МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. М.В. ЛОМОНОСОВА

Факультет вычислительной
математики и кибернетики

В.П. Иванников, Л.С. Корухова, В.Н. Пильщиков

ПИСЬМЕННЫЙ ЭКЗАМЕН ПО КУРСУ “АЛГОРИТМЫ И АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ”

Методическое пособие

Москва
2002

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М.В. Ломоносова

Факультет вычислительной математики и кибернетики

В.П. Иванников, Л.С. Корухова, В.Н. Пильщиков

ПИСЬМЕННЫЙ ЭКЗАМЕН ПО КУРСУ “АЛГОРИТМЫ И АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ”

(методическое пособие)

Москва
2002

УДК 519.6+510.6
ББК 22.18+22.12

Иванников В.П., Корухова Л.С., Пильщиков В.Н. Письменный экзамен по курсу “Алгоритмы и алгоритмические языки” (методическое пособие). — М. Издательский отдел факультета вычислительной математики и кибернетики МГУ (лицензия ИД № 05899 от 24.09.2001), 2002. – 47 с.

ISBN 5-89407-143-7

Пособие содержит варианты письменных экзаменов (1995-2002 годов) основного курса лекций “Алгоритмы и алгоритмические языки”, читаемого для студентов 1 курса факультета ВМК МГУ. Письменный экзамен по курсу преследует цель – на основе унифицированного набора задач по различным разделам курса и общих критериев объективно оценить знания студентов, уровень их подготовки. Варианты включают задачи различного уровня сложности: наряду с простейшими задачами и вопросами по конкретной теме, имеются задачи, предполагающие знание материала нескольких тем и умение владеть этим материалом. В пособии приведены ответы, а также продемонстрированы авторские решения некоторых задач, дана программа курса и рекомендуемая литература.

Методическое пособие предназначено для преподавателей, ведущих практические занятия в поддержку курса лекций “Алгоритмы и алгоритмические языки”, а также рекомендуется студентам при подготовке к письменному экзамену.

Рецензенты:
Профессор Томилин Александр Николаевич
Доцент Баула Владимир Георгиевич

Печатается по решению Редакционно-издательского Совета факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова

ISBN 5-89407-143-7

1. Программа курса "Алгоритмы и алгоритмические языки” (2001/2002 учебный год)

Часть 1. Введение в теорию алгоритмов.

Интуитивное понятие алгоритма. Свойства алгоритмов. Понятие об исполнителе алгоритма.

Уточнение понятия алгоритма. Алгоритм как преобразование слов из заданного алфавита. Машина Тьюринга. Тезис Тьюринга и его обоснование. Нормальные алгоритмы Маркова. Принцип нормализации и его обоснование. Композиции машин Тьюринга и нормальных алгоритмов Маркова. Понятие об алгоритмической неразрешимости.

Алгоритмическая сложность. Связь понятия алгоритма с понятием функции.

Часть 2. Алгоритмические языки. Язык Паскаль

Характеристика алгоритмических языков и их исполнителей. Понятие трансляции.

Понятие о формальных языках. Способы строгого описания формальных языков, понятие о метаязыках. Алфавит, синтаксис и семантика алгоритмического языка. Описание синтаксиса языка с помощью металингвистических формул и синтаксических диаграмм.

Язык программирования. Общие характеристики языков программирования. Алфавит, имена, служебные слова, стандартные имена, числа, текстовые константы, разделители.

Структура программы на Паскале. Заголовок программы. Блок.

Типы данных, их классификация. Переменные и константы. Скалярные типы данных и операции над ними, старшинство

операций, стандартные функции. Выражения и правила их вычисления. Оператор присваивания. Перечислимые и ограниченные типы данных.

Простые и сложные операторы. Пустой, составной, условный операторы и оператор перехода. Метки. Оператор варианта.

Файлы. Стандартные процедуры и функции ввода-вывода.

Операторы цикла. Программирование рекуррентных соотношений. Составные типы данных. Массивы.

Описание процедуры и оператор процедуры. Формальные и фактические параметры. Способы передачи параметров. Локализация имен. Разрешение коллизий. Функции, побочные эффекты.

Итерации и рекурсии.

Комбинированные типы. Оператор присоединения. Множества. Ссылочный тип данных. Динамические переменные.

Часть 3. Структуры данных

Абстрактные структуры данных: графы, деревья, таблицы. Отношения. Отображение абстрактных структур данных на структуры хранения: векторная память, списки. Стеки и очереди.

Таблицы. Последовательные таблицы. Деревья сравнений. Перемешанные таблицы. Оценки алгоритмической сложности.

Классические алгоритмы.

Этапы разработки программ.

2. РЕКОМЕНДОВАННАЯ ЛИТЕРАТУРА

1. М. Минский. Вычисления и автоматы. - М., “Мир”, 1971.

2. Т. Кормен, Ч. Лейзерсон, Р. Ривест. Алгоритмы: построение и анализ. - М., МЦНМО, 1999.

3. А.А. Марков, Н.М. Нагорный. Теория алгорифмов. - М., ФАЗИС, 1996.

4. Б.А. Трахтенброт. Алгоритмы и вычислительные автоматы. - М., Советское Радио, 1974.

5. А.И. Мальцев. Алгоритмы и рекурсивные функции. - М., Наука, 1986.

6. Л.С. Корухова, М.Р. Шура-Бура. Введение в алгоритмы. (учебное пособие для студентов 1 курса) - М., Издательский отдел факультета ВМК МГУ, 1997.

7. Н. Вирт. Алгоритмы + структуры данных = программы. - М., “Мир”, 1985.

8. К. Йенсен, Н. Вирт. Паскаль. Руководство для пользователя. - М., “Компьютер”, 1993.

9. В.Г. Абрамов, Н.П. Трифонов, Г.Н. Трифонова. Введение в язык Паскаль. - М., Наука. 1988.

10. Д. Кнут. Искусство программирования для ЭВМ.
Т.1. Основные алгоритмы.
Т.2. Сортировка и поиск.
- М., Наука, 1985.

11. Г. Лорин. Сортировка и системы сортировки. - М., “Наука”, 1983.

12. М. Сибуя, Т. Ямомото. Алгоритмы обработки данных. - М., “Мир”, 1986.

13. П. Холл. Вычислительные структуры. Введение в нечисленное программирование. - М., “Мир”, 1978.

14. Э.З. Любимский, В.В. Мартынюк, Н.П. Трифонов. Программирование. - М., “Наука”, 1980.

15. А. Ахо, Д. Хопкрофт, Д. Ульман. Структуры данных и алгоритмы. - М., изд-во Вильямс, 2000.

3. ВАРИАНТЫ ЭКЗАМЕНАЦИОННЫХ РАБОТ

3.1. ВАРИАНТ 1995

1. *а*) Сформулируйте тезис Тьюринга.
   б) Объясните, почему его нельзя доказать.
   в) Приведите доводы в пользу этого тезиса.

2. В алфавите A = {*a*,*b*,*c*} заданы алгоритмы P, Q и R:

$$P:\begin{cases} ca \to ac \end{cases} \qquad Q:\begin{cases} cb \to bc \end{cases} \qquad R:\begin{cases} cb \to bc \\ ca \to ac \end{cases}$$

Является ли алгоритм R композицией алгоритмов P и Q (R = P(Q))? Дать ответ (да/нет) и его обоснование.

3. Для каждого из приведенных выражений языка Паскаль укажите его значение (если выражение ошибочно, укажите в ответе слово "ошибка"):

1) -40 mod 7
2) false=true and false
3) chr(succ(ord('8')))
4) 5 * sin(0)
5) '4' <> 4

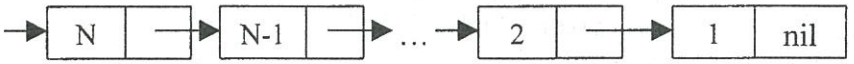
4. var p: ↑T; {T - некоторый тип}
Опишите действия стандартной процедуры DISPOSE(p).

5. Укажите, какие числа будут выданы на печать при выполнении следующего фрагмента программы:

```
var A, B, C, D: integer;
procedure P(X: integer;  var A: integer);
  var C: integer;
  begin X:=5;  A:=6;  C:=7;  D:=8  end;
begin ...
  A:=1;  B:=2;  C:=3;  D:=4;
  P(A, B);  write(A, ' ',  B, ' ', C, ' ',  D);  . . .
end.
```


6. Считая известными понятия <переменная> и <тип>, дать определение понятия <раздел описания переменных> в виде синтаксической диаграммы.

7. type неотр = 0..maxint;
   список = ↑звено;
   звено = record элем: неотр; след: список end;
Описать рекурсивную функцию constr(N) от N типа *неотр*, которая строит список вида:



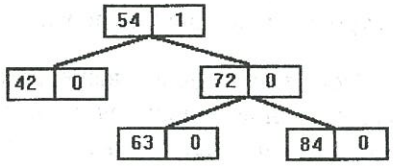
(при N=0 список пустой) и возвращает как свое значение ссылку на этот список.

8. Для построения перемешанной таблицы TAB[0..12] использованы функции первичного перемешивания I = K mod 13 и вторичного перемешивания I = (I+4) mod 13, где I – адрес, а K – ключ. Указать содержимое таблицы, если ключи в таблицу заносятся в следующем порядке: 19, 27, 23, 6, 18.

TAB:

Адрес	0	1	2	3	4	5	6	7	8	9	10	11	12
Ключ													

9. **а)** В указанное АВЛ-дерево (в каждой вершине слева - ключ, справа – характеристика) добавить новый ключ 93 и нарисовать дерево, полученное после вставки ключа с новыми характеристиками.



б) После этого привести дерево к виду АВЛ-дерева в соответствии с алгоритмом трансформации и нарисовать это дерево с новыми характеристиками

10. Записать в постфиксном виде (ПОЛИЗ) следующее выражение:

3 * ((A + B) * C - H) + (D - F) / 6 * A

3.2. ВАРИАНТ 1996

1, Определить, является ли самоприменимым указанный нормальный алгоритм **W** в расширенном алфавите **A**={a,b,c}. Дать ответ (да/нет) и привести обоснование.

$$\mathbf{W}:\begin{cases} *a \to a* \\ b \to bc \\ c \mapsto cc \\ \to * \end{cases}$$

2. Для указанного нормального алгоритма **W** в расширенном алфавите A={a,b} привести эквивалентный ему нормальный алгоритм **R** в алфавите A, состоящий только из одного правила подстановки.

$$\mathbf{W}:\begin{cases} +a \to a+ \\ +b \to b+ \\ + \to * \\ a* \to *a \\ b* \to *b \\ * \mapsto a \\ \to + \end{cases}$$

3. Дана последовательность ключей 1, 2, 3, 4, 5, 6, 7, 8, 9. Построить совершенное дерево с такими ключами.

4. Используя для вещественных чисел нормализованное представление вида: *знак порядка*, *десятичный порядок* (2 цифры), *знак мантиссы*, *мантисса* (4 десятичных цифры) и заданные вещественные числа **a** = +12+7153, **b** = +12-7153, **c** = +01-1002, вычислить значения ***r*** и ***p***: ***r*** = a+(b+c); ***p*** = (a+b)+c

5. Для заданной последовательности ключей 1, 2, 3, 4, 5, 6 построить АВЛ-дерево в соответствии с алгоритмом построения АВЛ-деревьев. Проставить характеристики в узлах дерева.

6. const n = … ; { n > 1 }
   type M = array [1..n] of 1..maxint;

Считая, что элементы массива X типа M расположены по убыванию (X[i] > X[i+1]), описать процедуру P(X,S), которая из X выделяет такой набор чисел (без повторов), чтобы их сумма равнялась целому числу S, и печатает этот набор. Если такого набора нет, процедура должна напечатать слово NO.

7. Указать, может ли в правильной программе на языке Паскале встретиться (вне строк и комментариев) следующий оператор присваивания:

r['h']↑.H[false].p := 'r' > 'p'

Дать ответ (да/нет) и его обоснование.

8. Для каждого из приведенных выражений языка Паскаль указать его значение или слово “ошибка”, если выражение ошибочно:

1) round(-0.5) = -round(0.5)
2) 20 / 4 div 2
3) 20 mod 3 / 5
4) true and false = true
5) ord('Q') > 5

9. type список = ↑звено;
   звено = record элем: char; след: список end;

Описать рекурсивную процедуру DEL(L), удаляющую последнее звено из непустого списка L. При удалении освободить память, занимаемую этим звеном.

10. Определить, что будет выдано на печать при выполнении следующей программы:

```
program A (output);
  var x, y, z : integer;
  procedure P (var x : integer);
    var z : integer;
```

```
  begin  z := x;  x := y;  y := z  end;
begin
   x := 1;   y := 2;   z := 3;
   P(z);  writeln(x, ' ', y, ' ', z)
end.
```

3.3. ВАРИАНТ 1997

1. Приведите несамоприменимый нормальный алгоритм в алфавите **A**={a,b,c}, в котором не более двух правил подстановки и который применим хотя бы к одному слову из **A***. Ответ обосновать.

2. Ниже приведены пять нормальных алгоритмов в расширенном алфавите A={a,b}. Есть ли среди них алгоритмы, являющиеся композицией двух других алгоритмов из заданных? Если "да", то указать все такие композиции. Если "нет" - ответ обосновать.

$$U:\{\ ab\to \quad T:\{\ b\to a \quad W:\begin{cases} ab\to \\ b\to a \end{cases} \quad Q:\begin{cases} ab\to \\ b\to a \\ \to * \\ *\mapsto \end{cases} \quad R:\begin{cases} *b\to a* \\ *a\to a* \\ *\mapsto \\ ab\to \\ \to * \end{cases}$$

3. Определить, что будет выдано на печать:

```
program PROG(output);
   var A, B, C, D: integer;
   procedure P(X: integer;  var A: integer);
     var C: integer;
   begin
     C:= 5;  X:=C;  A:=X+6;  B:=7+A;  D:=X+8
   end;
begin
   A:=1;  B:=2;  C:=3;  D:=4;
   P(A,B);  writeln(A, ' ', B, ' ', C, ' ', D);
end.
```


4. Определить, может ли приведенная таблица ключей быть перемешанной таблицей при следующих функциях первичного (F) и вторичного (G) перемешивания: A=F(K) = K mod 11, A=G(A) = (A+2) mod 11, где K – ключ, A – адрес. Если "да", то указать последовательность поступления ключей в таблицу. Если "нет", то доказать это. Предполагается, что исключений из таблицы не было.

Адрес	0	1	2	3	4	5	6	7	8	9	10
Ключ	20	-	22	-	29	-	31	18	17	7	-

5. Дать определение дерева Фибоначчи.
   Для множества ключей 1, 2, 13, 14, 25, 36, 55 нарисовать:
   а) дерево Фибоначчи,
   б) АВЛ-дерево, не являющееся деревом Фибоначчи.

6. type tree = ↑node;
   node = record elem: real; lev, prav: tree end;
Не используя операторы цикла и перехода, описать булевскую функцию AVL(t), определяющую, является ли двоичное дерево сравнений t (типа tree) АВЛ-деревом.

7. Всегда ли программы **а)** и **б)** на Паскале вычисляют одинаковый результат (S)? Дать ответ (да/нет) и обосновать его.

а)
```
program A(output);
  const N =19; A =0.0; B=1.0;
  var S, h, X: real;
      i: integer;
  begin  S:= 0;
   h:= (B-A)/N; X:= A - h;
   for i := 0 to N do
     begin
       X:=X+h;
       S:=S+sin(X)
     end ;
   writeln('S=', S)
end.
```

б)
```
program B(output);
  const N =19; A =0.0; B =1.0;
  var S, h, X: real;
  begin  S:= 0;
   h:= (B-A)/N; X:= A - h;
   while X <> B  do
     begin
       X:=X+h;
       S:=S+sin(X)
     end ;
   writeln('S=', S)
  end.
```

8. Определить, допустим ли в правильной программе на Паскале (вне комментариев и строк) следующий фрагмент:

```
; x:=y; y[2].x := 1+ true;
```

Если "да", то приведите подходящее описание имен, если "нет" – докажите это.

9.
```
var F: file of char;
```
Опишите действия стандартной процедуры RESET(F).

10.
```
const N =100;
var M: array[1..N, 1..N] of 0..1;
```
Имеется N городов, перенумерованных от 1 до N. В матрице M собрана информация о дорогах между этими городами: если из города i можно непосредственно попасть в город j, то M[i,j]=1, иначе M[i,j]=0 (не обязательно M[i,j]=M[j,i]). Описать процедуру PATH, которая, используя метод поиска с возвращением, находит путь, начинающийся в городе 1, проходящий через все остальные города (без повторов) и оканчивающийся в городе N. Если такой путь есть, процедура должна напечатать его (как последовательность номеров пройденных городов), а если нет, то должна напечатать слово НЕУСПЕХ.

3.4. ВАРИАНТ 1998

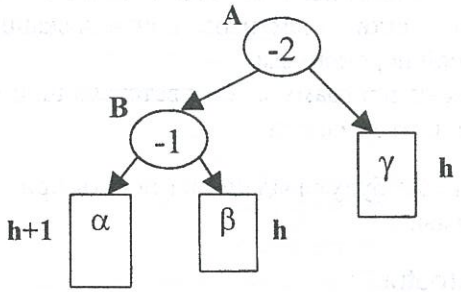
1.Дан алфавит **A**={*a*,*b*}. Приведите пример нормального алгоритма в расширенном алфавите **A**, который применим к любому слову из $\mathbf{A}^*$, содержит не более трех правил подстановки и несамоприменим.

2. Пусть R и Q - два нормальных алгоритма в алфавите **A**={0, 1}. Дайте определение эквивалентности R и Q в этом алфавите.

3. Пусть V - переменная типа integer. Определить, что будет напечатано при выполнении следующих операторов:

```
writeln;
V := ord(chr(ord('A') + 3)) - ord('A');
for V := 2 to V + 2 do write( V + 10, ' ,' );
writeln;
```


4. При занесении нового ключа в АВЛ-дерево возникла ситуация (см. рисунок, **h** и **h+1** - высоты поддеревьев), когда характеристика вершины **A** оказалось равной -2. Указать, в какое поддерево (α, β или γ) был занесен новый ключ, и нарисовать дерево после корректировки характеристик.



5.
```
type ключ = 0..maxint;
     тело = array[1..40] of char;
```
Каким должно быть **описание** перемешанной (хешированной) таблицы tab и каким должно было быть ее **содержимое** в момент перед занесением первого элемента, чтобы указанная процедура delete(x) всегда верно исключала из таблицы элемент с ключом x. Функция первичного перемешивания равна K mod (N+1) , где K - ключ, а N+1 - простое число. Коллизии ключей разрешаются с помощью списковой памяти переполнения.

```
procedure delete (x: ключ);
 var q, r: next;
     return: boolean;
begin
  q:= tab[x mod (N+1)];
  return := true;
  while (q↑.p <> nil) and return do
   if q↑.p↑.K = x then
     begin
        return:= false;
        r:= q↑.p; q↑.p:=r↑.p;
```

```
        dispose(r)
      end
    else q := q↑.p
  end;
```

6. Даны две динамические переменные множественного типа, имеющие конкретные значения. Требуется одной из них присвоить новое значение - множество элементов, принадлежащих одновременно первой и второй переменным.

Выписать фрагмент программы (соответствующие описания и действия) для решения этой задачи.

7. Укажите, какие числа будут выданы на печать при выполнении следующей программы:

```
program PR(output);
  var A, B, C, D: integer;
  procedure P(X: integer;  var A: integer);
    var C: integer;
  begin  X:=5;  A:=X+18;  C:=7;  D:=C+8  end;
begin
  A:=1;  B:=2;  C:=3;  D:=4;
  P(A, B);
  write(A, ' ',  B , ' ', C , ' ' ,  D)
end.
```

8. type pp = ↑node; {указатель на вершину}
node = record лев, прав: pp end; { вершина }

Приведите описание рекурсивной процедуры free(p), которая "ликвидирует" дерево p типа pp, освобождая при этом память, занятую вершинами удаляемого дерева.

9. Приведите описание рекурсивной процедуры PF(h,p), которая из заданного совершенного дерева p, имеющее высоту не менее h ($h \geq 1$), удаляет вершины, чтобы превратить его в дерево Фибоначчи высоты h, и освобождает при этом память, занимаемую удаляемыми вершинами. Считать, что в дереве Фибоначчи для любой нелистовой вершины высота её левого

поддерева меньше высоты правого. В решении использовать описания и процедуру free из задачи № 8.

10.
```
var a, b, c, d: char;
    p: record  a, b: char;  c: record d: char end  end;
    q: record  a: char;  c: record e: char end  end;
```
Записать без использования оператора with действия следующего оператора присоединения:
```
with p, q, c  do  begin  a := succ(b);  d := e  end;
```

3.5. ВАРИАНТ 1999

1. Назовем нормальный алгоритм Маркова укорачивающим, если в каждом его правиле подстановки длина левой части строго больше длины правой части. Доказать, что для укорачивающих алгоритмов проблема останова алгоритмически разрешима.

2. В алфавите **A**={**a**, **b**, **c**} заданы два алгоритма **R** и **W**. Алгоритм **R** имеет вид:

$\{a \rightarrow$

Алгоритм **W** задан машиной Тьюринга (предполагается, что управляющая головка в начале работы указывает на первый символ слова и находится в состоянии q_0; q_s - состояние останова):

	λ	a	b	c
q_0	λ,L,q_s	a,R,q_0	b,R,q_1	c,R,q_0
q_1	λ,L,q_s	a,R,q_0	b,R,q_1	b,L,q_2
q_2	λ,L,q_s	a,L,q_s	c,L,q_s	c,L,q_s
q_s				

Определить, существует ли алгоритм **Q**, являющийся композицией алгоритмов **W** и **R** (**Q=W(R)**)? Дайте ответ (да/нет). Если "да", то приведите алгоритм **Q**; если "нет " – объясните почему такого алгоритма нет.

3. Даны две перемешанные таблицы T1[0..12] и T2[0..12] с функцией первичного перемешивания I = K mod 13, функцией вторичного перемешивания I = (I+4) mod 13, где I - адрес, а K - ключ. В таблицы T1 и T2 занесены следующие ключи:

Адрес	0	1	2	3	4	5	6	7	
T1		23				27	19		
T2		40				14			

	8	9	10	11	12
	59				
		5	36		

Дополнить таблицу T2 ключами из таблицы T1 так, чтобы порядок записи ключей был таким же, как и при заполнении таблицы T1. Указать содержимое таблицы T2 после дополнения.

4. Считаем, что пустое дерево имеет высоту 0, дерево из одной вершины – высоту 1. Какова наибольшая высота у АВЛ–деревьев, содержащих 145 вершин? Привести ответ и обосновать его.

5. type типэл = … ;
   список = ↑звено;
   звено = record элем: типэл; след: список end;

Описать функцию modify(S), которая удаляет последнее звено из заданного непустого списка S и возвращает ссылку на обновленный список.

6. type неотр = 1..maxint;
   pp = ↑node; {указатель на вершину}
   node = record лев, прав: pp; элем: неотр end; {вершина}

Будем считать путем в двоичном дереве типа pp последовательность вершин, которые нужно "пройти", чтобы из

корневой вершины "попасть" в листовую вершину дерева. Привести описание булевской функции SUM(T, S), возвращающей значение TRUE, если в заданном дереве T существует путь, сумма элементов в вершинах которого равна заданному целому числу S, и FALSE – в противном случае. Считаем, что в пустом дереве сумма элементов пути равна 0. Операторы цикла и перехода не использовать.

7. Укажите, какие числа будут выданы на печать при выполнении следующей программы:

```
program PR(output);
  var A, B, C, D: integer;
  function P(var A: integer;  X: integer): integer;
    var C: integer;
  begin
    X:=5;  A:=6;  D:=8+B;  B:=A+10;  C:=7;  P:=A+X
  end;
begin
  A:=1;  B:=2;  C:=3;   D:=4;
  write( P(B,B) , ' ; ' , A, ' ; ',  B , ' ; ', C , ' ; ' ,  D)
end.
```

8. Указать, верно или нет в стандарте языка Паскаль каждое из следующих утверждений:

а) Любое нестандартное имя (переменной, типа и т.п.) всегда сначала должно быть описано и лишь затем использовано.

б) В операторе присваивания V:=E типы переменной V и выражения E должны быть одинаковыми.

в) Оператор процедуры является простым оператором.

г) Параметр стандартной процедуры вывода write может быть массивом.

д) Тип (integer или real) числа определяется по величине этого числа.

9. Не используя операторы цикла и перехода, написать программу для ввода непустой последовательности положительных целых чисел и вывода наибольшего значения этих чисел. Считать, что последовательность вводимых чисел заканчивается числом 0.

10. Может ли в правильной программе на языке Паскаль встретиться (вне строк и комментариев) следующий фрагмент текста:

```
; X := X↑.X;
```

Приведите ответ (да/нет) и его обоснование.

3.6. ВАРИАНТ 2000

1. Задан нормальный алгоритм Т в алфавите A={0,1}. Что можно сказать о применимости этого алгоритма к словам из A^* (A^* - множество слов из символов алфави-та А), т.е. каковы подмножества слов, к которым он применим и неприменим?

$$T:\begin{cases} 01 \to 10 \\ 10 \mapsto 01 \\ 1 \to 11 \end{cases}$$

2. В алфавите **A**={a,b} описать нормальный алгоритм, который выдает в качестве результата пустое слово, если буквы **a** и **b** входят во входное слово в равном количестве, и любое непустое слово иначе. В алгоритме должно быть не более четырех правил подстановки.

3. Дана перемешанная таблица T[0..16] с функцией первичного перемешивания I = K mod 17 и функцией вторичного перемешивания I = (I+5) mod 17, где I – адрес, а K - ключ. В первоначально пустую таблицу Т были последовательно занесены 6 ключей. В результате перемешанная таблица стала выглядеть так:

Адрес	0	1	2	3	4	5	6	7	
Ключ		35			38			24	

	8	9	10	11	12	13	14	15	16
	59					42	48		

Как выглядела бы таблица Т, если бы в первоначально пустую таблицу эти же ключи заносились в обратной последовательности? Ответ обосновать.

4. Для заданной последовательности ключей 4, 5, 6, 3, 1 построить АВЛ-дерево в соответствии с алгоритмом построения АВЛ-деревьев (путем последовательного применения процедуры INSERT). Нарисовать конечное АВЛ–дерево, указав в каждой вершине значения ключа и характеристики.

5. Описать функцию NOTEMP(f), которая считает число непустых строк в заданном текстовом файле f.

6. Назовем двоичное *дерево сравнений* **выровненным**, если для каждой его вершины число вершин в ее левом поддереве равно или на 1 меньше числа вершин в ее правом поддереве.

При следующих описаниях

```
const N = …;  {N ≥ 1}
type MASS = array[1..N] of integer;
     TREE = ↑node;  {указатель на вершину}
     node = record key: integer;  left, right: tree end;  {вершина}
```

и используя метод “разделяй и властвуй”, определить функцию TR(K), которая по заданному массиву К (типа MASS) из упорядоченных по убыванию различных ключей строит **выровненное** дерево (типа TREE) и возвращает ссылку на построенное дерево. Операторы цикла и перехода не использовать.

7. Каков результат выполнения следующей программы? Дайте ответ и объясните его.

```
program  PROG(input, output);
  var i: real; k: 10..20;
begin
   i:= round( - 5.75 );   k:= trunc( 10.75 );
   case i+k+10 of
     10, 12, 14, 16, 18, 20 :  write( ' Четное ' );
     11, 13, 15, 17, 19 :  write( ' Нечетное ' )
   end
end.
```

8. Назовем **выражением** в алфавите **A**={/, **s**} последовательность символов, окаймленную символом / (первый и последний символы - это /) и содержащую внутри символы / и s, причем символов / -

четное число. Привести синтаксическую диаграмму для понятия **выражение**.

9. Пусть J - переменная типа integer. Что будет напечатано при выполнении следующих операторов программы?:

```
writeln;
J := ord(chr(ord('F') + 11)) - ord('F');
for J := 11 downto J - 4 do write(J, ' ; ');
writeln;
```

10. Может ли в правильной программе на языке Паскаль встретиться (вне строк и комментариев) указанный фрагмент текста:

а) ; B := A↑ in [C +A↑, C] ;

б) ; R↑ := R↑ in [R↑+H] ;

Для каждого из вариантов а) и б) приведите ответ (да/нет) и его обоснование.

3.7. ВАРИАНТ 2001

1. В алфавите $A=\{|\}$ натуральное число n представляется n символами | . Приведите пример нормального алгоритма, который применим к любому натуральному четному числу и неприменим к любому нечетному числу. Алгоритм должен содержать не более четырех правил подстановки.

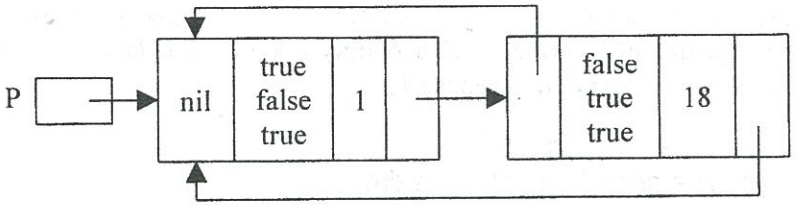
2. Дайте определение эквивалентности алгоритмов над заданным алфавитом. Пусть U, T, W, Q, R – указанные ниже нормальные алгоритмы над алфавитом A={a,b}. Укажите среди них группы эквивалентных алгоритмов.

$$U:\{\ ab\mapsto \quad T:\{\ b\to a \quad W:\begin{cases} ab\to \\ b\mapsto a \end{cases} \quad Q:\begin{cases} *b\to b* \\ *ab\mapsto \\ *a\to a* \\ *\mapsto \\ \to * \end{cases} \quad R:\begin{cases} *b\to a* \\ *a\to a* \\ *\mapsto \\ *ab\mapsto \\ \to * \end{cases}$$

3. Не используя операторы цикла и перехода и используя для вывода ***только*** вывод символов, описать процедуру REV(N), которая печатает в обратном порядке неотрицательное целое число N. Например, при N=1234 нужно напечатать 4321.

4. Существуют ли два АВЛ-дерева, у которых высота h ($0\le h\le 10$) одинакова, а число вершин различается на 800? Приведите ответ (да/нет) и его обоснование.

5. Описать переменную P и, если необходимо, другие переменные, а также выписать операторы, присваивающие P следующее значение:



6. Считая известными понятия *<выражение>*, *<константа>* и *<оператор>*, дать определение понятия *<оператор варианта>* в виде синтаксической диаграммы.

7. На плоскости (X,Y) нарисовать область, в которой булевское выражение ((X>0) = (Y<1)) принимает значение TRUE.

8. Какие изменения следует сделать ***только*** в операторах присваивания, имеющихся в теле программы и в теле процедуры, чтобы на печать были выданы числа: 11, 42, 7, 8. Изменить можно ***не более трех*** операторов. Отметьте изменения непосредственно в тексте программы.

```
program Print(output);
  var A, B, C, D: integer;
  procedure P(X: integer; var A: integer);
    var C: integer;
  begin
    X:=5;              A:=6;
    C:=D+B;            D:=8
```

```
    end;
{-------------------------------------------------------------------------}
begin
  A:=1;              B:=2;
  C:=3;              D:=4;
  P(A, B);
  write(A, ' ', B, ' ', C, ' ', D)
end.
```

9. type Natur = 1..maxint;
   tree = ↑ node; {указатель на вершину}
   node = record left, right: tree; key : Natur end; {вершина}

Привести описание процедуры delete(T,K), которая удаляет из дерева сравнений T (типа tree) вершину с ключом K (типа Natur), если такая вершина есть в дереве T.

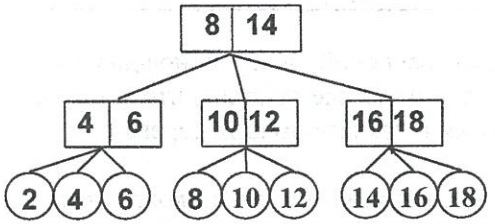
10. const n = ... ; { n > 1 }
    type A = array [1..n] of 1..maxint;

Считая, что элементы массива M (типа A) расположены по неубыванию (M[i]<=M[i+1]), привести рекурсивное описание булевской функции SUM(M,S), которая определяет, есть ли в массиве M набор чисел (без повторений), сумма которых равна заданному целому числу S.

3.8. ВАРИАНТ 2002

1. Дать описания типов для списков и описать процедуру Reorg12(L1,L2), осуществляющую преобразование (L1 → L2) заданного однонаправленного списка L1 элементов некоторого типа T в двунаправленный список L2. В полученном списке L2 должен быть сохранен порядок следования элементов, ссылка на предыдущий элемент для первого и ссылка на следующий элемент для последнего – nil. Заглавных звеньев нет. Вся освобождаемая память должна возвращаться в список свободной памяти.

2. В соответствии с алгоритмом вставки нового элемента в 2-3 дерево осуществить добавление ключа 11 в указанное 2-3 дерево. Нарисовать полученные на этапах перестройки деревья и результирующее 2-3 дерево.



3. Доказать, что при любом описании переменных **X** и **Y** в правильной программе на языке Паскаль не может встретиться (вне строк и комментариев) следующий оператор присваивания:

   Y := X[Y] + Y/2;

4. На плоскости (X,Y) нарисовать область, в которой булевское выражение

   (abs(x)+abs(y) < 2) and (trunc(x) = round(x))

принимает значение TRUE

5. var x, y: integer;

Для каждого из следующих пяти операторов укажите либо “ошибка”, либо то, что будет напечатано:

1) write(odd(sin(x-x)))
2) write(odd(cos(y-y)))
3) write(ord(x-y) = x–y)
4) write(y=y)
5) write(‘3.0E+3’ = ‘3000.0’)

6. Дайте также определение двоичного дерева сравнений.

Перемешанная таблица T[0..12] с функцией первичного перемешивания I = K mod 13 и с функцией вторичного перемешивания I = (I+5) mod 13 (где I – адрес, а K - ключ) заполнена следующими ключами:

Адрес	0	1	2	3	4	5	6	
Ключ	42	40		45			32	

	7	8	9	10	11	12
		66			27	

Как выглядит дерево сравнений, построенное для этих же ключей, если ключи поступали в том же порядке, что и в таблицу, и если не было балансировки и трансформации дерева?

7. Считая известными понятия *<имя>* и *<константа>*, приведите определение понятия *<раздел определения констант>* в виде синтаксической диаграммы.

8 Напишите нормальный алгоритм Маркова, в котором не более пяти правил подстановки и который применим ко всем словам в алфавите **A**={***a, b***}, кроме слова ***aba***.

9. Существуют ли два АВЛ-дерева, у которых высота h ($0 \le h \le 11$) одинакова, а число вершин различается на 1712? Приведите ответ (да/нет) и его четкое обоснование. (Замечание: высота дерева из одного узла равна 1).

10.
```
const N = …;   {N ≥ 1};
  type MASS = array[1..N] of integer;
       tree =↑node;  {указатель на вершину};
       node = record key: integer; left, right: tree end;
```
Описать функцию TR(K), которая по заданному массиву K (типа MASS) из *упорядоченных* по убыванию различных ключей строит *дерево сравнений* высоты $O(\log_2 N)$ и возвращает ссылку на построенное дерево. Операторы цикла и перехода не использовать.

4. ОТВЕТЫ, УКАЗАНИЯ И РЕШЕНИЯ

4.1. Вариант 1995

1. а) Любой алгоритм может быть реализован соответствующей машиной Тьюринга.

б) Понятие "любой алгоритм" не имеет точного определения, поэтому нельзя доказать тезис в целом.

в) Доводы:

- доказано, что композиции, разветвления и циклы алгоритмов (машин Тьюринга) являются алгоритмами (машинами Тьюринга), что дает основание строить сложные машины Тьюринга на основе более простых;
- все известные алгоритмы реализуются соответствующими машинами Тьюринга, и пока никто не придумал контрпримера, опровергающего этот тезис;
- разные уточнения понятия алгоритма (машина Тьюринга, машина Поста, нормальные алгоритмы Маркова и другие) оказались эквивалентными.

2. ОТВЕТ: нет.

Для обоснования ответа достаточно привести слово w из $\mathbf{A}^*$, для которого $P(Q(w)) \neq R(w)$. Принцип построения такого слова: для него оба правила подстановки из R должны применяться вперемежку, тогда как при последовательном применении сначала Q, а затем P этого не должно быть.

Один из возможных вариантов: $w = cab$, т.к.

$P(Q(cab)) = P(cab) = acb$

R: $cab \to acb \to abc$, т.е. $P(Q(w)) \neq R(w)$.

3. 1) -5 2) true 3) '9' 4) 0.0 (но не 0) 5) ошибка

4. dispose(p):

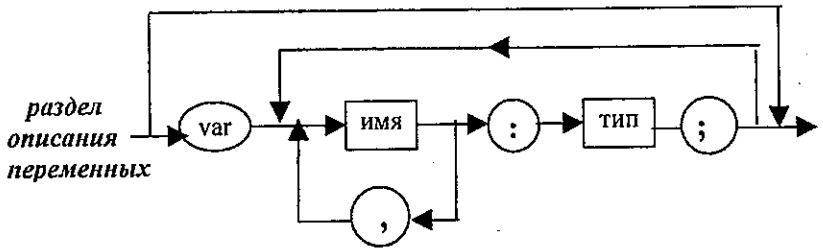
1) переменная p должна иметь значение, и это значение должно отличаться от nil, т.е. ссылочная переменная p должна ссылаться на какую-то динамическую переменную типа T;

2) процедура dispose(p) уничтожает эту динамическую переменную (освободившаяся память присоединяется к списку свободной памяти);

3) после dispose(p) значение p считается неопределенным.

5. Будет напечатано: 1 6 3 8

6.



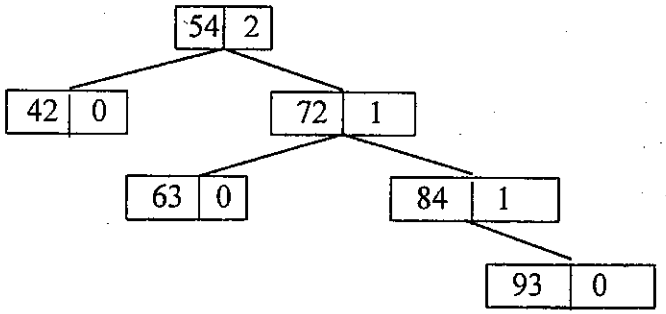
7.
```
function constr(N: неотр): список;
  var p: список;
begin
  if N=0 then constr:=nil else
    begin
      new(p); p↑.элем:=N; p↑.след:=constr(N-1);
      constr:=p
    end
end;
```

8. Таблица после вставки ключей 19, 27, 23, 6, 18:

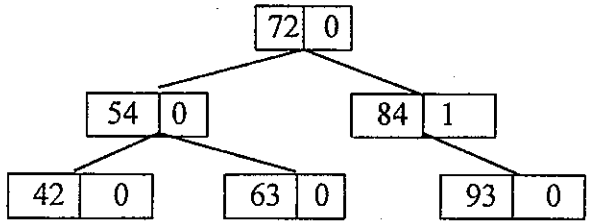
Адрес	0	1	2	3	4	5	6
Ключ		27				6	19

7	8	9	10	11	12
		18	23		

9. а) Дерево после вставки элемента:



б) Окончательное АВЛ-дерево:



10. 3 A B + C * H - * D F - 6 / A * +

4.2. Вариант 1996

1. Ответ: нет.

Алгоритм зацикливается на любом слове, содержащем букву *b* (см. второе правило подстановки). Поскольку в записи самого алгоритма присутствует буква *b*, то алгоритм на этой записи зацикливается, т.е. не является самоприменимым.

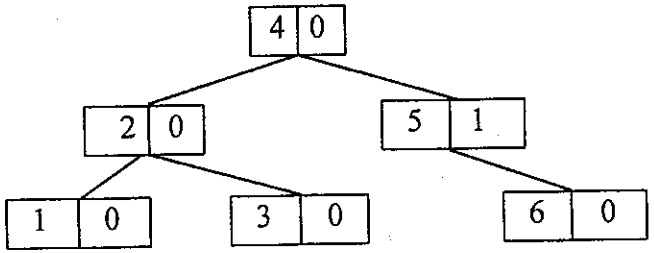
2. $R: \{\mapsto a$

3. Совершенного дерева с числом узлов, равным 9, не существует. Число узлов N в совершенном дереве высоты h определяется по

формуле N (h) = 2^h–1, поэтому существуют совершенные деревья с числом узлов 0, 1, 3, 7, 15, 31, …, но не с 9 узлами.

4. r = a + (b +. c) = 0
   p = (a + b) + c = c = +01–1002

5. АВЛ-дерево, построенное в соответствии с алгоритмом построения АВЛ-деревьев, имеет следующий вид:



6.

```
procedure P(var X: M;  S: integer);
      {Вспомогательная функция,  решающая задачу
       для элементов массива X с индексами от 1 до i }
   function Found(S, i: integer): boolean;
    label 1;
    var k: integer;
   begin
      if S = 0  then  Found := true else
         begin
           Found := false;
           for k := i downto  1 do
             if X[k] > S then goto 1 else
                if Found(S – X[k], k – 1) then
                  begin write(k,' ');  Found:=true;  goto 1  end;
         end;
     1:
   end;  {of  Found}
begin
   if not Found(S, N) then writeln ('NO') else writeln
```



```
 end;
```

7. Ответ: да.
   Например, указанный оператор присваивания верен при следующих описаниях:

```
 type ссылка = ↑элемент;
       T = record  p: boolean end;
       элемент = record H : array [boolean] of T end;
 var r: array ['a'..'z'] of ссылка;
```

8. 1) true 2) ошибка 3) 0.4 4) false 5) true

9.

```
 procedure DEL(var L: список);
   begin
     if L↑.след = nil then begin dispose(L); L:= nil  end
                    else  DEL(L↑.след)
   end;
```

10. Будет напечатано: 1 3 2

4.3. Вариант 1997

1.Возможны различные решения. Один из возможных алгоритмов:

$$\{\ \ b \to bb$$

Приведенный алгоритм зацикливается на своей записи, т.к. она содержит букву *b*, и применим к любому слову, не содержащему букву *b* .

2. Только одна композиция: R=T(U) .

3. Будет напечатано: 1 18 3 13

4. Да. Последовательность поступления ключей в таблицу:
   18, 7, 20, 22, 29, 31, 17

5. Определение:

1) Пустое дерево есть дерево Фибоначчи с высотой 0.

2) Дерево из одного узла есть дерево Фибоначчи с высотой 1.

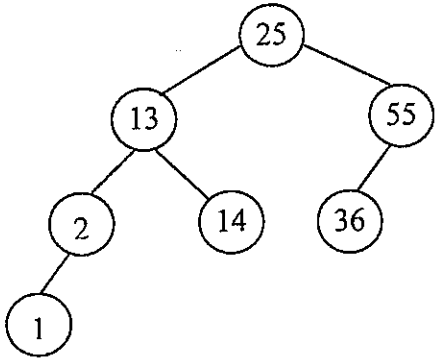
3) Если T_{h-1} и T_{h-2} - деревья Фибоначчи с высотой h-1 и h-2, то

$$T_h = < T_{h-1}, X, T_{h-2} >$$

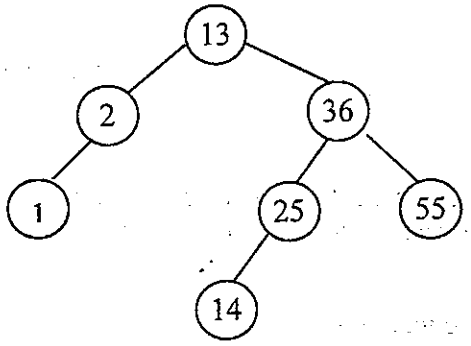
есть дерево Фибоначчи с высотой h и корнем X.

4) Никакие другие деревья не являются деревьями Фибоначчи.

а) Дерево Фибоначчи с ключами 1, 2, 13, 14, 25, 36, 55:



б) Можно привести разные АВЛ-деревья, например, такое:



6.

```
function AVL(t: tree): boolean;
   var h: integer; { h - высота дерева }
      {вспомогательная функция: }
```



```
function avlsub(t: tree;  var h: integer): boolean;
   {определяет, является ли t АВЛ-деревом, и
      возвращает его высоту h }
  var p1, p2: integer;
begin
   p1:=0;  p2:=0;  avlsub:= true;
   if  t = nil  then  h:=0  {пустое дерево -> АВЛ-дерево с h=0}
     else  {дерево непустое}
   if (t↑.lev=nil) and (t↑.prav= nil)
     then  h:=1 {только корень -> АВЛ –дерево с h=1}
     else  {есть поддеревья}
   if  t↑.lev= nil  then {левое пусто}
       begin avlsub:=avlsub(t↑.prav,p2); h:=p2+1 end
      else
   if t↑.prav=nil then {правое пусто}
       begin avlsub:=avlsub(t↑.lev,p1);  h:=p1+1 end
      else { оба поддерева непустые }
        begin
          avlsub:=avlsub(t↑.lev,p1) and avlsub(t↑.prav,p2) and
                  (abs(p1-p2) <= 1);
          if p1>p2 then h:=p1+1 else h:= p2+1
        end
  end;  {of AVLSUB }
begin
  AVL:= avlsub(t,h)
end;  {of AVL}
```

7. Нет.

Поскольку вещественные числа представляются в ЭВМ приближенно, то величина h=1/19 будет представлена, скорее всего, приближенно, из-за чего величина X+i*h никогда не будет точно равна B и while-цикл в программе *б)* будет выполняться бесконечно, эта программа зациклится. Целые же числа представляются точно, поэтому в программе *а)* for-цикл проработает ровно 20 раз и программа остановится.

8. Да.

Возможны различные верные описания, например, такое:

```
const true = 2;
var x, y: array [1..2 ] of record x: integer end;
```

9. RESET(F):

1) Маркер файла устанавливается на начало файла F.

2) В буферную переменную файла копируется значение первого элемента файла (если файл пуст, то значение этой переменной будет неопределенным).

3) Устанавливается режим чтения для файла F.

10.

```
procedure PATH;    {M – глобальная переменная}
  var r, i, j: integer;
      res: array[1..N] of 1..N;   {пройденный путь}
      res1: set of 1.. N ;   {множество пройденных городов}
      fail: boolean;

  function forw(j1:  integer): boolean;
    {сделать шаг вперед из города i в город с номером > j1;
     переходы из i в города 1, 2, ..., j1 уже исследованы }
  begin
    while (M[i, j1] = 0) and (j1 < N ) do  j1:= j1+1;
    if ((j1 = N ) and (M[i,n] = 1) or (j1 < N )) and (not  (j1 in res1))
       then
         begin
           i:=j1; j:=1; forw:=true;
           r:= r+1; res[r]:=i;
           res1:= res1+[i]
         end
       else
        if j1< N then forw:=forw(j1+1) else forw:= false
  end;   {of forw}

  procedure back;
    {откат назад из  i строки  j столбца }
  begin
   if (r = 1) and ( j>= N)
     then fail:= true {глобальный неуспех - исчерпаны все
                       варианты возврата }
```



```
    else
      begin
        j:=res[r]; res1:= res1-[res[r]];
        r:=r-1;  i:=res[r];
        if j = N  then back  {откат выше }
                  else  j:=j+1
      end
  end;   {of back}

begin
  r:=1;
   {индекс последнего города посещения в массиве res }
  res1:=[1];  res[1]:= 1;  i:= 1;  j:= 2;
  fail:= false;
  while (r < N) and not fail do
    if not forw(j) then { нельзя продвинуться вперед  -->
                          откат назад }back;
    if r= N then  {печать пути (номеров городов)}
     for r:=1 to N do write(' ', res[r])  else write(' Неуспех')
end;
```

4.4. Вариант 1998

1. Возможный пример:

$$\begin{cases} * \to * \\ \mapsto \end{cases}$$

2 Два алгоритма **R** и **Q** эквивалентны в алфавите **A**, если:

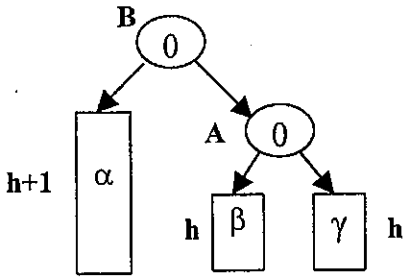
а) на множестве A^* (всех слов из символов алфавита A) области применимости алгоритма R и алгоритма Q совпадают;

б) для любого слова *w* из области применимости $\mathbf{R}(w)=\mathbf{Q}(w)$.

3. Будет напечатано: 12 , 13 , 14 , 15 ,

4. Занесение ключа произведено в поддерево α.

После корректировки характеристик дерево имеет следующий вид (в узлах проставлены новые характеристики):



5. Описание таблицы:

```
type next = ↑E;
       E = record  K: ключ;  S: тело;  p: next  end;
        {эта запись - элемент списка переполнения,
          p - ссылка на следующий элемент}
var tab: array [ 0..N ] of next;
```

В начале работы все элементы таблицы tab должны быть заполнены ссылками на элементы типа E, у которых значения полей K и S могут быть любыми, а значение поля p равно nil.

6.
```
type S = set of <базовый тип множества>;
var  A, B : ↑S;
 ....
A↑ := A↑ * B↑;
```

7. Будет напечатано: 1 23 3 15

8.
```
procedure free(p: pp);
begin
  if p <> nil then
      begin  free(p↑.лев);  free(p↑.прав);  dispose(p)  end
end;
```

9.
```
procedure PF(h: integer; p: pp);
```



```
begin
 if h=1 then
  begin
    free(p↑.лев);  p↑.лев:=nil;
    free(p↑.прав);  p↑.прав:= nil
  end
      else
 if h=2 then
    begin free(p↑.лев); p↑.лев:=nil; PF(h-1, p↑.прав) end
 else  {h>2}
    begin PF(h-2, p↑.лев); PF(h-1, p↑.прав)  end
end;
```

10.
```
begin  q.a := succ(p.b);  d:= q.c.e  end
```

4.5. Вариант 1999

1. Рассмотрим работу укорачивающего алгоритма над *любым* исходным словом ω (пусть длина слова |ω| = n). На каждом шаге работы такого алгоритма возможна одна из следующих ситуаций:

- ни одно из его правил подстановки не применимо, тогда алгоритм заканчивает работу,
- применяется одно из правил подстановки, при этом укорачивающий алгоритм только уменьшает длину текущего обрабатываемого слова. После применения правила алгоритм либо останавливается (если было применено заключительное правило), либо переходит к следующему шагу. Число таких шагов не больше числа символов (n) в исходном слове.

Следовательно, при обработке любого заданного слова длины n укорачивающий алгоритм остановится через конечное число ($\leq n$) шагов, т.е. он применим к любому слову.

В связи с этим, решение проблемы останова для укорачивающих алгоритмов сводится к предъявлению алгоритма, который для **любого** заданного слова получает результат (например, слово ДА) и прекращает работу. Такой алгоритм, очевидно, существует.

2. Ответ: Да.

Композицию указанных алгоритмов представляет следующий алгоритм:

$$\begin{cases} a \to \\ bc \mapsto cb \end{cases}$$

Он удаляет все буквы a в слове, заменяет первое вхождение bc на cb и останавливается. (Замечание: возможно более громоздкое описание композиции машиной Тьюринга.)

3. Порядок занесения ключей в таблицу T1 такой:

19, 6, 23, 27, 18.

После дополнения ключами из T1 таблица T2 будет иметь вид:

Адрес	0	1	2	3	4	5	6	
T2	6	40			23	14	19	

	7	8	9	10	11	12
		27	5	36		18

4. Ответ: высота равна 10.

Дерево Фибоначчи высоты h имеет число вершин N(h), равное величине F_{h+2}-1 (где F_{h+2} - число Фибоначчи с номером h+2), и является АВЛ-деревом с минимальным числом вершин при заданной высоте h.

Поскольку

F_1=1, F_2=1, F_3=2, F_4=3, F_5=5, F_6=8, F_7=13, F_8=21,

F_9=34, F_{10}=55, F_{11}=89, F_{12}=144, F_{13}=233, …,

то N(10)=F_{12}-1=144-1=143 < 145, а N(11)=F_{13}-1=233-1=232 > 145. Следовательно, есть АВЛ-дерево высоты 10 и с числом вершин 145, но нет АВЛ-дерева высоты 11 с таким числом вершин.

5.

```
{ рекурсивный вариант описания: }
function modify(S: список): список;
begin
  if S↑.след = nil then  {список из одного звена}
    begin dispose(S);  modify:=nil  end
  else {более 1 звена → ссылка на список не меняется}
    begin  modify:=S;  S↑.след:=modify(S↑.след)  end
end;
```


6.

```
function SUM(T: pp;  S: integer): boolean;
begin
  SUM:= false;
  if S>=0 then
     if T=nil then  SUM := S=0
           else
        if SUM(T↑.лев, S-T↑.элем)
             then SUM:=true
             else SUM:= SUM(T↑.прав, S-T↑.элем)
end;
```

7. Будет напечатано: 21 ; 1 ; 16 ; 3 ; 14

8. а) Нет б) Нет в) Да г) Да д) Нет

9.

```
program MAXIMUM (input, output);
  function MAX: integer;
    var a, b: integer;
  begin
    read(a);  {очередное число}
    if a=0 then MAX:=0 else
      begin
        b:=MAX;  {наибольшее из последующих чисел}
        if a > b then MAX:=a else MAX:=b
      end
  end;  {of  MAX}
begin
   writeln('Максимум  = ', MAX)
end.
```

10. Да. Например, при таком описании:

```
type  S  = ↑T;
```

```
        T = record E: char; X: S end;
var  X: S;
```

4.6. Вариант 2000

1. Для пустого слова и всех слов, содержащих хотя бы один символ 0, алгоритм Т применим, а для непустых слов из одних символов 1 – неприменим.

2. Возможный алгоритм:

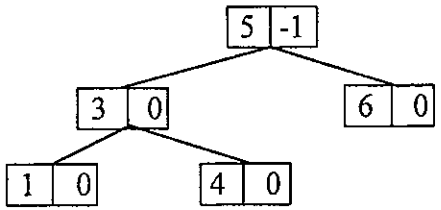
$$\begin{cases} ab\to \\ ba\to \\ \mapsto \end{cases}$$

3. Заданные ключи таковы, что при записи их в таблицу Т возникает только одна коллизия (59, затем 42), для остальных ключей порядок их занесения в таблицу Т не влияет на их расположение в Т. При обратном порядке поступления ключей (42, 59) таблица Т имеет вид:

Адрес	0	1	2	3	4	5	6	7	
Ключ		35			38			24	

	8	9	10	11	12	13	14	15	16
	42					59	48		

4. АВЛ-дерево, построенное в соответствии с алгоритмом построения АВЛ-деревьев для заданной последовательности ключей, будет иметь следующий вид (в узле слева – ключ, справа – характеристика):



5.

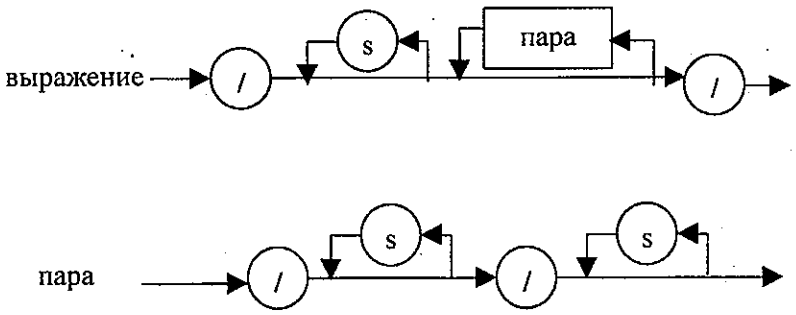
```
function NOTEMP(var f: text): integer;
{ функция, вычисляющая число непустых строк в
    текстовом файле f }
  var K : integer;
begin
  reset(f);
  K:=0;
  while not eof(f) do
    begin  if not eoln(f) then K:=K+1; readln(f) end;
  NOTEMP:= K;
end;
```

6.

```
function TR(var K: MASS): tree;
  function F(i, j: integer): tree;
   {построение поддерева для элементов массива K
    с индексами от i до j }
    var m: integer;   p: tree;
  begin
    if i>j then F:= nil else
      begin
        new(p); F:=p;
        if i=j then {один узел}
          begin p↑.key:=K[i];  p↑.left:= nil;  p↑.right:= nil end
        else
          begin
            m:= (i+j+1) div 2;
            p↑.key:=K[m];
            p↑.right:= F(i,m-1);  p↑.left:=F(m+1,j)
          end;
      end
 end; {of F}
begin
  TR:= F(1, n);
end;
```

7. Ошибка: в операторе варианта задан селектор (i+k+10) вещественного типа, т.к. i – переменная типа real.

8.



9. Будет напечатано:

11 ; 10 ; 9 ; 8 ; 7 ;

10. Ответ:

а) Да. B – типа boolean; A – ссылка на integer; C – типа integer;

б) Нет. Поскольку в правой части оператора присваивания указано отношение, то переменная R↑ должна быть логического типа, но для логических величин операция + не определена.

4.7. Вариант 2001

1.Возможны различные решения. Один из возможных алгоритмов:

$$\begin{cases} \| \rightarrow \\ | \rightarrow | \end{cases}$$

2. а) Два алгоритма эквивалентны над алфавитом **A**, если их области применимости совпадают и если для каждого слова *w* из области применимости оба этих алгоритма получают одинаковый результат;

б) эквивалентны U и Q (удаляют первое вхождение *ab*);
эквивалентны T и R (заменяют все вхождения *b* на *a*).

3.
```
procedure REV(N: integer);
begin
```



```
  if N < 10 then write(chr(ord('0')+N))
            else
              begin
               write(chr(ord('0')+(N mod 10)));
                REV(N div 10)
              end
end;
```

4. Да.

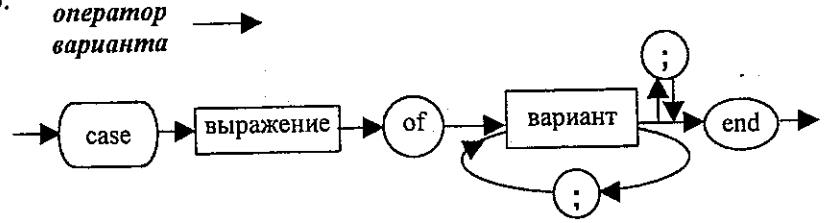
При высоте h=10 такие два АВЛ-дерева существуют.

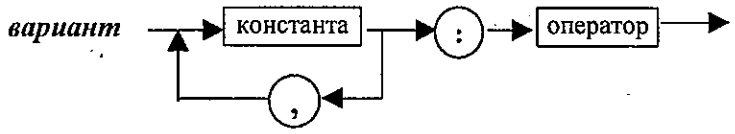
Например, при h=10 дерево Фибоначчи имеет 143 вершины ($N(h) = F_{h+2} - 1 = F_{12} - 1 = 144-1 = 143$), а совершенное дерево – 1023 ($N(h) = 2^h - 1 = = 2^{10} - 1 = 1024-1 = 1023$); оба этих дерева являются АВЛ-деревьями и различаются при этом по числу вершин на 880, что больше 800.

5. Возможный вариант описания:

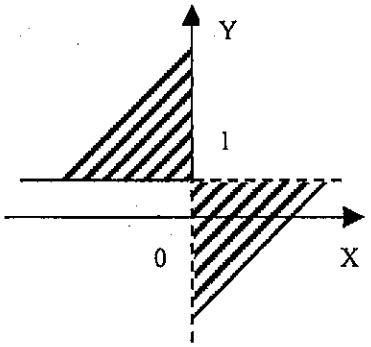
```
type SS = ↑rec;
     rec = record L, R: SS;   {левая и правая ссылки }
                  Mass: array[1..3] of boolean;
                  Numb: integer
           end;
var P, Q: SS;
  . . .
new(P); new(Q);  {создать обе записи}
    {Формирование первой записи}
P↑.L:=nil;  P↑.R:= Q;  P↑.Numb:= 1;
P↑.Mass[1]:=true;  P↑.Mass[2]:=false;  P↑.Mass[3]:=true;
    {Формирование второй записи}
Q↑.L:= P;   Q↑.R:= P;  Q↑.Numb:= 18;
Q↑.Mass[1]:=false; Q↑.Mass[2]:=true; Q↑.Mass[3]:=true;
  . . .
```

6.





7. Выражение имеет значение TRUE в двух случаях:
   а) X>0 и Y<1 (true = true)
   б) X≤0 и Y≥1 (false = false)



8. Приведенная программа печатает: 1 , 6 , 3 , 8, а нужно: 11, 42, 7, 8
   Изменения в ней показаны ***в фигурных скобках***:

```
. . .
begin X:=5;   A:=6; {42}    C:=D+B;   D:=8  end;
begin  A:=1; {11}     B:=2;     C:=3; {7}     D:=4;
. . .
```

9. *Указания к реализации процедуры delete*:
   а) В заголовке процедуры:
      procedure delete(var T: tree, K: natur);
параметр T обязательно задается как параметр - переменная (с **var**).
   б) При поиске вершины с ключом K обязательно сохранять ссылку на «родителя», чтобы затем, найдя узел-«потомок» с ключом K, можно было провести правильно удаление «потомка»;

в) Если ключ K найден, то возможен один из трех следующих вариантов:
   1) найденный ключ K – ключ в листовой вершине. Тогда применить dispose и записать nil в «родителе»;
   2) K – ключ в вершине, имеющей одного правого (левого) потомка. Тогда найти минимальный (максимальный) ключ в правом (левом) поддереве, переместить его на место ключа K, удалив со старого места (как в случае 1).
   3) K – ключ в вершине с двумя потомками. Тогда последовательно выполнить 2 и 1.

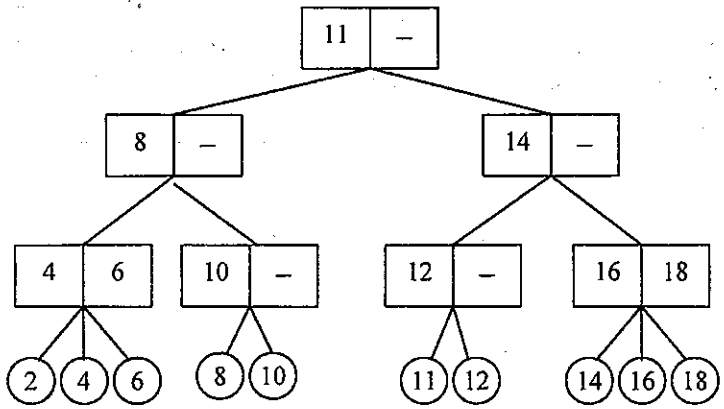
10. Решение аналогично решению задачи №6 из раздела 4.2.

4.8. Вариант 2002

```
1. type sp1 =↑node1;  sp2=↑node2;
        node1= record  el: T;  next: sp1  end;
        node2= record  el: T;  next, pred: sp2  end;
{ Процедура преобразования заданного однонаправленного
списка L1 в двунаправленный список L2.}
   procedure Reorg12(var L1: sp1;  var L2: sp2);
     var q1, p1: sp1;  q2, p2: sp2;
   begin
     if L1=nil then L2:=nil else
       begin
         p1:=L1;  {текущее звено в L1}
         q2:=nil;  {предыдущее звено в L2}
         repeat
           new(p2);   {новое звено в L2}
           p2↑.el:=p1↑.el;
           p2↑.pred:=q2;
           if q2=nil then L2:=p2 else q2↑.next:=p2;
           q2:=p2;
           q1:= p1;  p1:=p1↑.next;  dispose(q1)
         until p1=nil;
         q2↑.next:=nil;
```

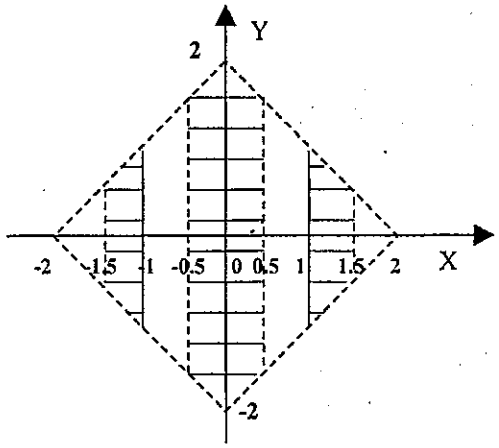
```
    end
  end;
```

2. Результирующее 2-3 дерево:



3 Справа в операторе присваивания стоит вещественное выражение, поскольку операция / дает вещественный результат. Следовательно, Y – вещественная переменная, но в переменной X[Y] она используется как индекс, а вещественные индексы недопустимы.

4.



5. 1) ошибка 2) ошибка 3) true 4) true 5) false

6. Двоичное дерево сравнений:

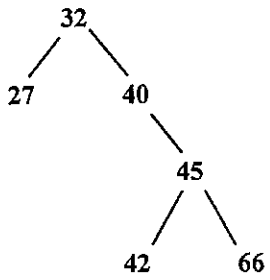
1) пустое дерево;
2) дерево из одной вершины;
3) дерево, в котором для любой вершины все ключи в ее левом поддереве меньше ключа в этой вершине, а все ключи в ее правом поддереве больше ключа в данной вершине.

Порядок поступления ключей в таблицу: сначала 40, 32 - в любом порядке (возможны 2 варианта), затем 27, 45, 66, 42. Следовательно, возможны ДВА варианта требуемых деревьев сравнений :

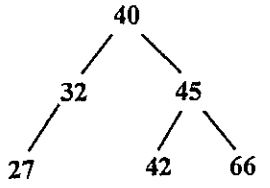
а) для последовательности ключей 40, 32, 27, 45, 66, 42, либо

б) для последовательности ключей 32, 40, 27, 45, 66 , 42:

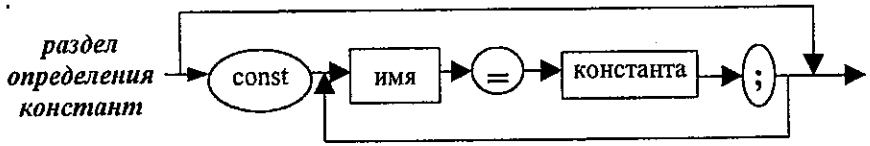
Вариант а)



Вариант б)



7.



8. Таких алгоритмов может быть несколько, как в расширенном алфавите, так и в исходном. Возможные варианты :

$$\begin{cases} *aba \to aba+ \\ +a \mapsto a \\ +b \mapsto b \\ * \mapsto \\ \to * \end{cases} \qquad \begin{cases} aaba \mapsto aaba \\ baba \mapsto baba \\ abaa \mapsto abaa \\ abab \mapsto abab \\ aba \to aba \end{cases}$$

9. Да.
При высоте $h = 11$ такие АВЛ-деревья существуют. При этой высоте АВЛ-дерево с минимальным числом вершин - дерево Фибоначчи - имеет 232 вершины ($N(h) = F_{h+2} - 1 = F_{13} - 1 = 233 - 1 = 232$), а АВЛ-дерево с максимальным числом вершин (совершенное дерево) имеет 2047 вершин ($N(h) = 2^h - 1 = 2^{11} - 1 = 2048 - 1 = 2047$), т.е. эти деревья различаются на 1815 вершин, что больше 1712.

10. См. решение задачи № 6 в разделе 4.6. (Описание функции TR(K)).

СОДЕРЖАНИЕ

Методическое пособие

Иванников В.П., Корухова Л.С., Пильщиков В.Н.

ПИСЬМЕННЫЙ ЭКЗАМЕН ПО КУРСУ “АЛГОРИТМЫ И АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ”

Издательский отдел
Факультета вычислительной математики и кибернетики
МГУ им. М.В. Ломоносова

Лицензия ИД N 05899 от 24.09.01 г.

Напечатано с готового оригинал-макета
в издательстве ООО “МАКС Пресс”
Лицензия ИД N 00510 от 01.12.99 г.
Подписано к печати 18.11.2002 г.
Формат 60x90 1/16. Усл.печ.л. 3,0. Тираж 600 экз. Заказ 877.
Тел. 939-3890, 939-3891, 928-1042. Тел./Факс 939-3891.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус, 627 к.

orlov@cs.msu.su