Глава 18. Макросредства

Макрос почти всегда указывает на недостаток языка программирования, программы или программиста.

Бьёрн Страуструп.

В большинстве языков программирования (по крайней мере императивного типа) существуют языковые возможности, называемые макросредствами (или часто просто макросами). Вообще говоря, макросредства могут включаться не только в языки программирования, но и в другие программные средства. Например, всем Вам должны быть известно макросы в текстовых редакторах (скажем в Word).

Полностью эту тему Вы будете изучать в курсе по архитектурам ЭВМ на примере макросредств языка Ассемблера. Дело в том, что в языках программирования высокого уровня макросредства примитивны (не развиты). Сейчас мы познакомимся с макросредствами в языке Free Pascal.

Сначала познакомимся с так называемыми константами времени компиляции (они часто и называются *макросами*), в программе они описываются так:

```
{$define X:=12}
```

Это не «настоящие» константы, на самом деле это строковые переменные, им присваивается новое значение при повторном описании:

```
{$define X:=12}
writeln(X); { вывод 12 }
{$define X:=3.1415}
writeln(X); { вывод 3.1415 }
```

Работа с этими константами в чём-то похожа на механизм текстовой подстановки Нормальных Алгоритмов Маркова: ниже по тексту программы имя константы заменяется на её текстовое значение, т.е. `X → 3.1415`. Как видим, это безтиповые переменные, аналог параменных языка Python или типа `variant` языка Free Pascal. Следующий пример:

```
var x: integer=1; y: integer=2;
  writeln(x); { вывод 1 }
{$define x:=y} { x → y }
  writeln(x); { вывод 2 }
```

Тонким моментом является то, что эти переменные существуют только на этапе компиляции, попытайтесь понять этот пример:

```
var i: integer=1;
  {$define x:=1}
  for i:=1 to 100 do
      {$define x:=x+1};
  writeln(x); { вывод 2 ⚠ }
```

Ценность этого механизма в том, что такие макросы можно определить не только в тексте программы, но и в командной строке при вызове компилятора, т.е. не изменяя текст исходной программы (и даже не имея возможности такого изменения!). Это делается путём задания так называемого ключа компилятора, например, ключ

```
-dsingle:=double
```

заменит в тексте программы все имена `single` на `double`, это эквивалентно заданию макроса

```
{$define single:=double}
```

Поймите, что так мы заменили типы всех вещественных переменных в программы с `single` на `double`.

Ещё пример

```
{$define Gipot:=sqrt(x*x+y*y);}
writeln('Гипотенуза=',Gipot);
```

В языке C есть и более развитые макросы *с параметрами*, например:

```
{$define Gipot(x,y) sqrt(x*x+y*y)}
double c=Gipot(a,b);
```

Когда макрос с некоторым именем больше не нужен, его можно уничтожить, например:

```
{$undef Gipot}
```

В языке определены и некоторые стандартные макросы, например:

`FPC_FULLVERSION` – выдаёт версию компилятора, например 20604 для Free Pascal 2.6.4.

Во время компиляции значения макроконстант можно сравнивать, например:

```
{$if FPC_FULLVERSION < 20604}
{$ERROR Надо Free Pascal версии >= 2.6.4}
{$endif}
{$if X<2}
  writeln('надо константу X>1');
  exit
{$endif}
{$if X=2}
  P(x,y);
{$else}
  Q(x,y);
{$endif}
```