробности см. здесь: http://info-master.su/books/cpp/
	**Как стать программистом** Бесплатная книга. Если вы новичок в программировании, то начать лучше именно с этой книги: http://info-master.su/mail/prog/

ОБ АВТОРЕ

На всякий случай представлюсь (вдруг кому интересно).

Это я – Поляков Андрей Валерьевич (можно просто Андрей))).

По образованию я инженер. Работаю руководителем инженерного отдела в крупном (по меркам нашего региона) агрохолдинге.

Это основная работа.

Но есть у меня и другие интересы. Я пишу книги, создаю обучающие курсы. А также немного (на уровне любопытства) занимаюсь инфо-бизнесом.

Кроме того у меня есть несколько сайтов с полезной информацией.

Итак, вот список ссылок на мои сайты, которые (во всяком случае, я на это надеюсь) могут оказаться Вам полезными.

Сначала личное:

Мой дневник (блог): http://av-inf.blogspot.ru

В контакте: http://vk.com/id185471101

Фейсбук: https://www.facebook.com/100008480927503

LinkedIn: http://ru.linkedin.com/pub/%D0%B0%D0%BD%D0%B4%D1%80%D0%B5%D0%B9-%D0%BF%D0%BE%D0%BB%D1%8F%D0%BA%D0%BE%D0%B2/86/906/6b1/

Одноклассники: http://ok.ru/polakoff

Мой круг: http://polyakovandrey44.moikrug.ru/

Гугл Плюс: https://plus.google.com/104478558796835580670/posts

Может сложиться впечатление, что я не вылажу из социальных сетей))) На самом деле это не так – я там редкий гость. Однако учётные записи в разных сетях нужны мне для общения с читателями.

P.S. Возможно, на тот момент, когда вы получите эту книгу, что-то изменится. Актуальную информацию можно найти здесь: http://info-master.su/contact.php

Бесплатные книги и видеокурсы:

http://info-master.su/mb.php - список всех моих бесплатных рассылок, где вы можете получить мои бесплатные книги и другие материалы.

Теперь сайты и проекты:

Реальные отзывы об инфо-продуктах
Здесь вы найдёте реальные (а не купленные и не придуманные) отзывы о разных инфо-продуктах. Кстати, можете поделиться своими впечатлениями о книге, тренинге, видеокурсе и т.п. – я буду только рад. Ссылка: http://realnie-otzivi.blogspot.ru

Инфо-мастер
Мой основной сайт, где вы можете найти все мои книги (и не только мои):
http://info-master.su

Основы программирования
Подраздел основного сайта, посвящённый программированию:
http://info-master.su/programming/index.php

Психология машин
Мой новый проект. Пока всё только начинается. Но посмотреть уже можно:
https://www.facebook.com/817221088320114

Автогид
Это ну ваще новый проект (на начало 2015 года). Пока практически пустой, но развитие планируется: http://avto-gid.blogspot.ru/

Ближе к железу
Сайт о программировании на ассемблере: http://av-assembler.ru

Информация
Тоже мой сайт, содержащий разнообразную информацию – от книг и рефератов до статей об автомобилях: http://av-mag.ru

Системы навигации и мониторинга
Подраздел предыдущего сайта, полностью посвящённый системам навигации, мониторинга, а также логистике и связанным со всем этим темам:
http://av-mag.ru/snm/sistemy-navigacii-i-monitoringa.php

Физика для всех
Сайт о физике: http://av-physics.narod.ru

Всё для студента
Этот сайт был создан, когда я учился в университете и был старостой группы.
http://tz-5133.narod.ru

В помощь студенту и инженеру
А это мой самый первый сайт. Создан аж в 2003 году. Я его давно не поддерживаю. Но народ заходит, поэтому и не удаляю: http://avprog.narod.ru

ЛИЦЕНЗИОННОЕ СОГЛАШЕНИЕ

1. Информационный продукт (далее – инфо-продукт), представляющий собой книгу, документ, видеокурс, подписку на почтовую рассылку и т.п., создан Поляковым Андреем Валерьевичем, г.Курган (далее – автор) и принадлежит ему. Инфо-продукт, приобретенный Вами у автора или у лица, уполномоченного автором, – это набор документов, книг, видеокурсов, вспомогательных файлов, программ и документации, который является объектом авторского права и охраняется законом. Под «документацией» подразумеваются печатные материалы, видеоматериалы и файлы с текстом, содержащие описание компонентов инфо-продукта и являющиеся неотъемлемой его частью. Все условия, оговоренные далее, относятся как к инфо-продукту в целом, так и ко всем его компонентам в отдельности.
2. Осуществляя копирование, установку или использование всего инфо-продукта или его части, Вы тем самым принимаете все условия и положения настоящего соглашения и признаете, что настоящее соглашение представляет собой обычный договор, составленный в письменном виде, подписанный Вами и имеющий юридическую силу.
3. Если Вы не принимаете условий этого соглашения, Вы не имеете права использовать этот инфо-продукт и должны удалить все его копии с Ваших носителей данных и/или отписаться от почтовой рассылки.
4. Имущественные права на данный инфо-продукт принадлежат исключительно автору. Вам соглашение предоставляет право на использование инфо-продукта в порядке и для целей, указанных в документации и как это оговорено ниже.
    а. Право на использование инфо-продукта предоставляется только Вам и никому более, если иное не оговорено особо.
    б. Вы можете использовать неограниченное количество копий инфо-продукта на каждом компьютере, при условии, что этим компьютером пользуетесь только Вы.
    в. Вы можете сделать разумное число запасных копий инфо-продукта, при условии, что запасные копии не используются для каких-либо целей, за исключением архивирования.
    г. Вы не можете изменять инфо-продукт иным способом, кроме как явно описанным в документации. Вы не можете переконструировать, декомпилировать, дизассемблировать или иным образом пытаться определить исходный код инфо-продукта.
    д. Вы не можете вносить какие-либо изменения в инфо-продукт.
    е. Вы не можете использовать инфо-продукт или его части, в своих книгах, сайтах, других Интернет-ресурсах, за исключением случаев, которые предусмотрены законом об авторском праве.
    ж. Вы не можете сдавать инфо-продукт в аренду или финансовую аренду, продавать, сублицензировать, уступать или передавать свои права на его использование, либо разрешать другим лицам копировать инфо-продукт либо его части на любые носители данных.
    з. Вы можете получать от автора новые версии инфо-продукта, но, возможно, с дополнительными условиями.
5. Если инфо-продукт не функционирует в соответствии с документацией, то полный объем ответственности разработчика ограничивается (по усмотрению разработчика) заменой инфо-продукта или возвратом суммы, уплаченной Вами за инфо-продукт.
6. В остальном инфо-продукт поставляется «таким, каков он есть». Автор не предоставляет Вам никаких гарантий, явных или подразумеваемых, что инфо-продукт не содержит ошибок, что инфо-продукт будет отвечать Вашим требованиям или ожиданиям, будет соответствовать Вашим целям и задачам. Ни автор, ни другие юридические или физические лица, имеющие отношение к созданию, производству или распространению инфо-продукта, не несут ответственности за прямые или косвенные убытки, которые могут возникнуть вследствие использования или невозможности использования инфо-продукта, даже если вы уведомили представителя автора о возможности возникновения таких убытков. Никакое другое письменное или устное соглашение, предоставленное Вам, не может расширить границы этой гарантии.