



Кафедра информатики и  
информационных технологий

## **Б1.О.17 АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ**

Методические указания к лабораторным работам и самостоятельной работе  
обучающихся

### **ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ПАСКАЛЬ**

для направления подготовки 09.03.03 Прикладная информатика  
профиль подготовки Прикладная информатика цифровой экономики

Квалификация выпускника  
**бакалавр**

**Уфа 2021**

Рекомендовано к изданию методической комиссией экономического факультета 25 марта 2021 г. (протокол №8).

Составитель: старший преподаватель Иванова Г.Р.

Ответственный за выпуск:

Заведующий кафедрой информатики и информационных технологий, д.т.н.  
доцент Беяева А.С.

г. Уфа, ФГБОУ ВО Башкирский ГАУ, кафедра информатики и  
информационных технологий

## Оглавление

Использование множеств для решения задач в Pascal .....	4
Использование записей для решения задач в среде Pascal .....	7
Использование записей с вариантами и вложенных записей для решения задач в Pascal.....	11
Типизированные файлы в Pascal.....	14
Нетипизированные файлы в Pascal.....	17
Текстовые файлы в Pascal.....	23
Динамические переменные. Указатели в Pascal .....	25
Динамические структуры данных в Pascal .....	30
Библиографический список .....	37
ПРИЛОЖЕНИЕ А .....	38
ПРИЛОЖЕНИЕ Б.....	39
ПРИЛОЖЕНИЕ В .....	40
ПРИЛОЖЕНИЕ Г .....	42
ПРИЛОЖЕНИЕ Д .....	43
ПРИЛОЖЕНИЕ Е.....	44
ПРИЛОЖЕНИЕ Ж.....	45
ПРИЛОЖЕНИЕ И .....	46
ПРИЛОЖЕНИЕ К .....	47

## Использование множеств для решения задач в Pascal

### Цель работы

Изучение принципов работы с множественным типом данных.

### Задачи работы

Освоение основных приемов и приобретение навыков в составлении программ с использованием множеств.

### Требования к организации рабочего места

Лабораторная работа выполняется в компьютерном классе. Рабочее место должно быть оборудовано персональным компьютером с процессором с тактовой частотой не менее 300 МГц, ОЗУ 256 Мб. В качестве программного обеспечения должна быть установлена среда программирования Pascal ABC.

## 1 Общие сведения

**Множество** – это структурированный тип данных, представляющий набор взаимосвязанных по какому-либо признаку или группе признаков объектов, которые можно рассматривать как единое целое. Каждый объект в множестве называется *элементом множества*. Все элементы множества должны принадлежать к одному из скалярных типов, кроме вещественного (в нем нет отношений порядка, нельзя ввести ограничения на тип). Этот тип называется *базовым типом множества*. Базовый тип задается диапазоном или перечислением. Область значений типа множество – набор всевозможных подмножеств, составленных из элементов базового типа.

В выражениях на языке Паскаль значения элементов множества указываются в квадратных скобках: [1, 2, 3, 4], ['a', 'b', 'c'], ['a'... 'z']. Если множество не имеет элементов, оно называется пустым и обозначается как [ ].

Мощностью множества называется количество неповторяющихся элементов, входящих в него.

Для описания множественного типа используется словосочетание *set of* (множество из ...).

Формат:

Type

<имя типа> = set of <элемент1, ... , элемент n>;

Var

<идентификатор, ... > : <имя типа>;

...

Можно задавать множественный тип и без предварительного описания:

Var

<идентификатор, ... >: set of <элемент1, ...>

## 2 Порядок выполнения работы

### 1) Решить задачу:

Пусть даны два множества  $X_1$  и  $X_2$ , содержащие целые числа из диапазона 1..100. Известно, что мощность каждого множества равна 10. Сформируйте и выведите на экран новое множество  $Y = X_1 + X_2$ .

### 2) Решить задачу:

Пусть даны три множества:  $X_1 = \{A_1, A_2, A_4, A_5\}$ ,  $X_2 = \{A_1, A_3, A_4, A_5\}$ ,  $X_3 = \{A_1, A_5\}$ . Нужно сформировать новое множество  $Y = (1 + X_2) * (X_1 - X_3)$  и вывести на экран его мощность.

### 3) Выполнить задание из приложения А по указанию преподавателя.

### 4) Выполнить задание из приложения Б по указанию преподавателя.

### *Требование к оформлению отчета*

Отчет по лабораторной работе должен быть выполнен на листах формата А4 и должен содержать название занятия, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

## 3 Контрольные вопросы

- 1) Что называют множеством?
- 2) Дайте определение элемента множества.
- 3) Что такое базовый тип множества?
- 4) Опишите формат типа множество.
- 5) Поясните понятие мощность множества.
- 6) В каких пределах можно изменять количество элементов, входящих во множество?
- 7) Какое множество имеет мощность 0?
- 8) Какова мощность множества, базовый тип которого Boolean?
- 9) Приведите пример множеств мощности 1; 10; 100.
- 10) Может ли базовый тип множества быть вещественным числом?
- 11) Какой объем памяти занимает один элемент множества?
- 12) Может ли множество содержать элементы различных типов?
- 13) Может ли множество содержать несколько одинаковых элементов?
- 14) Может ли тип диапазон быть базовым типом множества?
- 15) Могут ли два множества содержать одинаковые элементы?
- 16) Как работает операция In?
- 17) Какие множества считают равными; неравными? Имеет ли значение для сравниваемых множеств порядок следования элементов?
- 18) Каково назначение операций «больше или равно», «меньше или равно», примененных к данным множественного типа?
- 19) Дано описание множества X: `Var X: set of ['Q', 'W', 'E']`. Перечислите все возможные значения переменной X.
- 20) Найдите ошибки в описании множеств:

- a) Var M: set of ['Q', 'W', 'E', 'RT'];
  - b) Var M: set of ['a', 'o', 'e', 1..5];
  - c) Var M: set of [100..300];
  - d) Var M: set of [10, 10.5, 11.. 20];
  - e) Var M: set of byte.
- 21) Дан массив  $A = (1, 2, 3)$  и множество  $A = [1, 2, 3]$ . Чем они отличаются?
- 22) Какие операции определены над множествами?
- 23) В чем преимущества и недостатки использования множественного типа данных?
- 24) Даны два множества  $A=[1..10]$  и  $B=[5..15]$ . Найдите их объединение, пересечение и разность.
- 25) Определите истинность логических выражений:
- a)  $5 \text{ In } [3..7]$ ;
  - b)  $'a' \text{ In } ['A' .. 'Z']$ ;
  - c)  $4 \text{ in } ['p', 'c', '4', '3']$ .
- 26) Поясните суть процедур ввода и вывода элементов множества.
- 27) Пусть дан фрагмент программы.
- ```

type
bits = set of 0..1;
var
x: bits;
y: set of (a, b, c);
z: set of '!' .. '!';

```

Определите:

- a) базовый тип каждого из указанных множественных типов;
- б) сколько и каких значений может принимать каждая из переменных x, y, z.

- 28) Какие из следующих описаний не верны и почему?

```

type
точки = set of real;
байт = array[1..8]of 0..1;
данные = set of байт;
месяц = (янв, фев, мар, апр, май, июн, июл, авг, сен, окт, ноя, дек );

```

## Использование записей для решения задач в среде Pascal

### Цель работы

Изучение основных принципов работы с комбинированным типом данных запись.

### Задачи работы

Освоить основные приемы и приобрести навыки в составлении программ с использованием комбинированного типа данных запись.

### 1 Общие сведения

В программах часто возникает необходимость логического объединения данных. Например, база данных предприятия содержит для каждого сотрудника его фамилию, дату рождения, должность, оклад и т.п.; программа моделирования движения поездов – пункты отправления и прибытия, время, количество вагонов и многое другое. Однотипные данные организуются в массивы. Таким образом, для хранения каждого типа данных необходимо организовать массив и установить индекс для организации соответствия между разными массивами. Для объединения разнотипных данных предназначен тип «запись» или комбинированный тип данных.

**Запись(Record)** – это структура, состоящая из фиксированного числа компонент, называемых полями. Общий вид описания типа Record:

**Type**

<имя\_записи> = **Record**

<имя\_поля\_1> : <тип>;

<имя\_поля\_2> : <тип>;

...

<имя\_поля\_n> : <тип>;

**end;**

Поля записи могут быть любого типа, кроме файлового. Например, для каждого товара на складе требуется хранить его наименование, цену и количество единиц:

**Type**

товар = **Record**

name : string[20];

price : real;

number : integer;

**end;**

Переменные типа «запись» описываются обычным образом. Можно задать описание типа при описании переменной, создать массивы из записей, записи из массивов и т.д.:

**Var** t1, t2: товар;

sklad : array[1..20]of товар;

Student : record

name : string[20];

group : string[7];

marks : array[1..3]of byte;

end;

С записями целиком можно делать то же что и с массивами: присваивать одну запись другой, если они одного типа. Например:

```
t1:=t2;
t2:=sklad[5];
```

Все остальные действия выполняются с отдельными полями записи. Получить доступ к полю записи можно с помощью селектора записи либо с использованием оператора присоединения *with*.

**Селектор записи** – это составное имя:

<имя переменной>. <имя поля>

Для описанной выше переменной *t1* – товара *t1.name* – название товара, *t1.price* – цена товара, *t1.number* – количество товара.

Селектор используется в выражениях и обрабатывается как обычная переменная. Над полем записи могут производиться все операции, которые возможны над элементами этого типа данных.

Оператор *with* удобнее использовать, если требуется обращаться к нескольким полям одной и той же записи. Общий вид оператора присоединения таков:

```
With <имя_записи> do <оператор>;
```

Например, ввод данных о товарах на складе может быть организован:

– с помощью селектора записи:

```
for i:=1 to 20 do
  readln(sklad[i].name, sklad[i].price,
    sklad[i].number);
```

– с помощью оператора присоединения:

```
for i:=1 to 20 do
  with sklad[i] do
    readln(name, price, number);
```

**Пример 1.** Пусть нам необходимо заполнить сведения о студенте (Ф.И.О., дата рождения, адрес, курс и группа), а затем вывести эти сведения на экран.

```
program primer1;
type anketa=record
  fio: string[45];
  dat_r: string[8];
  adres: string[50];
  curs: 1..5;
  grupp: string[3]
end;
var student: anketa;
begin
  writeln('введите сведения о студенте');
  writeln('введите фамилию, имя и отчество');
  readln(student.fio);
  writeln('введите дату рождения');
  readln(student.dat_r);
  writeln('введите адрес');
  readln(student.adres);
  writeln('введите курс');
```



```

readln(student.curs);
writeln('введите группу');
readln(student.grupp);
writeln('ввод закончен');
writeln;
writeln('фамилия студента: ', student . fio );
writeln(' дата рождения : ', student.dat_r);
writeln(' адрес : ', student.adres);
writeln(' курс : ', student.curs);
writeln(' группа : ', student.grupp);
end.

```

**Пример 2.** Для каждого товара на складе известно его наименование, цена, вес и количество единиц. Составьте программу, которая по запросу выдает сведения о товаре или сообщение, о том, что товар не найден.

```

program primer;
const MaxN=100;
type
    tovar=record
        name: string[20];
        price:real;
        number:integer;
    end;
var
    sklad:array[1..MaxN] of tovar;
    name:string[20];
    n,i:integer;
    found:boolean;
begin
    writeln('Введите количество товаров на складе');
    readln(n);
    for i:=1 to n do
        begin
            writeln('Введите наименование товара');
            readln(sklad[i].name);
            writeln('Введите цену товару');
            readln(sklad[i].price);
            writeln('Введите количество товара');
            readln(sklad[i].number);
        end;
    writeln('Введите наименование искомого товара');
    readln(name);
    for i:=1 to n do
        if name=sklad[i].name then
            begin
                writeln('Сведения об искомом товаре');
                writeln(sklad[i].name:3,sklad[i].price:5:2,
                    sklad[i].number:3);
                found:=true;
                break;
            end;
        if not found then    writeln('Товар не найден');
    end;
end.

```

```
readln  
end.
```

## 2 Порядок выполнения работы

1. Дополните пример 1 сведениями о нескольких студентах, например, нашего факультета. Для этого необходимо организовать массив записей. Из общего списка вывести фамилии студентов 2-го курса.
2. Решить задачу примера 2, используя оператор присоединения.
3. Дополнить программу примера 2 вычислением общего количества единиц товара на складе.
4. Решить задачу по указанию преподавателя из приложения В.

### *Требование к оформлению отчета*

Отчет по лабораторной работе должен содержать название и цель работы, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

## 3 Контрольные вопросы

1. Чем отличается запись от массива?
2. Как описывается запись?
3. Что такое селектор записи и для чего он применяется?
4. Для чего используется оператор присоединения?
5. Каков общий вид оператора присоединения?
6. Может ли типом поля записи быть массив?
7. Верно ли, что все поля записи должны быть разных типов?
8. Опишите комбинированный тип для определения следующего понятия:
  - а. цена в рублях и копейках;
  - б. время в часах, минутах и секундах;
  - с. адрес ( город, улица, дом, квартира).

## Использование записей с вариантами и вложенных записей для решения задач в Pascal

### **Цель работы**

Изучение принципов работы с типом данных запись с вариантом.

### **Задачи работы**

Освоение основных приемов и приобретение навыков в составлении программ с использованием записей с вариантом.

### **1 Общие сведения**

В проблемных задачах так описываются объекты, характеризующиеся множеством разнородных признаков, причем часть признаков является общей, а часть – различной по составу для разных групп объектов (например, в описание литературных источников могут включаться выходные данные с одинаковой структурой, однако описания журналов должны содержать еще поле номера и т.д.)

В языке Паскаль такого рода описания отображаются конструируемым типом «запись с вариантами».

Ключевыми элементами работы являются:

- определение структуры описания объекта на уровне постановки задачи (проблемном) и адекватное отображение этого описания в тип «запись с вариантами»;
- создание и обработка текстовых и типизированных файлов с компонентами типа «запись с вариантами».

### **2 Порядок выполнения работы**

Имеются объекты V1, V2, V3, V4, V5, описываемые как записи с вариантами. Число объектов не задается.

- 1) Создать типизированный файл с компонентами типа «запись с вариантами» из описаний этих объектов.
- 2) Вывести в выходной документ информацию из этого файла.

Обработывая этот файл, создать другой типизированный файл из описаний объектов, удовлетворяющих заданному запросу (А, Б, В, Г). Вывести результаты в выходной документ.

*Рекомендация.* Пункты 1 и 2 можно реализовать как разные программы.

Таблица 1 -Варианты объектов

| Объект |                       | Варианты объекта | Их параметры                      |
|--------|-----------------------|------------------|-----------------------------------|
| V1     | Геометрическая Фигура | Треугольник      | Сторона<br>И два прилегающих угла |
|        |                       | Окружность       | Координаты центра<br>и радиус     |

|    |                                         |                                    |                                                                                         |
|----|-----------------------------------------|------------------------------------|-----------------------------------------------------------------------------------------|
|    |                                         | Параллелограмм                     | Две стороны<br>И угол между ними                                                        |
| V2 | Сотрудник<br>Завода                     | Инженерно-<br>технический работник | ФИО, пол, институт,<br>Специальность                                                    |
|    |                                         | Рабочий                            | ФИО, пол, профессия,<br>Разряд                                                          |
| V3 | Участник<br>конкурса<br>Красоты         | Юноша                              | ФИО, вес, рост,<br>Возраст                                                              |
|    |                                         | Девушка                            | ФИО, рост, 3 измерения (объем<br>груди, талии, бедер)                                   |
| V4 | Экологическое<br>Состояние<br>местности | Город                              | Название, количество предприятий,<br>площадь парков, концентрация<br>углерода в воздухе |
|    |                                         | Деревня                            | Название, площадь садов,<br>Площадь полей                                               |
| V5 | Издание: книга                          | Зарубежных авторов                 | Название, автор, переводчик, год<br>издания за рубежом и в СНГ                          |
|    |                                         | Отечественных авторов              | Название, автор, год первого издания,<br>количество переизданий                         |

Таблица 2 -Варианты заданий

| Объект | Вар. | Запрос |                                                                              |
|--------|------|--------|------------------------------------------------------------------------------|
| V1     | 1    | А      | Окружность, для кот. удаление центра от нач. координат $\leq p$ , и ее длина |
|        | 2    | Б      | Квадрат и его площадь                                                        |
|        | 3    | В      | Прямоугольный треугольник и длина его гипотенузы                             |
|        | 4    | Г      | Равносторонний треугольник и его высота                                      |
| V2     | 5    | А      | МЭИ, специальность «Теплотехника»                                            |
|        | 6    | Б      | Слесарь второго разряда                                                      |
|        | 7    | В      | Женщина-экономист                                                            |
|        | 8    | Г      | Мужчина-сталевар                                                             |
| V3     | 9    | А      | Манекенщица: рост больше 176 см, измерения 92-76-94                          |
|        | 10   | Б      | Манекенщик: рост больше 188 см, вес меньше 80 кг                             |
|        | 11   | В      | На кинороль: невысокая девушка (до 165 см) с голубыми глазами                |
|        | 12   | Г      | На роль киногероя: юноша возраста от 17 до 19 лет, рост 175 см               |
| V4     | 13   | А      | Деревня, площадь полей которой превосходит площадь садов                     |
|        | 14   | Б      | Город, в котором на одно предприятие не более 5 га парков                    |
|        | 15   | В      | Город, концентрация углерода в котором превышает допустимую                  |
|        | 16   | Г      | По отдельности параметры городов и деревень                                  |
| V5     | 17   | А      | Книга зарубежного автора, переводчик Пастернак                               |
|        | 18   | Б      | Книга отечественного автора, выдержавшая более 5 переизданий                 |
|        | 19   | В      | Книга зарубежного автора, изданная в СНГ не позже чем через год              |
|        | 20   | Г      | Книга отечественного автора, не переиздававшаяся с 1925 года                 |

### 3 Контрольные вопросы

1) Какой объем памяти резервирует компилятор под запись с вариантами, если варианты имеют разную длину?

2) В *turbo-pascal* поле тега может быть опущено и контроля правильности заполнения полей вариантной части нет. Проанализировать, к чему это может привести.

3) Сравните варианты использования перечисляемого и порядкового типов для описания поля тега.

#### *Требование к оформлению отчета*

Отчет по лабораторной работе должен содержать название и цель работы, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

## Типизированные файлы в Pascal

### Цель работы

Изучение принципов работы с файловым типом данных.

### Задачи работы

Освоение основных приемов и приобретение навыков в составлении программ с использованием типизированных файлов.

## 1 Общие сведения

Входная и выходная информация по отношению к программе на языке Паскаль представляется в виде файлов. Файл в общем случае состоит из нескольких однотипных *компонентов (элементов)* и представляется файловой переменной. Файл может существовать как до, так и после выполнения программы и занимать память намного больше, чем сама программа. Программу пользователя также можно рассматривать в виде совокупности данных, допускающей представление *файловой переменной*.

*Файл* – это поименованная область памяти на внешнем носителе, предназначенная для хранения информации.

*Файл* с точки зрения языка Паскаль – это структурированный тип данных, состоящий из последовательности компонентов в большинстве случаев одного типа и одной длины. Число компонентов, называемое длиной файла, определением типа файла *не* фиксируется.

В зависимости от способа объявления в Турбо Паскале можно выделить 3 вида файлов:

- типизированные, которые задаются предложением **File of** «тип»;
- текстовые, которые задаются предложением **Text**;
- нетипизированные, которые задаются предложением **File**.

Существует два способа доступа к компонентам файла: последовательный и произвольный (прямой). При последовательном способе доступа поиск требуемого элемента начинается с начала файла и проверяется по очереди каждый элемент, пока не будет найден нужный. Произвольный способ доступа позволяет обращаться к элементу файла по его порядковому номеру.

## 1.1 Типизированные файлы

### 1.1.1 Определение файлового типа

Файловый тип определяется в программе с помощью соответствующего описания:

Type

<имя типа> = File of <тип компонентов>;

где <тип компонентов> – любой базовый тип, кроме файлового.

Переменная файлового типа описывается в разделе определения переменных:

Var

<идентификатор> : <имя типа>;

Например, определение типов

```
Const
```

```
Max = 80;
```

**Type**

```
Line = string[Max];
```

```
F1 = File of Line;
```

задает файловый тип F1, элементами которого могут быть строки длиной до 80 символов. Файловая переменная описывается в следующем виде:

**Var**

```
List: F1;
```

Возможно описание файловой переменной непосредственно в разделе определения переменных без определения файлового типа. Однако базовый тип компонентов файла должен быть предварительно определен или же являться стандартным:

## 2 Порядок выполнения работы

2.1 Реализовать программу 1 и 2. Вычислить сумму всех чисел, заканчивающихся нулем.

2.2 Сформировать файл последовательности 15 чисел, в которой каждый  $i$ -й компонент определяется по формуле

$$y = \begin{cases} \sin(i\pi/8), & \text{если } i \leq 8; \\ 4\cos(i(\pi+1)/5), & \text{если } i > 8; \end{cases}$$

Определить количество отрицательных значений, содержащихся в сформированном файле.

2.3 Сформировать файл из значений вещественных случайных величин. Определить для данной последовательности сумму компонентов, значения которых меньше 0.5.

2.4 Выполнить задание из приложения Г по указанию преподавателя.

2.5 Выполнить задание Д по указанию преподавателя.

### *Требование к оформлению отчета*

Отчет по лабораторной работе должен содержать название и цель работы, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

## 3 Контрольные вопросы и задания

- 1) Что такое файл?
- 2) Какие виды файлов существуют в Турбо Паскале?
- 3) Сколько компонентов может содержать файл?
- 4) Как определяется длина файла?
- 5) Каким образом создается и открывается файл?

- 6) Как производится обмен информацией между оперативной памятью и файлом?
- 7) Как контролируется состояние файла?
- 8) Как производится пополнение файла?
- 9) Каким образом осуществляется прямой доступ к файлу?
- 10) Можно ли прочитать файл открытый для записи?
- 11) Приведите описание и фрагмент программы для формирования последовательного файла из целых чисел и последующего его считывания.
- 12) Может ли файл не содержать ни одной записи? Как об этом узнать?
- 13) Пусть дано описание файла. Можно ли из описания определить, какие операции над этим файлом разрешены?



## Нетипизированные файлы в Pascal

### Цель работы

Изучение принципов работы с файловым типом данных.

### Задачи работы

Освоение основных приемов и приобретение навыков в составлении программ с использованием нетипизированных файлов.

## 1 Общие сведения

Нетипизированным (бестиповым) файл объявляется предложением **File** и отличается тем, что для него не указан тип элементов. Отсутствие типа делает этот файл совместимым с любыми другими файлами и позволяет организовать высокоскоростной обмен данными между диском и памятью.

При работе с нетипизированными файлами могут применяться все процедуры и функции, доступные типизированным файлам, за исключением **Read** и **Write**. Эти процедуры заменяются соответствующими высокоскоростными процедурами **Blockread** и **Blockwrite**. Для вызова этих процедур используются следующие предложения:

**Blockread** ( $f_v$ , *Buf*, *N* [, *Result*])

**Blockwrite**( $f_v$ , *Buf*, *N* [, *Result*])

Здесь  $f_v$  – файловая переменная; *Buf* – имя буферной переменной, которая будет участвовать в обмене данными с дисками (ее размер должен быть не менее  $128 \cdot N$  байт); *N* – количество записей по 128 байт каждая, которые должны быть прочитаны или записаны за одно обращение к диску; *Result* – необязательный параметр, содержащий при выходе из процедуры количество фактически обработанных записей.

При использовании в работе с нетипизированными файлами процедуры *Seek* каждый компонент файла рассматривается как запись длиной 128 байт.

**Пример 1.** В качестве примера использования нетипизированных файлов приведем программу быстрого копирования файлов.

**Program** QuikCopy;

**Var**

```
NewFile, OldFile : file; { результирующий и
исходный файлы }
NewName, OldName : string[12];      { имена файлов }
OblBuf: array [1 ..200, 1 ..128] of byte;  {
буферная переменная)
Count: word; { счетчик прочитанных блоков }
Result: word; { счетчик записанных блоков }
lORes : byte; { результат работы с файлом }
Begin {QuikCopy}
    {Связь с исходным файлом для чтения}
```

```

repeat
  WriteLn ('Введите имя исходного файла');
  ReadLn(OldName);
  Assign(OldFile, OldName);
  {$I-}
  Reset(OldFile);
  {$I+}
  IORes := IOResult;
  if IORes <> 0
  then
    WriteLn('Файл', OldFile, 'не
    существует')
until IORes=0;
{Связь с результирующим файлом}
WriteLn('Введите имя выходного файла');
ReadLn(NewName);
Assign(NewFile, NewName);
{$I-}
Rewrite(NewFile);
{$I+}
if IOResult <> 0
then
  begin
    WriteLn('Невозможно создать файл', NewName);
    Halt
  end;
{Копирование файла}
repeat
  BlockRead(OldFile, OblBuf, 200, Count);
  {$I-}
  BlockWrite(NewFile, OblBuf, Count, Result);
  {$I+}
  if Result < Count
  then
    begin
      Writeln('Нет места на диске');
      Halt
    end
until Count < 200;
WriteLn('Копирование завершено');
Close(OldFile);
Close(NewFile)
End. {QuikCopy}

```

## Файлы произвольного доступа

Файл произвольного доступа создается для решения задач, требующих оперативного доступа к хранимой информации, или при наличии зависимости значения поля элемента от порядкового номера элемента в файле. Организовать файл произвольного доступа можно двумя способами:

1) Создать последовательный файл и обращаться к элементам по их порядковому номеру, трактуя последовательный файл как произвольный.

2) Создать файл фиктивных записей, затем загрузить его по ключу фактическими данными; обращение к элементам по ключу предполагает использование процедуры  $Seek(f, n)$ , где  $n$  - номер компонента файла, на который устанавливается указатель файла при выполнении процедуры  $Seek$ , причем нумерация компонентов начинается с нуля.

Обработка файла произвольного доступа должна осуществляться в следующей последовательности:

1. Присвоить имя файловой переменной (процедура Assign).
2. Открыть файл (процедура Reset) и запросить ключ.
3. Подвести указатель по ключу к нужному элементу (процедура Seek).
4. Считать нужный элемент (процедура Read).
5. Выполнить обработку считанной информации.
6. Закрыть файл (процедура Close).

**Пример 2.** Для каждого студента указаны фамилия и оценки (в баллах), полученные на экзаменах по пяти дисциплинам. Требуется вычислить средний балл для каждого студента и упорядочить список студентов по убыванию среднего балла.

Программа имеет следующий вид:

```
Program Sball; { Определение среднего балла и сортировка
}
```

```
Type
```

```
  stud = record
    fam : string[15]; b1,b2,b3,b4,b5:2..5;
    sb : real;
  end ;
  Fl = file of stud ;
```

```
Var
```

```
  tbl: Fl;
  t, y: stud;
  name : string[12];
  i, j, k, m : integer;
  x: real;
```

```
Procedure OutFile(Var F : Fl);
```

```
  Var
```

```
    y: stud;
```

```

        i, m : integer;
Begin {OutFile}
    Reset(F);
    m := FileSize(F);
    for i := 1 to m do
        begin
            Read(F, y);
            with y do
                Writeln(i: 3, fam :
                    15, sb : 6 : 3)
            end;
        end;
    Close(F)
End; {OutFile}
Begin {Sball}
    Writeln ('Введите имя файла');
    ReadLn(name);
    Assign(tbl, name);
    {ВВОД ИСХОДНЫХ ДАННЫХ}
    Writeln ('Введите компоненты записей');
    while not Eof do
        begin
            with y do
                begin
                    Write('Фамилия:');
                    ReadLn(fam);
                    Write('Оценки по пяти экзаменам:');
                    ReadLn(b1,b2,b3, b4,b5)
                end;
            Write(tbl, y)
        end;
    Close(tbl); { Эхо-печать }
    Writeln('Введенный список:');
    OutFile(tbl);
    { вычисление среднего балла }
    Reset(tbl);
    m := FileSize(tbl);
    for i := 1 to m do
        begin
            Read(tbl, y);
            with y do
                sb:=(b1+b2+b3+b4+b5)/5;
            Seek(tbl, i-1);
            Write(tbl, y)
        end;
    Close(tbl);

```

```

{сортировка списка группы студентов}
Reset(tbl);
for i := 0 to m - 2 do
  begin
    Seek(tbl, i);
    k := i;
    Read(tbl, y);
    x := y.sb ;
    for j := i+1 to m -1 do
      begin
        Read(tbl, t);
        if t.sb > x
          then
            begin
              k:=j;
              x := t. sb
            end
          end
      end;
    Seek(tbl, k);
    Read(tbl, t);
    Seek(tbl, i);
    Write(tbl, t);
    Seek(tbl, k);
    Write(tbl, y)
  end ;
Close(tbl);
{печать результатов}
WriteLn('Отсортированный список:');
OutFile(tbl)
End. {Sball}

```

Следует отметить, что текстовые файлы могут быть только последовательного доступа.

## 2 Порядок выполнения работы

При выполнении задания все преобразования производить только над исходным файлом без использования промежуточных файлов. Предусмотреть вывод содержимого файла до и после преобразования, если это возможно.

1) На примере студенческих групп создать файл-базу данных, содержащий запись следующего вида: номер группы, номер семестра, предметы, где поле «предметы» представляет собой список произвольной длины с элементами: название предмета, кафедра, обеспечивающая курс, фамилия лектора.

а) определить, какие предметы обеспечивает данная кафедра.

б) определить название предмета, который читает данный лектор, а также номер группы и номер семестра.

с) определить, в каких группах и в каких семестрах преподавался определенный предмет.

д) определить, какие предметы читались в заданном семестре в одной из групп.

2) Выполнить задание из приложения Е по указанию преподавателя.

*Требование к оформлению отчета*

Отчет по лабораторной работе должен содержать название и цель работы, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

### **3 Вопросы для самопроверки знаний**

1. Какова сущность нетипизированных файлов?
2. Каким образом осуществляется прямой доступ к файлу?
3. Какова последовательность действий при обработке файла произвольного доступа?
4. Опишите форматы высокоскоростных процедур Blockread и Blockwrite.

## Текстовые файлы в Pascal

### ***Цель работы***

Изучение основных принципов работы с текстовыми файлами.

### ***Задачи работы***

Освоить основные приемы и приобрести навыки в составлении программ с использованием текстовых файлов.

## **1 Общие сведения**

Текстовый файл – это файл, состоящий из элементов, являющихся строками. Каждая строка в текстовом файле завершается маркером конца строки. Текстовый файл завершается маркером конца файла. Для описания файловых переменных текстового типа используется стандартный идентификатор **Text**.

## **2 Порядок выполнения работы**

1. Наберите и отладьте следующую программу, производящую ввод символов в текстовый файл и вывод из него

```
program text_file;
var f:text;
    namefile:string[14];
    st:string[80];
begin
    write('Введите имя файла');
    readln(namefile);
    writeln('Вводите строки');
    writeln('Пустая строка – прекратить ввод в файл');
    assign(f,namefile);
    rewrite(f);
    repeat
        readln(st);
        writeln(f,st);
    until st="";
    close(f);
    writeln('Файл создан');
    assign(f,namefile);
    reset(f);
    while not eof(f) do
        begin
            readln(f,st);
            writeln(st);
```

```

end;
close(f);
end.

```

2. Используя следующий фрагмент нахождения количества цифр в файле, дополните исходную программу

```

s:='0123456789';
for i:=1 to length(st) do
  for j:=1 to 10 do
    if st[i]=s[j] then k:=k+1;

```

3. Решите задачу из приложения Ж по указанию преподавателя.

4. Решите задачу из приложения И по указанию преподавателя.

#### *Требование к оформлению отчета*

Отчет по лабораторной работе должен содержать название и цель работы, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

### **3 Вопросы для самопроверки знаний**

1. Что такое текстовый файл?
2. В чем состоит отличие текстовых файлов от других типов файлов?
3. Какие процедуры и функции определены только над текстовыми файлами?
4. Какие процедуры и функции, определенные для всех файлов применимы к текстовым файлам?
5. Какие процедуры и функции не применимы к текстовым файлам?
6. Возможен ли к компонентам файла произвольный доступ?  
Последовательный доступ?
7. Назначение функции **Eoln(f<sub>v</sub>)**?
8. Могут ли компоненты текстового файла быть записями?



## Динамические переменные. Указатели в Pascal

### **Цель работы**

Изучение принципов работы с динамическими типами данных.

### **Задачи работы**

Освоение основных приемов и приобретение навыков в составлении программ с использованием динамических переменных.

## **1 Общие сведения**

В IBM-совместимых компьютерах память условно разделена на сегменты. Компилятор формирует сегмент кода, сегмент данных и сегмент стека, а остальная доступная программе память называется динамической, или хипом, или кучей. Ее можно использовать во время выполнения программы.

В программах, приведенных ранее, для хранения данных использовались простые переменные, массивы или записи. По их описаниям в разделе `var` компилятор определяет, сколько места в памяти необходимо для хранения каждой величины. Такие переменные можно назвать *статическими*. Распределением памяти под них занимается компилятор, а обращение к этим переменным выполняется по имени. *Динамические переменные* создаются в хипе во время выполнения программы. Обращение к ним осуществляется через указатели.

С помощью динамических переменных можно обрабатывать данные, объем которых до начала выполнения программы не известен. Память под такие данные выделяется порциями, или блоками, которые связываются друг с другом. Такая организация хранения данных называется *динамическими структурами*, поскольку их размеры изменяются в процессе выполнения программы.

### **1.1 Указатели**

Имя переменной служит для обращения к области памяти, которую занимает ее значение. Каждый раз, когда в исполняемых операторах программы упоминается какое-либо имя, компилятор подставляет на его место обращение к соответствующей ячейке памяти.

Программист может определить собственные переменные для хранения адресов областей памяти. Такие переменные называются *указателями*. В указателе можно хранить адрес данных или программного кода (например, адрес точки входа в процедуру). Адрес занимает четыре байта и хранится в виде двух слов, одно из которых определяет сегмент, второе – смещение. Указатели в Паскале можно разделить на два вида: стандартные и определяемые программистом. Величины *стандартного типа **pointer*** предназначены для хранения адресов данных произвольного типа. Пример описания указателя стандартного типа:

```
var p : pointer;
```

Программист может определить *указатель на данные или подпрограмму конкретного типа*. Как и для других нестандартных типов, это делается в разделе **Type**, например:

```
type pword = ^word; { читается как "указатель на word" }
var pw : pword;
```

Здесь определяется тип pword как указатель на величины типа word. Переменной pw можно хранить только адреса величин указанного типа. Такие указатели называются *типизированными*. Можно описать указатель на любой тип данных, кроме файловых.

Аналогично другим типам, тип указателя на данные можно описать и непосредственно при описании переменной, например:

```
var pw : ^word; .
```

## 1.2 Операции с указателями

Для указателей определены только операции присваивания и проверки на равенство и неравенство. В Паскале, в отличие от других языков, запрещаются любые арифметические операции с указателями, их ввод-вывод и сравнение на больше-меньше.

Рассмотрим *правила присваивания указателей*.

- любому указателю можно присвоить стандартную константу *nil*, которая означает, что указатель не ссылается на какую-либо конкретную ячейку памяти.

- Указатели стандартного типа pointer совместимы с указателями любого типа.

- Указателю на конкретный тип данных можно присвоить только значение указателя того же или стандартного типа.

Операция @ и функция as!clg позволяют *получить адрес* переменной, например:

```
var x : word;      {переменная}
    pw : ^word;    {указатель на величины типа word}
```

...

```
pw := @w:      {или pw := addr(w); }
```

Для обращения к значению переменной, адрес которой хранится в указателе, примеряется *операция разадресации (разыменования)*, обозначаемая с помощью символа ^ справа от имени указателя, например:

```
pw^ := 2; inc(pw^); writeln(pw^);
```

В первом операторе в ячейку памяти, адрес которой хранится в переменной pw, заносится число 2. В результате выполнения оператора вывода на экране появится число 3.

С величинами, адрес которых хранится в указателе, можно выполнять любые действия, допустимые для значений этого типа. Для простоты восприятия можно считать имя указателя со следующей за ним «крышкой» просто именем переменной, на которую он указывает, хотя на самом деле это, конечно, не так.

Указатели стандартного типа разыменовывать нельзя.

-Указатели можно *сравнивать на равенство и неравенство*, например:

```
if p1 = p2 then ...
if p <> nil then ...
```

В Паскале определены *стандартные функции* для работы с указателями:

- addr(x) : pointer – возвращает адрес x (аналогично операции @), где x – имя переменной или подпрограммы;
- seg(x) : word – возвращает адрес сегмента для x;
- ofs(x) : word – возвращает смещение для x;
- cseg: word – возвращает значение регистра сегмента кода CS;
- dseg: word – возвращает значение регистра сегмента данных DS;
- ptr(seg, ofs : word): pointer – по заданному сегменту и смещению формирует адрес типа pointer.

### 1.3 Динамические переменные

*Динамические переменные* создаются в хипе во время выполнения программы с помощью подпрограмм new или getmem. Динамические переменные не имеют собственных имен – к ним обращаются через указатели.

– Процедура new(var p: тип\_указателя) выделяете динамической памяти участок размера, достаточного для размещения переменной того типа, на который ссылается указатель p, и адрес начала этого участка заносит в этот указатель.

– Функция new(тип\_указателя): pointer выделяет в динамической памяти участок размера, достаточного для размещения переменной базового типа для заданного типа указателя, и возвращает адрес начала этого участка.

**Пример 1.** Иллюстрация использования динамических объектов целого типа, динамических объектов типа Array.

```
Program D1;
Type T=Array [1..10] Of Integer;
Var rm:^T;
    P:^Integer;
    i, a, b, s: Integer;
```

```

Begin
  Write ('a=');
  ReadLn (a);
  Write('b=');
  ReadLn(b);
  New (p); { порождается новая переменная p^ }
  P^:=a+b; { присваивание p^ значения a+b }
  New(rm); { порождение rm^, как переменной массива }
  WriteLn('введите массив');
  For i:=1 to 10 do
    Begin
      Readln (rm[i]);
    End;
  S:=0;
  { нахождение суммы элементов массива rm^ }
  For i:=1 to 10 do
    Begin
      S:=s+rm^[i]
    End;
  WriteLn('s=',s);
  WriteLn('p=',p);
End.

```

**Пример 2.** Иллюстрация порождения и уничтожения массива.

```

Program D2; { сумма элементов каждого массива }
Const n=5; m=6;
Type massiv=Array[1..n,1..m] Of Real;
  ukaz=^massiv;
Var a,b,c:^massiv;
  s1,s2,s3:Real;
  Procedure WS(d:ukaz;Var s:Real);
    Var i,j:Integer;
    Begin
      New(d);
      s:=0;
      For i:=1 to n do
        For j:=1 to m do
          Begin
            d^[i, j]:=50*Random-25;
            s:=s+d^[i,j];
          End;
        Dispose(d);
      End;
    End;
  End;

```

Begin

```
WriteLn('Инициализация матрицы A'); WS(A,s1);
WriteLn('Инициализация матрицы B'); WS(B,s2);
WriteLn('Инициализация матрицы C'); WS(C,s3);
WriteLn('s1=', s1:5:2, 's2=', s2:5:2, 's3=',s3:5:2)
```

End.

## 2 Порядок выполнения работы

1. Проверить все программы, приведенные в работе.
2. Составить программу для решения задачи:

Дано некоторое количество матриц разных размеров. Элементы матриц выбираются случайно. Подсчитывается сумма элементов каждой матрицы, а затем находится их среднее арифметическое. Матрицы вырабатывать динамически.

### *Требование к оформлению отчета*

Отчет по лабораторной работе должен содержать название и цель работы, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

## 3 Вопросы для самопроверки знаний

- 1) Каковы особенности объявления данных динамической структуры?
- 2) Как определить, является ли данный элемент первым в цепочке данных?
- 3) Почему объекты (переменные), создаваемые процедурой new и уничтожаемые процедурой dispose, называют динамическими?
- 4) Почему динамическим переменным не дают имен?
- 5) Как определить, что список компонентов является пустым?
- 6) В чем различие между статическими и динамическими переменными?
- 7) Какую информацию содержит ссылочная переменная?
- 8) Какой процедурой создается указанная переменная?
- 9) Можно ли сформировать указатель, ссылающийся на статическую переменную?
- 10) Что выполняет операция разыменования?
- 11) С помощью каких процедур происходит распределение памяти под динамические переменные?
- 12) Какие состояния может принимать ссылочная переменная?
- 13) В каких случаях указатель может находиться в неопределенном состоянии?
- 14) В чем различие между состоянием nil и неопределенным состоянием?

## Динамические структуры данных в Pascal

### Цель работы

Изучение принципов работы с динамическими структурами данных.

### Задачи работы

Освоение основных приемов и приобретение навыков в составлении программ с использованием динамических структур данных.

### 1 Общие сведения

**Однонаправленный список** – это упорядоченная совокупность элементов, в которой возможен *последовательный* доступ к любому элементу, удаление или добавление элемента в любое место списка.

**Операции абстрактного уровня (функциональная спецификация)** определяют структуру «список» через действия с этой структурой независимо от физической реализации.

Однонаправленным списком является структура, над которой определены следующие операции.

- 1) Сделать список пустым.
- 2) Проверить, список пуст/не пуст.
- 3) Установить указатель в начало списка.
- 4) Передвинуть указатель вперед (переход к следующему элементу).
- 5) Добавить элемент: а) в начало; б) в конец; в) в середину.
- 6) Взять (прочитать) элемент: а) из начала; б) с конца; в) из середины.
- 7) Удалить элемент: а) из начала; б) с конца; в) из середины.

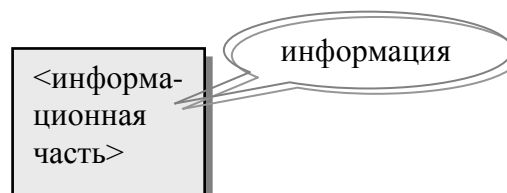
Операции добавления/удаления из середины списка могут иметь модификации, которые оговариваются позже.

**Логическое описание** определяет структуру списка с учетом реализации его в памяти компьютера как динамической структуры. Это означает, что память под элементы будет выделяться динамически, т.е. выделенные участки в общем случае расположены независимо, и упорядочены элементы могут быть только за счет организации ссылок между ними.

Тогда, *во-первых*, элемент списка должен включать *информационную часть*, так как список создается для хранения информации.

*Во-вторых*, все элементы списка создаются динамически по отдельности, но должны быть связаны в упорядоченную (один за другим) цепочку. Для этого каждый элемент должен ссылаться на следующий, т.е. в его структуре должно быть предусмотрено *поле ссылки*, содержащее адрес следующего элемента.

Изобразим соответствующую структуру элемента в общем виде, сразу дав имена полям элемента, типам этих полей и типу элемента в целом (рисунок 1).



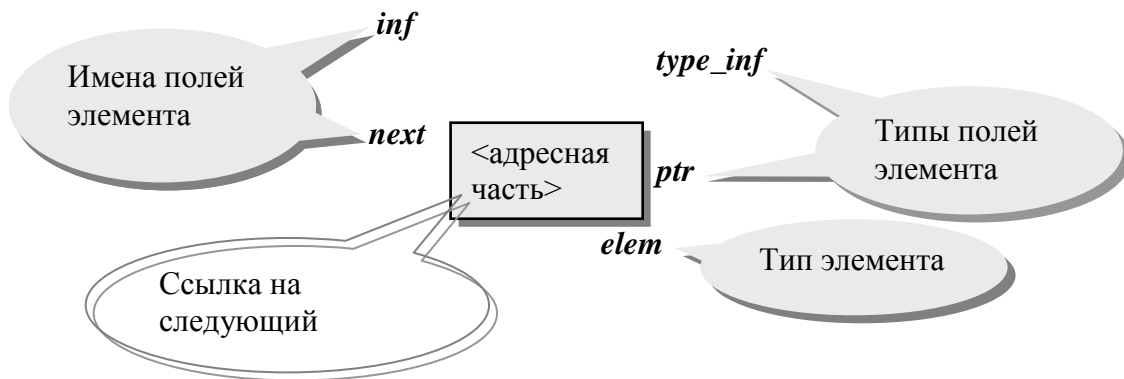


Рисунок 1 Структура элемента

Пусть *an* – адрес начала списка, т.е. адрес его первого элемента. Поскольку каждый элемент списка, кроме последнего, ссылается на следующий за ним, достаточно задать адрес начала списка, чтобы последовательно добраться до любого элемента.

За последним элементом ничего не следует, поэтому содержимое поля ссылки этого элемента должно быть равно *nil*.

Тогда структуру списка в целом можно изобразить следующим образом:

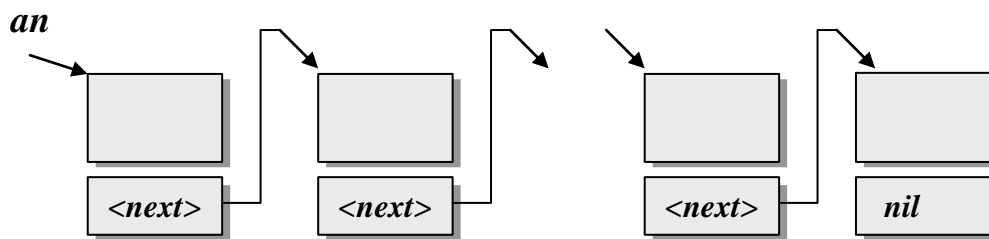


Рисунок 2 Структура списка

Итак, однонаправленный список задается:

- адресом первого элемента (обозначенным здесь *an*);
- типом элемента (этот тип назван *elem*); при этом элемент обязательно содержит по крайней мере два поля – информационное и поле ссылки на следующий элемент.

### 1.1 Принципы реализации в языке Паскаль

Тип *type\_inf* информационной части списка может быть любым, кроме файлового. Чтобы не ограничивать общности описания, будем полагать, что в каждом конкретном случае этот тип подлежит отдельному определению, а тип элемента включает всегда два поля, представленных на рисунок. 2. Очевидно, это тип-запись. Тип второго поля – тип-указатель на этот тип-запись.

```

{типы полей и элемента списка}
type
  type_inf = <описание типа информационной части>; {тип
информационной части }
  ptr = ^elem;                                     {тип-указатель на тип элемент
списка }
  elem = record                                     {тип элемента списка      }
    inf: type_inf;   {информационная часть элемента  }
  end;

```

```

        next: ptr;          {поле ссылки на следующий элемент }
    end;
{переменная – указатель на элемент списка}
var
    an: ptr; {адрес начала списка}

```

Описав далее операции функциональной спецификации с помощью средств Паскаля, **получим реализацию в Паскале информационно-логической структуры «однонаправленный список».**

Чтобы на уровне вызывающей программы каждая абстрактная операция реализовывалась как одна операция языка программирования, оформим абстрактные операции в виде процедур.

**Замечание.** По функциональным возможностям описываемая реализация (описание данных и процедуры-операции) может рассматриваться как тип данных «список», созданный нами, хотя средствами процедурного языка описать такой тип формально нельзя. Но это уже база для описания **класса «список»**, что вплотную подводит нас к **объектно-ориентированной парадигме**.

### 1.3 Реализация алгоритмов и процедур обработки списков

Каждая абстрактная операция оформлена как процедура языка Паскаль. Для ряда операций это сделано в методических целях, так как вряд ли в реальной задаче целесообразно использовать процедуру, содержащую один оператор.

«Передача списка» в процедуру имеет свою специфику: сам список не может быть параметром процедуры, так как не обладает ни типом, ни именем; задать список «как параметр» означает задать как параметр начальный адрес списка.

Тип элементов списка должен быть глобальным по отношению ко всем процедурам-операциям.

Отметим также, что:

- одни и те же функциональные операции могут быть реализованы разными способами, поэтому приведенный ниже список процедур, хотя и достаточно полон и систематизирован, не претендует на совершенство;
- операции, равноправные на абстрактном уровне, могут на физическом уровне реализовываться одна через другую или использовать общие части и иметь совершенно разные по сложности алгоритмы.

При описании процедур будем использовать введенные выше типы и имена.

Все процедуры размещены в модуле **unit**, который можно непосредственно использовать при работе, добавив описание типа информационного поля элемента.

#### **{Процедуры обработки однонаправленного списка}**

```

{ Тип type_inf информационного поля может быть любым, кроме файлового }
{ an,ak – адреса начала и конца списка }
{ am – некоторый заданный адрес элемента }
{ l – адрес элемента,предшествующего элементу с адресом am }

```



```

{ k – текущий адрес элемента списка }
{ val – переменная для записи\чтения информационного поля }
{ val_loc – локальная переменная процедур для обработки информационного поля }
{-----}
unit spis1;
interface
  {Типы полей и элемента списка}
  type
    type_inf = <описание типа информационной части>; {тип информационной части}
  }

  ptr = ^elem; {тип-указатель на тип элемент списка }
  elem = record {тип элемента списка }
    inf: type_inf; {информационная часть элемента }
    next: ptr; {поле ссылки на следующий элемент }
  end;

  {Заголовки процедур}
  {-----Вспомогательные процедуры обработки -----}
  { найти адрес последнего элемента }
  procedure adr_last(an:ptr;var ak:ptr);
  { найти адрес l элемента,предшествующего элементу с адресом am }
  procedure pre_am(an,am:ptr;var l:ptr);
  {найти адрес am элемента с инф. полем val; flag-индикатор наличия элемента }
  procedure val_inf(an:ptr;val:type_inf;var am:ptr;var flag:boolean);
  {создать список с начальным адресом an из файла f добавлением элемента в
начало}
  procedure creat_beg(var f:text;var an:ptr);
  {вывести список с начальным адресом an в файл r }
  procedure print_list(var r:text;an:ptr);
  {----- Абстрактные операции 1-4 -----}
  procedure del_list(var an:ptr); { сделать список пустым }
  function empty(an:ptr):boolean; { проверить, пуст ли список }
  procedure beg_list(an:ptr;var k:ptr); { установить указатель в начало }
  procedure next_ptr(var k:ptr); { установить указатель на следующий элемент }
  {----- Абстрактная операция 5: процедуры добавления в список -----}
  {-----элемента с информационным полем val -----}
  procedure add_beg(var an:ptr;val:type_inf); { в начало }
  procedure add_end(an:ptr;val:type_inf;var ak:ptr); { в конец }
  procedure add_middle(an,am:ptr;val:type_inf); { после элемента с адресом am }
  {----- Абстрактная операция 6: -----}
  {----- процедуры чтения информационной части элемента в переменную val -----}
  procedure take_beg(an:ptr;var val:type_inf); { из начала }
  procedure take_end(an:ptr;var val :type_inf); { из конца }
  procedure take_middle(an,am:ptr;var val:type_inf); { из элемента с адресом am }
  {----- Абстрактная операция 7: процедуры удаления элемента-----}
  procedure del_beg(var an:ptr); { из начала }
  procedure del_end(an:ptr;var ak:ptr); { из конца }
  procedure del_middle(an,am:ptr); { после элемента с адресом am }
implementation
  {Описания процедур}
  {-----Вспомогательные процедуры обработки -----}
  procedure adr_last(an:ptr;var ak:ptr);

```

```

var k:ptr;
begin
  k:=an;
  while k^.next<>nil do
    k:=k^.next;
    ak:=k;
  end;
procedure pre_am(an,am:ptr;var l:ptr);
var k:ptr;
begin
  k:=an;
  while k^.next <> am do
    k:=k^.next;
    l:=k;
  end;
procedure val_inf(an:ptr;val:type_inf;var am:ptr;var flag:boolean);
var k:ptr;
begin
  k:=an; flag:=false;
  while (k <> nil) and not flag do
    begin
      if k^.inf=val then
        begin
          am:=k; flag:=true
        end;
      k:=k^.next;
    end;
  end;
procedure creat_beg(var f:text;var an:ptr);
var k:ptr; val_loc:type_inf;
begin
  del_list(an);
  while not eof(f) do
    begin
      read(f,val_loc);
      add_beg(an,val_loc);
    end;
  end;
procedure print_list(var r:text;an:ptr);
var k:ptr;i:byte;
begin
  k:=an;
  while k<>nil do
    begin
      write(r,k^.inf);
      k:=k^.next;
    end;
  end;
{----- Абстрактные операции 1-4 -----}
procedure del_list(var an:ptr);
begin
  an:=nil

```

```

end;

function empty(an:ptr):boolean;
begin
  empty:=an=nil
end;
procedure beg_list(an:ptr;var k:ptr);
begin
  k:=an;
end;
procedure next_ptr(var k:ptr);
begin
  k:=k^.next;
end;
{----- Абстрактная операция 5: процедуры добавления в список -----}
{-----элемента с информационным полем val -----}
procedure add_beg(var an:ptr;val:type_inf);
var k:ptr;
begin
  new(k);
  k^.next:=an;
  k^.inf:=val;
  an:=k;
end;
procedure add_end(an:ptr;val:type_inf;var ak:ptr);
var k:ptr;
begin
  adr_last(an,ak);
  new(k);
  ak^.next:=k;
  k^.next:=nil;
  k^.inf:=val;
  ak:=k;
end;

procedure add_middle(an,am:ptr;val:type_inf);
var l,k:ptr;
begin
  new(k);
  k^.inf:=val;
  l:=am^.next;
  am^.next:=k;
  k^.next:=l;
end;
{----- Абстрактная операция 6: -----}
{----- процедуры чтения информационной части элемента в переменную val -----}
procedure take_beg(an:ptr;var val:type_inf);
begin
  val:=an^.inf;
end;
procedure take_end(an:ptr;var val :type_inf);
var ak:ptr;

```

```

begin
  adr_last(an,ak);
  val:=ak^.inf;
end;
procedure take_middle(an,am:ptr;var val:type_inf);
begin
  val:=am^.inf;
end;
{----- Абстрактная операция 7: процедуры удаления -----}
procedure del_beg(var an:ptr);
begin
  an:=an^.next;
end;
procedure del_end(an:ptr;var ak:ptr);
  var l:ptr;
begin
  adr_last(an,ak);
  pre_am(an,ak,l);
  l^.next:=nil;
  ak:=l;
end;
procedure del_middle(an,am:ptr);
  var l:ptr;
begin
  pre_am(an,am,l);
  l^.next:=am^.next;
end;
end {unit}.

```

## 2 Порядок выполнения работы

По указанию преподавателя решите задачу из приложения К

При формировании списков значения элементов задавать произвольно, каждый этап алгоритма выполнять в виде отдельной процедуры. Предусмотреть вывод на печать всех промежуточных результатов работы.

### *Требование к оформлению отчета*

Отчет по лабораторной работе должен содержать название и цель работы, результаты выполнения индивидуальных заданий, ответы на контрольные вопросы, а также основные теоретические положения по теме занятия.

## 3 Вопросы для самопроверки знаний

- 1) Можно ли назвать список студенческой группы (например, в журнале) однонаправленным списком в строгом смысле, как это определено в п. 2.2?
- 2) В процедурах удаления/добавления элементов сделан акцент на удаление/добавление элемента после элемента с заданным *адресом*, а в

списке вспомогательных процедур присутствуют процедуры поиска **адресов** различных элементов.

- 3) Реально нас интересуют не адреса, а **информационные поля** элементов. Может быть, целесообразно было бы разработать отдельные процедуры удаления элемента с заданным **информационным полем**? Добавления элемента до/после/вместо элемента с заданным **информационным полем**?
- 4) Почему такие процедуры отсутствуют в предложенном списке?
- 5) Почему в функциях изменения списка путем удаления или добавления элемента в середину отсутствуют выходные значения?
- 6) Какими достаточно универсальными вспомогательными процедурами можно, по Вашему мнению, дополнить предложенный список?
- 7) Какие модификации процедур вставки/удаления могли бы Вы предложить?

## Библиографический список

- 1) Аляев Ю.А. Алгоритмизация и языки программирования Pascal, C++, Visual Basic/ Ю.А. Аляев. – М.: Финансы и статистика, 2011. – 320 с.
- 2) Немнюгин С.А. Turbo Pascal. Программирование на языке высокого уровня. Учебник для ВУЗов. - СПб.: Питер, 2010. – 544 с.
- 3) Практикум по алгоритмизации на языке Паскаль/ Ю.А. Аляев, В.П. Гладков, О.А. Козлов. – М.: Финансы и статистика, 2009. – 527 с.
- 4) Фаронов В.В. Турбо Паскаль. – СПб.: Питер, 2011. - 368 с.
- 5) Шамсутдинова, Т. М. Электронный учебник по основам алгоритмизации и программирования на языке Паскаль 7 [Электронный ресурс]: [электрон. учеб.]: подгот. по свидетельству об отраслевой регистрации разработки № 2366 от 20.02.2003; ИнРФ 50200300141 от 28.02.2003. – Электрон. дан. и прогр. – Уфа: БГАУ, 2003. – 1 электрон. опт. диск (CD-ROM); 12 см. – Систем. требования: ПК Pentium; Windows 95/98/2000/XP; ОЗУ 32 Мб. – Загл. с экрана.
- 6) Новичков В.С. Алгоритмизация и программирование на Турбо Паскале: Учебное пособие. – М.: Горячая линия – Телеком, 2009 – 438 с.
- 7) Ускова О.Ф. Программирование на языке Паскаль: задачник – СПб.: Питер, 2011. – 336с.

## ПРИЛОЖЕНИЕ А

### Задачи для самостоятельного решения

Даны несколько слов из русских букв. Вывести на экран:

- 1) все гласные буквы, которые входят в каждое слово;
- 2) все согласные буквы, которые не входят ни в одно слово;
- 3) все звонкие согласные буквы, которые входят хотя бы в одно слово;
- 4) все глухие согласные буквы, которые не входят хотя бы в одно слово;
- 5) все согласные буквы, которые входят только в одно слово;
- 6) все глухие согласные буквы, которые не входят только в одно слово;
- 7) все звонкие согласные буквы, которые входят больше чем в одно слово;
- 8) все звонкие согласные буквы, которые входят в каждое нечетное слово и не входят ни в одно четное слово;
- 9) все глухие согласные буквы, которые входят в каждое нечетное слово и не входят хотя бы в одно четное слово;
- 10) все гласные буквы, которые не входят более чем в одно слово.

Примечание: гласные буквы – а, е, и, о, у, ы, э, ю, я;  
согласные – все остальные, кроме й, ь, ъ;  
звонкие согласные – б, в, г, д, ж, з, л, м, н, р;  
глухие согласные – к, п, с, т, ф, х, ц, ч, ш, щ.

## ПРИЛОЖЕНИЕ Б

### Задачи для самостоятельного решения

1) Даны три множества  $X_1$ ,  $X_2$ ,  $X_3$ , содержащие целые числа из диапазона  $[1..100]$ . Сформировать новое множество  $Y = (X_1 \cup X_2) \cap (X_2 \cup X_3)$ , из которого выделить подмножество нечетных чисел.

2) Даны три множества  $X_1$ ,  $X_2$ ,  $X_3$ , содержащие целые числа из диапазона  $[1..100]$ . Сформировать новое множество  $Y = (X_1 \cup X_2) - (X_2 \cup X_3)$ , из которого выделить подмножество чисел кратных 3.

3) Дано множество  $X_1$ , содержащее целые числа из диапазона  $[1..255]$ . Сформировать новое множество  $Y$  путем выделения из множества  $X_1$  нечетных чисел и чисел, делящихся без остатка на 17.

4) Даны множества  $X_1$  и  $X_2$ , содержащие целые числа из диапазона  $[1..255]$ . Сформировать новое множество  $Y = (X_1 \cap X_2)$  и выделить из него все четные числа и числа, делящиеся без остатка на 19.

5) Дано множество  $X_1$ , содержащее целые числа из диапазона  $[50..100]$ . Сформировать новое множество  $Y_1$  путем выделения из множества  $X_1$  нечетных чисел и множество  $Y_2$  путем выделения из множества  $X_1$  чисел, кратных 5. На экран вывести множество  $Y_3 = (Y_1 \cap Y_2)$ .

6) Дано множество  $X_1$ , содержащее символы из диапазона  $[a..z]$ . Сформировать новое множество  $Y_1$  путем выделения из множества  $X_1$  всех символов, расположенных в алфавите позже  $f$  и раньше  $m$ , и множество  $Y_2$  путем выделения из множества  $X_1$  символов, расположенных в алфавите раньше  $g$  и позже  $j$ . На экран вывести множество  $Y_3 = (Y_1 - Y_2)$ .

7) Дано множество, состоящее из различных символов. Вывести на экран упорядоченные по убыванию символы русского алфавита.

8) Ввести с клавиатуры множество – последовательность символов из диапазона от А до Я. Определить число различных (без повторений) букв, входящих в данную последовательность.

9) Подсчитать во введенном текстовом отрывке отдельно количество цифр, латинских букв и знаков пунктуации.

10) Дан текст. Вывести в алфавитном порядке все буквы текста, входящие в него более двух раз.

## ПРИЛОЖЕНИЕ В

Имеются некоторые сведения об абитуриентах поступающих в университет. Организовать ввод определенных данных о нескольких абитуриентах и поиск искомых записей по критерию.

1. *Запись:* Ф.И.О. год рождения, год окончания школы, средний балл по аттестату.  
*Результаты обработки данных:* список со средним баллом  $> 4.5$  и количество отличников.
2. *Запись:* Ф.И.О. год рождения, какое учебное заведение окончил, место проживания (город, село).  
*Результаты обработки данных:* список абитуриентов, проживающих не в Уфе и количество окончивших техникум.
3. *Запись:* Ф.И.О. год окончания школы, средний балл по аттестату, служба в армии.  
*Результаты обработки данных:* список окончивших школу до 1990 года и количество служивших в армии.
4. *Запись:* Ф.И.О. год рождения, пол, средний балл по аттестату.  
*Результаты обработки данных:* список абитуриентов моложе 18 лет и количество абитуриентов мужского пола.
5. *Запись:* Ф.И.О. год рождения, средний балл по аттестату, баллы на вступительных экзаменах (4 экзамена).  
*Результаты обработки данных:* список абитуриентов набравших проходной балл и количество сдавших экзамены без троек.
6. *Запись:* Ф.И.О., пол, средний балл по аттестату, баллы на вступительных экзаменах (4 экзамена).  
*Результаты обработки данных:* список абитуриентов со средним баллом по аттестату  $> 4.75$  и количество сдавших экзамены только на "отлично".
7. *Запись:* Ф.И.О. год рождения, год окончания школы, средний балл по аттестату.  
*Результаты обработки данных:* список со средним баллом  $> 4.5$  и количество отличников.
8. *Запись:* Ф.И.О. год рождения, какое учебное заведение окончил, место проживания (город, село).  
*Результаты обработки данных:* список абитуриентов, проживающих не в Уфе и количество окончивших техникум.
9. *Запись:* Ф.И.О. год окончания школы, средний балл по аттестату, служба в армии.  
*Результаты обработки данных:* список окончивших школу до 1990 года и количество служивших в армии.



10. *Запись:* Ф.И.О., год рождения, пол, средний балл по аттестату.  
*Результаты обработки данных:* список абитуриентов моложе 18 лет и количество абитуриентов мужского пола.
11. *Запись:* Ф.И.О., год рождения, средний балл по аттестату, баллы на вступительных экзаменах (4 экзамена).  
*Результаты обработки данных:* список абитуриентов набравших проходной балл (18) и количество сдавших экзамены без троек.
12. *Запись:* Ф.И.О., пол, средний балл по аттестату, баллы на вступительных экзаменах (4 экзамена)  
*Результаты обработки данных:* список абитуриентов со средним баллом по аттестату  $>4.75$  и количество сдавших экзамены только на "отлично".

## ПРИЛОЖЕНИЕ Г

- 1 Дан файл, компонентами которого являются действительные числа. Найти разность первой и последней компонент.
- 2 Дан файл, компонентами которого являются действительные числа. Найти сумму наибольшего и наименьшего из значений компонент.
- 3 Дан файл, компонентами которого являются действительные числа. Найти наибольшее из значений компонент с четными номерами.
- 4 Дан файл, компонентами которого являются действительные числа. Найти наименьшее из значений компонент с нечетными номерами.
- 5 Дан файл, компонентами которого являются действительные числа. Найти наибольший по модулю компонент.
- 6 Дан файл, компонентами которого являются целые числа. Найти количество четных чисел среди компонент файла.
- 7 Дан файл, компонентами которого являются действительные числа. Найти последнюю компоненту файла.
- 8 Дан файл, компонентами которого являются действительные числа. Найти модуль суммы и квадрат произведения компонент файла.
- 9 Дан файл, компонентами которого являются действительные числа. Найти сумму квадратов компонент файла.
- 10 Дан файл, компонентами которого являются действительные числа. Найти произведение компонент файла.
- 11 Дан файл, компонентами которого являются целые числа. Найти количество чисел кратных 3.
- 12 Дан файл, компонентами которого являются целые числа. Найти количество компонент файла кратных 2 и не кратных 4.
- 13 Дан файл, компонентами которого являются целые числа. Найти сумму положительных чисел среди компонент файла.
- 14 Дан файл, компонентами которого являются целые числа. Вычислить произведение отрицательных компонент.
- 15 Дан файл, компонентами которого являются целые числа. Найти сумму компонент файла, расположенных между первым и последним элементом.

## ПРИЛОЖЕНИЕ Д

- 1 Сформировать файл, содержащий фамилии нескольких студентов. Добавить к полученному файлу фамилии еще 2-3 студентов.
- 2 Записать в файл оценки (в баллах), полученные некоторым студентом на экзаменах в течение всех сессий. Добавить в начало файла оценки, полученные на вступительных экзаменах.
- 3 Записать в файл оценки (в баллах), полученные некоторым студентом на экзаменах в течение всех сессий, и определить средний балл.
- 4 Сформировать два файла. В один из них поместить фамилии пяти ваших знакомых, а в другой - номера их телефонов. Составить программу, которая по фамилии вашего знакомого определяет номер его телефона.
- 5 Сформировать два файла. В один из них поместить фамилии пяти ваших знакомых, а в другой - номера их телефонов. Составить программу, которая по номеру телефона вашего знакомого определяет его фамилию.
- 6 Сформировать файл, компоненты которого являются записями, содержащими информацию о фамилии и дате рождения 10 ваших друзей. Составить программу определения даты рождения по фамилии вашего друга.
- 7 Сформировать файл, компоненты которого являются записями, содержащими информацию о фамилии и дате рождения 10 ваших товарищей. Составить программу определения фамилии вашего товарища по дате его рождения.
- 8 Записать в текстовый файл первое предложение из параграфа 1.1. Определить число слов в данном предложении.
- 9 Сформировать файл, состоящий из пяти записей, каждая из которых содержит фамилию любимого вами актера и название фильма, в котором он снимался. Составить программу определения названия фильма по фамилии актера, который в нем снимался.
- 10 Сформировать файл, состоящий из пяти записей, каждая из которых содержит фамилию любимого вами актера и название фильма, в котором он снимался. Составить программу определения фамилии актера по названию фильма, в котором он снимался.
- 11 Записать в текстовый файл предложение «В ЛЕСУ РОДИЛАСЬ ЕЛОЧКА». Определить число гласных в данном предложении.
- 12 Записать в текстовый файл предложение «У ЛУКОМОРЬЯ ДУБ ЗЕЛЕНый». Определить число согласных букв в данном предложении.
- 13 Сформировать файл, компонентами которого являются названия нескольких троллейбусных остановок по некоторому маршруту. Добавить в конец файла названия еще нескольких остановок данного маршрута.
- 14 Сформировать файл, элементами которого являются значения функции  $y = \sin(x_i) + 2\cos(x_i)$  в точках  $X = (0.1, 0.2, 0.25, 0.33, 1.78, 2.05, 2.23)$ . Определить компонент файла, имеющий минимальное значение.

## ПРИЛОЖЕНИЕ Е

- 1) Создать файл-список идентификаторов произвольной длины.
- 2) Преобразовать файл, содержащий текст программы, таким образом, чтобы каждый внутренний оператор был сдвинут на две позиции вправо по сравнению с внешним.
- 3) Для программы, записанной в файле в виде непрерывного текста, преобразовать файл таким образом, чтобы каждый оператор располагался на отдельной строке.
- 4) В столбцах матрицы произвольного размера, размещенной во внешнем файле, провести перестановку ее элементов таким образом, чтобы максимальный элемент каждого столбца оказался на главной диагонали.
- 5) Перемножить два сверхдлинных целых числа, записанные в файле. Результат записать в тот же файл.
- 6) Произвести сортировку файла целых чисел методом «пузырька».
- 7) Файл целых чисел циклически сдвинуть влево или вправо на  $K$  элементов в зависимости от знака числа  $K$ .
- 8) Во внешний файл записать два многочлена в виде последовательности пар чисел - коэффициента и показателя соответствующей степени. Сложить многочлены и результат поместить в исходный файл.
- 9) Дан символьный файл  $f$ . Записать в этот же файл компоненты файла в обратном порядке.
- 10) В файле целых чисел  $f$  исключить повторное вхождение одних и тех же чисел.
- 11) Удалить из символьного файла  $f$  все однобуквенные слова, не используя дополнительные файлы.
- 12) Дан файл целых чисел. Преобразовать его таким образом, чтобы вначале шли все отрицательные числа, а затем - неотрицательные.
- 13) Заданный файл целых чисел отсортировать методом максимального элемента, не используя дополнительный файл.
- 14) В некотором файле содержится таблица футбольного чемпионата, в котором участвовало  $n$  команд. Перестроить эту таблицу таким образом, чтобы команды располагались в соответствии с занятыми ими местами.

**ПРИЛОЖЕНИЕ Ж**

1. Дан текстовый файл. Подсчитать число строк в нем.
2. Дан текстовый файл. Подсчитать количество символов в каждой строке.
3. Задан текстовый файл. Дописать в него 6 одинаковых строк.
4. Дан текстовый файл. Подсчитать число вхождений заданного символа в текст.
5. Дан текстовый файл. Подсчитать количество гласных в тексте.
6. Дан текстовый файл. Подсчитать количество строчек с длиной не более 20.
7. Дан текстовый файл. Подсчитать количество строк, содержащих введенного слово.
8. Дан текстовый файл. Найти длину самой длинной строки.
9. Дан текстовый файл. Найти номер самой короткой строки.
10. Дан текстовый файл. Вывести самую длинную строку.
11. Дан текстовый файл. Вывести все его строки, начинающиеся с буквы «Т».
12. Дан текстовый файл. Вывести все строки, вторых имеется более 3 пробелов.
13. Дан текстовый файл. Найти количество строк, в которых число символов равно 5.
14. Дан текстовый файл. Найти количество строк, которые заканчиваются точкой.
15. Дан текстовый файл. Найти количество букв «Е» в нем.

**ПРИЛОЖЕНИЕ И**

1. Дан текстовый файл, в котором содержатся и числа. Найти количество цифр в файле.
2. Дан текстовый файл, в котором содержатся и числа. Найти сумму всех этих цифр.
3. Дан текстовый файл, в котором содержатся и числа. Найти среди них наибольшее.
4. Дан текстовый файл, в котором содержатся и числа. Найти среди них наименьшее.
5. Дан текстовый файл, в котором содержатся и числа. Найти их произведение.
6. Дан текстовый файл, в котором содержатся и числа. Найти число цифр отличных от 0.
7. Дан текстовый файл, в котором содержатся и числа. Найти число цифр превышающих число 7.
8. Дан текстовый файл, в котором содержатся и числа. Найти среди них наименьшее по модулю.
9. Дан текстовый файл, в котором содержатся и числа. Найти их среднее арифметическое.
10. Дан текстовый файл, в котором содержатся и числа. Найти сумму их кодов.
11. Дан текстовый файл, в котором содержатся и числа. Найти удвоенное произведение цифр, не превышающих цифру 5 .
12. Дан текстовый файл, в котором содержатся и числа. Определить, есть ли среди них равные.
13. Дан текстовый файл, в котором содержатся и числа. Определить, различны ли первая и последняя цифра.
14. Дан текстовый файл, в котором содержатся и числа. Найти все возможные сочетания из этих цифр.
15. Дан текстовый файл. Найти количество гласных букв в тексте.

## ПРИЛОЖЕНИЕ К

- 1) Составить процедуру нахождения среднего арифметического элементов непустого списка  $L$ . Используя данную процедуру, найти максимальное среднее арифметическое в списках  $K, M, N$ .
- 2) Составить процедуру проверки упорядоченности символьных элементов списка  $L$  по алфавиту. Используя данную процедуру, проанализировать элементы списков  $M, N, K$ .
- 3) Составить функцию, подсчитывающую количество слов списка, которые начинаются и оканчиваются одной и той же литерой. Используя данную функцию, найти сумму числа слов, начинающихся и оканчивающихся одной и той же литерой в списках  $M, K, L$ .
- 4) Составить процедуру, которая помещает в начало списка  $L$  количество четных элементов, а в конец списка - количество нечетных элементов. С использованием данной процедуры преобразовать списки  $M, N$  и  $K$ .
- 5) Составить процедуру, проверяющую на равенство значения элементов списков  $L1, L2$  и подсчитывающую количество одинаковых элементов в них. Используя процедуру, проанализировать пары списков  $M1$  и  $M2, N1$  и  $N2$ .
- 6) Составить процедуру, определяющую вхождение списка  $L1$  в список  $L2$  и наоборот. Если один из списков длиннее, удалить лишние элементы из его начала. Используя процедуру, проанализировать пары списков  $M1$  и  $M2, N1$  и  $N2$ .
- 7) Составить процедуру, определяющую порядковый номер наибольшего элемента последовательности натуральных чисел. Используя данную процедуру, проанализировать последовательности натуральных чисел  $M$  и  $N$ .
- 8) Составить процедуру вставки элемента  $E$  после каждого элемента списка, превышающего некоторое значение  $P$ . Подсчитать количество вставленных элементов.
- 9) В списке натуральных чисел переставить элементы по следующему правилу: если текущий элемент больше некоторого числа  $P$ , то поместить следующий за ним элемент в конец цепочки; если текущий элемент меньше или равен числу  $P$ , перенести в начало цепочки текущий элемент (первый оставить без изменения).
- 10) Построить список  $L1$  - копию списка  $L$ , расположив элементы в обратном порядке (первый элемент списка  $L$  - последний элемент списка  $L1$ ). Заменить элементы списка  $L$ , имеющие четные значения, на элементы списка  $L1$ , имеющие нечетные значения.
- 11) Построить список  $L$ , упорядочив его по возрастанию, из двух неупорядоченных списков  $L1$  и  $L2$ .
- 12) Определить, входит ли элемент  $E$  в список  $L$ , подсчитать количество

вхождений данного элемента в список. Вставить первый элемент цепочки после каждого вхождения  $E$  в список.

13) Построить список  $L$ , упорядочив его по убыванию, из четных элементов  $L1$  и нечетных элементов  $L2$ .

14) Сформировать список  $L$  из элементов, которые входят одновременно в списки  $L1$  и  $L2$ . Дописать в начало элементы, которые входят в  $L1$ , но не входят в  $L2$ , а в конец - элементы, которые входят в  $L2$ , но не входят в  $L1$ .