Московский государственный
университет им. М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Трифонов Н.П., Пильщиков В.Н.

# Задания практикума на ЭВМ

(1 курс)

Москва
2001

УДК 681.325.5
ББК 22.18
Т67

**Трифонов Н.П., Пильщиков В.Н.**

Задания практикума на ЭВМ (1 курс). Учебное пособие, 2-е исправленное издание. — М.: МГУ, 2001. — 32 с.

Издательский отдел факультета ВМК (лицензия ЛР №040777 от 23.07.96),

Приводятся описания заданий практикума на ЭВМ для студентов 1 курса факультета вычислительной математики и кибернетики МГУ. Эти задания относятся к языку программирования Паскаль и языку ассемблера для персональных компьютеров.

В разработке заданий принимали участие И.А Волкова, Е.В. Зима, В.В. Игнатов, В.Н. Пильщиков, В.И. Родин, Т.В. Руденко, Н.П. Трифонов.

В новом издании исправлены замеченные ошибки и неточности первого издания (1992 г.), в качестве используемой версии языка Паскаль взят язык Турбо Паскаль 7.0. При подготовке нового издания большую помощь оказала Е.А. Бордаченкова.

Рецензенты:

доц. Баула В.Г.
доц. Корухова Л.С.

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

**ISBN 5-89407-102-X**

# Задание 1. ЯЗЫК ПАСКАЛЬ. ВЫЧИСЛЕНИЕ КОРНЕЙ УРАВНЕНИЙ И ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

## 1.1. ПОСТАНОВКА ЗАДАЧИ

С заданной точностью $\varepsilon$ вычислить площадь плоской фигуры, ограниченной тремя кривыми, уравнения которых $y = f_1(x)$, $y = f_2(x)$ и $y = f_3(x)$ определяются вариантом задания.

При решении задачи необходимо:

— с некоторой точностью *eps1* вычислить абсциссы точек пересечения кривых, используя предусмотренный вариантом задания метод приближенного решения уравнения $F(x)=0$; отрезки, где программа будет искать точки пересечения и где применим используемый метод, определить вручную;

— представить площадь заданной фигуры как алгебраическую сумму определенных интегралов и вычислить эти интегралы с некоторой точностью *eps2* по квадратурной формуле, предусмотренной вариантом задания.

Величины $\varepsilon_1$ и $\varepsilon_2$ подобрать вручную так, чтобы гарантировалось вычисление площади фигуры с точностью $\varepsilon$.

## 1.2. ВАРИАНТЫ ЗАДАНИЯ

Во всех вариантах $\varepsilon = 0.001$.

### А. Уравнения кривых

$y = f_i(x)$:

| | | | |
|---|---|---|---|
| 1) | $f_1 = 2^x + 1$ | $f_2 = x^5$ | $f_3 = (1 - x)/3$ |
| 2) | $f_1 = 3(0.5/(x + 1) + 1)$ | $f_2 = 2.5x - 9.5$ | $f_3 = 5/x \quad (x > 0)$ |
| 3) | $f_1 = \exp(-x) + 3$ | $f_2 = 2x - 2$ | $f_3 = 1/x$ |
| 4) | $f_1 = \exp(x) + 2$ | $f_2 = -1/x$ | $f_3 = -2(x + 1)/3$ |
| 5) | $f_1 = 0.35x^2 - 0.95x + 2.7$ | $f_2 = 3^x + 1$ | $f_3 = 1/(x + 2)$ |
| 6) | $f_1 = 0.6x + 3$ | $f_2 = (x - 2)^3 - 1$ | $f_3 = 3/x$ |
| 7) | $f_1 = \ln(x)$ | $f_2 = -2x + 14$ | $f_3 = 1/(2 - x) + 6$ |
| 8) | $f_1 = \exp(x) + 2$ | $f_2 = -2x + 8$ | $f_3 = -5/x$ |
| 9) | $f_1 = 3/((x - 1)^2 + 1)$ | $f_2 = \text{sqrt}\,(x + 0.5)$ | $f_3 = \exp(-x)$ |
| 10) | $f_1 = 1 + 4/(x^2 + 1)$ | $f_2 = x^3$ | $f_3 = 2^{-x}$ |

### Б. Методы приближенного решения уравнений:

1) метод деления отрезка пополам;

2) метод хорд (секущих);

3) метод касательных (Ньютона);

4) комбинированный метод (хорд и касательных).

В. Квадратурные формулы:

1) формула прямоугольников;

2) формула трапеций;

3) формула Симпсона (парабол).

## 1.3. ТРЕБОВАНИЯ К ПРОГРАММЕ

1. В программе предусмотреть печать как площади заданной фигуры, так и абсцисс точек пересечения кривых.

2. Описать в программе процедуру *root* (*f, g, a, b, eps1, x*), вычисляющую с точностью $\varepsilon_1$ корень $x$ уравнения $f(x) = g(x)$ на отрезке $[a, b]$. (Если используется метод касательных или комбинированный метод, то у *root* должны быть еще параметры $f_1$ и $g_1$ — производные функций $f$ и $g$.)

3. Описать в программе функцию *integral* (*f, a, b, eps2*), вычисляющую с точностью $\varepsilon_2$ величину определенного интеграла от функции $f(x)$ на отрезке $[a, b]$.

4. Процедуру *root* и функцию *integral* следует предварительно протестировать.

## 1.4. ЛИТЕРАТУРА

[1]. Ильин В.А., Садовничий В.А., Сендов Бл.Х. Математический анализ. Т.1 — М.: Наука, 1985.

[2]. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. — М.: Наука, 1988.

[3]. Епанешников А.М., Епанешников В.А. Программирование в среде Turbo Pascal 7.0 — М.: «ДИАЛОГ-МИФИ», 2000.

## 1.5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

1. Для корректного применения предложенных методов приближенного решения уравнения $F(x) = 0$ (где $F(x) = f(x) - g(x)$) необходимо найти отрезок $[a, b]$, на котором уравнение имеет ровно один корень. Достаточное условие для этого таково: на концах отрезка функция $F(x)$ имеет разные знаки и на всем отрезке производная функции не меняет знак. Кроме того, для методов хорд и касательных, а также комбинированного метода обязательно требуется, чтобы на данном отрезке первая и вторая производные функции не меняли свой знак (не обращались в ноль).

2. В *методе деления отрезка пополам* определяется средняя точка $c$ отрезка $[a, b]$ и из двух отрезков $[a, c]$ и $[c, b]$ выбирается тот, на концах которого функция $F(x)$ имеет разные знаки. К выбранному отрезку применяется та же процедура. Процесс деления отрезков прекращается, когда длина очередного отрезка станет меньше требуемой точности $\varepsilon$; за корень уравнения можно принять любую точку этого отрезка.

3. В остальных трех методах следует различать два случая:

   случай 1 — первая и вторая производные функции $F(x)$ имеют одинаковый знак ($F'(x)F''(x) > 0$);

   случай 2 — эти производные имеют разные знаки.

В *методе хорд* концы $(a, F(a))$ и $(b, F(b))$ кривой $y = F(x)$ соединяются прямой линией и определяется точка пересечения этой линии с осью абсцисс:

$$c = \frac{aF(b) - bF(a)}{F(b) - F(a)}.$$

Далее выбирается отрезок $[c, b]$ в случае 1 или отрезок $[a, c]$ в случае 2 и к нему применяется та же процедура. Тем самым происходит постепенное приближение к искомому корню слева (в случае 1) или справа (в случае 2).

В *методе касательных* проводится касательная к кривой $y = F(x)$ в точке $(b, F(b))$ в случае 1 или в точке $(a, F(a))$ в случае 2 и определяется точка $c$ пересечения этой касательной с осью абсцисс:

$$c = d - \frac{F(d)}{F'(d)},$$

где $d = b$ в случае 1 и $d = a$ в случае 2. Далее проводится касательная к кривой в точке $(c, F(c))$ и процедура повторяется. Тем самым происходит приближение к искомому корню справа (в случае 1) или слева (в случае 2).

В методах хорд и касательных можно использовать следующий критерий завершения процесса приближения к корню. Если приближение «идет» слева, то на очередном шаге надо сравнить величины $F(c)$ и $F(c + \varepsilon)$: если они одного знака, то процесс продолжается, иначе на отрезке $[c, c + \varepsilon]$ имеется корень и потому процесс завершается. При приближении справа надо проверять знаки $F(c - \varepsilon)$ и $F(c)$.

В *комбинированном методе* одновременно применяется метод хорд и метод касательных, в связи с чем приближение к корню происходит с двух сторон. Критерий окончания — длина очередного отрезка меньше $\varepsilon$.

4. При использовании метода хорд, метода касательных или комбинированного метода процедура *root* должна самостоятельно распознавать, какой из двух случаев, указанных в п. 2, имеет место при текущем обращении к ней. Это можно сделать проверкой следующих двух условий:

   — функция возрастает или убывает;

   — график функции расположен выше хорды, соединяющей концы графика, или ниже.

   Поскольку производные $F'(x)$ и $F''(x)$ на отрезке $[a, b]$ не меняют знак, для проверки первого условия достаточно сравнить $F(a)$ с 0 (при $F(a) < 0$ функция возрастает). Для проверки же второго условия надо сравнить в какой-то внутренней точке отрезка значения функции и хорды; например, если взять среднюю точку $(a + b)/2$ отрезка, то соотношение $F\left(\frac{a+b}{2}\right) > \frac{F(a)+F(b)}{2}$ означает, что график функции расположен выше хорды. Если функция возрастает и ее график расположен ниже хорды или если функция убывает и ее график расположен выше хорды, то имеет место случай 1, иначе — случай 2.

5. Квадратурные формулы для приближенного вычисления интеграла $I$ от функции $F(x)$ на отрезке $[a, b]$ имеют следующий вид ($n$ — число разбиений отрезка $[a, b]$):

   *Формула прямоугольников*:

   $I \cong I_n = h(F_0 + F_1 + ... + F_{n-1})$, где $F_i = F(a + (i + 0.5)h)$, $h = (b - a)/n$

*Формула трапеций:*

$I \cong I_n = h\,(0.5F_0 + F_1 + F_2 + \ldots + F_{n-1} + 0.5F_n)$, где $F_i = F(a + ih)$, $h = (b - a)/n$

*Формула Симпсона* ($n$ — чётное):

$I \cong I_n = h/3\,(F_0 + 4F_1 + 2F_2 + 4F_3 + \ldots + 4F_{n-3} + 2F_{n-2} + 4F_{n-1} + F_n)$,

где $F_i = F(a + ih)$, $h = (b - a)/n$

6. Для обеспечения требуемой точности $\varepsilon$ при приближенном вычислении интеграла $I$ по квадратурной формуле нужно подобрать соответствующее число $n$ разбиений отрезка интегрирования. Известны формулы, выражающие $n$ через $\varepsilon$, но в эти формулы входят производные подынтегральной функции, что неудобно на практике. Поэтому для достижения требуемой точности обычно используется следующий метод: берется некоторое начальное число разбиений $n_0$ (например, 10 или 20) и последовательно вычисляются значения $I_n$ при $n$, равном $2n_0$, $4n_0$, $8n_0$ и т.д. Известно *правило Рунге*

$$| I - I_n | \cong p\,| I_n - I_{2n} |$$

(для формул прямоугольников и трапеций $p = 1/3$, для формулы Симпсона $p = 1/15$). Согласно этому правилу, когда на очередном шаге величина $p\,|I_n - I_{2n}|$ окажется меньше $\varepsilon$, в качестве приближенного значения для $I$ можно взять $I_n$ или, что лучше, $I_{2n}$.

7. При реализации функции *integral* следует учитывать, что в формулах трапеций и Симпсона в сумму $I_{2n}$ входят значения $F_i$, вычисленные ранее для суммы $I_n$, поэтому их не следует перевычислять заново.

8. В процедуре *root* и функции *integral* используются параметры-функции ($f$, $g$ и др.). В языке Турбо Паскаль такие параметры описываются следующим образом.

   В разделе типов необходимо описать т.н. функциональный тип:

```
type TF = function (x: real): real;
```

   (вместо *TP* и *x* можно использовать любые другие имена), в который автоматически включаются все описанные в программе вещественные функции от одного вещественного аргумента, а затем имя данного типа нужно указать в спецификации формального параметра-функции, например:

```
procedure root (f, g: TF;  a, b, eps1: real; var x: real);
```

   При обращении же к процедуре *root* или функции *integral* указываются, как обычно, имена фактических функций (не стандартных!), например:

```
root(f1, f2, -0.1, 3.5, 0.0001, x12);
```

   Что касается самих фактических функций ($f_1$, $f_2$ и др.), то перед их описанием необходимо разместить директиву {$F+} транслятору:

```
{$F+}
function f1 (x: real): real;  begin f1:=ln(x) end;
function f2 (x: real): real;  begin f2:=2*x+14 end;
. . .
```

# Задание 2. ЯЗЫК ПАСКАЛЬ. ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ.

## 2.1. ПОСТАНОВКА ЗАДАЧИ

### Варианты 1-11

Дана непустая последовательность слов, в каждом из которых содержится от 1 до 6 заглавных латинских букв; соседние слова разделены запятой, за последним словом следует точка.

Требуется ввести эту последовательность слов в память ЭВМ, преобразовав ее во внутреннее представление (см. ниже), а затем распечатать в алфавитном порядке:

1) все слова последовательности;
2) все слова с указанием для каждого из них его порядкового номера в исходной последовательности;
3) все различные слова с указанием для каждого из них числа его вхождений в исходную последовательность;
4) все различные слова с указанием для каждого из них порядкового номера его первого вхождения в исходную последовательность;
5) сначала все однобуквенные слова, затем все двухбуквенные слова и т.д.;
6) сначала все однобуквенные слова с указанием для каждого из них его порядкового номера в исходной последовательности, затем аналогичным образом все двухбуквенные слова и т.д.;
7) сначала все различные однобуквенные слова с указанием для каждого из них числа его вхождений в исходную последовательность, затем аналогичным образом все различные двухбуквенные слова и т.д.;
8) сначала все различные однобуквенные слова с указанием для каждого из них порядкового номера его первого вхождений в исходную последовательность, затем аналогичным образом все различные двухбуквенные слова и т.д.;
9) та же задача, что и в варианте 1;
10) та же задача, что и в варианте 3;
11) та же задача, что и в варианте 4.

В качестве внутреннего представления последовательности слов использовать:

— в вариантах 1-4 — однонаправленный список из слов, упорядоченных по алфавиту;

— в вариантах 5-8 — массив из 6 списков, в *k*-ом из которых хранятся *k*-буквенные слова, упорядоченные по алфавиту;

— в вариантах 9-11 — двоичное дерево поиска (в нем слева от каждой вершины-слова располагаются только те слова, что предшествуют ему по алфавиту, а справа — следующие за ним по алфавиту).

## Варианты 12-18

Дана символьная запись (см. ниже) многочлена (многочленов) от одной переменной $X$ с целыми коэффициентами. Требуется ввести этот многочлен в память ЭВМ, преобразовав во внутреннее представление (см. ниже), выполнить операцию, определяемую вариантом задания, и распечатать полученный результат.

Операции над многочленами:

12) вычислить значение заданного многочлена при заданном целочисленном значении переменной $X$;

13) проверить, имеет ли заданный многочлен хотя бы один целый корень (такими корнями могут быть только положительные и отрицательные делители свободного члена, а при нулевом свободном члене корнем является 0);

14) проверить на равенство два заданных многочлена;

15) получить многочлен, являющийся производной заданного многочлена по переменной $X$;

16) получить многочлен, являющийся суммой двух заданных многочленов;

17) получить многочлен, являющийся произведением двух заданных многочленов;

18) привести подобные члены в заданном многочлене, в котором могут повторяться одночлены с одними и теми же степенями $X$.

Исходный многочлен от переменной $X$ с целыми коэффициентами записывается как алгебраическая сумма одночленов любого из следующих видов (^ — возведение в степень):

```
aX^k,
X^k,
aX,
X
a
```

где $k$ и $a$ — целые числа ($k \geq 2$, $a \geq 1$). При этом по степеням $X$ одночлены могут быть не упорядочены, но одночлены одной и той же степени не повторяются (кроме варианта 18). За последним одночленом следует пробел — признак конца записи многочлена. (Особый случай: нулевой многочлен записывается как 0).

Примеры записи многочленов:

```
7x^30
x^8+1–139x+5x^46
```

Если результатом операции является многочлен, то он должен быть распечатан в указанном виде (без нулевых слагаемых, без коэффициентов 1 и без показателей степени 0 и 1), но обязательно по убыванию степеней $X$.

В памяти ЭВМ многочлен должен быть представлен в виде однонаправленного списка, в котором каждому одночлену соответствует звено, содержащее степень этого одночлена и его коэффициент. Звенья списка должны быть упорядочены по убыванию степеней, звеньев с нулевыми коэффициентами не должно быть.

### Варианты 19-22

Дана формула, содержащая переменную *X* (см. ниже). Требуется ввести эту формулу в память ЭВМ, преобразовав ее в двоичное дерево (см. ниже), выполнить операцию, указанную в варианте задания, и распечатать полученный результат.

Операции над формулами:w

19) по заданным формуле и целому числу вычислить значение этой формулы при значении переменной *X*, равном этому числу;

20) получить производную заданной формулы по переменной *X*;

21) напечатать заданную формулу в префиксном виде;

22) напечатать заданную формулу в постфиксном виде.

Под «формулой» понимается запись следующего вида:

```
<формула> ::= <число> | <переменная> | (<формула><знак><формула>)
<знак>    ::= + | - | *
<число>   ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<переменная> ::= X
```

Примеры формул:

```
5
((2*X)+(X–(8+X)))
```

Если результатом операции является снова формула, то она должна быть распечатана в таком же виде.

Формулу можно представить в виде двоичного дерева согласно следующим правилам:

— числу или переменной соответствует дерево из одной вершины, содержащей это число или переменную,

— формуле со знаком операции соответствует дерево, корневая вершина которого — знак, левое поддерево — соответствующее представление первого операнда, а правое поддерево — второго операнда.

Префиксной записью формулы (*a#b*), где # — знак некоторой операции, называется конструкция (*#ab*), а постфиксной записью — конструкция (*ab#*). Примеры записи формул:

| *обычная (инфиксная)* | *префиксная* | *постфиксная* |
|---|---|---|
| 7 | 7 | 7 |
| (a–(b*c)) | (–a(*bc)) | (a(bc*)–) |
| ((a–b)*c) | (*(–ab)c) | ((ab–)c*) |

## 2.2. ТРЕБОВАНИЯ К ПРОГРАММЕ

1. В программе должна быть предусмотрена проверка правильности задания исходной информации.

2. В вариантах 1-11 в программе должны быть определены процедура выделения очередного слова из исходной последовательности и процедура вставки слова в упорядоченный список (в дерево поиска). Кроме того, в вариантах 9-11 должна быть определена рекурсивная процедура печати слов, входящих в дерево.

3. Аналогичные процедуры (выделение очередного одночлена, вставка нового звена в упорядоченный список) должны быть определены в вариантах 12-18.

4. В вариантах 19-22 должны быть определены процедура ввода исходной формулы и построения соответствующего двоичного дерева, а также процедура вычисления значения формулы при заданной величине $X$ (в варианте 19) или процедура печати формулы в нужном виде (варианты 20-22); все эти процедуры должны быть рекурсивными.

## 2.3. ЛИТЕРАТУРА

[1]. Вирт Н. Алгоритмы + структуры данных = программы. — М.: Мир, 1985.

[2]. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. — М.: Наука, 1988.

[3]. Епанешников А.М., Епанешников В.А. Программирование в среде Turbo Pascal 7.0 — М.: «ДИАЛОГ-МИФИ», 2000.

## 2.4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

1. Вводить последовательность слов, многочлен или формулу следует посимвольно. Конец ввода определяется соответственно по точке, по пробелу или по балансу скобок. Ввод целого числа в вариантах 12 и 19 можно осуществлять с помощью процедуры read(n), где n — целочисленная переменная.

2. В вариантах задания, где в качестве внутреннего представления используются списки, следует упорядочивать списки (по алфавиту или по степеням одночленов) одновременно с вводом слов или одночленов: введенное слово или одночлен следует сразу вставлять на «свое» место в ранее упорядоченный список.

3. Для упрощения операций над списками, рекомендуется использовать списки с заглавными звеньями. В этом случае следует в начале выполнения программы построить список (списки) с одним заглавным звеном.

4. В вариантах 1-11 в звеньях списков (вершинах дерева) следует хранить не только слова, но и дополнительную информацию (порядковый номер слова в исходной последовательности или число его вхождений в эту последовательность). В частности, даже в вариантах 1 и 5 для каждого слова лучше иметь только одно звено, фиксируя в нем число вхождений этого слова в последовательность; при печати же надо продублировать слово данное число раз. В варианте 9 такой прием является единственно возможным, поскольку в дереве поиска не должно быть нескольких вершин с одним и тем же словом.

# Задание 3. ЯЗЫК ПАСКАЛЬ. МЕТОДЫ СОРТИРОВКИ.

## 3.1. ПОСТАНОВКА ЗАДАЧИ

Требуется реализовать два метода сортировки, определяемых вариантом задания, и провести их экспериментальное сравнение.

Под «сортировкой» понимается упорядочение элементов последовательности $X_1, X_2, ..., X_n$ ($1 \le n \le 100$) по неубыванию, т.е. такая перестановка элементов, после которой будет выполняться условие $X_i \le X_{i+1}$ для всех $i$. В качестве элементов $X_i$ использовать даты вида 26.11.94, 9.5.00 и т.п.

Сравнение реализованных методов сортировки нужно проводить на одних и тех же последовательностях. Следует рассмотреть последовательности разной длины (например, $n$ = 10, 20, 50, 100), используя при каждом $n$ по крайней мере следующие исходные расстановки элементов:

1) элементы уже упорядочены по неубыванию;
2) элементы упорядочены в обратном порядке (по невозрастанию);
3) элементы с нечетными индексами упорядочены по неубыванию, а с четными индексами — по невозрастанию;
4) одна случайная расстановка элементов;
5) другая случайная расстановка элементов.

Оценку методов производить по следующим двум параметрам:

— число сравнений элементов последовательности, выполненных в процессе упорядочения;

— число перемещений (перестановок пар элементов или пересылок элементов на новые места — в зависимости от метода), выполненных в процессе упорядочения.

Результаты экспериментов оформить в виде двух таблиц, например, такого формата (средние значения округлять до целой части):

Название метода сортировки

| n | параметр | номер последовательности 1 | 2 | 3 | 4 | 5 | среднее значение |
|---|---|---|---|---|---|---|---|
| 10 | сравнения | 9 | 45 | 33 | 25 | 30 | 28 |
| | перемещения | | | ... | | | ... |
| 20 | сравнения | | | ... | | | ... |
| | ... | | | | | | |
| | ... | | | | | | |

## 3.2. ВАРИАНТЫ ЗАДАНИЯ (методы сортировки)

Методы 1-5 — простые, но медленные (с числом сравнений порядка $n^2$), а методы 6-10 — сложные, но быстрые (порядка $n \log_2 n$). Названия методов даны по книге [1],

кроме метода 4. В скобках указаны номера книг по списку литературы и страницы, где приводятся описания данных методов.

1) Сортировка простыми вставками ([1] 101-103; [3] 95-96).
2) Сортировка бинарными вставками ([1] 103-104; [3] 97-98).
3) Метод пузырька ([1] 130-132; [2] 27-28; [3] 101-102).
4) Челночная сортировка ([2] 30-31).
5) Сортировка простым выбором ([1] 169-171; [2] 15-16; [3] 99-100).
6) Метод Шелла ([1] 105-107; [2] 37-40; [3] 105-107).
7) Быстрая сортировка, рекурсивный вариант ([3] 114-117).
8) Быстрая сортировка, нерекурсивный вариант ([1] 140-144; [2] 88-97; [3] 114-121).
9) Сортировка простым слиянием ([1] 198-199; [2] 106-111).
10) Сортировка естественным слиянием ([1] 193- 197; [2] 112-117).

## 3.3. ТРЕБОВАНИЯ К ПРОГРАММЕ

1. Сравнение двух элементов (дат) должно быть реализовано в виде логической функции, а перемещение (пересылка или перестановка) элементов — в виде процедуры. Как побочный эффект, при каждом обращении эти функция и процедура должны увеличивать на 1 значения глобальных счетчиков сравнений и перемещений, соответственно. Перед началом каждой сортировки эти счетчики нужно обнулять.

   (*Замечание*. Под одним «перемещением» в методах 1, 2, 9 и 10 понимается пересылка элемента на новое место, а в остальных методах — перестановка значений двух элементов.)

2. Каждый из предложенных методов сортировки должен быть реализован в виде процедуры от двух параметров: массива ($X$), в котором вначале находится неупорядоченная последовательность, а в конце работы процедуры должна оказаться упорядоченная последовательность, и длины ($n$) упорядочиваемой последовательности. Эти процедуры должны правильно работать при любом $n \leq 100$.

   Прежде чем проводить эксперименты, следует отладить каждую из этих процедур (например, на двух-трех последовательностях длины 10).

   Основная программа должна генерировать все необходимые последовательности, обращаться к процедурам сортировки и печатать таблицы результатов.

   (*Замечание*. Для генерации случайных последовательностей можно использовать функцию *random(k)* языка Турбо Паскаль, которая при каждом обращении к ней в качестве своего значения выдает новое случайное целое число из отрезка [0, $k$-1].)

## 3.4. ЛИТЕРАТУРА

[1]. Кнут Д. Искусство программирования для ЭВМ. Т.3. — М.: Мир, 1978.

[2]. Лорин Г. Сортировка и системы сортировки. — М.: Наука, 1983.

[3]. Вирт Н. Алгоритмы и структуры данных. — М.: Мир, 1989.

[4]. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. — М.: Наука, 1988.

[5]. Епанешников А.М., Епанешников В.А. Программирование в среде Turbo Pascal 7.0 — М.: «ДИАЛОГ-МИФИ», 2000.

## 3.5. КРАТКОЕ ОПИСАНИЕ МЕТОДОВ СОРТИРОВКИ

### СОРТИРОВКА ВСТАВКАМИ

Идея методов сортировки вставками состоит в том, что в ранее упорядоченную подпоследовательность $X_1, X_2, ..., X_{k-1}$ вставляется $X_k$ так, чтобы упорядоченными оказались уже $k$ первых элементов исходной последовательности. В зависимости от способа поиска места для элемента $q = X_k$ различаются следующие методы.

1) *ПРОСТЫЕ ВСТАВКИ*. Если $q < X_{k-1}$, то величина $q$ по очереди сравнивается с $X_{k-1}, X_{k-2}, ...$, пока не будет найдена такая пара элементов $X_{i-1}$ и $X_i$, что $X_{i-1} \le q < X_i$. Это означает, что местом для $q$ является $i$-я позиция. Поэтому элементы $X_i, \ldots, X_{k-1}$ сдвигаются на одну позицию вправо и в освободившуюся $i$-ю позицию вставляется $q$.

2) *БИНАРНЫЕ ВСТАВКИ*. Величина $q$ сравнивается со средним элементом подпоследовательности $X_1, X_2, ..., X_{k-1}$. Если $q$ меньше этого элемента, то место для $q$ ищется тем же способом в левой половине подпоследовательности, иначе - в правой половине. Когда место для $q$ будет найдено, правые (от этого места) элементы сдвигаются на одну позицию вправо, а в освободившуюся позицию вставляется $q$.

### СОРТИРОВКА ОБМЕНАМИ

В методах сортировки обменами переставляются элементы неупорядоченных пар, т.е. таких пар, в которых левый элемент больше правого. В зависимости от порядка перебора пар различаются следующие методы.

1) *МЕТОД ПУЗЫРЬКА*. По очереди просматриваются пары соседних элементов ($X_1$ и $X_2$, $X_2$ и $X_3$, $X_3$ и $X_4$ и т.д.) и в неупорядоченных парах ($X_i > X_{i+1}$) переставляются элементы; в результате просмотра всей последовательности ее максимальный элемент окажется на своем окончательном месте - в конце. Далее аналогичная процедура применяется ко всем элементам последовательности, кроме последнего. (*Замечание*: если при очередном просмотре последовательности не было ни одной перестановки, то последовательность уже упорядочена и потому следует прекратить сортировку.)

2) *ЧЕЛНОЧНАЯ СОРТИРОВКА (метод просеивания)*. Здесь также сравниваются пары $X_1$ и $X_2$, $X_2$ и $X_3$ и т.д., но только до обнаружения неупорядоченной пары $X_k > X_{k+1}$. В этом случае осуществляется «движение назад»: сравниваются и переставляются пары $X_{k+1}$ и $X_k$, $X_k$ и $X_{k-1}$ и т.д. до тех пор, пока $X_{k+1}$ не попадет на свое место, т.е. пока не окажется упорядоченной подпоследовательность из $k + 1$ первых элементов. Далее возобновляется «движение вперед»: сравниваются пары $X_{k+1}$ и $X_{k+2}$, $X_{k+2}$ и $X_{k+3}$ и т.д.

### СОРТИРОВКА ПРОСТЫМ ВЫБОРОМ

В сортировке простым выбором находится минимальный (максимальный) элемент последовательности и он переставляется с первым (последним) элементом. Далее аналогичная процедура применяется ко всем элементам, кроме первого (последнего). И так далее.

## МЕТОД ШЕЛЛА

Пусть $k$ — целое от 1 до $n/2$. Независимо друг от друга упорядочиваются (одним из описанных выше методов, например, простых вставок) подпоследовательности из элементов, отстоящих друг от друга на $k$ позиций:

$$X_i, X_{i+k}, X_{i+2k}, X_{i+3k}, \ldots \ (i = 1, 2, \ldots, k)$$

Затем $k$ уменьшается и процесс повторяется заново. Последний шаг обязательно должен быть выполнен при $k = 1$.

Значение $k$ можно менять разными способами, один из них таков: вначале $k$ равно (целой части от) $n/2$, а затем $k$ каждый раз уменьшается вдвое.

## БЫСТРАЯ СОРТИРОВКА

Выбирается некоторый элемент (например, средний) и все элементы последовательности переставляются так, чтобы выбранный элемент оказался на своем окончательном месте, т.е. чтобы слева от него были только меньшие или равные ему элементы, а справа — только большие или равные. Затем этот же метод применяется к левой и правой частям последовательности, на которые ее разделил выбранный элемент. (*Замечание*: если в части оказалось два-три элемента, то упорядочивать ее следует более простым способом.)

Требуемая перестановка элементов выполняется так. Выбранный элемент копируется в некоторую переменную $q$. Последовательность просматривается слева направо, пока не встретится элемент, больший или равный $q$, а затем просматривается справа налево до элемента, меньшего или равного $q$. Оба этих элемента меняются местами, после чего просмотры с обоих концов последовательности продолжаются со следующих элементов, и т.д. В итоге выбранный элемент окажется в той позиции, где просмотры сошлись, это и есть его окончательное место.

Быструю сортировку можно реализовать рекурсивно и нерекурсивно. Во втором случае границы одной из двух частей (лучше - более длинной), на которые выбранный элемент разделил последовательность, запоминаются в стеке, а другая часть упорядочивается описанным способом. После ее упорядочения из стека извлекаются границы первой части, и теперь уже она упорядочивается.

## СОРТИРОВКА СЛИЯНИЕМ

Основная идея такой сортировки — разделить последовательность на уже упорядоченные подпоследовательности (назовем их «отрезками») и затем объединять эти отрезки во все более длинные упорядоченные отрезки, пока не получится единая упорядоченная последовательность. Отметим, что при этом необходима дополнительная память (массив $Y[1..n]$).

Различаются следующие варианты сортировки слиянием.

1) *ПРОСТОЕ СЛИЯНИЕ*. Считается, что вначале отрезки состоят только из одного элемента, и они сливаются в отрезки из двух элементов (из $X_1$ и $X_2$, из $X_3$ и $X_4$, ...), которые переносятся в массив $Y$. На втором этапе соседние двухэлементные отрезки ($Y_1$, $Y_2$ и $Y_3$, $Y_4$; $Y_5$, $Y_6$ и $Y_7$, $Y_8$; ...) объединяются в отрезки из 4 элементов, которые записываются в массив $X$. На третьем этапе строятся отрезки из 8 элементов, и они заносятся в массив $Y$, и т.д.
2) *ЕСТЕСТВЕННОЕ СЛИЯНИЕ*. Берутся наиболее длинный (по неубыванию) отрезок в начале массива $X$ и наиболее длинный (также по неубыванию, но при просмотре справа налево) отрезок в конце массива, и они сливаются в

один отрезок, который записывается в *начало* массива Y. Затем сливаются следующие максимально длинные отрезки с обоих концов, и полученный отрезок записывается (справа налево) в *конец* массива *Y*. Третьи по порядку отрезки после слияния записываются снова в *начало Y* (вслед за первым объединенным отрезком), четвертые — в *конец Y* (перед вторым объединенным отрезком) и т.д. Первый этап сортировки оканчивается, когда все элементы из *X* будут перенесены в массив *Y*. На втором этапе применяется та же процедура, только массивы *X* и *Y* меняются ролями. И так далее.

*Замечание*: в обоих вариантах следует учитывать, что в конце концов упорядоченные элементы должны оказаться в массиве *X*.

# Задание 4. ЯЗЫК ПАСКАЛЬ. ИНТЕРФЕЙС ПРОГРАММЫ СОРТИРОВКИ.

## 4.1. ПОСТАНОВКА ЗАДАЧИ

Требуется дополнить программу сортировки, реализованную в задании 3, подсистемой диалогового взаимодействия с пользователем (интерфейсом), которая с помощью экранных окон, меню и других средств упрощает общение пользователя с этой программой. Работу с экраном подсистема должна осуществлять в текстовом режиме.

Объединенная система должна предусматривать следующие возможности.

— Пользователь должен иметь возможность выбрать из нескольких предъявленных ему методов сортировки тот метод, которым затем будут упорядочиваться последовательности.

— Пользователь должен иметь возможность задавать свою длину последовательностей, которые будут упорядочиваться выбранным методом.

— В системе должна быть предусмотрена возможность работы в двух режимах — отладки и счета. В режиме отладки пользователь сам вводит элементы (даты) последовательности, которая должна быть упорядочена, а в ответ система демонстрирует пошаговую работу процедуры сортировки для этой последовательности (это нужно для показа правильности работы процедуры). В режиме счета система должна работать так же, как и в задании 3: должна сформировать несколько последовательностей заданной длины, отсортировать их и вывести на экран таблицу результатов.

— Система должна позволять пользователю менять сделанный им ранее выбор (метода, длины или режима) и должна повторять свою работу при новом выборе.

— При вводе длины и элементов последовательности пользователь должен иметь возможность редактировать вводимую информацию.

## 4.2. ВОЗМОЖНЫЕ СЦЕНАРИИ РАБОТЫ С СИСТЕМОЙ

Ниже описаны два возможных сценария работы пользователя с системой. Любой из них можно взять за основу и со своими изменениями реализовать в задании.

### Сценарий 1.

*Этап 1. Выбор метода.*

Работа системы начинается с очистки экрана и показа на нем списка названий различных (не менее 4-5) методов сортировки. Пользователь выбирает один из них либо прекращает работу с системой.

Этот список должен быть реализован в виде вертикального меню — в виде окна (закрашенного своим цветом прямоугольника), в котором друг под другом выписаны названия методов (см. рис. 1). Вначале должно быть выделено (высвечено особым цветом) первое из этих названий, а затем при нажатии пользователем клавиши со стрелкой вниз или вверх система должна выделить следующее или предыдущее (по кругу) название, сняв выделение текущего названия.

Одновременно с меню методов на экране (в его нижней строке) должен высвечиваться текст, подсказывающий пользователю, что он должен сейчас делать, какие клавиши может нажимать и что они означают. Подсказка может быть, например, такой:

```
ВЫБЕРИТЕ МЕТОД: ↓,↑-сдвиг  Enter-выбор  Esc-выход
```

Нажатие пользователем клавиши Enter означает, что он выбрал тот метод, название которого сейчас выделено. (*Замечание*: поскольку в системе реализовано лишь два метода сортировки, то при выборе нереализованного метода система должна как-то сообщить об этом, например звуковым сигналом или выдачей сообщения «Метод не реализован» и позволить пользователю выбрать иной метод. Желательно реализованные методы как-то пометить, например звездочкой.)

Нажатие клавиши *Esc* означает конец работы с системой.

На этом этапе система не должна реагировать на другие клавиши, а курсор должен быть невидимым.

*Этап 2. Выбор режима.*

После того как пользователь выбрал метод сортировки, система спрашивает его о режиме дальнейшей работы — режиме отладки или режиме счета. Пользователь выбирает нужный режим, но может и вернуться на предыдущий этап. В нижней строке экрана должна высвечиваться соответствующая подсказка.

Запрос режима может быть реализован следующим образом. В свободной части экрана появляется окно с текстом «РЕЖИМ: ОТЛАДКА» (см. рис. 1). Если пользователь нажимает клавишу пробела, тогда слово «ОТЛАДКА» заменяется на слово «СЧЕТ»; при новом нажатии этой клавиши снова появляется слово «ОТЛАДКА» и т.д. Нажатие клавиши *Enter* означает выбор того режима, название которого указано сейчас в окне.

Допустимо также нажатие клавиши *Esc*, что означает отказ от выбора режима и возврат на предыдущий этап (с удалением окна режима и восстановлением подсказки, соответствующей предыдущему этапу). На иные клавиши система в это время не должна реагировать. Курсор должен быть невидимым.

*Этап 3. Выбор длины.*

При любом выбранном режиме система далее спрашивает у пользователя, последовательности какой длины он желает упорядочивать. В режиме счета длина может меняться от 1 до 100, а в режиме отладки — от 1 до 15.

Запрос длины можно реализовать так. На экране появляется окно с текстом «ДЛИНА (<=15): » (или «<=100» в режиме счета), за которым следует «поле ввода» — 3-4 позиции, в которые пользователь будет вводить длину (система не должна разрешать ему выходить за рамки этого поля). Курсор на этом этапе видим и показывает позицию, в которую пользователь вводит очередной символ (вначале это первая позиция поля ввода, а затем курсор сдвигается вправо по мере ввода числа). В нижней строке экрана должна высвечиваться соответствующая подсказка. Экран в этот момент может выглядеть так:

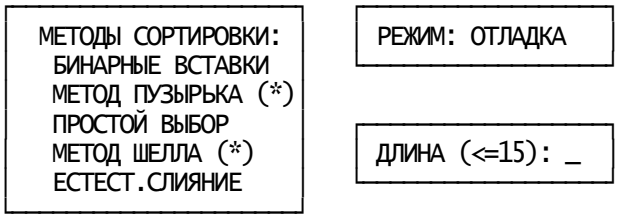


*Рис. 1*

При вводе числа (до нажатия клавиши *Enter*) пользователь должен иметь возможность вносить изменения в набранный текст. Рекомендуется использовать следующие клавиши для редактирования вводимого текста:

| | |
|---|---|
| ← , → | перемещение курсора на одну позицию влево или вправо (без выхода за границы поля ввода) |
| *Del* | удаление символа, на который указывает курсор (со сдвигом влево на одну позицию правой части уже набранного текста) |
| *Backspace* | удаление символа слева от курсора (если только курсор не находится в начале поля ввода) со сдвигом на одну позицию влево самого курсора и правой части уже набранного текста |
| *Ins* | переключение с режима вставки на режим замены или наоборот (начальный режим — вставка) |

(*Замечание*. В режиме замены введенный символ заменяет на экране тот символ, на который указывает курсор, а в режиме вставки часть текста (от курсора и вправо) сдвигается на одну позицию вправо и в освободившуюся позицию вставляется введенный символ. Далее (в любом режиме) курсор перемещается на одну позицию вправо, если только он не находится в конце поля ввода.)

При вводе длины нажатие клавиши *Enter* (при любом положении курсора) означает, что ввод окончен. Система должна считать набранные цифры и перевести их в соответствующую числовую величину. Если число набрано верно и не выходит за определенный диапазон, то система переходит к следующему, четвертому, этапу своей работы, а иначе она должна каким-то образом сообщить об ошибке и предоставить пользователю возможность исправить ранее набранную длину.

Помимо указанных выше клавиш система должна реагировать и на клавишу *Esc*, нажатие которой означает отказ от ввода и возврат системы на предыдущий этап (с удалением окна длины и восстановлением прежней подсказки).

*Этап 4а. Работа в режиме счета.*

Если пользователь выбрал режим счета, тогда система (после запроса длины) сама генерирует несколько последовательностей заданной длины (какие именно — см. задание 3) и сортирует каждую из них выбранным методом. В результате на экране должна появиться таблица, в которой для каждой последовательности указано число сравнений и число перемещений, выполненных во время ее сортировки, а также усредненные значения этих характеристик. Таблица должна сохраняться на экране до тех пор, пока пользователь не нажмет на какую-нибудь клавишу, после чего следует восстановить состояние экрана, соответствующее 3-му этапу, чтобы пользователь мог задать новую длину.

*Этап 4б. Работа в режиме отладки.*

В этом режиме система (после запроса длины) очищает экран и высвечивает в его левой части окно из *n* строк, где *n* — заданная длина (см. рис. 2). Это окно предназначено для ввода пользователем дат той последовательности, которую он хочет упорядочить (поэтому ширина окна должна быть выбрана с расчетом на самую «длинную» дату). Курсор в это время видим и показывает место, куда будет помещен очередной набранный символ. Пользователь должен иметь возможность редактировать набираемый текст — так же, как и при вводе длины (см. выше).

Желательно в верхней части окна указать, в каком порядке должны вводиться элементы дат (например: день, месяц, год) и какой символ (например, точка) используется как разделитель. При этом формат вводимых дат должен быть свободным: не надо требовать, чтобы каждый элемент даты содержал ровно две цифры, не надо заранее расставлять точки.


*Рис. 2*

Нажатие клавиши *Enter* или ↓ является признаком конца ввода текущей даты. В этот момент система должна проверить, правильно ли была набрана дата, и, если да, перейти к следующей строке окна. Иначе система должна сообщить (звуковым сигналом или каким-то сообщением) пользователю об ошибке и остаться в текущей строке окна, чтобы пользователь мог исправить дату. Смысл клавиши ↑ аналогичен, но по ней происходит переход к предыдущей дате (этот возврат нужен, чтобы пользователь мог изменить ранее набранную дату).

Правильный ввод даты в нижней строке окна означает конец ввода всей исходной последовательности (можно сделать и так, чтобы признаком конца ввода всей последовательности было нажатие какой-нибудь особой клавиши). В этом случае в правой части экрана должны появиться два новых окна и должна начать свою работу процедура сортировки, которая в одном из этих окон будет показывать состояние упорядочиваемой последовательности перед каждым очередным шагом сортировки, а в другом - после этого шага. Содержимое этих окон должно оставаться на экране до тех пор, пока пользователь не нажмет какую-нибудь клавишу, иначе нельзя будет уследить за работой процедуры. Когда последовательность будет полностью упорядочена, следует указать на экране число выполненных при сортировке сравнений и перемещений. И эта "картинка" должна сохраняться на экране до тех пор, пока пользователь не нажмет какую-нибудь клавишу. Далее следует убрать с экрана два правых окна и очистить левое окно, чтобы пользователь мог набрать в левом окне новую последовательность дат (при той же длине *n*).

(*Замечание.* Под «шагом сортировки» в методах челнока, простых и бинарных вставок понимается установка очередного элемента на новое место, в быстрой сортировке — установка выбранного элемента на свое окончательное место, а в остальных методах — очередной просмотр всей сортируемой последовательности.)

Если при вводе дат (в любой момент) нажата клавиша *Esc*, то это означает отказ от задания новой исходной последовательности и возврат к предыдущему этапу работы

программы (запросу длины); экран должен быть приведен в состояние, соответствующее предыдущему этапу.

*Сценарий 2*

Работа программы начинается с очистки экрана и высвечивания в его верхней строке основного меню из 5 разделов (см. рис. 3), один из которых должен быть выделен особым цветом. Курсор в этот момент невидим. В нижней строчке экрана должна появиться строка-подсказка примерно такого вида:

```
ВЫБЕРИТЕ РАЗДЕЛ: ←,→ - сдвиг     Enter-выбор
```

При нажатии клавиши со стрелкой выделяется соседний (по кругу) раздел меню, а при нажатии клавиши *Enter* выбирается выделенный раздел.

При выборе раздела МЕТОД на экране (ниже основного меню) должно появиться меню методов, а в нижней строке экрана — соответствующая подсказка (см. рис. 3). Как и в сценарии 1, пользователь выбирает один из методов. После чего экран очищается от меню методов и происходит возврат в основное меню (с восстановлением его подсказки).

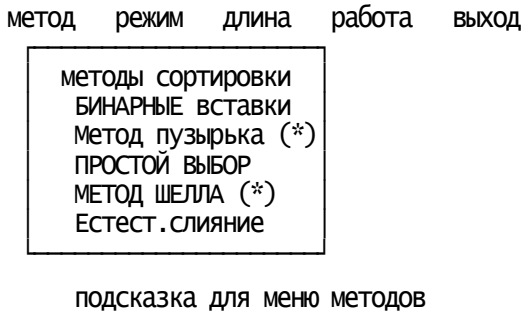


*Рис. 3*

При выборе в основном меню раздела РЕЖИМ на экране появляется окно режима (со своей подсказкой), и пользователь выбирает нужный ему режим (см. этап 2 в сценарии 1), после чего следует убрать это окно с экрана и вернуться в основное меню. Аналогичные действия производятся при выборе раздела ДЛИНА (см. этап 3 в сценарии 1). Выбор раздела РАБОТА означает запуск в работу процедуры сортировки (см. этапы 4а и 4б в сценарии 1), а выбор раздела ВЫХОД — завершение всей работы системы.

*Замечание.* Следует учитывать, что разделы основного меню могут выбираться независимо друг от друга, поэтому возможны неприятные ситуации. Например, раздел РАБОТА может быть выбран до того, как будут заданы метод, режим или длина. Или в режиме счета была установлена длина 60, но затем этот режим был заменён на режим отладки, где такая длина недопустима. Поэтому при использовании этого сценария необходимо заранее продумать реакцию системы на подобные ситуации.

## 4.3. ЛИТЕРАТУРА

[1]. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. — М.: Наука, 1988.

[2]. Епанешников А.М., Епанешников В.А. Программирование в среде Turbo Pascal 7.0 — М.: «ДИАЛОГ-МИФИ», 2000.

## 4.4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

### Модули CRT и CRT1

Для реализации подсистемы интерфейса следует использовать возможности, которые предоставляют стандартный модуль CRT системы программирования Турбо Паскаль 7.0 и модуль CRT1, специально созданный для данного задания (ниже процедуры и функции из модуля CRT1 помечены звездочкой). Для того чтобы можно было воспользоваться процедурами, константами и т.п. из этих модулей, следует в начале текста программы сразу за ее заголовком поместить строку

```
uses crt, crt1;
```

### Работа с окнами и цветом

Окном называется прямоугольный участок экрана. Установка окна (с помощью процедуры *window*) не вызывает никаких изменений на экране, но означает, что все последующие действия с экраном ведутся только в рамках окна, как будто бы и нет иной части экрана. В частности, все координаты позиций отсчитываются от левого верхнего угла окна; исключение составляет обращение к процедуре *window*, координаты для которой всегда задаются как абсолютные, т.е. отсчитываются от верхнего левого угла экрана. В начале работы программы текущим окном является весь экран.

Координаты позиций на экране (в текущем окне) задаются парой (*x, y*), где *x* означает номер колонки экрана (окна), а *y* — номер строки. Отсчет координат ведется от левого верхнего угла экрана (окна), который имеет координаты (1,1). Правый нижний угол всего экрана имеет координаты (80,25).

При высвечивании любого символа на экране используются два цвета: фоновый цвет, которым закрашивается тот участок экрана, где показывается символ, и передний цвет, которым высвечивается сам символ. Все допустимые цвета (их всего 16) нумеруются целыми числами от 0 до 15. При этом передний цвет может быть любым, а фоновый цвет должен иметь номер от 0 до 7. Для более наглядного обозначения цветов в модуле CRT описаны следующие константы:

```
black=0;      {черный}          darkgray=8;      {темно-серый}
blue=1;       {синий}           lightblue=9;     {светло-синий}
green=2;      {зеленый}         lightgreen=10;   {салатовый}
cyan=3;       {сине-зеленый}    lightcyan=11;    {голубой}
red=4;        {красный}         lightred=12;     {светло-красный}
magenta=5;    {малиновый}       lightmagenta=13; {светло-малиновый}
brown=6;      {коричневый}      yellow=14;       {желтый}
lightgray=7;  {серый}           white=15;        {белый}
```

Рекомендуется использовать эти названия вместо числовых кодов цветов; например, вместо *textcolor*(14) лучше писать *textcolor*(*yellow*).

Для работы с окнами и цветом в модулях *CRT* и *CRT1* имеются следующие процедуры (тип *byte* обозначает целые числа от 0 до 255):

*textmode*(*3*)

> процедура при фактическом параметре 3 устанавливает цветной текстовый режим работы с экраном из 80 колонок и 25 строк; с обращения к этой процедуре надо начинать работу подсистемы интерфейса;

*window*(*x1,y1,x2,y2*:*byte*)

процедура объявляет прямоугольную часть экрана с левым верхним углом в точке ($x_1$, $y_1$) и правым нижним углом в точке ($x_2$, $y_2$) текущим окном; все координаты здесь — абсолютные, т.е. отсчитываются от левого верхнего угла всего экрана;

*windcoord*(*var x1,y1,x2,y2*:*byte*)

процедура присваивает своим параметрам (абсолют-ные) координаты текущего окна;

*textcolor*(*fc*:*byte*)

процедура делает цвет с номером *fc* (от 0 до 15) цветом символов (передним цветом) для всех последующих выдач на экран, осуществляемых стандартной процедурой вывода *write*;

*textbackground*(*bc*:*byte*)

процедура делает цвет с номером *bc* (от 0 до 7) фоновым цветом для всех последующих выдач на экран, осуществляемых процедурой вывода *write*;

*getcolors*(*var fc*,*bc*:*byte*)

процедура присваивает параметрам *fc* и *bc* номера текущих переднего и фонового цветов, соответственно;

*clrscr*

процедура очищает текущее окно (записывает во все его позиции пробелы), закрашивая его текущим фоновым цветом.

## Работа с курсором

Курсор всегда показывает позицию на экране, в которую будет помещен очередной символ, вводимый с клавиатуры стандартной процедурой ввода *read* или выводимый стандартной процедурой вывода *write*.

Для работы с курсором полезны следующие процедуры из модулей CRT и CRT1:

*gotoxy*(*x,y*:*byte*)

процедура перемещает курсор в позицию (*x, y*) текущего окна;

*wherexy*(*var x,y:byte*)

процедура присваивает своим параметрам координаты текущей позиции курсора;

*crsoff*

процедура делает курсор невидимым на экране;

*crson*

процедура восстанавливает видимость курсора;

## Непосредственный доступ к экрану

Вывод на экран можно осуществлять без обращения к стандартной процедуре вывода *write*. В этом случае используется тот факт, что информация, отображаемая на экране, хранится в специальном месте оперативной памяти, называемом видеопамятью. Каждую секунду видеопамять многократно просматривается и ее содержимое отображается на экране, поэтому любая запись в нее означает вывод на экран. Чтение же из ячеек ви-

деопамяти позволяет узнать текущее содержимое экрана. (*Замечание*: в этих операциях положение курсора не учитывается и не меняется.)

Каждый элемент видеопамяти состоит из двух байтов: один — это кода символа, который сейчас высвечивается в соответствующей позиции экрана, а другой — так называемый цветовой атрибут, указывающий фоновый и передний цвета, которые используются при высвечивании символа в этой позиции экрана, и признак «мерцания» символа. Более точно, цветовой атрибут — это величина *fc* + 16 *bc* + 128 *blink*, где *fc* — номер переднего цвета, *bc* — номер фонового цвета, а *blink* равен 1, если символ должен мерцать на экране, и равен 0 в противном случае. Например, атрибут «белый символ на синем фоне, без мерцания» задается так: *white* + 16 *blue* (или 15+16*1).

Для работы с видеопамятью в модуле CRT1 имеются следующие процедуры и функции:

*putch*(*x,y*:*byte*; *c*:*char*)

процедура записывает символ *c* в позицию (*x*, *y*) текущего окна, не меняя цветовой атрибут в этой позиции;

*putattr*(*x,y*:*byte*; *a*:*byte*)

процедура меняет цветовой атрибут позиции (*x*, *y*) текущего окна на новое значение *a*, не меняя сам символ в этой позиции

*getch*(*x,y*:*byte*):*char*

значением функции является символ, высвечиваемый в данный момент в позиции (*x*, *y*) текущего окна

*getattr*(*x,y*:*byte*):*byte*

значением функции является цветовой атрибут позиции (*x*, *y*) текущего окна

### Ввод с клавиатуры «без эха»

Использовать в задании для ввода длин и элементов (дат) сортируемых последовательностей стандартную паскалевскую процедуру *read* нельзя: она не учитывает границы поля ввода, трактует даты как неправильные числа и т.д. Поэтому ввод длины и дат (с возможностью редактирования вводимой информации) необходимо реализовать иначе, используя так называемый ввод «без эха».

Когда на клавиатуре нажимается клавиша (вводится символ), то данный символ автоматически не высвечивается на экране — это дополнительное действие, которое может быть выполнено, а может быть и не выполнено. В связи с этим различаются два вида ввода — «с эхом», когда введенный символ тут же высвечивается на экране (именно так работает процедура *read*, которая сама и высвечивает символ), и «без эха», когда символ не высвечивается. Во втором случае вопрос о том, высвечивать символ или нет, а если высвечивать, то в какой позиции экрана, должен решаться дополнительно. Например, если нажата управляющая клавиша (типа *Enter* или ←), то на экране обычно ничего не высвечивается, а если нажата клавиша с обычным символом, то этот символ, как правило, надо выводить на экран. Такое отделение операции ввода от операции вывода на экран позволяет запрограммировать свой редактор ввода, который «в темную» вводит символ за символом и определяет, что делать с каждым из них.

Для ввода «без эха» можно использовать следующую процедуру из модуля *CRT1*:

*inkey*(*var c:char; var spec:boolean*)

процедура вводит «без эха» символ первой из клавиш, нажатых на клавиатуре, и присваивает его параметру *c* (если никакая клавиша еще не была нажата, процедура ждет нажатия первой же клавиши); при нажатии клавиши с обычным символом параметру *spec* присваивается значение *false*, при нажатии управляющей клавиши — значение *true*

*Замечание*. Количество символов, которые можно ввести с клавиатуры (с учетом управляющих клавиш и комбинаций типа *Ctrl+Shift*), столь велико, что для них не хватает имеющихся 256 кодов. Поэтому все эти символы разделены на две группы, в одну из которых включены обычные символы (буквы, цифры, знаки операций и т.п.), а в другую — управляющие символы (и их комбинации). При этом в каждой группе символы имеют одни и те же коды — от 0 до 255 (например, код 83 имеет и буква 'S', и клавиша *Del*), поэтому знание только кода еще не определяет символ. Именно из-за этого процедура *inkey* и сообщает через свой параметр *spec*, символ из какой группы был введен.

Ниже приведены (десятичные) коды некоторых управляющих символов, выдаваемые процедурой *inkey*:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ↑ | 72 | ↓ | 80 | ← | 75 | → | 7 |
| Enter | 13 | Del | 83 | Backspace | 8 | Ins | 82 |
| Esc | 27 | Tab | 9 | Home | 71 | End | 79 |
| F1 | 59 | F2 | 60 | . . . | | F10 | 68 |

Рекомендуется ввести в программе константы с подходящими именами, например:

```
const
  Enter  = #13;
  Left   = #75;
  EndKey = #79;
  F3     = #61;
  ...
```

и уже ими пользоваться.

## Другие процедуры модуля CTR1

В модуле CRT1 имеются также следующие две процедуры без параметров:

*wait*

ожидать нажатия на клавиатуре любой клавиши (введенный символ «глотается»)

*bell*

подача звукового сигнала

## Работа с полями и окнами

На основе процедур и функций из модулей *CRT* и *CTR1* рекомендуется реализовать более крупные операции для работы с экраном. (При этом желательно использовать тип *string*, описывающий в языке Турбо Паскаль строки переменной длины.)

## Операции над полями

Прежде всего следует реализовать в виде процедур операции над полями. Под «полем» понимается часть одной строки экрана; оно задается тремя параметрами: *x*, *y* — координатами начальной точки поля, *n* — длиной поля. Процедуры могут быть следующими:

*putfield*(*x,y,n*:*byte*; *s*:*string*)

> записать в поле, определяемое параметрами *x*, *y* и *n*, все символы строки *s*; если длина *s* больше *n*, то лишние символы строки *s* не выводятся, если меньше - в конец поля дописываются пробелы (в частности, при пустой строке *s* происходит очистка поля);

*getfield*(*x,y,n*:*byte*; *var s*:*string*)

> скопировать в строку *s* все символы из поля, заданного параметрами *x*, *y* и *n*;

*paintfield*(*x,y,n*:*byte*; *a*:*byte*)

> закрасить поле заданным цветом; точнее, записать цветовой атрибут *a* во все позиции поля, заданного параметрами *x*, *y* и *n* (не меняя при этом символы поля);

*readfield*(*x,y,n*:*byte*; *stopkeys*:*string; var s*:*string*; *var key*:*char*)

> ввод любого текста (с возможностью редактирования) в рамках поля, заданного параметрами *x*, *y* и *n*, и запись введенного текста в строку *s*; в параметре *stopkeys* перечисляются все управляющие клавиши (Enter, Esc, ↓, ...), нажатие которых означает конец ввода; та из них, которая была действительно нажата, присваивается параметру *key*

*Замечания относительно процедуры readfield:*

- процедура не должна предварительно очищать поле ввода (если надо, такую очистку следует делать до обращения к процедуре); это нужно для того, чтобы при неправильно набранном ранее тексте пользователь мог вносить исправления в уже набранный текст, а не вводить весь текст заново;
- процедура должна высвечивать курсор, установив его вначале в левую позицию поля ввода, а затем перемещая по мере ввода и редактирования текста; курсор не должен выходить за рамки поля ввода;
- процедура должна реагировать на клавиши редактирования (←, →, *Del*, *Backspace*, *Ins*), смысл которых объяснен в описании этапа 3 сценария 1;
- процедура не должна проверять правильность набираемого текста (например, не должна «ловить» нецифровые символы при вводе числа), такую проверку надо делать только по окончании работы процедуры; это позволяет использовать данную процедуру для ввода любого текста — длины, даты или еще чего-либо.

## Операции над окнами

На основе перечисленных операций над полями рекомендуется реализовать следующие процедуры для работы с окнами и меню (названия процедур и их параметры определить самим):

- выдача в нижней строке экрана (не окна!) строки-подсказки;

— выдача на экран (например, в его нижней строчке) текста «Нажмите любую клавишу», ожидание нажатия пользователем любой клавиши и последующее восстановление соответствующей части экрана;

— показ на экране основного меню и выбор раздела в этом меню; результат — номер выбранного раздела (для сценария 2);

— показ на экране меню методов и выбор раздела в этом меню; результаты — номер выбранного метода и код клавиши выхода (Enter или Esc);

— показ окна режима и выбор режима; результаты — номер выбранного режима и код клавиши выхода (Enter или Esc);

— показ окна длины и ввод длины (с проверкой); результаты — введенная длина и код клавиши выхода (Enter или Esc);

— показ окна для ввода дат (в режиме отладки), сам ввод дат (с проверкой) и заполнение ими соответствующего массива; результат должен указывать, завершен ли полностью ввод дат или была нажата клавиша Esc;

— выдача текущего содержимого упорядочиваемого массива (для режима отладки);

— выдача таблицы с характеристиками (сравнениями и перемещениями) работы процедуры сортировки в режиме счета.

*Замечания:*

— установка окна и последующая запись в него не сохраняют прежнее содержимое (символы и цветовые атрибуты) этой части экрана; поэтому, если позже надо будет восстановить это содержимое, то следует его предварительно где-то сохранить, а затем снова воспроизвести на экране;

— для выдачи текста (например, строки-подсказки) вне текущего окна следует запомнить координаты текущего окна, затем сделать окном весь экран и вывести текст в нужном месте экрана, а затем по запомненным координатам снова установить текущее окно;

— для рисования рамок вокруг окон следует с помощью процедуры *putchar* записывать в граничные позиции окон символы так называемой псевдографики. Они имеют следующие (десятичные) коды:

```
218: ┌ 196: ─ 191: ┐      201: ╔ 205: ═ 187: ╗
179: │ 197: ┼ 179: │      186: ║ 206: ╬ 186: ║
192: └ 196: ─ 217: ┘      200: ╚ 205: ═ 188: ╝
```

# Задание 5. ЯЗЫК ПАСКАЛЬ. РАБОТА С ФАЙЛАМИ.

## 5.1. ПОСТАНОВКА ЗАДАЧИ

Во внешнем текстовом файле, специально подготовленном для данного задания, хранится информация о некотором множестве студентов 3-го курса. Количество студентов заранее не известно, данные о них в файле никак не упорядочены. Сведения о каждом студенте записаны (без ошибок) в одной строке файла и имеют следующий формат:

<фамилия> <имя> <отчество> <пол> <дата рождения> <место рождения>
<номер группы> <1-я оценка> <2-я оценка> <3-я оценка>

Причем в фамилии, имени, отчестве и месте рождения (названии города) содержится не более 12 (заглавных) букв, пол указывается буквой М или Ж, дата задается в виде 31.5.80 и т.п., номер группы — это целое от 301 до 329, а оценки (за экзамены) — целые от 3 до 5. Элементы строки разделены одним или несколькими пробелами. Возможные примеры:

```
МАРКОВ  РОМАН  ПЕТРОВИЧ  М  1.10.81  УФА  309  4  3  3
ГОЛОВКО  ВЕРА  ОЛЕГОВНА  Ж  2.1.82  КИЕВ  325  5  4  5
```

Требуется среди всех этих студентов сначала отобрать студентов тех групп, которые удовлетворяют свойству А, а затем из их числа отобрать студентов, удовлетворяющих свойству В, и напечатать соответствующие данные о них. Для контроля следует также напечатать некоторые промежуточные сведения (см. вариант задания).

## 5.2. ВАРИАНТЫ ЗАДАНИЯ

### Свойство А (искомые группы):

1) В группе юношей больше, чем девушек.
2) В группе больше половины студентов, которым на 1 сентября текущего года было более 19 лет.
3) Средний возраст студентов группы (на 1 сентября текущего года) меньше 19.5 лет.
4) В группе более трети студентов из городов Владимир, Иваново, Калуга, Москва, Рязань, Смоленск, Тверь.
5) Средняя успеваемость в группе выше 4.0.
6) В группе есть хотя бы два «хорошиста» (т.е. не имеющих троек, но и не отличников).

В качестве промежуточной информации напечатать по каждой (непустой) группе курса общее число студентов, а также: в 1-м варианте — число юношей, во 2-м варианте — число студентов старше 19 лет, в 3-м — средний возраст (в годах) студентов группы, в 4-м — число студентов из указанных городов, в 5-м — среднюю успеваемость в группе, в 6-м — число «хорошистов». Кроме того, напечатать номера отобранных групп и общее число студентов во всех этих группах.

### Свойство В (искомые студенты из отобранных групп):

1) Студенты, модальные по следующему набору параметров:

   фамилия, имя, год рождения.

2) Студенты, модальные по следующему набору параметров:
   имя, пол, возраст (в полных месяцах).
3) Студенты, модальные по следующему набору параметров:
   отчество, город, месяц рождения.
4) Студенты, модальные по следующему набору параметров:
   город, 1-я оценка, 2-я оценка, 3-я оценка.
5) Студенты, средние по следующему набору параметров:
   1-я оценка, 2-я оценка, 3-я оценка.
6) Студенты, средние по следующему набору параметров:
   средняя успеваемость (с точностью до 0.1), возраст (в полных месяцах).

В качестве промежуточной информации напечатать все модальные значения или среднее арифметическое значение каждого из указанных в варианте параметров.

Как основной результат работы программы напечатать (упорядочив по ФИО) следующие сведения о найденных студентах: фамилия, имя, отчество, номер группы и значение каждого из указанных параметров.

## 5.3. ОПРЕДЕЛЕНИЯ

1. Пусть имеется множество объектов, каждый из которых характеризуется некоторым параметром (признаком). «Модальным» называется то значение параметра, которое присуще наибольшему числу объектов множества (таких значений может быть несколько). Объект с таким значением параметра называется «модальным по этому параметру». Например, если среди студентов наиболее часто и в равном количестве встречаются имена Елена и Наталья, то все Елены и Натальи являются модальными по параметру «имя».

   Объект называется «модальным по набору параметров», если число его модальных параметров из этого набора максимально среди всех других объектов. Если, например, модальными значениями параметра «фамилия» являются Иванов и Петров, параметра «имя» — Сергей, Андрей и Евгений, а параметра «город» — Москва, то в множестве студентов {Петров Андрей из Тулы, Иванов Олег из Москвы, Сидоров Сергей из Курска} модальными по набору параметров {фамилия, имя, город} будут два первых студента, т.к. у них из трех признаков набора два являются модальными, а у третьего студента такой признак только один.

2. Пусть каждый объект множества характеризуется некоторым числовым параметром и пусть $!P$ — среднее арифметическое значение этого параметра на данном множестве. «Средним» называется то значение $P$ этого параметра, которое наиболее близко к числу !P, т.е. при котором величина $\mathrm{abs}(P - !P)$ минимальна (таких значений может быть два). Объект с таким значением называется «средним по этому параметру». Например, если средняя оценка (по какому-то экзамену) равна 3.5, то все студенты, получившие оценки 3 и 4, будут средними по этому параметру.

   Объект называется «средним по набору параметров», если количество его средних параметров из этого набора максимально среди всех других объектов.

## 5.4. ТРЕБОВАНИЯ К ПРОГРАММЕ

1. Информация обо всех студентах, введенная из внешнего файла, должна быть представлена в виде списка. Звено списка должно содержать сведения об одном студенте и представлять собой запись со следующими полями: фамилия, имя, отчество (все — строки из 12 литер), пол (перечислимый тип), дата (запись из трех полей), город (строка из 12 литер), номер группы (целое), оценки (массив из трех целых чисел). Вся дальнейшая работа программы должна вестись только с этим списком.
2. В программе должны быть описаны процедура чтения из файла слов и записи их в 12-литерные строки (с пробелами справа) и процедура чтения целых чисел (стандартная процедура ввода не подойдет из-за точек в датах).
3. Для тестирования программы подготовить свой небольшой внешний файл.

## 5.5. ЛИТЕРАТУРА

[1]. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. — М.: Наука, 1988.

[2]. Епанешников А.М., Епанешников В.А. Программирование в среде Turbo Pascal 7.0 — М.: «ДИАЛОГ-МИФИ», 2000.

## 5.6. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

1. Целесообразно избрать следующий план решения задачи:
2. Чтение информации из файла и построение списка.
3. Просмотр списка и определение нужных сведений по каждой группе (завести массив по числу групп). Выбор групп, удовлетворяющих свойству А. Промежуточная печать. (Пункты *а* и *б* можно выполнять одновременно).
4. Просмотр списка и удаление из него сведений о студентах из других групп.
5. Просмотр списка оставшихся студентов и определение числа «встречаемости» каждого значения по каждому параметру из заданного набора. Определение наиболее часто встречающихся, т.е. модальных, значений по каждому из этих параметров. (Для средних значений — подсчет среднего арифметического по каждому параметру, а затем — при новом просмотре списка — определение минимального отклонения.) Промежуточная печать.
6. Новый просмотр списка и подсчет для каждого студента числа модальных или средних параметров из заданного набора, определение максимума этих чисел.
7. Просмотр списка и выбор студентов с наибольшим числом модальных или средних параметров. Печать окончательного результата.
8. Особенности работы с файлами в языке Турбо Паскаль.

   — В этом языке нет понятия буферной переменной файла и нет процедур *get* и *put*. Поэтому для чтения из файлов следует пользоваться процедурами *read* и *readln*, а для записи — *write* и *writeln*.

   — В языке нет внутренних файлов, все файлы — внешние. Поэтому перечислять внешние файлы в заголовке программы не надо, и заголовок выглядит так:

   ```
   program <имя программы>;
   ```

— Поскольку названия файлов на диске (типа *C:\COURSE3.TXT*) не являются идентификаторами в смысле языка Паскаль, то их нельзя использовать в качестве имен (файловых) переменных. В связи с этим в Турбо Паскале названию дискового файла ставят в соответствие некоторое имя (например, *t*), законное с точки зрения языка, и далее в программе пользуются только этим именем: именно его описывают как имя файла, именно его указывают в процедурах и функциях *reset*, *rewrite*, *read*, *write*, *eof*, *eoln* и т.п. Соответствие же между этим именем и названием дискового файла устанавливается следующей процедурой:

```
assign(t, 'C:\COURSE3.TXT')
```

(второй параметр здесь - строка), причем обращение к этой процедуре должно быть выполнено до любых других операций над файлом.

— По завершению работы с файлом его надо «закрыть»:

```
close(t)
```

# Задание 6. ЯЗЫК АССЕМБЛЕРА.ОБРАБОТКА ЛИТЕРНЫХ ДАННЫХ.

## 6.1. ПОСТАНОВКА ЗАДАЧИ.

Дан непустой текст (последовательность литер), содержащий не более 100 элементов, с точкой в качестве признака конца текста.

Требуется:

— ввести с клавиатуры данный текст и записать его в память ЭВМ;

— определить, обладает ли этот текст заданным свойством (свойство определяется вариантом задания);

— преобразовать текст по правилу 1, если он обладает заданным свойством, и по правилу 2 в противном случае (правила преобразования определяются вариантом задания);

— вывести на экран дисплея: исходный текст, номер примененного правила, преобразованный текст.

## 6.2. ВАРИАНТЫ ЗАДАНИЯ

### А. Свойство исходного текста:

1) Текст оканчивается заглавной латинской буквой, которая больше не встречается в тексте.
2) Текст начинается цифрой и оканчивается цифрой, причем эти цифры различны.
3) Текст начинается латинской буквой и оканчивается латинской буквой.
4) Текст содержит не менее трех латинских букв.
5) Текст содержит равное количество заглавных и строчных латинских букв.
6) Текст не содержит иных литер, кроме цифр и латинских букв.

### Б. Правило 1 преобразования текста:

1) Заменить каждую заглавную латинскую букву на следующую по алфавиту букву (букву Z заменять на букву A).
2) Заменить каждую ненулевую цифру на соответствующую ей по порядковому номеру строчную букву латинского алфавита ($1 \to a$, $2 \to b$ и т.д.).
3) Заменить каждую заглавную латинскую букву на цифру, числовое значение которой равно величине $N \bmod 10$, где $N$ — порядковый номер буквы в алфавите (от 1 до 26).
4) Заменить каждую строчную латинскую букву на соответствующую заглавную букву.
5) Заменить все заглавные латинские буквы на соответствующие строчные буквы.
6) Заменить каждую заглавную латинскую букву на заглавную букву, симметричную ей в алфавите (A ↔ Z, B ↔ Y, ...).

В. Правило 2 преобразования текста:

1) Перенести в начало текста все входящие в него цифры с сохранением порядка их следования.
2) Перевернуть текст, не используя дополнительную память.
3) Удвоить каждую литеру текста.
4) Удалить из текста все повторные вхождения его первой литеры.
5) Оставить в тексте только те литеры, которые входят в него ровно один раз.
6) В каждой группе следующих подряд одинаковых литер оставить только одну из них.

## 6.3. ТРЕБОВАНИЯ К ПРОГРАММЕ

1. Вывод исходного текста должен быть выполнен сразу после его записи в память — до его анализа и преобразования.
2. Вывод преобразованного текста должен быть выполнен только после его окончательного формирования.
3. Алгоритмы преобразования текста по правилам 1 и 2 должны быть оформлены в виде процедур.
4. В программе использовать операции ввода-вывода *INCH*, *OUTCH*, *OUTSTR* и операцию останова *FINISH*, описанные в [1].

## 6.4. ЛИТЕРАТУРА

[1]. Пильщиков В.Н. Программирование на языке ассемблера IBM PC. — М.: Диалог-МИФИ, 1999.

# Задание 7. ЯЗЫК АССЕМБЛЕРА. ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ.

## 7.1. ПОСТАНОВКА ЗАДАЧИ

Дана последовательность от 1 до 20 слов, каждое из которых содержит от 1 до 8 заглавных латинских букв; соседние слова разделены запятой, за последним словом следует точка.

Требуется ввести эту последовательность и преобразовать ее во внутреннее представление, а затем напечатать по алфавиту определенные слова с дополнительной информацией о каждом из них.

## 7.2. ВАРИАНТЫ ЗАДАНИЯ

### А. Внутреннее представление последовательности слов:

7) Список слов, упорядоченных по алфавиту.
8) Массив списков: список из однобуквенных слов, список из двухбуквенных слов и т.д. (в каждом из них слова упорядочены по алфавиту).
9) Двоичное дерево поиска (в нем слева от каждой вершины-слова должны находиться только те слова, что предшествуют этому слову по алфавиту, а справа - следующие за ним по алфавиту).

### Б. Какие слова и в каком порядке печатать:

10) Все слова (с печатью всех повторных вхождений).
11) Все различные слова (без повторений).
12) Все слова, входящие в последовательность только один раз.
13) Все слова (с повторениями), входящие в последовательность более одного раза.
14) Все различные слова, входящие в последовательность более одного раза.
15) Сначала все слова, входящие в последовательность один раз, затем все слова (с повторениями), входящие два раза, и т.д.
16) Сначала все слова, входящие в последовательность один раз, затем все различные слова, входящие два раза, потом все различные слова, входящие три раза, и т.д.
17) Сначала все однобуквенные слова (с повторениями), затем все двухбуквенные слова и т.д.
18) Сначала все различные однобуквенные слова, затем все различные двухбуквенные слова и т.д.

*Замечание:* в каждой группе слова печатать по алфавиту.

### В. Дополнительная информация о слове:

1) Нет дополнительной информации.
2) Порядковый номер слова в последовательности.

3) Число вхождений слова в последовательность.

## 7.3. ТРЕБОВАНИЯ К ПРОГРАММЕ

1. Для размещения звеньев списков (вершин дерева) выделить в памяти область подходящего размера — «кучу» (см. [1]). Описать процедуру (аналогичную процедуре New языка Паскаль), которая при каждом обращении к ней выделяет из кучи свободные ячейки под новое звено (вершину).
2. Описать в программе следующие процедуры:
   - чтение очередного слова, дополнение его справа пробелами (до 8 символов) и запись его в фиксированное место памяти;
   - вставка нового слова в упорядоченный список (в дерево);
   - просмотр списка (дерева) и печать нужных слов.
3. Программа должна быть протестирована на различных исходных данных.
4. В программе использовать операции ввода-вывода *INCH*, *OUTCH*, *OUTSTR*, *OUTWORD* и операцию останова *FINISH*, описанные в [1].

## 7.4. ЛИТЕРАТУРА

[1]. Пильщиков В.Н. Программирование на языке ассемблера IBM PC. — М.: Диалог-МИФИ, 1999.

## 7.5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

1. Исходную последовательность слов сначала целиком ввести в память и лишь затем выделять из нее слова.
2. Структуру звеньев списков (вершин дерева) определить с учетом особенностей решаемой задачи.
3. Для упрощения операций над списками следует использовать списки с заглавными звеньями.

СОДЕРЖАНИЕ