Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет»

Кафедра «Информационные технологии и автоматизированные системы»

Полякова О.А.

«Информатика 1».
«Информатика 2».

Теоретические материалы и методические указания для выполнения лабораторных работ

4-5 ЗЕ
7 ЗЕ

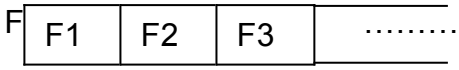
Часть 2

Пермь 2017

Оглавление

9. Файловые типы данных

Файл представляет собой произвольные последовательности элементов одного и того же типа, причём длина этих последовательностей заранее не определена, а конкретизируется в процессе выполнения программы. Этот тип значения получил в Паскале название *файлового*. Условно файл можно изобразить как некоторую ленту, у которой есть начало, а конец не фиксируется. Элементы файла записываются на эту ленту последовательно, друг за другом:



где F- имя файла, а F1, F2, F3- его элементы.

В программировании существует несколько разновидностей файлов, отличающихся методом доступа к его компонентам. Мы рассмотрим простейший метод доступа, состоящий в том, что по файлу можно двигаться только последовательно, начиная с первого его элемента, и, кроме этого, всегда существует возможность начать просмотр файла с его начала. Таким образом, чтобы добраться до пятого элемента файла, необходимо, начав с первого элемента, пройти через предыдущие четыре элемента. Такие файлы называются *файлами* последовательного доступа, или последовательными файлами. Так что, например, невозможно прочитать 100-й элемент последовательного файла, не прочитав предыдущие 99.

9.1. Инициализация файла

Имя файла дает возможность программе работать одновременно с несколькими файлами, длина файла ограничивается только емкостью устройств внешней памяти.

Файловый тип или переменную файлового типа можно задать одним из трех способов:

<имя>=FILE OF <тип>;
<имя>=TEXT;
<имя>=FILE,
где <имя> - имя файлового типа,
FILE,OF - зарезервированные слова (файл, из);
TEXT - имя стандартного типа текстовых файлов.

От способа объявления можно выделить три вида файлов:

1. Типизированные файлы (задаются предложением FILE OF);
2. Текстовые файлы (тип TEXT);
3. Нетипизированные файлы (тип FILE).

Файлы, а также логические устройства, становятся доступны программе только после выполнения особой процедуры открытия файла(логического устройства). Эта процедура заключается в связывании ранее объявленной файловой переменной с именем существующего или вновь создаваемого файла, а также в указании направления обмена информацией: чтение из файла или запись в него.

Файловая переменная связывается с именем файла в результате обращения к стандартной процедуре ASSIGN:

ASSIGN(<ф.п.>,<имя файла или л.у.>);

здесь <ф.п.> - файловая переменная (правильный идентификатор, объявленный в программе как переменная файлового типа);

<имя файла или л.у.> - текстовое выражение, содержащее имя файла или л.у.

Например:

```
Var: data:file of integer ;                {задаём файловую переменную data содержащую целые числа типа integer}
     begin
       assign(data, ' c:\tp\user.me ');   {связываем файловую переменную с существующим файлом или с файлом который будет создан}
     end.
```

Инициировать файл означает указать для этого файла направление передачи данных. В TP можно открыть файл для чтения, для записи информации, и для чтения и записи одновременно.

Для чтения файл инициируется с помощью стандартной процедуры RESET:

RESET(<ф.п.>);

где <ф.п.> - файловая переменная, связанная ранее процедурой ASSIGN с уже существующим файлом.

Также можно обращаться к типизированным файлам, открытым процедурой RESET, с помощью процедуры REWRITE (для текстовых - нельзя).

Стандартная процедура:

REWRITE(<ф.п.>)

инициирует запись информации в файл, связанный ранее с файловой переменной. При выполнении этой процедуры старый файл уничтожается если таков был и создаётся новый файл.

Стандартная процедура:

APPEND(<ф.п.>)

инициирует запись в ранее существовавший текстовый файл для его расширения - эту процедуру можно использовать только для текстовых файлов.

CLOSE(<ф.п.>)

закрывает файл, но связь с <ф.п.> с именем файла сохраняется, при выходе из программы все файловые переменные задействованы процедурами RESET(<ф.п.>), REWRITE(<ф.п.>), APPEND(<ф.п.>), должны бать закрыты процедурой CLOSE(<ф.п.>).

ERASE(<ф.п.>)

уничтожение файла. Перед выполнением процедуры необходимо закрыть файл.

Например:

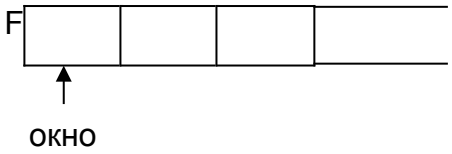
```
Var: data:file of integer ;                {задаём файловую переменную data содержащую целые числа типа integer}
     begin
       assign(data, ' c:\tp\user.me ');   {связываем файловую переменную с существующим файлом или с файлом который будет создан}
       reset(data);
       ……..                              {тело программы которая в своей работе                    }
       ……..                              {использует данные записанные ранее в файл user.me }
       ……..
       ……..
       close(data);
     end.
```

9.2. Файлы и работа с ними

Для удобства описания действий над файлами введём понятие «окно файла» или просто «окно». Окно представляет позицию доступа, т.е. ту позицию файла, которая доступна для чтения в режиме чтения, либо для записи в режиме записи. Позиция файла, следующая за последней компонентой файла (или первая позиция пустого файла), помечается специальным маркером. Благодаря этому маркеру определяется конец файла.

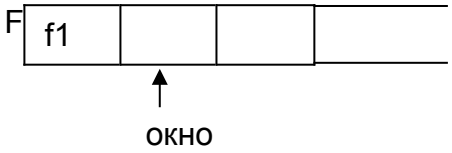
Оператор RESET(F) или REWRITE(F) устанавливает файл с именем F в начальное состояние режима записи или чтения, в результате чего окно устанавливается на первую позицию файла.

После выполнения процедуры REWRITE(F) файл с именем F переходит в режим записи. Результат выполнения выглядит следующим образом:

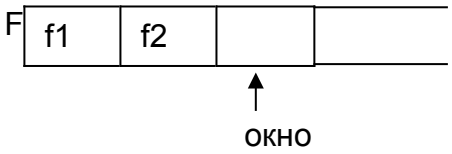


Оператор WRITE(F, X) записывает в файл (в ту позицию, на которую указывает окно) очередную компоненту, содержащуюся в переменной X, после чего окно сдвигается на следующую позицию. Естественно тип переменной X должен совпадать с типом компонента файла F. Результат выполнения выглядит следующим образом:

X:=f1;



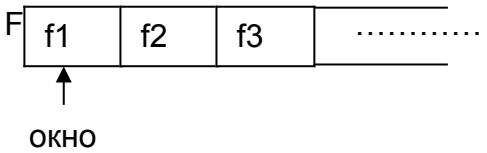
X:=f2;



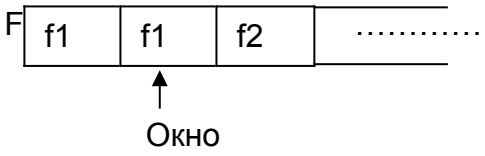
и так далее.

Запись в файл с помощью процедуры WRITE(F, X) можно производить только после выполнения процедуры REWRITE(F).

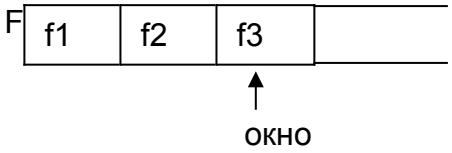
После выполнения процедуры RESET(F) файл с именем F переходит в режим чтения. Результат выполнения выглядит следующим образом:



Оператор READ(F, X) читает из файла в переменную X (из той позиции, на которую указывает окно) очередную компоненту, после чего окно сдвигается на следующую позицию. Естественно тип переменной X должен совпадать с типом компонента файла F. Результат выполнения выглядит следующим образом:



X=f1;



X=f2;

и так далее.

Чтение из файла с помощью процедуры READ(F, X) можно производить только после выполнения процедуры RESET(F).

При чтении из файла нужно определять, указывает ли окно на какую-то компоненту файла или указывает на маркер конца файла. Для определения этого факта в паскале введена в употребление стандартная логическая функция с именем EOF (от *end of file*), обращение к которой имеет вид eof(F).

Значение этой функции равно TRUE, если окно указывает на маркер конца файла с именем F, и значению FALSE в противном случае.

Недопустимо использовать процедуру READ(F, X) если eof(F)=TRUE.

Например:

```
Var: data:file of integer ;              {задаём файловую переменную data содержащую
целые}
      x:integer ;                        {числа типа integer
}
    begin
      assign(data, ' c:\tp\user.me ');   {связываем файловую переменную с
существующим }
      reset(data);                       {файлом или с файлом который будет создан
}
      while not eof(data) {если не уверены что файл содержит данные сначала
проверяем}
        begin                 {а потом читаем
}
          read(data,x);       {читаем все данные из файла до конца
}
          ……..
          ……..
         end;
      close(data);
    end.
```

или

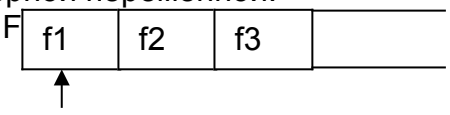
Например:

```
Var: data:file of integer ;                {задаём файловую переменную data содержащую
целые}
      x:integer ;                          {числа типа integer
}
    begin
      assign(data, ' c:\tp\user.me ');   {связываем файловую переменную с
существующим }
      reset(data);                         {файлом или с файлом который будет создан
}
      repeat
        read(data,x);      {читаем все данные из файла до конца
}
        ……..
        ……..
      until eof(data);    {если уверены что файл содержит хоть одну компоненту, можно}
      close(data);        {сначала прочитать её в переменную X а потом проверить       }
    end.
```

Стандартная процедура чтения компоненты из файла READ(F, X) выполняет два действия: первое-это копирования компоненты файла в переменную X, а второе-это передвижение окна на следующую компонента. В некоторых задачах удобно иметь возможность производить эти два действия отдельно. Для таких случаев удобно использовать буферные переменные файлов.

Предположим, что файл F установлен в режим чтения. Тогда буферной переменной будем называть конструкцию F^, т.е. к имени файловой переменной справа приписывается символ ^. Эту переменную не надо описывать в разделе описания переменных, она определяется автоматически с введением в употребление файловой переменной. Тип буферной переменной совпадает с типом компоненты файла. Со значением этой буферной переменной можно выполнять любые действия, которые можно выполнять с любыми переменными.

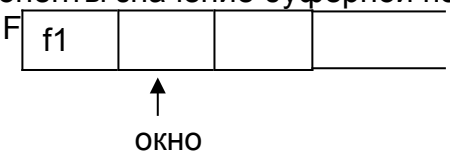
В режиме чтения значение переменной F^ всегда является та компонента файла на которую указывает окно. При выполнении процедуры RESET(F) происходит не только установка окна на начало файла, но и присваивание значения первой компоненты файла буферной переменной:



F^ = f1;

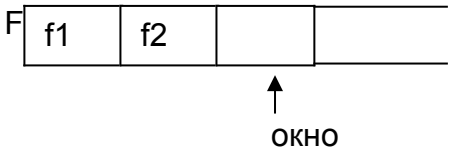
Для передвижения окна на следующую компоненту непустого файла предусмотрена процедура GET(F), параметром которой является имя файловой переменной. Результат этой процедуры состоит в передвижении окна на следующую позицию файла и присваиванием значения этой следующей компоненты буферной переменной.

В режиме записи буферная переменная выполняет роль поставщика значений компонент файла. Процедура PUT(F) производит запись в файл F в качестве очередной компоненты значение буферной переменной F^ и сдвигает окно на следующую позицию:



до выполнения процедуры GET(F)

```
F^:=f2;      {присваиваем значение буферной переменной}
Get(F);
```



после выполнения процедуры GET(F)

Лабораторная работа №11.

Работа с внешними файлами

Цель задания.

1. Ознакомление с возможностями организации файлов на внешних носителях в ЭВМ.
2. Получение навыков работы с внешними файлами.

Постановка задачи.

Подготовить данные об абитуриентах, поступающих в институт. Информацию о каждом абитуриенте оформить в виде записи, содержащей следующие поля:

1. ФИО.
2. Год рождения.
3. Год окончания школы.
4. Оценки в аттестате.
5. Признак - нуждается ли в общежитии.
6. Оценки вступительных экзаменов.

Разработать программу записи подготовленных данных во внешний файл и программу обработки созданного внешнего файла.

Удалить из внешнего файла все записи, удовлетворяющие условию, заданному в варианте, и распечатать записи , оставшиеся в файле.

Добавить N записей в начало(конец) внешнего файла и распечатать записи полученного файла согласно конкретному варианту.

Содержание отчета.

1. Постановка задачи.
2. Анкетные данные абитуриентов.
3. Тексты программ.
4. Распечатка результатов выполнения программы.

Методические указания.

При подготовке исходных данных необходимо учесть, что выходная информация программы обработки внешнего файла должна составлять не менее одной четверти от входной.

Образец выполнения задания.

Лабораторная работа №11, вариант № 5.

Работа с внешними файлами

Постановка задачи.

Подготовить данные об абитуриентах, поступающих в институт. Информацию о каждом абитуриенте оформить в виде записи, содержащей следующие поля:

1. ФИО.
2. Год рождения.
3. Год окончания школы.
4. Оценки в аттестате.
5. Признак - нуждается ли в общежитии.
6. Оценки вступительных экзаменов.

Разработать программу записи подготовленных данных во внешний файл и программу обработки созданного внешнего файла.

Удалить из внешнего файла все карточки иногородних студентов которым больше 18 лет, и распечатать записи оставшиеся в файле.

Добавить 4 записи в начало(конец) внешнего файла и распечатать список студентов не нуждающихся в общежитии.

Анкетные данные на абитуриентов в конце методического пособия.

Тексты программ №1.

Программа записи подготовленных данных во внешний файл.

```
program lab11{вариант № 5};
uses crt;
type data=record
        fio:string[30];
        godr,godo:integer;
        ates:record
             mat,fiz,rus:integer;
             end;
        haus:boolean;
        ekz:record
            mat,fiz,rus:integer;
            end;
        end;
var stu:data;
   files:file of data;
   keys:char;
begin
 assign(files,'c:\tp\data.dat');
 rewrite(files);
 repeat;
  clrscr;
  writeln('      Вводим данные об абитуриентт');
  write('Фамилия Имя Отчество..');readln(stu.fio);
  write('Год рождения.....');readln(stu.godr);
  write('Год окончания школы...');readln(stu.godo);
  writeln('      Оценки в атестате');
  write('Метематика.....');readln(stu.ates.mat);
  write('Физика.....');readln(stu.ates.fiz);
```

```
 write('Русский язык...');readln(stu.ates.rus);
 writeln('Нуждается ли в общежитии (1-да/2-нет)');
 keys:=readkey;if keys='1' then stu.haus:=true
                          else stu.haus:=false;
 writeln('      Оценки на вступительных экзаменах');
 write('Метематика.....');readln(stu.ekz.mat);
 write('Физика.........');readln(stu.ekz.fiz);
 write('Русский язык...');readln(stu.ekz.rus);
 write(files,stu);
 clrscr;
 writeln('      Данные об абитуриенте введены');
 writeln('1-Добавить данные');
 writeln('2-Выход');
 writeln('      (нажмите 1 или 2)');
 keys:=readkey;
until keys='2';
close(files);
end.
```

Тексты программ №2.

Программа обработки созданного внешнего файла.

Удалить из внешнего файла все карточки иногородних студентов которым больше 18 лет, и распечатать записи оставшиеся в файле.

```
program lab11{вариант № 5};
uses crt;
type data=record
        fio:string[30];
        godr,godo:integer;
        ates:record
             mat,fiz,rus:integer;
             end;
        haus:boolean;
        ekz:record
            mat,fiz,rus:integer;
            end;
        end;
var stu:data;
   files,filee:file of data;
   keys:char;
begin
 assign(files,'c:\tp\data.dat');
 assign(filee,'c:\datae.dat');
 reset(files);rewrite(filee);
 while not eof(files)do
 begin
  read(files,stu);write(filee,stu);
 end;
 reset(filee);rewrite(files);
 while not eof(filee) do
 begin
  read(filee,stu);
  if (stu.godr>=1983)or not(stu.haus) then write(files,stu);
```

```
 end;
 close(files);close(filee);erase(filee);
end.
```

Тексты программ №3.

Программа обработки созданного внешнего файла.

Добавить 4 записи в начало(конец) внешнего файла и распечатать список студентов не нуждающихся в общежитии.

```
program lab11{вариант № 5};
uses crt;
type data=record
        fio:string[30];
        godr,godo:integer;
        ates:record
             mat,fiz,rus:integer;
             end;
        haus:boolean;
        ekz:record
            mat,fiz,rus:integer;
            end;
        end;
var stu:data;
   files,filee:file of data;
   keys:char;
procedure add(n:integer);
begin
 assign(files,'c:\tp\data.dat');assign(filee,'c:\date.dat');
 reset(files);rewrite(filee);
 while not eof(files) do begin read(files,stu);write(filee,stu);end;
 reset(filee);rewrite(files);
 while not eof(filee) do begin read(filee,stu);write(files,stu);end;
 for n:=1 to n do
 begin
  clrscr;
  writeln('      Вводим данные об абитуриентt');
  write('Фамилия Имя Отчество..');readln(stu.fio);
  write('Год рождения.....');readln(stu.godr);
  write('Год окончания школы...');readln(stu.godo);
  writeln('      Оценки в атестате');
  write('Метематика.....');readln(stu.ates.mat);
  write('Физика.....');readln(stu.ates.fiz);
  write('Русский язык...');readln(stu.ates.rus);
  writeln('Нуждается ли в общежитии (1-да/2-нет)');
  keys:=readkey;if keys='1' then stu.haus:=true
                          else stu.haus:=false;
  writeln('      Оценки на вступительных экзаменах');
  write('Метематика.....');readln(stu.ekz.mat);
  write('Физика.....');readln(stu.ekz.fiz);
  write('Русский язык...');readln(stu.ekz.rus);
  write(files,stu);
 end;
```

```
 close(files);close(filee);erase(filee);
end;
procedure print;
var n:byte;
begin
 writeln('      ',stu.fio);
 writeln('Год рождения.....',stu.godr);
 writeln('Год окончания школы...',stu.godo);
 writeln('      Оценки в атестате');
 writeln('Метематика.....',stu.ates.mat);
 writeln('Физика.....',stu.ates.fiz);
 writeln('Русский язык.....',stu.ates.rus);
 writeln('      Оценки на вступительных экзаменах');
 writeln('Метематика.....',stu.ekz.mat);
 writeln('Физика.....',stu.ekz.fiz);
 writeln('Русский язык.....',stu.ekz.rus);
 if stu.haus then writeln('Нуждается в общежитии')
             else writeln('Не нуждается в общежитии');
 for n:=1 to 79 do write('*');
 writeln('Для продолжения нажмите Enter !');
 readln;
end;
begin
 add(4);
 clrscr;
 reset(files);
 while not eof(files) do
 begin
  read(files,stu);
  if not stu.haus then print;
 end;
 close(files);
end.
```

Варианты заданий.

Из внешнего файла, содержащего исходные данные, удалить записи, соответствующие:

1) абитуриентам, получившим хотя бы одну оценку 2.
2) иногородним абитуриентам, получившим все оценки 3.
3) абитуриентам, имеющим средний балл меньше 4 и хотя бы оценку 3 в аттестате.
4) абитуриентам, имеющим средний балл в аттестате меньше 4.
5) абитуриентам, имеющим средний балл больше 4,5 и нуждающимся в общежитии.
6) иногородним абитуриентам, не получившим ни одной оценки 5.
7) абитуриентам, имеющим отличный аттестат и получившим все оценки 3.
8) абитуриентам, нуждающимся в общежитии и получившим хотя бы одну оценку 3.
9) абитуриентам, которые старше 18 лет и имеющим отличный аттестат.
10) абитуриентам, имеющим средний балл в аттестате меньше 4,5.
11) абитуриентам, имеющим средний балл за экзамены меньше 4,5.
12) абитуриентам, имеющим отличный аттестат и получившим за экзамены одну оценку 3.

13) абитуриентам, не имеющим в аттестате оценок 5.
14) абитуриентам, имеющим средний балл в аттестате 4,5 и получившим хотя бы одну оценку 3.
15) абитуриентам, имеющим отличный аттестат и нуждающимся в общежитии.
16) абитуриентам, у которых больше одной оценки 3 в аттестате.
17) абитуриентам, у которых больше одной оценки 3 за экзамены.
18) список абитуриентов, сдавших вступительные экзамены на оценки 4 и 5 и возраст менее 20 лет.
19) список абитуриентов, у которых две оценки 3 за экзамены и отличный аттестат.
20) список абитуриентов, у которых все экзамены сданы на 4 и нуждающимся в общежитии.
21) список абитуриентов, у которых не более двух оценок 5 в аттестате.
22) список абитуриентов, у которых менее двух оценок 5 за экзамены.
23) список абитуриентов, у которых менее одной оценки 5 за экзамены.
24)список абитуриентов, у которых менее одной оценки 5 за экзамены и нуждающимся в общежитии.

Используя внешний файл, содержащий исходные данные, добавить N записей и распечатать:
1) список абитуриентов, имеющих в аттестате только оценки 5 (N=2).
2) список абитуриентов, имеющих в аттестате одну оценку 4, а остальные 5 (N=3).
3) список абитуриентов, имеющих средний балл больше 4,5 (N=4).
4) список абитуриентов, имеющих средний балл меньше 4 (N=1).
5) список абитуриентов, нуждающихся в общежитии (N=3).
6) список абитуриентов, сдавших вступительные экзамены только на оценки 5 (N=4).
7) список абитуриентов, сдавших вступительные экзамены на оценки 4 и 5 (N=2).
8) список абитуриентов, сдавших экзамены с 2-мя оценками 4 и остальными оценками 5 (N=3).
9) список абитуриентов, имеющих средний балл в аттестате 4,5(N=3).
10) список абитуриентов, имеющих в аттестате две оценки 4, а остальные 5(N=2).
11) список абитуриентов, имеющих средний балл меньше 4(N=3).
12) список абитуриентов, у которых все экзамены сданы на 4 (N=4).
13) список абитуриентов, у которых одна оценка 4, а остальные 5(N=3).
14) список абитуриентов, у которых одна оценка 5, а остальные 4(N=5).
15) список абитуриентов, у которых одна оценка 3 в аттестате. (N=3).
16) список абитуриентов, имеющих больше двух оценок 3 в аттестате. (N=2).
17) список абитуриентов, имеющих две оценки 3 в аттестате(N=4).
18) список абитуриентов, имеющих средний балл в аттестате ниже 4,5 (N=3).
19) список абитуриентов, у которых две оценки 3 за экзамены и отличный аттестат (N=2).
20) список абитуриентов, у которых нет ни одной оценки 5 в аттестате (N=4).
21) список абитуриентов, у которых отличный аттестат и средний балл за экзамены меньше 4(N=3).
22) список абитуриентов, имеющих средний балл больше 4 и оценки 3 в аттестате (N=4).
23) абитуриентов возраст которых больше 18 лет и все оценки 5 за экзамены(N=2).
24) абитуриентов, у которых средний балл больше 4,5 и одна оценка 3 в аттестате(N=3).

9.3. Сортировка файлов.

9.3.1. Слияние упорядоченных последовательностей.

К последовательному файлу нельзя непосредственно применить метод внутренней сортировки (сортировки массивов). Это объясняется тем, что в последовательном файле в каждый момент имеется доступ только к одному элементу. Это строгое ограничение, по сравнению с возможностями, которые даёт массив, поэтому для файлов приходится применять другие методы сортировки.

Разумеется, в этом случае, когда размер файла не велик и объём оперативной памяти достаточен для его размещения, можно прочитать файл в память, отсортировать полученный массив одним из методов внутренней сортировки, а затем отсортированный массив записать в файл. Однако, это не является типичным случаем; более того, в системах обработки данных такая ситуация встречается крайне редко.

Далее мы рассмотрим общий случай, когда сортируемый файл имеет объём значительно больше, чем объём доступный оперативной памяти. Такой файл сортируется поэтапно.

Все методы внешней сортировки сначала применяют внутреннюю сортировку, а затем используют стратегию слияния. Общая идея состоит в том, чтобы прочесть в оперативную память как можно больше записей, отсортировать их одним из методов внутренней сортировки, записать их во внешнюю память как уже отсортированный блок и повторять этот процесс до тех пор, пока весь файл не превратится в последовательность отсортированных блоков. Затем производится слияние в более крупные блоки до тех пор, пока не останется один блок, на этом сортировка файла завершится.

Итак, основным методом внешней сортировки является слияние упорядоченных последовательностей. Рассмотрим слияние более подробно.

Пусть даны две упорядоченные последовательности (файл) A и B со следующими значениями числовых ключей:

A: 06 12 18 42 44 56
B: 07 14 15 17 19

Читаем первые элементы каждого из файлов, сравниваем их и записываем меньший элемент в выходной файл C, а на его место читаем следующий элемент из соответствующего файла. После этой операции имеем следующее (элементы, прочитанные в память, подчёркнуты):

A: 12 18 42 44 55
B: 07 14 15 17 19
C: 06

Повторяем данную операцию:

A: 12 18 42 44 55
B: 14 15 17 19
C: 06 07

Эта операция повторяется до тех пор, пока один из исходных файлов не закончится:
A: 42 44 55
B: пустой
C: 06 07 12 14 15 17 18 19

После этого оставшиеся элементы непустого файла переписываются в файл C, который будет иметь окончательный вид:

C: 06 07 12 14 15 17 18 19 42 44 55

Таким образом, мы получили отсортированный файл C. рассмотренная схема называется двухпутевым слиянием. Обобщая эту схему, можно получить p-путевое слияние для упорядоченных последовательностей, размещённых в P-файлах.

Алгоритм P-путевого слияния

1. Установить r:=P.
2. Если r>1, то перейти к следующему пункту, иначе к пункту 5.
3. Прочитать начальные элементы оставшихся r непустых последовательностей, записать минимальный элемент в выходной файл, а на его место прочитать следующий элемент из соответствующего файла.
4. Если последовательность, из которой читался элемент, станет пустой, запомнить этот и уменьшить r на 1. Перейти к пункту 2.
5. Переписать оставшиеся элементы непустой последовательности в выходной файл.

Закончить выполнения алгоритма.

Нетрудно видеть, что использование для сортировки файла P-путевого слияния требует предварительно разделения исходного файла на P частей, сортировки каждого из частей в оперативной памяти и размещение их во внешней памяти в виде P файлов. Если размещать каждый файл на отдельном устройстве (ленте), то потребуется P лент. Поэтому приходится на одной ленте (в одном файле) размещать несколько отсортированных в оперативной памяти блоков. Такие блоки, записи в которых упорядочены, будем называть сериями.

Вернёмся к двух путевому слиянию и попробуем распространить его на случай, когда в каждом из двух входных файлов имеется несколько серий. Пусть исходные файлы A и B имеют следующие значения (серии подчёркнуты):

A: 44 55 18 15 17 07
B: 12 42 94 06 67 14 15 19

Используя алгоритм 2x-путевого слияния, выполним слияние первой серии файла A и первой серии файла B. получим следующее значение файла C:

C: 12 42 44 55 94

Далее выполним слияние вторых серий файлов A и B, запишем полученную последовательность в файл C.

C: 12 42 44 55 94 06 18 67

Аналогично работаем с третьими сериями, итак до тех пор, пока один из файлов не окажется пустым (в нашем примере файл B). Оставшуюся серию непустого файла (четвёртую серию файла A) переписываем в выходной файл C. В результате получим файл:

C: 12 42 44 55 94 06 18 67 14 15 15 17 19 07

Мы получили выходной файл, состоящий из упорядоченных последовательностей (серий).

Алгоритм слияния серий двух файлов

1. Прочитать начальные элементы каждого файла. Если один из файлов пустой, перейти к пункту 5.

2. Сравнить два элемента и записать минимальный в выходной файл, а на его место прочитать следующий элемент.

3. Если конец серии не достигнут, перейти к пункту 2.

4. Если достигнут конец серии, переписать элементы другого файла до окончания серии и перейти к пункту 1.

5. Если в другом файле имеются серии (хотя бы одна) переписать элементы этих серий в выходной файл. Закончить выполнения алгоритма.

В отличие от предыдущего алгоритма в рассмотренном необходимо обнаружить конец серии.

Это можно сделать одним из следующих способов:

Ключ прочитанной записи сравнивается с ключом следующей записи. Если значение ключа следующей записи меньше, то серия закончена.

В качестве метки, указывающей на конец серии, вставляют специальную запись с определённым значением ключа. Это значение не должно входить в диапазон значений ключей записи файла. Как только будет прочитано это значение, конец серии достигнут.

Серии в файлах делают фиксированной длины.

Вернёмся к последнему примеру. В результате слияний серий мы получили файл C, состоящий из нескольких серий. Однако, в целом файл ещё не отсортирован.

Как продолжить его сортировку? Необходимо разделить этот файл на два файла, переписывая в них поочерёдно серии. В результате такого разделения мы получим новые файлы A и B, состоящие из серий в среднем в два раза большей длины, чем исходные. Затем вновь выполняем слияние и разделение и так до тех пор, пока не получим выходной файл C, состоящий из одной серии. Естественно, такой файл будет отсортирован.

Алгоритм внешней сортировки

1. Используя оперативную память, сформировать как можно более длинные серии из элементов заданного файла. Распределить эти серии на несколько файлов.

2. Сформировать более длинные серии путём слияния серий нескольких файлов. Вновь распределить эти серии на несколько файлов. Повторять эту операцию.

3. Если в конце образуется одна серия, закончить сортировку.

9.3.2. Сортировка сбалансированным слиянием

Сортировка сбалансированным слиянием является наиболее простым методом сортировки, предполагающим многократное распределение и слияние серий.

Для того, чтобы представить себе как выполняется сортировка, рассмотрим следующий пример.

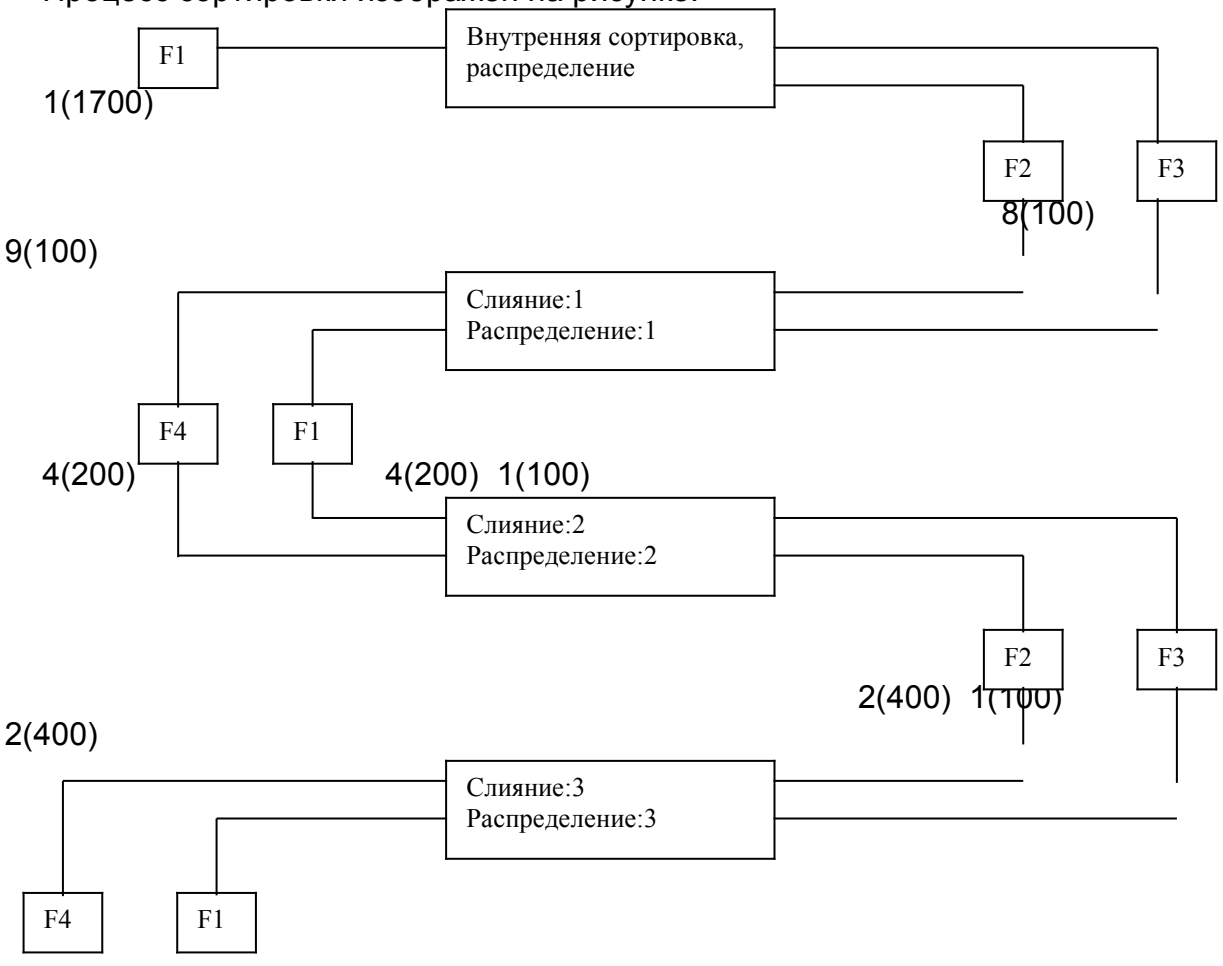
Пусть исходный файл содержит 1700 записей. Допустим, что в оперативной памяти можно разместить не более 100 записей. Предположим также, что мы можем использовать для сортировки 4 файла, которые мы обозначим f1 f2 f3 f4. Исходным является файл f1. Будем использовать 2-путевое слияние. Начинаем сортировку. Считываем 100 записей, сортируем их одним из методов внутренней сортировки и записываем в файл f2. Затем считываем следующие 100 записей, сортируем и записываем в файл f3, следующие 100 сортированных записей снова записываем в файл f2 и т.д. В результате мы получим 9 серий по 100 записей в файле f3.

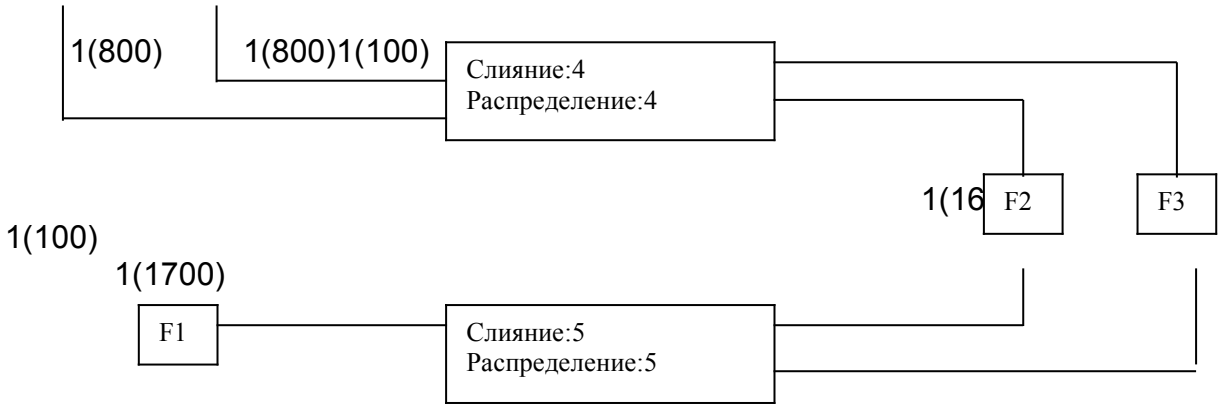
Далее серии файлов f2 и f3 сливаем и вновь в процессе слияния попеременно записываем, формируя новые серии, на файле f1 и f4. В итоге в файле f1 мы будем иметь 4 серии по 200 записей и 1 серию из 100 записей, а в файле f4 – 4 серии по 200 записей.

Аналогичный цикл слияний серий с f1 и f4 и записи на f2 и f3 приведёт к 2 сериям по 400 записей и 1 серии из 100 записей в f2 и к 2 сериям по 400 записей в f3.

После ещё трёх циклов в f1 окажется полностью отсортированный файл.

Процесс сортировки изображён на рисунке:





Программа: Сортировка сбалансированным слиянием.

Программа использует уже готовый файл содержащий не отсортированные данные и не распечатывает отсортированный файл (программа только сортирует файл file1).

```
program balans;
const n=10;{количество записей в сериях при внутренней(начальной) сортировке и распределении}
type item=record
        key:integer;
        {описание других полей}
        end;
     filetype=file of item;
var f1,f2,f3,f4:filetype;
    z,l,all:integer;
    eor,per:boolean;
    mas:array[1..n]of integer;
    data:item;
{z-для подсчёта числа серий}
{eor-индикатор конца серии}
procedure readfile;{процедура считывает 100 записей в массив mas}
begin
 l:=0;
 while not eof(f1)and not(l=n) do
 begin l:=l+1;read(f1,data);mas[l]:=data.key;end;
 all:=l;
end;
procedure massort;{процедура сортировки массива mas}
var buf,units:integer;
    m:boolean;
begin
 units:=all;
 repeat
  m:=true;;
  units:=units-1;
  for l:=1 to units do
  begin
   if mas[l]>mas[l+1]
    then begin buf:=mas[l];mas[l]:=mas[l+1];mas[l+1]:=buf;m:=false;end;
  end;
 until m or (units=1);
```

```
    end;
    procedure sort(var x:filetype);{процедура считывает, сортирует а затем записывает на
ленту серии по 100 записей}
    begin
     readfile;massort;
     for l:=1 to all do begin data.key:=mas[l];write(x,data);end;
    end;
    procedure sortrun;{прцедура первоночальной (внутренней) сортировки и
распределении}
    begin
     reset(f1);rewrite(f2);rewrite(f3);
     while not eof(f1) do
    {распределяем сначала на ленту f2 а затем на ленту f3 и т.д.}
     begin sort(f2);if not eof(f1) then sort(f3);end;
     close(f1);close(f2);close(f3);
    end;
    procedure copy(var x,y:filetype);{процедура копирования записи и определения конца
серии}
    var buf,buf1:item;
    begin
     read(x,buf);write(y,buf);
     if eof(x) then eor:=true
               else begin
                    {заглядываем вперёд}
                    read(x,buf1);
                    {возвращаемся на исходную запись}
                    seek(x,filepos(x)-1);
                    eor:=buf1.key<buf.key
                   end;
    end;
    procedure copyrun(var x,y:filetype);{процедура копирования серий}
     {переписать серии из X в Y}
    begin
     repeat
      copy(x,y);
     until eor;
    end;
    procedure merge(var a,b,c:filetype);{процедура слияния серии}
    var bufa,bufb:item;
    begin
     repeat
      read(a,bufa);seek(a,filepos(a)-1);
      read(b,bufb);seek(b,filepos(b)-1);
      if bufa.key<bufb.key
       then begin;copy(a,c);if eor then copyrun(b,c);end
       else begin;copy(b,c);if eor then copyrun(a,c);end;
     until eor
    end;
    procedure ende(var a,b,c:filetype);
    begin
     while not eof(a) do begin;copyrun(a,c);z:=z+1;end;
     while not eof(b) do begin;copyrun(b,c);z:=z+1;end;
```

```
    end;
    procedure mer(var a,b,d,e:filetype);{процедура сливает серии из a и b и распределяет
между d и e}
    var con:boolean;
    begin
     con:=false;
     reset(a);reset(b);rewrite(d);rewrite(e);
     while (not eof(a))and(not eof(b)) do
     begin
      merge(a,b,d);z:=z+1;con:=true;
      if (not eof(a))and(not eof(b)) then begin merge(a,b,e);z:=z+1;con:=false;end;
     end;
     if con then ende(a,b,e);
     ende(a,b,d);
     close(a);close(b);close(d);close(e);
    end;
    begin
     assign(f1,'c:\tp\file1');
     assign(f2,'c:\tp\file2');
     assign(f3,'c:\tp\file3');
     assign(f4,'c:\tp\file4');
     sortrun;
     per:=false;
     repeat
      z:=0;
      if per then begin mer(f1,f4,f2,f3);per:=false;end
             else begin mer(f2,f3,f1,f4);per:=true;end;
     until z=1;
    end.
```

Результат работы:

```
 0  3  86  20  27  67  32  16  37  43  8  47  7  84  6  29  92  37  77  33  70
84  72  31  16  33  47  25  83  28  48  15  87  29  77  98  49  89  83  2  14  1
4  50  2  59  1  77  65  77  71  56  21  68  59  96  64  100  24  68  30  9  77
 50  88  51  57  95  68  34  1  71  99  77  75  20  14  91  78  59  86  69  29
9  63  28  88  16  27  54  96  17  16  27  18  58  50  29  16  61  74
Нажмите Enter для продолжения !

 0  1  1  2  2  3  6  7  8  9  9  14  14  14  15  16  16  16  16  16  17  18  20
  20  21  24  25  27  27  27  28  28  29  29  29  29  30  31  32  33  33  34  37
  37  43  47  47  48  49  50  50  50  51  54  56  57  58  59  59  59  61  63  64
  65  67  68  68  68  69  70  71  71  72  74  75  77  77  77  77  77  77  78  83
  83  84  84  86  86  87  88  88  89  91  92  95  96  96  98  99  100
Нажмите Enter для продолжения !
```

Рассмотренный пример позволяет нам сделать ряд выводов. Процесс сортировки можно разделить на два существенно различных этапа. На первом этапе выполняется внутренняя сортировка записей и формирование из них серий возможно большего размера. Эти серии распределяются по двум или нескольким лентам, причём точный способ распределения зависит от конкретного метода последующего слияния.

На втором этапе выполняется несколько циклов слияния этих серий во всё большие и большие серии, пока все записи не окажутся в одной серии.

Возможно ли при сортировке обойтись только тремя магнитными лентами?

Да, возможно. Для этого начальные серии распределяем на файлы T2 и T3 сливаем в файл T1. Затем полученные серии снова распределяем по файлам T2 и T3, до тех пор, пока не получим в файле T1 одну серию.

Недостатком такого метода является излишнее копирование, что увеличивает время сортировки.

Анализируя рассмотренный алгоритм сортировки, мы можем выделить в нём повторяющиеся операции, которые однократно обрабатывают все записи. Такие операции называются фазой, а наименьший процесс, который, повторяясь, образует процесс сортировки, называют проходом. В приведённом примере сортировка производится за пять проходов, каждый проход состоит из фазы распределения и фазы слияния.

Сколько требуется файлов для сбалансированного P-путевого слияния?

В общем случае требуется 2P файлов. Однако, если после распределения серий сливать их в один файл, то количество требуемых файлов сократится почти в два раза и будет равно P+1, но за это приходится расплачиваться увеличением времени сортировки вдвое, за счёт дополнительного копирования, о чём уже говорилось.

9.3.3. Сортировка простым слиянием

Рассмотрим простой метод, когда не обращая внимания на содержимое заданного исходного файла, его рассматривают как состоящий из серий длиной 1. Разделяя этот файл на два и производя их слияние, получают упорядоченные пары, т.е. серии длиной 2. На следующих шагах упорядоченные пары сливаются в упорядоченные четвёрки, четвёрки сливаются в восьмёрки и т.д. Весь процесс повторяется до тех пор, пока не будет упорядочен весь файл. Таким образом на каждом проходе длина серии удваивается, причём если длина исходного файла H не является степенью 2, то последняя серия будет иметь число элементов равное H-2^k. В данном случае не требуется выполнения первого этапа сортировки, т.е. внутренней сортировки для формирования начальных серий. Такой метод называется простым слиянием.

Как было показано выше, простым слиянием состоит из двух фаз: фазы разделения серий по двум файлам и фазы слияния упорядоченных серий. В соответствии с этим напишем две процедуры: разделения и слияния.

Исходные данные представим следующим образом.

```
type item=record
        key:integer;
        {описание других полей}
        end;
    filetype=file of item;
```

```
var a,b,c:filetype;
    l:integer;
```

Исходные данные разместим в файле C, в него же будем записывать результаты слияний серий. В файл A и B будем распределять серии из файла C.

Пусть M-длина серии (M=1,2,4,8,……). Тогда распределение серий длиною M по файлам A и B можно выполнить следующим образом:

```
begin
reset(c);rewrite(a);rewrite(b);
repeat
{переписать очередную серию из файла C в файл A}
{переписать следующую серию из файла C в файл B}
until eof(c);
close(a);close(b);close(c);
end;
```

Посмотрим как можно переписать серию. Для этого необходимо последовательно переписывать записи до тех пор, пока не кончится либо серия, либо файл C. Поскольку серии имеют фиксированную длину, определить конец серии не представляет труда, для этого достаточно организовать счётчик. Тогда переписать серию из файла C в A можно следующим образом.

```
k:=0; {счётчик}
ok:=eof(c);
while not ok do
 begin
  read(c,buf);write(a,buf);
  k:=k+1;
  ok:=eof(c) or (k=l);
 end;
```

Аналогично переписывается серия из C в B. запишем окончательно процедуру распределения серий из C в A и B.

{процедура распределения серий}

```
procedure Distribute;
var buf:item;
    k:integer;
    ok:boolean;
begin
 reset(c);rewrite(a);rewrite(b);
 repeat
  k:=0;ok:=eof(c);
  while not ok do
   begin
    read(c,buf);write(a,buf);
    k:=k+1;ok:=eof(c) or (k=l);
   end;
```

```
  k:=0;ok:=eof(c);
  while not ok do
   begin
    read(c,buf);write(b,buf);
    k:=k+1;ok:=eof(c) or (k=l);
   end;
 until eof(c);
close(a);close(b);close(c);
end;
```

Теперь рассмотрим процедуру слияния серий фиксированной длины M из файлов A и B в файл C. Вспомним алгоритм слияния. Читаем начальные элементы очередной серии из файлов A и B, сравниваем их и записываем не больший элемент в C, а на его место читаем следующий элемент из соответствующего файла. И так до тех пор, пока не закончится серия в каком-нибудь из файлов.

Попробуем записать алгоритм следующим образом:

```
read(a,bufa);read(bufb);k1:=0;k2:=0;
while (k1<l) and (k2<l) do {повторять, пока не закончится серия}

if bufa.key<bufb.key
 then
  begin;write(c,bufa);read(a,bufa);k1:=k1+1;end
 else
  begin;write(c,bufb);read(b,bufb);k2:=k2+1;end
```

Если внимательно проанализировать записанный выше алгоритм, можно увидеть следующие ошибки.

Во первых, когда записывается в файл C последний элемент серии (K1 или K2 равны M), не надо читать следующий элемент файла, т.к. он относится уже к другой серии.

Во вторых, мы не учитываем то обстоятельство, что последняя серия в файле может содержать элементов меньше, чем M. Поэтому, необходимо обрабатывать не только конец серии, но и конец файла.

В третьих, когда мы записываем в файл C последний элемент очередной серии, например, из файла A, в буфер файла B (bufb) остаётся элемент, который необходимо переписать в файл C.

С учётом вышесказанного, алгоритм запишем следующим образом:

```
ok:=eof(a) or eof(b);
if not ok
 then
  begin;read(a,bufa);read(b,bufb);end;
k1:=0;k2:=0;
while not ok do {повторять, пока не закончится}
begin
 if bufa.key<bufb.key
  then
```

```
    begin
     write(c,bufa);
     k1:=k1+1;
     ok:=(k1=l)or eof(a);
     if not ok then read(a,bufa)
               else begin
                     write(c,bufb);
                     k2:=k2+1;
                    end;
    end
   else
    begin
     write(c,bufb);
     k2:=k2+1;
     ok:=(k2=l)or eof(b);
     if not ok then read(b,bufb)
               else begin
                     write(c,bufa);
                     k1:=k1+1;
                    end;
    end;
  end;
```

После того, как полностью переписана серия какого-либо файла, необходимо переписать в файл С остаток серии из другого файла.

```
переписать остаток серии файла A}
while (k1<l) and not eof(a) do
begin
 read(a,bufa);write(c,bufa);k1:=k1+1;
end;
{переписать остаток серии файла B}
while (k2<l) and not eof(b) do
begin
 read(b,bufb);write(c,bufb);k2:=k2+1;
end;
```

Окончательно процедура слияния серий длиною M из файла A и B в файл C будем иметь следующий вед.

{процедура слияния серий длиной M}

```
procedure Merge;
var bufa,bufb:item;
    k1,k2:integer;
    ok:boolean;
begin
 reset(a);reset(b);rewrite(c);
 repeat   {повторять, пока не закончатся оба файла A и B}
  z:=z+1; {подсчитываем число полученных серий}
  ok:=eof(a) or eof(b);
```

```
  if not ok then begin;read(a,bufa);read(b,bufb);end;
  k1:=0;k2:=0;
  while not ok do {повторять, пока не закончится серия или файл}
  begin
   if bufa.key<bufb.key
    then
     begin
      write(c,bufa);
      k1:=k1+1;
      ok:=(k1=l)or eof(a);
      if not ok then read(a,bufa)
              else begin
                    write(c,bufb);
                    k2:=k2+1;
                   end;
     end
    else
     begin
      write(c,bufb);
      k2:=k2+1;
      ok:=(k2=l)or eof(b);
      if not ok then read(b,bufb)
              else begin
                    write(c,bufa);
                    k1:=k1+1;
                   end;
     end;
  end;
  {переписать остаток серии файла A}
  while (k1<l) and not eof(a) do
  begin
   read(a,bufa);write(c,bufa);k1:=k1+1;
  end;
  {переписать остаток серии файла B}
  while (k2<l) and not eof(b) do
  begin
   read(b,bufb);write(c,bufb);k2:=k2+1;
  end;
 until eof(a) and eof(b);
 close(a);close(b);close(c);
end;
```

Здесь Z-глобальная переменная, которая необходима для определения конца процесса сортировки. Сортировка заканчивается, когда на очередном проходе мы получим одну серию (Z=1).

Теперь мы можем написать программу внешней сортировки методом простого слияния.

Программа использует уже готовый файл содержащий не отсортированные данные и не распечатывает отсортированный файл (программа только сортирует файл file1).

```
programma primemerge;
type item=record
```

```
          key:integer;
          {описание других полей}
          end;
     filetype=file of item;
var a,b,c:filetype;
    l,z:integer;
procedure Distribute;
var buf:item;
    k:integer;
    ok:boolean;
begin
 reset(c);rewrite(a);rewrite(b);
 repeat
  k:=0;ok:=eof(c);
  while not ok do
   begin
    read(c,buf);write(a,buf);
    k:=k+1;ok:=eof(c) or (k=l);
   end;
  k:=0;ok:=eof(c);
  while not ok do
   begin
    read(c,buf);write(b,buf);
    k:=k+1;ok:=eof(c) or (k=l);
   end;
 until eof(c);
close(a);close(b);close(c);
end; {distribute}
procedure Merge;
var bufa,bufb:item;
    k1,k2:integer;
    ok:boolean;
begin
 reset(a);reset(b);rewrite(c);
 repeat   {повторять, пока не закончатся оба файла A и B}
  z:=z+1; {подсчитываем число полученных серий}
  ok:=eof(a) or eof(b);
  if not ok then begin;read(a,bufa);read(b,bufb);end;
  k1:=0;k2:=0;
  while not ok do {повторять, пока не закончится серия или файл}
  begin
   if bufa.key<bufb.key
    then
     begin
      write(c,bufa);
      k1:=k1+1;
      ok:=(k1=l)or eof(a);
      if not ok then read(a,bufa)
               else begin
                     write(c,bufb);
                     k2:=k2+1;
                    end;
```

```
     end
    else
     begin
      write(c,bufb);
      k2:=k2+1;
      ok:=(k2=l)or eof(b);
      if not ok then read(b,bufb)
               else begin
                     write(c,bufa);
                     k1:=k1+1;
                    end;
     end;
  end;
  {переписать остаток серии файла A}
  while (k1<l) and not eof(a) do
  begin
   read(a,bufa);write(c,bufa);k1:=k1+1;
  end;
  {переписать остаток серии файла B}
  while (k2<l) and not eof(b) do
  begin
   read(b,bufb);write(c,bufb);k2:=k2+1;
  end;
 until eof(a) and eof(b);
 close(a);close(b);close(c);
end; {merge}
begin {main}
 assign(a,'{имя внешнего файла}');
 assign(b,'{имя внешнего файла}');
 assign(c,'{имя внешнего файла}');
 l:=1;
 repeat {повторять, пока не получим одну серию}
  distribute;z:=0;
  merge;l:=(2*l);
 until z=1;
end.
```

Результат работы:

```
 0  3  86  20  27  67  32  16  37  43  8  47  7  84  6  29  92  37  77  33  70
84  72  31  16  33  47  25  83  28  48  15  87  29  77  98  49  89  83  2  14  1
4  50  2  59  1  77  65  77  71  56  21  68  59  96  64  100  24  68  30  9  77
 50  88  51  57  95  68  34  1  71  99  77  75  20  14  91  78  59  86  69  29
9  63  28  88  16  27  54  96  17  16  27  18  58  50  29  16  61  74
Нажмите Enter для продолжения !

 0  1  1  2  2  3  6  7  8  9  9  14  14  14  15  16  16  16  16  16  17  18  20
  20  21  24  25  27  27  27  28  28  29  29  29  29  30  31  32  33  33  34  37
  37  43  47  47  48  49  50  50  50  51  54  56  57  58  59  59  59  61  63  64
  65  67  68  68  68  69  70  71  71  72  74  75  77  77  77  77  77  77  78  83
  83  84  84  86  86  87  88  88  89  91  92  95  96  96  98  99  100
Нажмите Enter для продолжения !
```

В качестве примера в таблице показан файл C в исходном состояние и после каждого прохода. В таблице показаны только значения ключей, причём серии разделены апострофом.

Заметим, что требуется четыре прохода.

Пример сортировки простым слиянием.

Исходный файл

C: 44 55 12 42 94 18 06 67 15 17 14 15 19 07

Первый проход (M=1)
A: 44 12 94 06 15 14 19
B: 55 42 18 67 17 15 07
C: 44 55 12 42 18 94 06 67 15 17 14 15 07 19

Второй проход (M=2)
A: 44 55 18 94 15 17 07 19
B: 12 42 06 67 14 15
C: 12 42 44 55 06 18 67 94 14 15 15 17 07 19

Третий проход (M=4)
A: 12 42 44 55 14 15 15 17
B: 06 18 67 94 07 19
C: 06 12 18 42 44 55 67 94 07 14 15 15 17 19

Четвёртый проход (M=8)
A: 06 12 18 42 44 55 67 94
B: 07 14 15 15 17 19
C: 06 07 12 14 15 15 17 18 19 42 44 55 67 94

Более совершенным методом сбалансированного слияния является сортировка естественным слиянием.

9.3.4. Сортировка естественным слиянием.

В случае простого слияния мы ничего не выигрываем, если данные уже частично отсортированы. На К-ом проходе длина всех сливаемых серий меньше или равна 2^К без учёта того обстоятельства, что могут быть упорядочены и более длинные серии и их можно было бы сливать. Можно было бы сразу сливать какие-либо серии длиною M и H в одну серию длиною M+H. Метод сортировки, при котором каждый раз сливаются две самые длинные упорядоченные последовательности, называется естественным слиянием.

Следующим нашим упражнением будет разработка алгоритма естественного слияния методом структурного программирования «сверху-вниз».

Запишем программу следующим образом:

```
program naturalmerge;
type item=record
        key:integer;
```

```
            {описание других полей}
            end;
        filetype=file of item;
    var a,b,c:filetype;
        z:integer;  {для подсчёта числа серий}
        eor:boolean;{индикатор конца серии}
    begin
     assign(a,'{имя внешнего файла}');
     assign(b,'{имя внешнего файла}');
     assign(c,'{имя внешнего файла}');
     repeat
      distribute;z:=0;
      merge;
     until z=1;
    end.
```

Здесь две фазы сортировки (разделения и слияния) реализуются отдельными процедурами: Distribute и Merge. Запишем эти процедуры.

{процедура распределения серий}

```
    procedure distribute; {из C в A и B}
    begin
     reset(c);rewrite(a);rewrite(b);
     repeat
      copyrun(c,a);
      if not eof(c) then copyrun(c,b);
     until eof(c);
     close(a);close(b);close(c);
    end;
```

{процедура слияния серий}

```
    procedure merge;
    begin
     reset(a);reset(b);rewrite(c);
     while (not eof(a))and(not eof(b)) do
      begin
       mergerun;z:=z+1;
      end;
     while not eof(a) do begin;copyrun(a,c);z:=z+1;end;
     while not eof(b) do begin;copyrun(b,c);z:=z+1;end;
     close(a);close(b);close(c);
    end;
```

Здесь Copyrun(x,y)-процедура копирования серий из файла X в файл Y, а Mergerun-процедура слияния двух серий из файлов A и B в файл C. Опишем эти процедуры. Будем использовать булевскую переменную eor, значение которой показывает, достигнут ли конец серии. Введём также процедуру Copy(x,y), которая копирует очередную запись их файла X в файл Y и определяет, достигнут ли конец серии.

{процедура копирования серий}

```
    procedure copyrun(var x,y:filetype);
     {переписать серии из X в Y}
    begin
     repeat
      copy(x,y);
     until eor;
```

```
end;
```

При реализации процедуры Copy надо находить конец серии. Для этого нужно сравнить ключ последней переписанной записи с ключом следующей. То есть мы должны видеть следующую запись. Это «заглядывание вперёд» достигается использованием буферной переменной файла X^. Однако, не все реализации языка Паскаль поддерживает буферную переменную. В частности буферные переменные отсутствуют в Турбо Паскаль. В этом случае наиболее просто задача решается, если использовать прямой доступ к файлу, который реализуется в Турбо Паскале. Так это и сделано в процедуре Copy.

{процедура копирования записи и определения конца серии}

```
procedure copy(var x,y:filetype);
var buf,buf1:item;
begin
 read(x,buf):write(y,buf);
 if eof(x) then eor:=true
        else begin
             {заглядываем вперёд}
             read(x,buf1);
             {возвращаемся на исходную запись}
             seek(x,filepos(x)-1);
             eor:=buf1.key<buf.key
            end;
end;
```

Здесь seek-процедура, которая устанавливает указатель файла на требуемую компаненту, filepos-функция, возвращающая номер текущей компоненты файла.

Теперь запишем процедуру Mergerun. Здесь мы также будем использовать процедуру seek для того, чтобы указатель файла всегда находился на записях, которые сравниваются. Процесс сравнения и выбора по ключу при слиянии серий завершается, как только будет исчерпана одна из двух серий. После этого остаток другой серии нужно скопировать в выходной файл C. это осуществляется вызовом процедуры Copyrun.

{процедура слияния двух серий}

```
procedure mergerun;
{слияние серий из A и B в C}
var bufa,bufb:item;
begin
 repeat
  read(a,bufa);seek(a,filepos(a)-1);
  read(b,bufb);seek(b,filepos(b)-1);
  if bufa.key<bufb.key
   then begin;copy(a,c);if eor then copyrun(b,c);end
   else begin;copy(b,c);if eor then copyrun(a,c);end;
 until eor
end;
```

В качестве примера в таблице показан файл C в исходном состоянии и после каждого прохода. Также как и в предыдущем примере серии в таблице разделены. Заметим, что здесь для сортировки требуется только 3 прохода.

Пример сортировки естественным слиянием

Исходный файл

C: 44 55 12 42 94 18 06 67 15 17 14 15 19 07

Первый проход

A: 44 55 18 15 17 07

B: 12 42 94 06 67 14 15 19

C: 12 42 55 94 06 18 67 14 15 15 17 19 07

Второй проход

A: 12 42 44 55 94 14 15 15 17 19

B: 06 18 67 07

C: 06 12 18 42 44 55 67 94 07 14 15 15 17 19

Третий проход

A: 06 12 18 42 44 55 67 94

B: 07 14 15 15 17 19

C: 06 07 12 14 15 15 17 18 19 42 44 55 67 94

В заключении заметим, что хотя и предполагается, что процедура Distribute посылает серии поровну в оба файла, действительное количество выходных серий в «a» и «b» могут различаться больше чем на 1. Например, рассмотрим также исходные данные

C: 08 06 09 11 09 13 15 05 07 20 19 27 31 13 25 (6 серий)

После разделения получим

A: 08 09 13 15 19 27 31 (1 серия)

B: 06 09 11 05 07 20 13 25 (3 серии)

Этот пример показывает, что простое распределение серий в несколько файлов может дать в результате меньшее число выходных серий, чем входных. Это происходит потому, что первый элемент (i+1)-й серии может быть больше, чем последний элемент i-й серии, что приведёт к автоматическому слиянию двух серий в одну, что и наблюдается в примере при распределении серий в файл A. Это следует учитывать при последующем слиянии серий, а именно, после достижения конца одного из файлов копировать весь остаток другого файла, а не только одну серию. Именно та и написана процедура Merge.

Конечно, отказ от требования, чтобы серии распределились поровну на два файла может привести к неоптимальной работе программы. Однако в худшем варианте она сохраняет те же характеристики, кроме того случай существенно неравномерного распределения статистически крайне маловероятен.

Программа использует уже готовый файл содержащий не отсортированные данные и не распечатывает отсортированный файл (программа только сортирует файл file1).

```
program naturalmerge;
```

```
type item=record
          key:integer;
          {описание других полей}
          end;
     filetype=file of item;
var a,b,c:filetype;
    z:integer;   {для подсчёта числа серий}
    eor:boolean; {индикатор конца серии}
procedure copy(var x,y:filetype);
var buf,buf1:item;
begin
 read(x,buf):write(y,buf);
 if eof(x) then eor:=true
           else begin
                {заглядываем вперёд}
                read(x,buf1);
                {возвращаемся на исходную запись}
                seek(x,filepos(x)-1);
                eor:=buf1.key<buf.key
               end;
end;
procedure copyrun(var x,y:filetype);
 {переписать серии из X в Y}
begin
 repeat
  copy(x,y);
 until eor;
end;
procedure mergerun;
{слияние серий из A и B в C}
var bufa,bufb:item;
begin
 repeat
  read(a,bufa);seek(a,filepos(a)-1);
  read(b,bufb);seek(b,filepos(b)-1);
  if bufa.key<bufb.key
   then begin;copy(a,c);if eor then copyrun(b,c);end
   else begin;copy(b,c);if eor then copyrun(a,c);end;
 until eor
end;
procedure distribute; {из C в A и B}
begin
 reset(c);rewrite(a);rewrite(b);
 repeat
  copyrun(c,a);
  if not eof(c) then copyrun(c,b);
 until eof(c);
 close(a);close(b);close(c);
end;
procedure merge;
begin
 reset(a);reset(b);rewrite(c);
```

```
 while (not eof(a))and(not eof(b)) do
  begin
   mergerun;z:=z+1;
  end;
 while not eof(a) do begin;copyrun(a,c);z:=z+1;end;
 while not eof(b) do begin;copyrun(b,c);z:=z+1;end;
 close(a);close(b);close(c);
end;
begin {main}
 assign(a,'{имя внешнего файла}');
 assign(b,'{имя внешнего файла}');
 assign(c,'{имя внешнего файла}');
 repeat
  distribute;z:=0;
  merge;
 until z=1;
end.
```

Результат работы:

```
 0  3  86  20  27  67  32  16  37  43  8  47  7  84  6  29  92  37  77  33  70
84  72  31  16  33  47  25  83  28  48  15  87  29  77  98  49  89  83  2  14  1
4  50  2  59  1  77  65  77  71  56  21  68  59  96  64  100  24  68  30  9  77
 50  88  51  57  95  68  34  1  71  99  77  75  20  14  91  78  59  86  69  29
9  63  28  88  16  27  54  96  17  16  27  18  58  50  29  16  61  74
Нажмите Enter для продолжения !

 0  1  1  2  2  3  6  7  8  9  9  14  14  14  15  16  16  16  16  16  17  18  20
  20  21  24  25  27  27  27  28  28  29  29  29  29  30  31  32  33  33  34  37
  37  43  47  47  48  49  50  50  50  51  54  56  57  58  59  59  59  61  63  64
  65  67  68  68  68  69  70  71  71  72  74  75  77  77  77  77  77  77  78  83
  83  84  84  86  86  87  88  88  89  91  92  95  96  96  98  99  100
Нажмите Enter для продолжения !
```

В том случае, когда не используются средства прямого доступа (процедура типа Seek) алгоритм усложняется.

При слиянии серии основная сложность связана с правильной обработкой конца серии. Признаком конца серии может быть либо прочитанная следующая запись с ключом меньше, чем ключ предыдущей записи, либо конец соответствующего файла. При этом если в первом случае для правильного слияния серий необходимо сравнивать ключи записей в bufa и в bufb, то во втором-достаточно переписать в выходной файл C остаток серии другого файла.

Здесь также необходимо учитывать, что когда достигнут конец файла, в буфере остаётся его последняя запись, которую надо правильно переписать в выходной файл C.

При распределении серий в файла A и B необходимо правильно определять конец серии. Здесь для определения конца серии сравниваем ключ записи, находящийся в буфере (переменная buf) с ключом предыдущей записи (хранится в переменной X). Если buf.key<x, то в буфере находится запись уже другой серии, которую надо переписать в другой файл.

Ниже приведены тексты процедур разделения и слияния серий, а также основной части программы с подробными комментариями, которые помогут Вам лучше понять предложенный здесь алгоритм.

{сформировать признак конца сортировки}

```
procedure ended;
begin
 reset(a);reset(b);
 if eof(a) or eof(b) then {получили одну серию} z:=1;
 close(a);close(b);
end;
```

{разделение файла C на файлы A и B}

```
procedure distribute;
var buf:item;
    x:integer;
    pt:boolean;{переключатель выходных файлов}
    {если pt=true, то запись в файл A, иначе в файл B}
    ok:boolean;{признак нахождения в буфере последней записи серии}
begin
 reset(c);rewrite(a);rewrite(b);
 pt:=true;
 ok:=false;
 if not eof(c) then {переписываем в A первую запись из C, её ключ запоминаем в X}
                begin
                 read(c,buf);write(a,buf);x:=buf.key;
                end;
 if not eof(c) then
                begin
                 read(c,buf);{читаем в буфер вторую запись из C}
                 repeat      {повторять пока не закончится файл C}
                  while not eof(c) and ((buf.key>=x) or ok) do
{пока не конец C и либо не конец серии, либо в буфере находится запись другой серии}
                   begin
                    ok:=false;
                    if pt then write(a,buf) else write(b,buf);
                    x:=buf.key;read(c,buf);
                   end;
                  if buf.key<x then
{в буфере находится запись уже другой серии и её надо переписать в другой файл}
                              begin;pt:=not pt;ok:=true;end;
                 if eof(c) then
{записать в выходной файл последнюю запись файла C, уже прочитанную в буфер}
                            if pt then write(a,buf) else write(b,buf);
                until eof(c);
                end;
 close(a);close(b);close(c);
end;
```

{процедура слияния серий}

```
procedure merge;
var bufa,bufb:item;
    xa,xb:integer;
    ok,pr1,pr2:boolean;
    k1,k2:boolean;
{pr1-признак того, что было прочитано не меньше двух записей файла A}
{pr2-признак того, что было прочитано не меньше двух записей файла B}
{k1-признак того, что в буфере есть запись файла A}
{k2-признак того, что в буфере есть запись файла B}
begin
 reset(a);reset(b);rewrite(c);
 pr1:=false;pr2:=false;k1:=false;k2:=false;
 if not eof(a) then begin;read(a,bufa);k1:=true;end;
 if not eof(b) then begin;read(b,bufb);k2:=true;end;
 repeat {повторять пока не закончится либо файл A, либо файл B}
  ok:=eof(a) or eof(b);
  while not ok do
{повторять пока не закончится либо один из файлов, либо серия}
   begin
    if bufa.key<bufb.key
     then begin
           write(c,bufa);xa:=bufa.key;read(a,bufa);pr1:=true;
           ok:=eof(a) or (xa>bufa.key);
          end
     else begin
           write(c,bufb);xb:=bufb.key;read(b,bufb);pr2:=true;
           ok:=eof(b) or (xb>bufb.key);
          end;
   end;
  if (xa>bufa.key)and pr1 {конец текущей серии файла A}
   then while (xb<bufb.key)and not eof(b) do
{переписать в файл C остаток текущей серии файла B}
         begin
          write(c,bufb);xb:=bufb.key;read(b,bufb)
         end;
  if (xb>bufb.key)and pr2 {конец текущей серии файла B}
   then while (xa<bufa.key)and not eof(a) do
{переписать в файл C остаток текущей серии файла A}
         begin
          write(c,bufa);xa:=bufa.key;read(a,bufa)
         end;
 until eof(a) or eof(b);
 if not eof(a) and k2
{в bufb есть запись и файл A не закончился}
{переписать в файл C записи из файла A с ключами меньше, чем ключ записи в bufb}
  then while(bufa.key<bufb.key)and not eof(a) do
       begin;write(c,bufa);read(a,bufa);end;
  else if not eof(b) and k1
{в bufa есть запись и файл B не закончился}
{переписать в файл C записи из файла B с ключами меньше, чем ключ записи в bufa}
       then while(bufb.key<bufa.key) and not eof(b) do
```

```
          begin;write(c,bufb);read(b,bufb);end;
 if k1 and k2{переписать в C записи из буферов bufa и bufb}
  then if bufa.key<bufb.key
       then begin;write(c,bufa);write(c,bufb);end;
       else begin;write(c,bufb);write(c,bufa);end;
  else if k1 then write(c,bufa)
           else if k2 then write(c,bufb);
{переписать в C остаток файла A}
 while not eof(a) do begin;read(a,bufa);write(c,bufa);end;
{переписать в C остаток файла B}
 while not eof(b) do begin;read(b,bufb);write(c,bufb);end;
 close(a);close(b);close(c);
end;
```

Внешняя сортировка файлов естественным слиянием (используется только последовательный доступ к файлам).

Программа использует уже готовый файл содержащий не отсортированные данные и не распечатывает отсортированный файл (программа только сортирует файл file1).

```
program natmeg;
type item=record
         key:integer;
         {описание других полей}
         end;
     tfile=file of item;
var a,b,c:tfile;
procedure ended;
begin
 reset(a);reset(b);
 if eof(a) or eof(b) then {получили одну серию} z:=1;
 close(a);close(b);
end;
procedure distribute;
var buf:item;
    x:integer;
    pt:boolean;{переключатель выходных файлов}
    {если pt=true, то запись в файл A, иначе в файл B}
    ok:boolean;{признак нахождения в буфере последней записи серии}
begin
 reset(c);rewrite(a);rewrite(b);
 pt:=true;
 ok:=false;
 if not eof(c) then {переписываем в A первую запись из C, её ключ запоминаем в X}
               begin
                read(c,buf);write(a,buf);x:=buf.key;
               end;
 if not eof(c) then
               begin
                read(c,buf);{читаем в буфер вторую запись из C}
                repeat      {повторять пока не закончится файл C}
                 while not eof(c) and ((buf.key>=x) or ok) do
{пока не конец C и либо не конец серии, либо в буфере находится запись другой серии}
```

```
                 begin
                  ok:=false;
                  if pt then write(a,buf) else write(b,buf);
                  x:=buf.key;read(c,buf);
                 end;
                if buf.key<x then
{в буфере находится запись уже другой серии и её надо переписать в другой файл}
                        begin;pt:=not pt;ok:=true;end;
                if eof(c) then
{записать в выходной файл последнюю запись файла C, уже прочитанную в буфер}
                      if pt then write(a,buf) else write(b,buf);
               until eof(c);
              end;
 close(a);close(b);close(c);
end;
procedure merge;
var bufa,bufb:item;
    xa,xb:integer;
    ok,pr1,pr2:boolean;
    k1,k2:boolean;
{pr1-признак того, что было прочитано не меньше двух записей файла A}
{pr2-признак того, что было прочитано не меньше двух записей файла B}
{k1-признак того, что в буфере есть запись файла A}
{k2-признак того, что в буфере есть запись файла B}
begin
 reset(a);reset(b);rewrite(c);
 pr1:=false;pr2:=false;k1:=false;k2:=false;
 if not eof(a) then begin;read(a,bufa);k1:=true;end;
 if not eof(b) then begin;read(b,bufb);k2:=true;end;
 repeat {повторять пока не закончится либо файл A, либо B}
  ok:=eof(a) or eof(b);
  while not ok do
{повторять пока не закончится либо один из из файлов, либо серия}
   begin
    if bufa.key<bufb.key
     then begin
           write(c,bufa);xa:=bufa.key;read(a,bufa);pr1:=true;
           ok:=eof(a) or (xa>bufa.key);
          end
     else begin
           write(c,bufb);xb:=bufb.key;read(b,bufb);pr2:=true;
           ok:=eof(b) or (xb>bufb.key);
          end;
   end;
  if (xa>bufa.key)and pr1 {конец текущей серии файла A}
   then while (xb<bufb.key)and not eof(b) do
{переписать в файл C остаток текущей серии файла B}
         begin
          write(c,bufb);xb:=bufb.key;read(b,bufb)
         end;
  if (xb>bufb.key)and pr2 {конец текущей серии файла B}
   then while (xa<bufa.key)and not eof(a) do
```

```
{переписать в файл C остаток текущей серии файла A}
        begin
         write(c,bufa);xa:=bufa.key;read(a,bufa)
        end;
 until eof(a) or eof(b);
 if not eof(a) and k2
{в bufb есть запись и файл A не закончился}
{переписать в файл C записи из файла A с ключами меньшими, чем ключ записи в bufb}
  then while(bufa.key<bufb.key)and not eof(a) do
       begin;write(c,bufa);read(a,bufa);end;
  else if not eof(b) and k1
{в bufa есть запись и файл B не закончился}
{переписать в файл C записи из файла B с ключами меньшими, чем ключ записи в bufa}
then while(bufb.key<bufa.key) and not eof(b) do
          begin;write(c,bufb);read(b,bufb);end;
 if k1 and k2{переписать в C записи из буферов bufa и bufb}
  then if bufa.key<bufb.key
       then begin;write(c,bufa);write(c,bufb);end;
       else begin;write(c,bufb);write(c,bufa);end;
  else if k1 then write(c,bufa)
           else if k2 then write(c,bufb);
{переписать в C остаток файла A}
 while not eof(a) do begin;read(a,bufa);write(c,bufa);end;
{переписать в C остаток файла B}
 while not eof(b) do begin;read(b,bufb);write(c,bufb);end;
 close(a);close(b);close(c);
end;
begin {main}
 assign(a,'{имя внешнего файла}');
 assign(b,'{имя внешнего файла}');
 assign(c,'{имя внешнего файла}');
 repeat
  z:=0;
  distribute;
  merge;
  ended;
 until z=1;
end.
```

Результат работы:

```
 0  3  86  20  27  67  32  16  37  43  8  47  7  84  6  29  92  37  77  33  70
84  72  31  16  33  47  25  83  28  48  15  87  29  77  98  49  89  83  2  14  1
4  50  2  59  1  77  65  77  71  56  21  68  59  96  64  100  24  68  30  9  77
 50  88  51  57  95  68  34  1  71  99  77  75  20  14  91  78  59  86  69  29
9  63  28  88  16  27  54  96  17  16  27  18  58  50  29  16  61  74
Нажмите Enter для продолжения !

 0  1  1  2  2  3  6  7  8  9  9  14  14  14  15  16  16  16  16  16  17  18  20
  20  21  24  25  27  27  27  28  28  29  29  29  29  30  31  32  33  33  34  37
  37  43  47  47  48  49  50  50  50  51  54  56  57  58  59  59  59  61  63  64
  65  67  68  68  68  69  70  71  71  72  74  75  77  77  77  77  77  77  78  83
  83  84  84  86  86  87  88  88  89  91  92  95  96  96  98  99  100
```

Нажмите Enter для продолжения !

9.3.5. Сортировка многофазным слиянием.

Рассмотренные выше методы сортировки простым и естественным слиянием достаточно просты, однако не эффективны, так как сортировка выполняется за значительное число проходов. Например, при сортировке M записей простым слиянием

$$K = \lfloor Log_2 M \rfloor$$

требуется:

K -проходов.

Естественный путь повышения эффективности сортировки заключается в предварительной внутренней сортировке записей, формирование из них серий возможно большей длинны и распределение этих серий по нескольким файлам. При P-путевом

$$K = \lfloor Log_P R \rfloor$$

сбалансированным слияний число проходов

Где R-начальное число серий. Поскольку на каждом проходе обрабатывается N записей, то эффективность всего процесса сортировки будет пропорциональна

А с учётом внутренней сортировки

$$N * Log_P R$$

$$N * Log_p R + 1$$

Из этой формулы виден второй путь повышения эффективности сортировки, а именно увеличивать число файлов P. Однако при сбалансированном многофазном слиянии одновременно используются лишь чуть больше половины имеющихся файлов, а остальные простаивают. Возникает вопрос, можно ли ещё лучше использовать имеющиеся файлы? Да, это возможно, если распределить начальные серии не поровну между всеми файлами, а более хитрым способом, так чтобы на каждой фазе слияния, кроме самой последней, только один файл оказывался пустым. Такой метод называется многофазным слиянием.

Вернёмся к предыдущему примеру сортировки 1700 записей, предварительно отсортированных в серии по 100 записей. Однако на первом этапе распределим эти серии на 3 файла следующим образом: в файл 1-7 серий, в файл 2-6 серий, в файл 3-4 серии.

Далее будем сливать эти серии в файл 4. В результате в файле 4 мы получим 4 серии длиною 300 записей, в файле 1 останется 3 серии, в файле 2 будет 2 серии, а файл 3 станет пустой. Далее сливаем серии в файл 3 и так далее, пока не получим отсортированный файл, состоящий из одной серии длиною 1700 записей. Процесс сортировки показан на рисунке.

1700(1)

F4

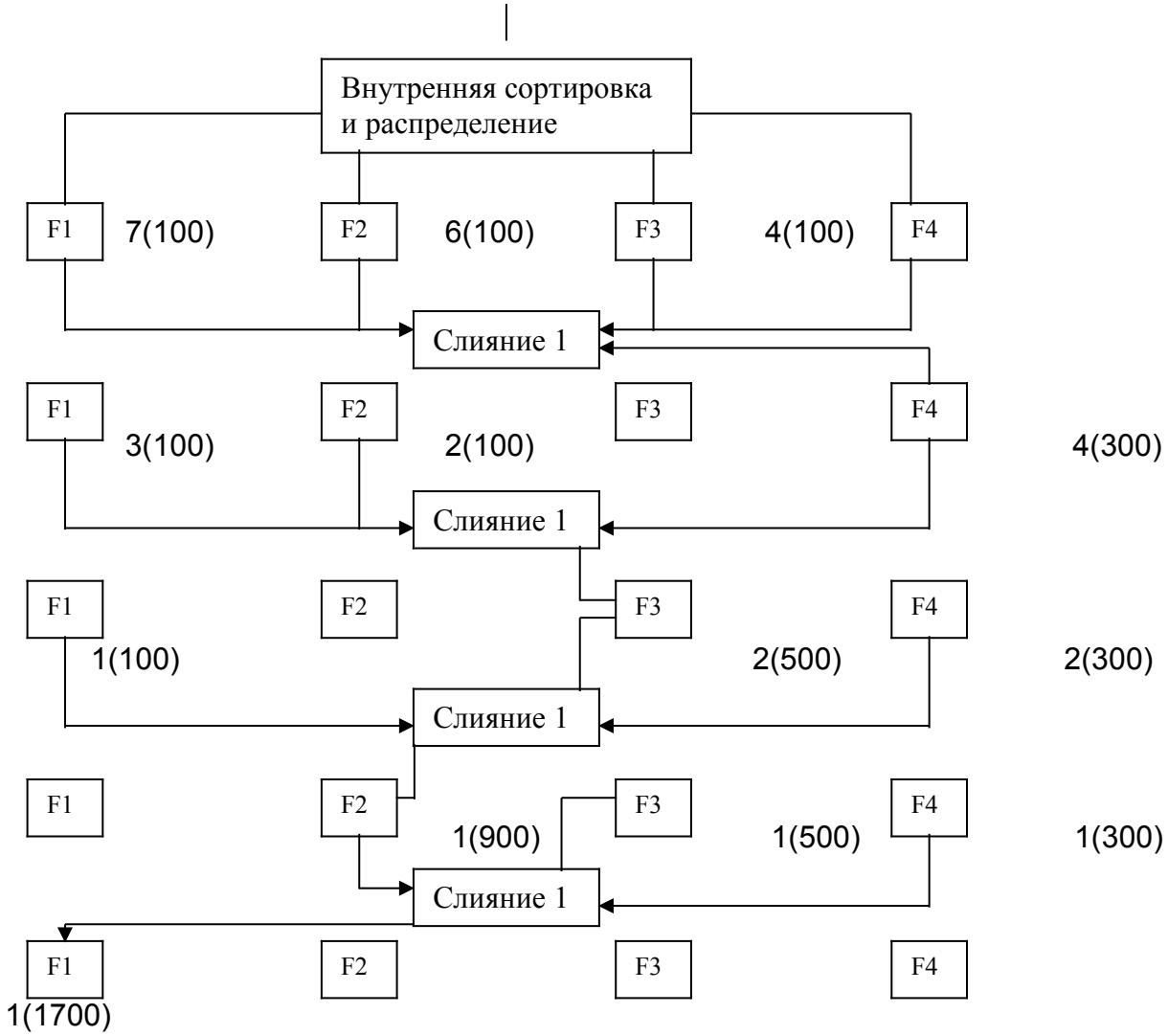


Рисунок. Сортировка 1700 записей с использованием 4 файлов методом многофазного слияния.

Например, для 4 файлов имели следующую последовательность Фибоначчи 2-го порядка

0 , 0 , 1 , 1 , 2 , 4 , 7 , 13 , 24 , 44 , 81.

Возможные начальные распределения серий показаны в таблице.

Таблица.
Начальное распределение серий для 4 лент.

F1	4	7	13	24	44
F2	6	11	20	37	68
F3	7	13	24	44	81
Количество серий в исходном файле	17	31	57	105	193

Из сказанного выше следует, что алгоритм многофазного слияния применим только к таким входным данным, в которых число начальных серий имеет волне определённое значение. А что же делать, если число начальных серий отличается от требуемого? В этом случае поступают следующим образом: вводится соответствующее число фиктивных пустых серий, так, чтобы сумма реальных и фиктивных серий давала нужное значение. Однако возникает сразу следующий вопрос: как распределить фиктивные серии между файлами? Давайте посмотрим, как сливаются реальные и фиктивные серии. Понятно, что выбор фиктивной серии с i-ый файла означает, что i-ый файл не участвует в слиянии; в результате слияние происходит с менее чем N-1 файлами. Слияние фиктивных серий со всех N-1 входных файлов не предполагает никакой действительной операции слияния, а означает просто запись фиктивных серий на входной файл. Из этого можно сделать вывод, что фиктивные серии нужно распределить на N-1 файлов как можно более равномерно, так как мы заинтересованы в слиянии с возможно большего числа файлов. Можно также показать, что фиктивные серии лучше располагать в начале файлов.

Пример программы Сортировка многофазным слиянием:

Программа использует уже готовый файл содержащий не отсортированные данные и не распечатывает отсортированный файл (программа только сортирует файл file0).

```
program Многофазное слияние;
uses crt;
const n=10;
type item=record
        key:integer;
        {описания других полей}
        end;
    filetype=file of item;
var f0,f1,f2,f3,a,b,c,d:filetype;
ser0,ser1,ser2,ser3,kolser,disk0,disk1,disk2,disk3,z,l,all,zero0,zero1,zero2,zero3:intege
r;
   eora,eorb,eorc,per,eor:boolean;
   mas:array[1..n]of integer;
   data:item;
procedure fibonachi;
 procedure fib;
 begin
  disk0:=disk1+disk2+disk3;
  while disk0<kolser do
  begin
   disk3:=disk2;disk2:=disk1;disk1:=disk0;disk0:=disk1+disk2+disk3;
  end;
 end;
begin
```

```
 disk1:=1;disk2:=1;disk3:=1;fib;kolser:=disk0;
 disk1:=0;disk2:=0;disk3:=1;fib;disk1:=disk2;disk2:=kolser-(disk1+disk3);
end;
procedure readfile;
begin
 l:=0;
 while not eof(f0)and not(l=n) do
 begin l:=l+1;read(f0,data);mas[l]:=data.key;end;
 all:=l;
end;
procedure massort;
var buf,units:integer;
    m:boolean;
begin
 units:=all;
 repeat
  m:=true;;
  units:=units-1;
  for l:=1 to units do
  begin
   if mas[l]>mas[l+1]
    then begin buf:=mas[l];mas[l]:=mas[l+1];mas[l+1]:=buf;m:=false;end;
  end;
 until m or (units=1);
end;
procedure sort(var x:filetype);
begin
 readfile;massort;
 for l:=1 to all do begin data.key:=mas[l];write(x,data);end;
end;
procedure sortrun;
begin
 reset(f0);rewrite(f1);rewrite(f2);rewrite(f3);
 kolser:=1;zero0:=0;zero1:=0;zero2:=0;zero3:=0;ser0:=0;ser1:=0;ser2:=0;ser3:=0;
 repeat
  fibonachi;
  if disk1<>ser1 then begin
                      if not eof(f0) then sort(f1)
                                     else zero1:=zero1+1;
                      ser1:=ser1+1;
                     end;
  if disk2<>ser2 then begin
                      if not eof(f0) then sort(f2)
                                     else zero2:=zero2+1;
                      ser2:=ser2+1;
                     end;
  if disk3<>ser3 then begin
                      if not eof(f0) then sort(f3)
                                     else zero3:=zero3+1;
                      ser3:=ser3+1;
                     end;
  if (not eof(f0))and(kolser=(ser1+ser2+ser3)) then kolser:=kolser+1;
```

```
 until eof(f0)and(kolser=(ser1+ser2+ser3));
 close(f0);close(f1);close(f2);close(f3);
end;
procedure copy(var x,y:filetype);
var buf,buf1:item;
begin
 read(x,buf);write(y,buf);
 if eof(x) then eor:=true
          else begin
               {заглядываем вперед}
               read(x,buf1);
               {возвращаемся на исходную запись}
               seek(x,filepos(x)-1);
               eor:=buf1.key<buf.key
              end;
end;
procedure mergerun(var d,a,b,c:filetype;var
serd,sera,serb,serc,zerod,zeroa,zerob,zeroc:integer);
{слияние серий из A и B и C в D}
var bufa,bufb,bufc:item;
    zapis:boolean;
begin
 repeat
  eora:=true;eorb:=true;eorc:=true;
  serd:=serd+1;sera:=sera-1;serb:=serb-1;serc:=serc-1;
  if zeroa<>0 then begin eora:=false;zeroa:=zeroa-1;end;
  if zerob<>0 then begin eorb:=false;zerob:=zerob-1;end;
  if zeroc<>0 then begin eorc:=false;zeroc:=zeroc-1;end;
  if (not eora)and(not eorb)and(not eorc) then zerod:=zerod+1;
  while (eora or eorb or eorc) do
  begin
   bufa.key:=1999;bufb.key:=1999;bufc.key:=1999;
   if eora then begin read(a,bufa);seek(a,filepos(a)-1);end;
   if eorb then begin read(b,bufb);seek(b,filepos(b)-1);end;
   if eorc then begin read(c,bufc);seek(c,filepos(c)-1);end;
   if bufa.key<bufb.key then
                    if bufa.key<bufc.key then
                                    begin
                                     copy(a,d);
                                     if eor then eora:=false;
                                    end
                                   else
                                    begin
                                     copy(c,d);
                                     if eor then eorc:=false;
                                    end
                    else
                    if bufb.key<bufc.key then
                                    begin
                                     copy(b,d);
                                     if eor then eorb:=false;
                                    end
```

```
                                    else
                                     begin
                                      copy(c,d);
                                      if eor then eorc:=false;
                                     end;
  end;
 until (sera=0)or(serb=0)or(serc=0);
end;
procedure merge;
begin
 reset(f0);reset(f1);reset(f2);reset(f3);
 while (ser0+ser1+ser2+ser3)<>1 do
 begin
  if (ser0=0) then begin
rewrite(f0);mergerun(f0,f1,f2,f3,ser0,ser1,ser2,ser3,zero0,zero1,zero2,zero3);reset(f0);
end;
  if (ser1=0)and((ser0+ser1+ser2+ser3)<>1) then
   begin

rewrite(f1);mergerun(f1,f2,f3,f0,ser1,ser2,ser3,ser0,zero1,zero2,zero3,zero0);reset(f1);
   end;
  if (ser2=0)and((ser0+ser1+ser2+ser3)<>1) then
   begin

rewrite(f2);mergerun(f2,f3,f0,f1,ser2,ser3,ser0,ser1,zero2,zero3,zero0,zero1);reset(f2);
   end;
  if (ser3=0)and((ser0+ser1+ser2+ser3)<>1) then
   begin

rewrite(f3);mergerun(f3,f0,f1,f2,ser3,ser0,ser1,ser2,zero3,zero0,zero1,zero2);reset(f3);
   end;
 end;
 close(f0);close(f1);close(f2);close(f3);
end;
begin
 clrscr;
 assign(f0,'c:\tp\file0');
 assign(f1,'c:\tp\file1');
 assign(f2,'c:\tp\file2');
 assign(f3,'c:\tp\file3');
 sortrun;
 merge;
end.
```

Результат работы:

```
  0  3  86  20  27  67  32  16  37  43  8  47  7  84  6  29  92  37  77  33  70
 84  72  31  16  33  47  25  83  28  48  15  87  29  77  98  49  89  83  2  14  1
 4  50  2  59  1  77  65  77  71  56  21  68  59  96  64  100  24  68  30  9  77
  50  88  51  57  95  68  34  1  71  99  77  75  20  14  91  78  59  86  69  29
 9  63  28  88  16  27  54  96  17  16  27  18  58  50  29  16  61  74
 Нажмите Enter для продолжения !
```

```
 0  1  1  2  2  3  6  7  8  9  9  14  14  14  15  16  16  16  16  16  17  18  20
 20  21  24  25  27  27  27  28  28  29  29  29  29  30  31  32  33  33  34  37
 37  43  47  47  48  49  50  50  50  51  54  56  57  58  59  59  59  61  63  64
 65  67  68  68  68  69  70  71  71  72  74  75  77  77  77  77  77  77  78  83
 83  84  84  86  86  87  88  88  89  91  92  95  96  96  98  99  100
Нажмите Enter для продолжения !
```

Файл в котором будут отсортированные данные в зависимости от количества данных будут будет меняться(file0 или file1 или file2 или file3).

Итак, мы рассмотрели основные методы внешней сортировки.

Мы приходим к довольно сложным алгоритмам сортировки файлов, поскольку более простые методы внутренней сортировки требуют достаточно большой оперативной памяти, чтобы хранить все сортируемые данные. Внутренняя сортировка, как мы видим, используется на этапе распределения начальных серий так, чтобы в результате эти серии имели как можно большую длину. Очевидно, что на последних проходах никакой внутренней сортировки не дадут какого-либо улучшения, так как длинна участвующих в них серий постоянно растёт и, следовательно, всегда будет больше имеющейся оперативной памяти. Поэтому следует сосредоточить внимание на оптимизацию алгоритма, который формирует начальные серии. Наиболее подходящим является логарифмические методы сортировки массивов и среди них сортировка с помощью дерева или пирамидальная сортировка.

Программа, представляющая собой комбинацию многофазной и пирамидальной сортировки достаточно сложна, хотя, по существу, она выполняет туже самую легко определимую задачу переупорядочения множества элементов, что и любая из коротких программ простых методов сортировки массивов. Из всего сказанного можно следующий выводы:

1.Между алгоритмом и структурой данных существует тесная связь: структура данных оказывает большое влияние на вид программы.

2.При помощи усложнения программы можно значительно повысить её эффективность, даже когда структура данных, с которой она работает, плохо соответствует поставленной задаче.

Лабораторная работа №12.

Сортировка файлов.

Цель работы:

1. Ознакомление с возможностями сортировки файлов на внешних носителях в ЭВМ.
2. Получение навыков работы с внешними файлами.

Постановка задачи:

Подготовить данные об абитуриентах, поступающих в институт. Информацию о каждом абитуриенте оформить в виде записи, содержащей следующие поля:

1. ФИО.
2. Год рождения.
3. Год окончания школы.
4. Оценки в аттестате.
5. Признак - нуждается ли в общежитии.

6. Оценки вступительных экзаменов.

Разработать программу сортировки подготовленных данных во внешнем файле, методом заданном в варианте, и распечатать записи в файле.

Содержание отчёта:

1. Постановка задачи.
2. Анкетные данные абитуриентов.
3. Тексты программ.
4. Распечатка результатов выполнения программы.

Методические указания.

При подготовке исходных данных необходимо учесть, что выходная информация программы обработки внешнего файла должна составлять не менее одной четверти от входной.

Образец выполнения задания.

Лабораторная работа №12.

Сортировка файлов.

Постановка задачи:

Подготовить данные об абитуриентах, поступающих в институт. Информацию о каждом абитуриенте оформить в виде записи, содержащей следующие поля:

1. ФИО.
2. Год рождения.
3. Год окончания школы.
4. Оценки в аттестате.
5. Признак - нуждается ли в общежитии.
6. Оценки вступительных экзаменов.

Разработать программу сортировки подготовленных данных во внешнем файле по фамилии, методом сортировки естественным слиянием, и распечатать фамилии и даты рождения и даты окончания школы, до сортировки и после.

Методические указания.

При подготовке исходных данных необходимо учесть, что выходная информация программы обработки внешнего файла должна составлять не менее одной четверти от входной.

Анкетные данные на абитуриентов в конце методического пособия.

Текст программы:

```
program sortirovka;
uses crt;
type data=record
        fio:string[30];
        godr,godo:integer;
        ates:record
             mat,fiz,rus:integer;
             end;
        haus:boolean;
        ekz:record
           mat,fiz,rus:integer;
```

```
            end;
          end;
     filetype=file of data;
var files,a,b,c:filetype;
    z,n:integer;   {для подсчета числа серий}
    eor:boolean; {индикатор конца серии}
    keys:char;
    buf,stu:data;
procedure copys(var x,y:filetype);
var buf1:data;
begin
 read(x,buf);write(y,buf);
 if eof(x) then eor:=true
          else begin
               {заглядываем вперед}
               read(x,buf1);
               {возвращаемся на исходную запись}
               seek(x,filepos(x)-1);
               eor:=buf1.fio<buf.fio;
              end;
end;
procedure copyrun(var x,y:filetype);
 {переписать серии из x в y}
begin
 repeat
  copys(x,y);
 until eor;
end;
procedure mergerun;
{слияние серий из A и B в C}
var bufa,bufb:data;
begin
 repeat
  read(a,bufa);seek(a,filepos(a)-1);
  read(b,bufb);seek(b,filepos(b)-1);
  if bufa.fio<bufb.fio
                  then begin;copys(a,c);if eor then copyrun(b,c);end
                  else begin;copys(b,c);if eor then copyrun(a,c);end;
 until eor
end;
procedure distribute; {из C в A и B}
begin
 reset(c);rewrite(a);rewrite(b);
 repeat
  copyrun(c,a);
  if not eof(c) then copyrun(c,b);
 until eof(c);
 close(a);close(b);close(c);
end;
procedure merge;
begin
 reset(a);reset(b);rewrite(c);
```

```
 while (not eof(a))and(not eof(b)) do
  begin
   mergerun;z:=z+1;
  end;
 while not eof(a) do begin;copyrun(a,c);z:=z+1;end;
 while not eof(b) do begin;copyrun(b,c);z:=z+1;end;
 close(a);close(b);close(c);
end;
procedure sort;
begin
 repeat
  distribute;z:=0;
  merge;
 until z=1;
 erase(b);erase(a);
end;
procedure print;
begin
 reset(c);
 while not eof(c) do
 begin
  read(c,stu);
  writeln(stu.fio,'--',stu.godr,'--',stu.godo);
 end;
 writeln('Для продолжения нажмите Enter !');
 readln;
end;
begin {main}
 assign(c,'c:\data.dat'); {файл с данными}
 assign(b,'c:\temp1');
 assign(a,'c:\temp2');
 clrscr;
 writeln('Не отсортированные данные');
 print;
 writeln('Сортировка  данных по фамилии');
 writeln('Отсортированные данные');
 sort;
 print;
end.
```

Результат выполнения программы:

```
Неотсортированные данные:
Анисимов Пётр Иванович--1982--1999
Синилов Сергей Иванович—1983--2000
Шарапов Евгений Владимирович--1984--2001
Бажин Никита Андреевич--1983--2000
Созинов Алексей Петрович—1984--2001
Малышев Василий Владимирович--1982--1999
Тихонов Сергей Геннадьевич—1982--1999
Еговцев Иван Артурович--1983--2000
Михайлов Артём Егорович—1983--2000
Ползунова Елена Андреевна—1982--1999
```

```
Смирнов Никита Владимирович--1984--2001
Мельникова Лариса Анатольевна--1984--2001
Токарева Надежда Александровна--1983--2000
Маслова Нина Михайловна—1984--2001
Молчановский Ильнар Ирекович--1982--1999
Корягина Нина Плахова--1984—2001
Егорова Пелагея Луповна--1983--2000
Гаспер Валентина Александровна--1982--1999
Теплоухов Юрий Леонидович—1982--1999
Кирьянов Антон Алексеевич—1983--2000
Для продолжения нажмите Enter !


Сортировка  данных по фамилии
Отсортированные данные
Анисимов Пётр Иванович--1982--1999
Бажин Никита Андреевич--1983--2000
Гаспер Валентина Александровна--1982--1999
Еговцев Иван Артурович--1983--2000
Егорова Пелагея Луповна--1983--2000
Кирьянов Антон Алексеевич--1983--2000
Корягина Нина Плахова--1984--2001
Малышев Василий Владимирович--1982--1999
Маслова Нина Михайловна--1984--2001
Мельникова Лариса Анатольевна--1984--2001
Михайлов Артём Егорович--1983--2000
Молчановский Ильнар Ирекович--1982--1999
Ползунова Елена Андреевна--1982--1999
Синилов Сергей Иванович--1983--2000
Смирнов Никита Владимирович--1984--2001
Созинов Алексей Петрович--1984--2001
Теплоухов Юрий Леонидович--1982--1999
Тихонов Сергей Геннадьевич--1982--1999
Токарева Надежда Александровна--1983--2000
Шарапов Евгений Владимирович--1984--2001
Для продолжения нажмите Enter !
```

Варианты заданий.

1. Сортировать файл с данными по оценкам на вступительных экзаменах и средней оценке в атестате, методом многофазного слияния, распечатать файл с данными до сортировки и после: фамилия имя отчество, год рождения, оценки на вступительных экзаменах и атестате.

2. Сортировать файл с данными по году окончания школы, методом сбалансированного слияния, распечатать файл с данными до сортировки и после: фамилия имя отчество, год рождения, год окончания школы и надобность в общежитие.

3. Сортировать файл с данными по году рождения, методом простого слияния, распечатать файл с данными до сортировки и после: фамилия имя отчество, год рождения и оценки в атестате.

4. Сортировать файл с данными по оценкам на вступительных экзаменах, методом естественного слияния, распечатать файл с данными до сортировки и после: фамилия имя отчество, оценки на вступительных экзаменах и надобность в общежитие.

5. Сортировать файл с данными по фамилиям, методом многофазного слияния, распечатать файл с данными до сортировки и после: фамилия имя отчество, год рождения, год окончания школы и надобность в общежитие.

10. Динамическая память.

Все переменные, объявляемые в программе, размещаются в одной непрерывной области оперативной памяти - сегменте данных, длина которого определяется архитектурой микропроцессора.

Динамическая память (ДП) - оперативная память персонального компьютера(ПК), предоставляемая программе при ее работе за вычетом сегмента данных стека и тела программы. Ее размер определяется всей доступной памятью ПК.

ДП - единственная возможность обработки массивов данных большой размерности.

ОП ПК - это совокупность элементарных ячеек для хранения информации - байтов , каждый из которых имеет собственный номер. Эти номера называются адресами , которые позволяют обращаться к любому байту памяти.

10.1. Указатели.

Turbo Pascal предоставляет гибкое средство управления динамической памятью - так называемые указатели .

Указатель - это переменная, которая в качестве своего значения содержит адрес байта памяти.

В TP указатель связывается с некоторым типом данных, их называют типизированными. Для объявления типизированного указателя используется значок ^ , который помещается перед типом:

```
var p1:^integer;
      p2:^real;
```

Также можно объявлять указатель, не связывая его при этом с конкретным типом данных. Для этого существует стандартный тип POINTER:

```
var pp: pointer;
```

Указатели такого рода называют нетипизированными. Так как они не связаны с конкретным типом, то с их помощью удобно размещать динамические данные, структура и тип, которых меняются в ходе работы программы.

```
var
pp: pointer;
p1,p2: ^integer;
p3: ^real;
.....
p1:=p2;   - связь указателей одного типа данных.
p1:=p3;    - запрещено.
```

pp:=p3; p1:=pp; - с нетипизированными указателями таких ограничений не существует.

Вся ДП рассматривается как массив байтов, который называется кучей. Она располагается в старших адресах сразу же за областью памяти, которую занимает тело программы.

HEAPORG - начало кучи хранится в этой переменной.

HEAPEND - конец кучи.

HEAPPTR - текущая граница незанятой динамической памяти.

Память под любую динамически размещаемую переменную выделяется процедурой NEW. Параметром обращения к этой процедуре является типизированный указатель. В результате обращения указатель приобретает значение , соответствующее динамическому адресу, начиная с которого можно разместить данные:

```
var
    i: ^integer;
     j: ^real;
begin
    new(i);
.....
```

После выполнения этого фрагмента указатель i приобретает значение, которое перед этим имел указатель кучи HEAPPTR, а сам HEAPPTR увеличивает свое значение на 2 , так как внутреннее представление типа integer, с которым связан указатель i, составляет 2 байта.

После того как указатель приобрел некоторое значение, т. е. стал указывать на конкретный физический адрес памяти, по этому адресу можно разместить любое значение соответствующего типа. Для этого сразу за указателем без пробелов ставится значок ^, например:

i^:=2;{в область памяти i помещено значение 2}.

Значением любого указателя является адрес, а чтобы указать что речь идет не об адресе, а о тех данных, которые расположены по этому адресу, за указателем ставится ^.

Чтобы вернуть байты обратно в кучу, используется процедура DISPOSE.

DISPOSE(i) - возвращает в кучу 2 байта, которые ранее были выделены указателем i.

Для работы с нетипизированными указателями используются процедуры:

GETMEM(P,SIZE) - резервирование памяти.

FREEMEM(P,SIZE) - освобождение памяти.

Указательная переменная P может быть в 3 –х состояних:

1. Содержать адрес какой-либо переменной, память под которую уже выделена.
2. Содержать постоянный пустой адрес nil.
3. Находится в неопределенном состоянии.

В неопределённом состоянии указатель бывает в начале работы программы до первого присвоения ему или конкретного адреса или пустого адреса nil , а также после освобождения области памяти на которую он указывает.

Для организации связей между элементами динамических структур данных, требуется чтобы каждый элемента содержал кроме информационных значений, как минимум один указатель. Отсюда следует, что в качестве элементов таких структур необходимо использовать записи, которые могут объединять в одно целое разнородные элементы.

```
Type
    TPtr = ^Telem;
    Telem = record
            Inf: Real;
            Link: TPtr
    End;
```

10.2. Списки.

Указатели являются простым механизмом, позволяющим строить данные со сложной и меняющейся структурой. Используя указатели можно создавать и обрабатывать структуры данных, компоненты которых связаны явными ссылками. Самый простой способ связать множество элементов – это расположить их линейно в списке. В этом случае каждый элемент содержит только один указатель на следующий элемент списка.

Пусть тип point описан следующим образом:

```
Type point = ^ item;
   item = record
          number: integer;
          next: point
          end;
```

Каждая переменная типа point состоит из двух компонентов: индентифицирующего номера и указателя на следующий элемент. Из этих переменных, связав их указателями, можно создать линейный список.

Способ построения линейного списка: начиная с пустого списка, последовательно добавлять элементы в его начало.

Процесс формирования списка из n элементов:

```
First: = nil; {начало с пустого списка}
While n>0 do begin
New (r); r^. Next: = first; r^. Number: = n;
First: = r; n: = n-1 end;
```

Основные операции со списками

Просмотр списка

Напишем процедуру, которая вывод на экран значение поля number элементов списка.

```
{просмотр списка}
procedure Print (first: point);
Var r: point
Begin
R: = first;
While r<>nil do begin
Writeln (‘number = ‘ , r^. Number);
R:= r^. Next;   End;
```

Поиск в списке

Очень частая операция – поиск в списке элементов с заданным значением ключевого поля x. Так же как в случае файлов, поиск ведется последовательно. Он заканчивается либо когда элемент найден, либо когда достигнут конец списка.

```
{поиск в линейном списке}
Procedure Search (first: point; x: integer; var q: point);
{q – звращает указатель на найденный элемент;  q – nil, если элемент с ключем  x в списке нет}
var r: point;
   ok: boolean;
begin
r: = first; ok: = true;
while (r<>nil) and ok do
if r^. Number = x then
ok: = false;
else r: = r^. Next; q: = r
```

```
end;
```

Включить элемент в список

Элемент нужно включить в середину списка после элемента, на который указывает ссылка q. Процедура включения записывается следующим образом.

```
{включить элемент в середину списка перед q^}
Procedure Insert (Var q: point; x: integer):
{x – значение информационного поля включаемого элемента}
Var r: point
Begin
New  (r); {размещаем элемент в памяти}
R^. Number: = x;
{меняем ссылку}
r^/ next: = q^. Next; q^. Next: = r
end;
```

Если требуется включить перед элементом q^, а не после него, то кажется, что однонаправленная цепочка связей создает трудность, поскольку нет “прохода” к элементам, предшествующим данному. Однако эту проблему можно решить, используя простой прием, который состоит в том, что новый элемент вставляется после q^, но затем происходит обмен значениями между новым элементом и q^.

```
{включить элемент в середину списка перед q^}
Procedure insert Before (Var q: point; x: integer);
Var r: point;
Begin
New   ( r ); {размещаем элемент памяти}
{включаем элемент после  q^}
r^. Next: = q^. Next; q^. Next: = r;
{выполняем обмен значениями}
r^. Number: = q^. Nunber;
q^. Number: = x
end;
```

Удалить элемент из списка

Посмотрим как удаляется элемент из середины списка. Следует иметь в виду, что этот довольно очевидный и простой прием можно применять только в случае, когда у q^ есть последующий элемент, т.е он не является последним в списке.

Предположим, что надо удалить элемент, расположенный после элемента, на который указывает ссылка q

```
{удаление элемента из середины списка после q^}
Procedure Del (Var q: point);
Var r: point;
Begin
r: = q^. Next;
q^. Next: = q^. Next;
r^. Next: = nil
End;
```

Если следует удалить элемент на который указывается ссылка q, то следует в начале присвоить элементу q^ значение следующего за ним элемента, а затем этот элемент удалить.

```
{удаление элемента q^}
Prrocedure Deiet (Var q: point):
Var r: point;
```

```
Begin
r: = q^. next;
q^: = r^;
r^. Next: = nil;  End;
```

Обратите внимание на то, что удаляемый из списка элемент остается в памяти и к нему имеется доступ к указателю r, так что в дальнейшем этот элемент можно вставить, например, в другой список. Если требуется освободить занимаемую этим элементом память, то следует выполнить

```
Dispose ( r );
r : = nil
```

Лабораторная работа № 13.

Исключение элементов списка.

Цель задания:

1. Ознакомиться с возможностью выполнения операции исключения элементов из списка.
2. Закрепление навыков использования переменных ссылочных типов данных.

Постановка задачи:

1. Составить список учебной группы, содержащей 20 учащихся.
2. Указать для каждого учащегося оценки, полученные на четырех экзаменах.
3. Разработать программу, которая вводит с экрана данные о каждом учащемся и заносит эти данные в однонаправленный список.
4. Обработать список согласно конкретному варианту.

Содержание отчета:

1. Постановка задачи.
2. Текст программы и результаты ее выполнения.

Образец выполнения работы.

Лабораторная работа № 13.

Исключение элементов списка.

Цель задания:

1. Ознакомиться с возможностью выполнения операции исключения элементов из списка.
2. Закрепление навыков использования переменных ссылочных типов данных.

Постановка задачи:

Составить список учебной группы, содержащей 20 учащихся.

Указать для каждого учащегося оценки, полученные на четырех экзаменах. Разработать программу, которая вводит с экрана данные о каждом учащемся и заносит эти данные в однонаправленный список.

Обработать список согласно конкретному варианту.

Содержание отчета:

1. Постановка задачи.
2. Текст программы и результаты ее выполнения.

Вариант задания:

Одна оценка 4, а остальные 3.

Текст программы:

```
                {Исключение элементов из списка}
Program ExludingelementsFromList;
Uses CRT;
Type
    PStudents= ^TStudents;
    TStudents= Record
                Name: String[20];
                Marks: Array [1..4] of ShortInt;
                Next:PStudents;
            End;

Var
  PS:PStudents; {указатель на последний элемент списка в статической памяти}

    { Процедура заполнения списка }
Procedure Init;

Var
  i,y:Integer;
  pro:PStudents;
Label
  Exits;

Begin
PS^.Next:=nil; {последний элемент списка}
y:=1;
  While true Do Begin
     New(pro);      {выделяем память под переменную с указателем pro}

     {Присваиваем значение переменной}
     WriteLn('Введите Ф.И.О. ',y,'-го студента, "Enter" - завершение программы');
     ReadLn(pro^.Name);
     If pro^.Name='' Then GoTo Exits;
     WriteLn('Введите оценки студента (всего 4)');
```

```
        For i:=1 To 4 Do ReadLn(pro^.Marks[i]);

      pro^.Next:=PS; {записываем в поле Next указатель на предыдущий элемент}
      PS:=pro; {указателю на голову списка присваиваем новое значение
            т.е значение текущего элемента}
   Inc(y);
   End;

Exits : End;

   {Процедура удаления элементов из списка }
Procedure Removing;
Var
   Head,p1,p2:PStudents;
   i,e3,e4:ShortInt;
Label
   Exits;

Begin
head:=PS; {первый элемент-голова списка}
p2:=PS; {текущий указатель}
p1:=PS; {указатель на предыдущий элемент}

 While True Do Begin
  e4:=0; e3:=0;
   For i:=1 to 4 Do Begin  {подсчет оценок}
      If p2^.Marks[i]=4 Then inc(e4);
      If p2^.Marks[i]=3 Then inc(e3);
   End;
   If (e4=1) And (e3=3) Then  {проверка условия на удаление}
      If (Head=P2) Then Begin  {если удаляемый элемент - голова списка}
         PS:=PS^.Next; {новая голова}
         Dispose(p2); p2:=PS; p1:=PS; Head:=PS;
         End
      Else Begin         {если элемент в середине списка}
         p1^.Next:=p2^.Next; {полю Next предыдущего элемента
                        присваиваем указатель следующего за текущим}
         Dispose(p2); p2:=p1^.Next; {уничтожаем ссылку на текущий элемент}

      End
   Else Begin
          p2:=p2^.Next;{если ничего не удалялось
                   передвигаем указатель на следующий элемент}
          p1:=p1^.Next; {передвигаем указатель предыдущего элемента}
        End;
  If  (p2=nil) Then GoTo Exits;

 End;
Exits:End;


  {Процедура вывода на печать списка }
```

```
Procedure PrintOut;
var
   p1:PStudents;
Label
   Exits;
Begin
p1:=PS;
  While True Do Begin
    WriteLn(p1^.Name);
    If p1^.Next=nil Then GoTo Exits;
    p1:=p1^.Next;
  End;
Exits:End;


        {Тело программы }
Begin
Init;
Removing;
PrintOut;
WriteLn('Нажмите любую клавишу...');
Repeat Until KeyPressed;
End.
```

Результат работы программы:

```
Введите Ф.И.О. 1-го студента, "Enter" - завершение программы
Иванов И.И.
Введите оценки студента (всего 4)
4
2
3
5
Введите Ф.И.О. 2-го студента, "Enter" - завершение программы
Петров П.П.
Введите оценки студента (всего 4)
5
5
5
5
Введите Ф.И.О. 3-го студента, "Enter" - завершение программы
Сидоров С.С.
Введите оценки студента (всего 4)
4
3
3
3
Введите Ф.И.О. 4-го студента, "Enter" - завершение программы
Иваненко И.И.
Введите оценки студента (всего 4)
4
3
```

```
3
3
Введите Ф.И.О. 5-го студента, "Enter" - завершение программы
Петренко
Введите оценки студента (всего 4)
5
3
2
1
Введите Ф.И.О. 6-го студента, "Enter" - завершение программы
Сидоренко С.С.
Введите оценки студента (всего 4)
3
4
3
3
Введите Ф.И.О. 7-го студента, "Enter" - завершение программы
Иванчук И.И.
Введите оценки студента (всего 4)
2
3
4
2
Введите Ф.И.О. 8-го студента, "Enter" - завершение программы
Петрук П.П.
Введите оценки студента (всего 4)
5
5
5
3
Введите Ф.И.О. 9-го студента, "Enter" - завершение программы
Сидорчук С.С.
Введите оценки студента (всего 4)
3
3
3
4
Введите Ф.И.О. 10-го студента, "Enter" - завершение программы
Самосадкин С.С.
Введите оценки студента (всего 4)
4
3
3
3
Введите Ф.И.О. 11-го студента, "Enter" - завершение программы
Самоделкин С.С.
Введите оценки студента (всего 4)
4
3
4
3
Введите Ф.И.О. 12-го студента, "Enter" - завершение программы
```

```
Самопалкин С.С.
Введите оценки студента (всего 4)
3
3
3
4
Введите Ф.И.О. 13-го студента, "Enter" - завершение программы
Самохвалкин С.С.
Введите оценки студента (всего 4)
3
4
3
3
Введите Ф.И.О. 14-го студента, "Enter" - завершение программы
Самострелкин С.С.
Введите оценки студента (всего 4)
2
3
3
3
Введите Ф.И.О. 15-го студента, "Enter" - завершение программы
Самоедкин С.С.
Введите оценки студента (всего 4)
4
3
3
3
Введите Ф.И.О. 16-го студента, "Enter" - завершение программы
Самогонкин С.С.
Введите оценки студента (всего 4)
3
3
4
3
Введите Ф.И.О. 17-го студента, "Enter" - завершение программы
Самокаткин С.С.
Введите оценки студента (всего 4)
5
3
3
3
Введите Ф.И.О. 18-го студента, "Enter" - завершение программы
Самолеткин С.С.
Введите оценки студента (всего 4)
2
3
5
4
Введите Ф.И.О. 19-го студента, "Enter" - завершение программы
Самоходкин С.С.
Введите оценки студента (всего 4)
5
```

```
2
3
4
Введите Ф.И.О. 20-го студента, "Enter" - завершение программы
Самоучкин С.С.
Введите оценки студента (всего 4)
3
3
3
4
Введите Ф.И.О. 21-го студента, "Enter" - завершение программы

Самоходкин С.С.
Самолеткин С.С.
Самокаткин С.С.
Самострелкин С.С.
Самоделкин С.С.
Петрук П.П.
Иванчук И.И.
Петренко
Петров П.П.
Иванов И.И.
Нажмите любую клавишу...
```

Варианты задания.

Исключить из списка элементы, относящиеся к учащимся, у которых:

1) Средний балл меньше среднего балла группы.
2) Средний балл меньше 4,5.
3) Средний балл больше 4.
4) Все оценки 5.
5) Одна оценка 4, а остальные - 5.
6) Оценка, полученная на первом экзамене - 2.
7) Оценка, полученная на втором экзамене - 5.
8) Нет удовлетворительных и неудовлетворительных оценок.
9) Больше одной оценки 2.
10) Одна оценка 3, а остальные 4 и 5.
11) Одна оценка 5, а остальные 4 .
12) Оценки только 4 и 5.
13) Больше одной оценки 3.
14) Две оценки 3.
15) Одна оценка 4, а остальные 3.
16) Все оценки 3.
17) Оценки 3 и 4.
18) Все оценки 4.
19) Оценка, полученная на первом экзамене - 3.
20) Оценки за первый и третий экзамен 3.
21) Одна оценка 5, а остальные 3.

22) Только одна оценка 5.
23) Две оценки 5, а остальные 2.
24) Одна оценка 2, а остальные 5.
Распечатать оставшийся список.

Лабораторная работа № 14.

Работа со списками.

Цель работы:

1. Закрепить навыки работы с исключением элементов из списка.
2. Ознакомиться с возможностью добавления элементов в список.

Постановка задачи:

Подготовить данные об абитуриентах, поступающих в институт. Информацию о каждом абитуриенте оформить в вид записи со следующими полями:

1.ФИО.
2. Год рождения.
3. Год окончания школы.
4. Оценки в аттестате.
5. Оценки вступительных экзаменов (3).

Оформить информацию в виде внешнего файла.

Занести информацию из внешнего файла в однонаправленный список.
Обработать список согласно варианту.

1. Исключить из списка все элементы, удовлетворяющие условию, заданному в варианте и распечатать полученный список.
2. Добавить N элементов в начало (конец) списка согласно конкретному варианту и распечатать полученный список.

Содержание отчета:

1. Постановка задачи.
2. Тексты программ и результаты их выполнения.

Образец выполнения работы.

Лабораторная работа № 14.

Работа со списками.

Цель работы:

1. Закрепить навыки работы с исключением элементов из списка.
2. Ознакомиться с возможностью добавления элементов в список.

Постановка задачи:

Подготовить данные об абитуриентах, поступающих в институт. Информацию о каждом абитуриенте оформить в вид записи со следующими полями:

1.ФИО.

2. Год рождения.
3. Год окончания школы.
4. Оценки в аттестате.
5. Оценки вступительных экзаменов (3).

Оформить информацию в виде внешнего файла.

Занести информацию из внешнего файла в однонаправленный список.

Обработать список согласно варианту.

1. Исключить из списка все элементы, удовлетворяющие условию, заданному в варианте и распечатать полученный список.

2. Добавить N элементов в начало (конец) списка согласно конкретному варианту и распечатать полученный список.

Вариант задания:

1. Все оценки 4.
2. Добавить в список элементы, относящиеся к абитуриентам, у которых : все оценки 3 за экзамены и отличный аттестат.

Текст программы:

```
{Заполнение внешнего файла данными, вводимыми с клавиатуры}

Program P8_5_2;
Uses CRT;

Const
   GradMarks=5; {количество оценок в аттестате}
   ExamenMarks=3; {количество оценок на экзамене}

Type
  Data = Record
    Name:String[20];  {Имя}
    BirthDate,GradDate:Integer;  {год рождения, год окончания школы}
    SchoolGrad: Array[1..GradMarks] of Byte;  {оценки в аттестате}
    ExamGrad: Array[1..ExamenMarks] of Byte;  {оценки на экзамене}
    End;

Var
   Telega: Data;
   FileOfData: File of Data ;

        (* Процедура заполнения файла 'data.dat' *)

Procedure Initialising;
Var i,y:Integer;
Label
   Exits;
```

```
Begin
  Assign(FileOfData,'data.dat');
  ReWrite(FileOfData);
  i:=1;

   While True Do Begin
    WriteLn('_____ Д А Н Н Ы Е  ',i,' -ГО  А Б И Т У Р И Е Н Т А_____');
     WriteLn;

    With Telega Do Begin

        WriteLn('Введите Ф.И.О. абитуриента:');
        WriteLn('"Enter"-завершение программы');
          ReadLn(Name);
        If (Name='') Then Goto Exits;

        WriteLn('Введите год рождения абитуриента:');
          ReadLn(BirthDate);

        WriteLn('Введите год окончания школы абитуриентом:');
          ReadLn(GradDate);

        WriteLn('_______Введите оценки из аттестата абитуриента_______');
        WriteLn('всего ',GradMarks,' штук');
        WriteLn;
          For y:=1 to GradMarks Do Begin
             ReadLn(SchoolGrad[y]);
          End;


        WriteLn('Введите оценки, полученные абитуриентом на экзамене:');
        WriteLn('Всего ',ExamenMarks,' штук');
        WriteLn;
           For y:=1 to ExamenMarks Do Begin
              ReadLn(ExamGrad[y]);
           End;

    End;
  Write(FileOfData,Telega);
  inc(i);
   End;
Exits: End;



          {*************** Тело программы ***************}
Begin
Initialising;
Close(FileOfData);
End.
```

Результат работы программы:

_____ Д А Н Н Ы Е 1 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Иванов И.И.
Введите год рождения абитуриента:
1970
Введите год окончания школы абитуриентом:
1988
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
5
5
5
5
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

4
4
4
_____ Д А Н Н Ы Е 2 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Петров
Введите год рождения абитуриента:
1971
Введите год окончания школы абитуриентом:
1989
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
5
5
5
5
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

3
3
3
_____ Д А Н Н Ы Е 3 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Сидоров С.С.
Введите год рождения абитуриента:

1972
Введите год окончания школы абитуриентом:
1990
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

4
3
5
4
5
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

3
2
4
_____ Д А Н Н Ы Е 4 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Васильев В.В.
Введите год рождения абитуриента:
1973
Введите год окончания школы абитуриентом:
1991
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
4
3
4
3
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

4
4
4
_____ Д А Н Н Ы Е 5 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Сергеев С.С.
Введите год рождения абитуриента:
1974
Введите год окончания школы абитуриентом:
1992
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
5
5
5
5
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

3
3
3
_____ Д А Н Н Ы Е 6 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Юрьев Ю.Ю.
Введите год рождения абитуриента:
1974
Введите год окончания школы абитуриентом:
1994
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
5
5
5
5
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

3
5
5
_____ Д А Н Н Ы Е 7 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Ильин И.И.
Введите год рождения абитуриента:
1975
Введите год окончания школы абитуриентом:
1993
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

4
4
4
3
3
Введите оценки, полученные абитуриентом на экзамене:

Всего 3 штук

3
3
3
_____ Д А Н Н Ы Е 8 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Михайлов М.М.
Введите год рождения абитуриента:
1976
Введите год окончания школы абитуриентом:
1998
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

4
4
4
5
4
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

4
4
4
_____ Д А Н Н Ы Е 9 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Савельев С.С.
Введите год рождения абитуриента:
1977
Введите год окончания школы абитуриентом:
1999
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
5
5
5
5
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

3
3
3
_____ Д А Н Н Ы Е 10 -ГО А Б И Т У Р И Е Н Т А_____

```
Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Николаев Н.Н.
Введите год рождения абитуриента:
1979
Введите год окончания школы абитуриентом:
1997
_______Введите оценки из аттестата абитуриента_______
всего 5 штук

4
5
4
4
4
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

4
4
4
__________ Д А Н Н Ы Е  11 -ГО  А Б И Т У Р И Е Н Т А_____________

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
```

```
Program T8_5_2_1;
Uses CRT;

Const
    GradMarks=5; {количество оценок в аттестате}
    ExamenMarks=3; {количество оценок на экзамене}

Type
  Data = Record
    Name:String[20];  {Имя}
    BirthDate,GradDate:Integer;  {год рождения, год окончания школы}
    SchoolGrad: Array[1..GradMarks] of Byte;  {оценки в аттестате}
    ExamGrad: Array[1..ExamenMarks] of Byte;  {оценки на экзамене}
    End;
  DataPointer=^DataList;
  DataList = Record
    Name:String[20];  {Имя}
    BirthDate,GradDate:Integer;  {год рождения, год окончания школы}
    SchoolGrad: Array[1..GradMarks] of ShortInt;  {оценки в аттестате}
    ExamGrad: Array[1..ExamenMarks] of ShortInt;  {оценки на экзамене}
    Next: DataPointer;
    End;
```

```
Var
    Telega: Data;
    P:DataPointer;
    Stop:Boolean;

{**** Копирование записей из внешнего файла в однонаправленный список *****}
Procedure CopyToList;
Var i,ii:Integer;
    P1:DataPointer;
    DataFile: File of Data;
Begin
    p1:=nil;
  Assign(DataFile,'Students.dat');
  Reset(DataFile);

   While (Not EOF(DataFile)) Do Begin
     Read(DataFile,Telega);
     New(P1);
     P1^.Name:=Telega.Name;
     P1^.BirthDate:=Telega.BirthDate;
     P1^.GradDate:=Telega.GradDate;
     For i:=1 To GradMarks Do P1^.SchoolGrad[i]:=Telega.SchoolGrad[i] ;
     For i:=1 To ExamenMarks Do P1^.ExamGrad[i]:=Telega.ExamGrad[i];
     P1^.Next:=P;    {ссылка на предыдущий элемент}
     P:=P1           {новая голова списка}
   End;
Close(DataFile);
End;


    {***** Удаление элементов списка, удовлетворяющих условию *****}

Function Removing(Head:DataPointer):DataPointer;
var
  i,e4: ShortInt;
Begin
   e4:=0;
   If Head=nil Then Removing:=nil     {если список пустой}
   Else Begin
      For i:=1 to ExamenMarks Do If Head^.ExamGrad[i]=4 Then inc(e4);
      If e4=3 Then Begin
         Removing:=Head^.Next;    {обработка головы списка}
         Dispose(Head); Head:=nil;
      End
      Else Begin {если удаляемый элемент-не голова,тогда рекурсия}
         Head^.Next:=Removing(Head^.Next); {по следующим за ним}
         Removing:=Head;
         If Head^.Next=nil Then Stop:=True; {выход из функции и из цикла}
      End;                              {по достижении конца списка}
   End;
End;
```

```
  {*************** Процедура вывода на печать списка *********************}
Procedure PrintOut;
var
   p1:DataPointer;
   s1,s2:string;
   i,n:integer;
Label
   Finita;
Begin
p1:=P;
WriteLn('_______  Ф.И.О. абитуриентов, оставшихся в списке  _________');
WriteLn;
  While True Do Begin
    s1:='';
    s2:='';
    For i:=1 to ExamenMarks Do Begin
      str(p1^.ExamGrad[i],s1);
      s2:=s2+s1+', ';
    End;
    WriteLn('имя: ',p1^.Name,' (оценки: ',s2,')');
    If p1^.Next=nil Then GoTo Finita;
    p1:=p1^.Next;
  End;
Finita:End;



          {**************** Тело программы ***************}
Begin
P:=nil;
CopyToList;
Stop:=False;
While (Not Stop) Do P:=Removing(P);  {вызов рекурсивной функции}
PrintOut;
WriteLn('Press any key...');
Repeat Until KeyPressed;
End.
```

Результат работы программы:

```
_______  Ф.И.О. абитуриентов, оставшихся в списке  _________

имя: Савельев С.С. (оценки: 3, 3, 3, )
имя: Ильин И.И. (оценки: 3, 3, 3, )
имя: Юрьев Ю.Ю. (оценки: 3, 5, 5, )
имя: Сергеев С.С. (оценки: 3, 3, 3, )
имя: Сидоров С.С. (оценки: 3, 2, 4, )
имя: Петров (оценки: 3, 3, 3, )
Press any key...
```

{Добавление в список элементов с условием}

```
Program T8_5_2_2;
Uses CRT;

Const
    GradMarks=5; {количество оценок в аттестате}


    ExamenMarks=3; {количество оценок на экзамене}

Type
  Data = Record {тип элементов в файле}
    Name:String[20];  {Имя}
    BirthDate,GradDate:Integer;  {год рождения, год окончания школы}
    SchoolGrad: Array[1..GradMarks] of Byte;  {оценки в аттестате}
    ExamGrad: Array[1..ExamenMarks] of Byte;  {оценки на экзамене}
    End;
  DataPointer=^DataList;
  DataList = Record {тип элементов в списке}
    Name:String[20];  {Имя}
    BirthDate,GradDate:Integer;  {год рождения, год окончания школы}
    SchoolGrad: Array[1..GradMarks] of Byte;  {оценки в аттестате}
    ExamGrad: Array[1..ExamenMarks] of Byte;  {оценки на экзамене}
    Next: DataPointer;
    End;


Var
   Telega: Data;
   P:DataPointer;
   StopCondition:Boolean;

{**** Копирование записей из внешнего файла в однонаправленный список *****}
Procedure LoadFromFileToList;
Var i,y:Integer;
   P1:DataPointer;
   FileOfData: File of Data;
Begin
   p1:=nil;
  Assign(FileOfData,'data.dat');
  Reset(FileOfData);

   While (Not EOF(FileOfData)) Do Begin
    Read(FileOfData,Telega);
    New(P1);
    P1^.Name:=Telega.Name;
```

```
    P1^.BirthDate:=Telega.BirthDate;
    P1^.GradDate:=Telega.GradDate;
    For i:=1 To GradMarks Do P1^.SchoolGrad[i]:=Telega.SchoolGrad[i] ;
    For i:=1 To ExamenMarks Do P1^.ExamGrad[i]:=Telega.ExamGrad[i];
    P1^.Next:=P;    {ссылка на предыдущий элемент}
    P:=P1           {новая голова списка}
  End;
Close(FileOfData);
End;



   {***** Удаление элементов списка, удовлетворяющих условию *****}
{удалить всех кроме тех, у которых все оценки "3" за экзамены}
{и отличный аттестат}
Function Filtering(Head:DataPointer):DataPointer;
var
  i,Mark_0,Mark_1: ShortInt;
Begin
  Mark_0:=0;
  Mark_1:=0;
  If Head=nil Then Filtering:=nil     {если список пустой}
  Else Begin
    For i:=1 to ExamenMarks Do If Head^.ExamGrad[i]=3 Then inc(Mark_0);
    For i:=1 to GradMarks Do If Head^.SchoolGrad[i]=5 Then inc(Mark_1);

    If not ((Mark_0=ExamenMarks) and (Mark_1=GradMarks)) Then Begin
      Filtering:=Head^.Next;    {обработка головы списка}
      Dispose(Head); Head:=nil;
    End
    Else Begin {если удаляемый элемент-не голова,тогда рекурсия}
      Head^.Next:=Filtering(Head^.Next); {по следующим за ним}
      Filtering:=Head;
      If Head^.Next=nil Then StopCondition:=True; {выход из функции и из цикла}
    End;                              {по достижении конца списка}
  End;
End;


  {*************** Процедура вывода на печать списка **********************}
Procedure PrintResult;
var
  p1:DataPointer;
  str1,str2,str3:string;
  i,n:integer;
Label
  Exits;
Begin
p1:=P;
HighVideo;
WriteLn('       _______  Ф.И.О. абитуриентов, оставшихся в списке  ________');
NormVideo;
```

```
WriteLn;
  While True Do Begin
    str1:='';
    str2:='';
    str3:='';
    For i:=1 to ExamenMarks Do Begin
      str(p1^.ExamGrad[i],str1);
      str2:=str2+str1+', ';
    End;
    For i:=1 to GradMarks Do Begin
      str(p1^.SchoolGrad[i],str1);
      str3:=str3+str1+', ';
    End;
    WriteLn('имя: ',p1^.Name,' (оценки в аттестате: ',str3,'на экзамене: ',str2,')');
    If p1^.Next=nil Then GoTo Exits;
    p1:=p1^.Next;
  End;
Exits:End;


Function CheckOut:Boolean;
var
  Mark_1,e3,i:ShortInt;
Begin
 e3:=0; Mark_1:=0;
 For i:=1 to ExamenMarks Do If Telega.ExamGrad[i]=3 Then inc(e3);
 For i:=1 to GradMarks Do If Telega.SchoolGrad[i]=5 Then inc(Mark_1);
 If ((e3=ExamenMarks) and (Mark_1=GradMarks)) Then CheckOut:=true
 Else CheckOut:=false;
End;

Procedure AddictionToList;
var
   P1:DataPointer;
   i:ShortInt;
Begin
 New(P1);
 P1^.Name:=Telega.Name;
 P1^.BirthDate:=Telega.BirthDate;
 P1^.GradDate:=Telega.GradDate;
 For i:=1 To GradMarks Do P1^.SchoolGrad[i]:=Telega.SchoolGrad[i] ;
 For i:=1 To ExamenMarks Do P1^.ExamGrad[i]:=Telega.ExamGrad[i];
 P1^.Next:=P;    {ссылка на предыдущий элемент}
 P:=P1           {новая голова списка}
End;


Procedure Initialising;
Var
   i,y:Integer;
Label
    Exits;
```

```
Begin
WriteLn('     Дополнение данного списка новыми элементами: ');
WriteLn;
  i:=1;

   While True Do Begin
    WriteLn('          __________ Д А Н Н Ы Е  ',i,' -ГО  А Б И Т У Р И Е Н Т А__________');
     WriteLn;

     With Telega Do Begin

         WriteLn('Введите Ф.И.О. абитуриента:');
         WriteLn('"Enter"-завершение программы');
           ReadLn(Name);
         If (Name='') Then Goto Exits;

         WriteLn('Введите год рождения абитуриента:');
           ReadLn(BirthDate);

         WriteLn('Введите год окончания школы абитуриентом:');
           ReadLn(GradDate);

         WriteLn('    _______Введите оценки из аттестата абитуриента_______');
         WriteLn('всего ',GradMarks,' штук');
         WriteLn;
           For y:=1 to GradMarks Do Begin
              ReadLn(SchoolGrad[y]);
           End;


         WriteLn('     Введите оценки, полученные абитуриентом на экзамене:');
         WriteLn('Всего ',ExamenMarks,' штук');
         WriteLn;
            For y:=1 to ExamenMarks Do Begin
               ReadLn(ExamGrad[y]);
            End;

     End;
  If CheckOut Then AddictionToList;
  inc(i);
   End;
Exits: End;


          {**************** Тело программы ***************}
Begin
ClrScr;
P:=nil;
LoadFromFileToList;
StopCondition:=False;
While (Not StopCondition) Do P:=Filtering(P);  {вызов рекурсивной функции}
PrintResult;
```

```
WriteLn;
Initialising;
PrintResult;
WriteLn('Press any key...');
While not KeyPressed do;
End.
```

Результат работы программы:

_____ Ф.И.О. абитуриентов, оставшихся в списке _____

имя: Савельев С.С. (оценки в аттестате: 5, 5, 5, 5, 5, на экзамене: 3, 3, 3,)
имя: Сергеев С.С. (оценки в аттестате: 5, 5, 5, 5, 5, на экзамене: 3, 3, 3,)
имя: Петров (оценки в аттестате: 5, 5, 5, 5, 5, на экзамене: 3, 3, 3,)

Дополнение данного списка новыми элементами:

_____ Д А Н Н Ы Е 1 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Смирнов В.В.
Введите год рождения абитуриента:
1970
Введите год окончания школы абитуриентом:
1978
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
4
5
4
5
Введите оценки, полученные абитуриентом на экзамене:
Всего 3 штук

3
3
3
_____ Д А Н Н Ы Е 2 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы
Пермяков
Введите год рождения абитуриента:
1980
Введите год окончания школы абитуриентом:
1998
_____Введите оценки из аттестата абитуриента_____
всего 5 штук

5
5
5
5
5
Введите оценки, полученные абитуриентом на экзамене:

Всего 3 штук

3
3
3
_____ Д А Н Н Ы Е 3 -ГО А Б И Т У Р И Е Н Т А_____

Введите Ф.И.О. абитуриента:
"Enter"-завершение программы

_____ Ф.И.О. абитуриентов, оставшихся в списке _____

имя: Пермяков (оценки в аттестате: 5, 5, 5, 5, 5, на экзамене: 3, 3, 3,)
имя: Савельев С.С. (оценки в аттестате: 5, 5, 5, 5, 5, на экзамене: 3, 3, 3,)
имя: Сергеев С.С. (оценки в аттестате: 5, 5, 5, 5, 5, на экзамене: 3, 3, 3,)
имя: Петров (оценки в аттестате: 5, 5, 5, 5, 5, на экзамене: 3, 3, 3,)
Press any key...

Варианты задания.

1. Исключить из списка элементы, относящиеся к абитуриентам, у которых:
1) одна оценка 2;
2) все оценки 3;
3) средний балл меньше 4,5 и одна оценка 3 в аттестате;

4) средний балл больше 4 и одна оценка 3 в аттестате;
5) средний балл меньше 4;
6) возраст больше 18 лет;
7) нет ни одной оценки 5 в аттестате;
8) нет отличного аттестата, но все оценки за экзамены 5;
9) хотя бы одна оценка 3;
10) отличный аттестат;
11) больше одной оценки 3 в аттестате;
12) нет ни одной оценки 5 в аттестате;
13) хотя бы одна оценка 3 в аттестате;
14) отличный аттестат и больше одной оценки 3 за экзамены;
15) меньше 2-х оценок 5;
16) аттестат с хотя бы одной оценкой 3 и нет оценки 5 за экзамены;
17) оценки 3 и 4 за экзамены;
18) все оценки 3 за экзамены и отличный аттестат;
19) средний балл меньше 4,5;
20) средний балл больше 4 и оценки 3 в аттестате;
21) средний балл меньше 4;
22) отличный аттестат и возраст больше 18 лет;
23) отличный аттестат и возраст меньше 17 лет;
24) Нет удовлетворительных и неудовлетворительных оценок;
25) все оценки 4;

2. Добавить в список элементы, относящиеся к абитуриентам, у которых :

1) отличный аттестат;
2) одна оценка 4, а остальные 5;
3) средний балл в аттестате 4,5;
4) две оценки 4,а остальные 5;
5) средний балл меньше 4;
6) все экзамены сданы на 5;
7) все экзамены сданы на 4 и 5;
8) все экзамены сданы на 4;
9) одна оценка 4, а остальные 5;

10) одна оценка 5, а остальные 4;
11) две оценки 4, а остальные 5;
12) одна оценка 3;
13) одна оценка 3 в аттестате;
14) больше одной оценки 3 в аттестате;
15) средний балл в аттестате ниже 4,5;
16) две оценки 3 за экзамены и отличный аттестат;
17) нет ни одной оценки 5 в аттестате;
18) отличный аттестат и средний балл меньше 4;
19) средний балл больше 4 и оценки 3 в аттестате;
20) возраст больше 18 лет и все оценки 5 за экзамены;
21) средний балл больше 4,5 и одна оценка 3 в аттестате;
22) отличный аттестат и одна оценка 3 за экзамены;
23) одна оценка 3 за экзамены, а остальные 4;
24) одна оценка 3 за экзамены, а остальные 5;
25) все оценки 3 за экзамены и отличный аттестат.

Лабораторная работа № 15.

Выполнение операций над списковыми структурами.

Цель задания:

1. Ознакомление с возможностями представления строк символов в виде списков.
2. Закрепление навыков выполнения операций над списками.

Постановка задачи:

Ввести с клавиатуры строку символов, формируя из ее элементов однонаправленный список. Обработать список согласно конкретному варианту. Распечатать результат.

Содержание отчета:

1. Постановка задачи.
2. Входная строка символов.
3. Текст программы.

Выходная строка символов.

Образец выполнения работы.

Лабораторная работа № 15.

Выполнение операций над списковыми структурами.

Цель задания:

1. Ознакомление с возможностями представления строк символов в виде списков.
2. Закрепление навыков выполнения операций над списками.

Постановка задачи:

Ввести с клавиатуры строку символов, формируя из ее элементов однонаправленный список. Обработать список согласно конкретному варианту. Распечатать результат.

Вариант задания:

Поменять местами первый символ и символ, стоящий посередине строки;
{Представление строк символов в виде списков}

```
Program T853;
Uses CRT;
Type
   PListHead=^TList;
   TList=Record
     Sym:Char;
     Next:PListHead;
     End;
Var
  str1, str2:String;
  P:PListHead; {указатель на голову списка}


Procedure InitString;
Begin
WriteLn('Введите строку символов');
ReadLn(str1);
End;


      {формируем из строки символов однонаправленный список}
Procedure ConvertStringToList;
Var
  i:integer;
  Head,P1,P2:PListHead;
Label
  Exits;
Begin
```

```
P:=nil;
P1:=nil;
P2:=nil;

 {создаем и заполняем голову списка и следующий за ней элемент}
Head^.Sym:=str1[1];
New(P1);
P1^.Sym:=str1[2];
Head^.Next:=P1;

 {создаем и заполняем остальные элементы}
For i:=3 to 255 Do Begin
  If Ord(str1[i])=0 Then Goto Exits;
  New(P2);
  P2^.Sym:=str1[i];
  P1^.Next:=P2;
  P1:=P2;
  End;

Exits:
P2^.Next:=nil;
P:=Head;
End;


Procedure EditList;
var
   i,i1,i2:integer;
   P1,P2,P3,PMiddle,PEnd:PListHead;
Begin
P1:=P;
i1:=-1; i2:=0;

While P1^.Next<>nil Do Begin {находим общее количество элементов(i1)}
  PEnd:=P1;      {PEnd - предпоследний элемент}
  P1:=P1^.Next;  {P1 - последний элемент}
  inc(i1);
End;

  i2:=i1 div 2; {порядковый номер элемента в середине списка}
  P2:=P;

For i:=0 to i2 Do Begin
  PMiddle:=P2; {PMiddle - элемент, предшествующий середине списка}
  P2:=P2^.Next; {P2 - середина списка}
End;

{перестановка ссылок}
PMiddle^.Next:=P1;
PEnd^.Next:=P2;
P3:=P2^.Next;
P2^.Next:=P1^.Next;
```

```
P1^.Next:=P3;

End;

  {*************** Процедура вывода на печать списка *********************}
Procedure Result;
var
   P1:PListHead;
   i:integer;
Begin
p1:=P;
i:=1;

{ формирование выходной строки }
  Repeat
    str2[0]:=Chr(i); {устанавливаем новую длину строки}
    str2[i]:=P1^.Sym; {присваиваем значение элементу строки}
    inc(i);
    P1:=P1^.Next;
  Until P1=nil;

WriteLn('       _______ Результат работы программы _____________        ');
WriteLn;
WriteLn('Входная строка символов: ',str1);
WriteLn('Выходная строка символов: ',str2);
WriteLn;
End;


Begin
InitString;
ConvertStringToList;
EditList;
Result;
WriteLn('Press any key...');
Repeat Until KeyPressed;
End.
```

Результат работы программы:

```
 Введите строку символов произвольной длины
 ABCDEFG

    Укажите какой элемент поменять местами с элементом в середине строки:
            цифра "1" - первый, цифра "2" - последний.

       _______ Результат работы программы _____________

 Входная строка символов: ABCDEFG
 Выходная строка символов: DBCAEFG

 Press any key...
```

Варианты заданий.

1) удалить два первые символа строки;
2) удалить последние три символа строки;
3) удалить все буквы К;
4) удвоить все символы *;
5) добавить в конец строки слово END;
6) поменять местами первый и последний символы строки;
7) поменять местами первый и второй символы строки;
8) поменять местами первый и предпоследний символы строки;
9) подсчитать в строке число букв А и В, и если букв А больше, чем букв В, то удалить в строке все символы В;
10) удалить все символы, равные первому символу строки;
11) подсчитать число символов в строке , и если число нечетное, то удалить символ, стоящий посередине строки;
12) поменять местами второй и предпоследний символы строки;
13) удалить два последние символа строки;

14) удалить все символы строки, повторяющиеся более двух раз;
15) устранить дублирование символов;
16) добавить в начало строки слово BEGIN;
17) увеличить количество символов на 5;
18) подсчитать число символов в строке , и если число четное, то добавить в конец строки один символ;
19) удалить все буквы Д;
20) добавить в конец строки символ, равный первому символу строки;
21) удалить из строки символы, порядковый номер которых кратен 3;
22) удалить из строки все символы с нечетным порядковым номером;
23) удалить из строки символы с четным порядковым номером;
24) удалить из строки символы, порядковый номер которых является простым числом;
25) поменять местами первый символ и символ, стоящий посередине строки;

10.3. Деревья.

Как известно, рекурсию можно эффективно использовать для определения сложных структур. Распространенным примером применения рекурсии служат деревья. Дерево определяется следующим образом: дерево с базовым типом Т - это либо:

1) пустая структура, либо
2) элемент типа Т, с которым связано конечное число деревьев с базовым типом Т , называемых поддеревьями.

Отсюда видно, что список есть дерево, в котором каждая вершина имеет не более одного поддерева, поэтому список называют вырожденным деревом.

Существует несколько способов изображения структуры дерева. Структура, представленная в виде графа и явно отражающая разветвления, привела к появлению термина “дерево”.

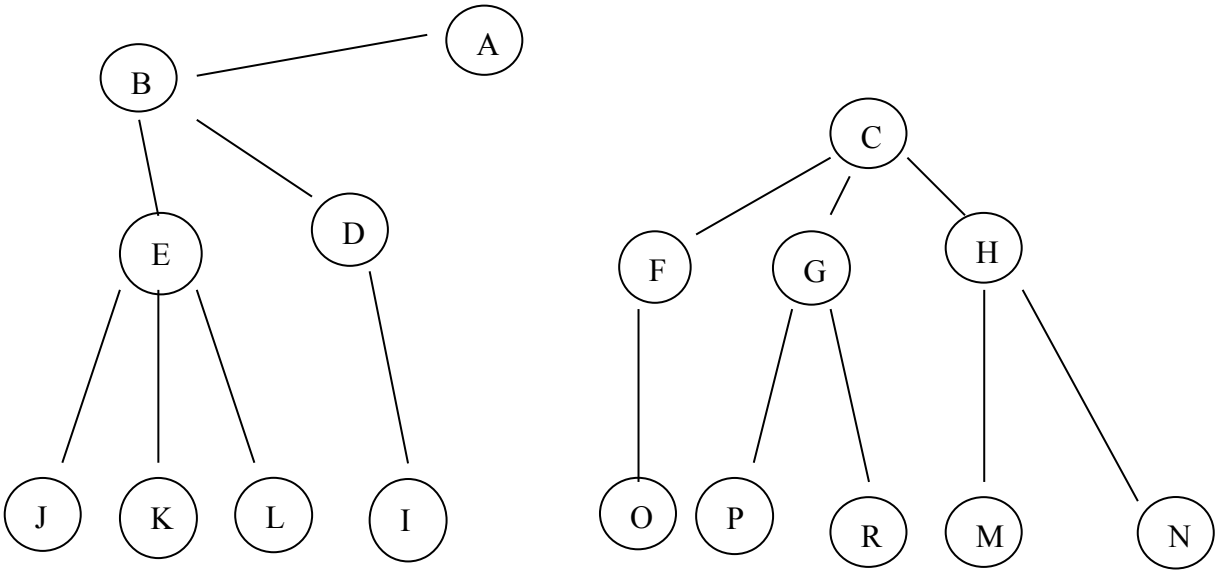


Рис. 1. Представление древовидной структуры в виде графа.

Элемент типа T называется узлом дерева. Связи между узлами дерева называются ветвями. Дерево, являющееся частью другого дерева называется поддеревом. Самый верхний узел называется корнем (на рис. 1 это узел A). Рассматривая пример дерева, приведенного на рисунке 1, можно также ввести понятие потомка и предка. Узел B, который находится непосредственно под узлом A, называется непосредственным потомком A. И наоборот, узел A является непосредственным предком узла B.

Узел, не имеющий потомков называется терминальным узлом или листом. Узел, не являющийся листом, называется внутренним узлом. Число потомков внутреннего узла называется его степенью. Максимальная степень всех узлов есть степень дерева. Например степень дерева на рис. 1 равна 3.

Упорядоченное дерево - это дерево, у которого ветви каждого узла упорядочены. Упорядоченные деревья степени 2 называются бинарными деревьями. Бинарное дерево - это упорядоченное дерево, в котором каждый узел связан не более, чем с двумя деревьями. Эти деревья называют левыми и правыми поддеревьями.

Идеально сбалансированное дерево

Дерево называется идеально сбалансированным, если для каждого из узла количество узлов в левом и правом поддереве различается не более, чем на один.

Что достичь максимальной высоты при данном числе узлов нужно располагать максимально возможное число узлов на всех уровнях, кроме самого нижнего. Это можно сделать очень просто, если распределить узлы поровну слева и справа от каждого узла.

```
{построение идеально сбалансированного бинарного дерева для  n -узлов}
Procedure Tree (n: integer; Var p: point);
Var r: point;
   Nl, nr: integer;
Begin
If n = 0 then {это лист} p: = nil
Else   begin
Nl: = nd1v2; nr: = n – n1 – 1;
{взять один узел в качестве корня}
new ( r); read (r^. Data);
```

```
tree (nl, left): {построить левое поддерево}
tree (nr, left):  {построить правое поддерево}
{включить узел в дерево}
p: = r
  end
End;
```

Обратите внимание, что узлы включаются в дерево при обратном ходе в порядке, обратном их формированию, т.е. снизу вверх (от листьев к корню). И все это достигается одим оператором p: = r.

Обход дерева

Имеется много задач, которые можно выполнять на дереве; распространенная задача – выполнения заданной операции P с каждым элементом дерева. Здесь P рассматривается как параметр более общей задачи посещения всех узлов, или, как это обычно называют, обход дерева.

При обходе дерева его узлы посещаются в определенном порядке и могут считаться расположенными линейно. Можно говорить о переходе к следующему узлу, имея ввиду некоторое упорядочение.

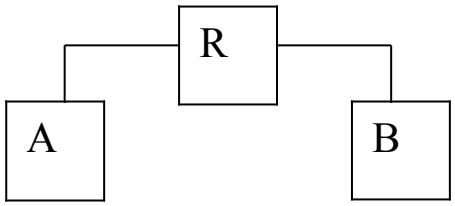


Рис. 2

Существует три принципа упорядочения, которые естественно вытекают из структуры дерева Рис 2.

Сверху вниз: R, A, B (посетить корень, обойти левое поддерево, обойти правое поддерево).

Слева направо: A, R, B (обойти левое поддерево, посетить корень, обойти правое поддерево).

Снизу вверх: A, B, R (обойти левое поддерево, обойти правое поддерево, посетить корень).

Обходя дерево на рис.1 и выписывая символы, находящиеся в узлах в том порядке, в котором они встречаются мы получим следующие последовательности:

```
сверху вниз - +a*bc/d+ab
слева  направа  a+b*c-d/a+b
снизу вверх abc*dab+/-
{обход дерева сверху вниз}
Procedure Preorder  (r: point);
Begin
   If r<>nil then
         Begin
               P ( r ); {посетить узел r}
               Preorder (r^. Left); {обойти левое поддерево}
               Preorder (r^.Right); {обойти правое поддерево}
```

```
            End
    End;

    {обход дерева слева направо}
    Procedure Inorder (r: point)
    Begin
       If r<> nil then
            Begin
                Inorder (r^. Left); {обойти левое поддерево}
                P( r ); {посетить узел}
                Inorder (r^. Right); {обойти правое поддерево}
            End
    End.

    {обход дерева снизу вверх}
    Procedure Postorder (r: point)
    Begin
       If r<> nil then
            Begin
                Postorder (r^. Left); {обойти левое поддерево}
                Postorder(r^.Right);{обойти правое поддерево}
                P( r ); {посетить узел}
            End
    End.
```

Теперь напишем процедуру которая печатает дерево, обходя его слева направо. Для этого надо просто заменить операцию P процедурой печати. Для наиболее наглядной печати потребуем, чтобы дерево печаталось с соответствующим сдвигом, выделяющим каждый уровень узлов.

```
    {печать дерева обходя его слева направо}
    {вначале печатается левое поддерево, затем узел, который выделяется
предшествующими l пробелами, затем правое поддерево}
    Procedure PrintTree (r: point; l: integer);
    Var  i: integer;
    Begin
       If   r<>nil   then
            Begin
                PrintTree (r^.left, l+1);
                For   i:=1 to  l  do  write (‘ ‘);
                Writeln (r^. Data);
                PrintTree (r^. Right, l+1)
            End;
    End.
```

Дерево поиска

Бинарные деревья часто используются для представления множества данных, элементы которого ищутся по уникальному ключю. Если дерево организовано таким образом, что для каждого узла P все ключи в левом поддереве меньше ключа P , а ключи в правом поддереве больше ключа P ,то это дерево называется деревом поиска. В дереве поиска можно найти место каждого ключа, двигаясь от корня и переходя на левое или правое поддерево каждого узла в зависимости от его ключа.

```
    {инициализация дерева}
    Procedure  IniTree (Var  p: pointer; k : char);
```

```
Begin
   New(p); p^.data :=k; p^.left:=nil; p^.right:=nil;
End.

{добавить узел слева}
Procedure  Inleft  (Var  p: pointer; k : char);
Var  r : point;
Begin
   New(r); p^.data := k; r^.left:= nil; r^.right:= nil;
   p^.left := r;
End.

{добавить узел справа}
Procedure  Inright (Var  p: pointer; k : char);
Var   r : point;
Begin
   New(r); p^.data := k; r^.left:= nil; r^.right:= nil;
   p^.right:= r;
End.
```

Для построения дерева поиска используем следующий алгоритм.Первый элемент помещаем в корень (инициализируем дерево). Далее поступаем следующим образом. Если добавленный в дерево элемент имеет ключ в больший, чем ключ узла, то, если узел не лист, обходим его справа. Если добавленный элемент имеет ключ не больший чем ключ узла, то, если узел не лист, обходим его слева.

Если дошли до узла, то добавляем элемент соответственно справа или слева.

```
{включить элемент в дерево}
Procedure  Tree (Var  p: pointer; k : char);
Var   ok : boolean;
Begin
   Ok : false ;
   While   not  ok   do
  Begin
   If  k>p^.data   then  {посмотреть направо}
   If  p^.right <>nil  {правый узел  не лист}
   Then  p: = p^. Right. {обход справа}
   Else begin {правый узел – лист и надо добавить к нему  элемент}
            InRight (p,k); {и конец};
 ok: = true;
 end;
                  else {посмотреть налево}
                  if  p^. Left <> nil {левый узел не лист}
                  then  p:=p^.left   {обход слева}
                  else   begin{левый узел –лист и надо добавить к нему элемент}
Inleft(p,k);{и конец} ok :=true  end
            end  {цикла while}

   end.
```

Поиск дерева с включением

Хорошим примером использования дерева поиска является задача построения частотного словаря. В этой задаче задана последовательность слов и нужно установить число появлений каждого слова. Начиная с пустого дерева каждое слово ищется в дереве. Если оно найдено увеличивается счетчик его появлений, если нет в дерево добавляется новое слово с начальным значением счетчика, равным единице. Будем называть эту задачу поиском по дереву с включением.

Процедура поиска включения запишется следующим образом.

```
{поиск по дереву с включением}
Procedure Search (Var p: point; x: integer);
Begin
   If   p = nil  then {новое слово}
   Begin {оно добавляется в дерево}
   New (p), p^.  Left: = nil;  p^. Right: = nil;
         P^. Key: = x; p^. Count: = 1;
      End;
   Else  if  x> p^. Key then {слово ищется в правом поддереве}
   Search (p^. Right. x)
         Else  if  x<p^.  Key  then  {слово ищется в левом поддереве}
         Search (p^. left. x)
         Else {слово найдено и его счетчик увеличивается на 1}
         P^. Count: = p^. Count + 1
   End;
```

Удаление из дерева

Переходим к задаче, обратной включению, а именно удалению. Необходимо построить алгоритм для удаления узла с ключем x из дерева с упорядоченными ключами (дерево поиска).

Удаление узла довольно просто если он является листом или имеет одного потомка. Например, если требуется удалить узел с ключем M надо просто заменить правую ссылку узла K на указатель на L. Трудность заключается в удалении узла с двумя потомками, поскольку мы не можем указать одним указателем на два направления.

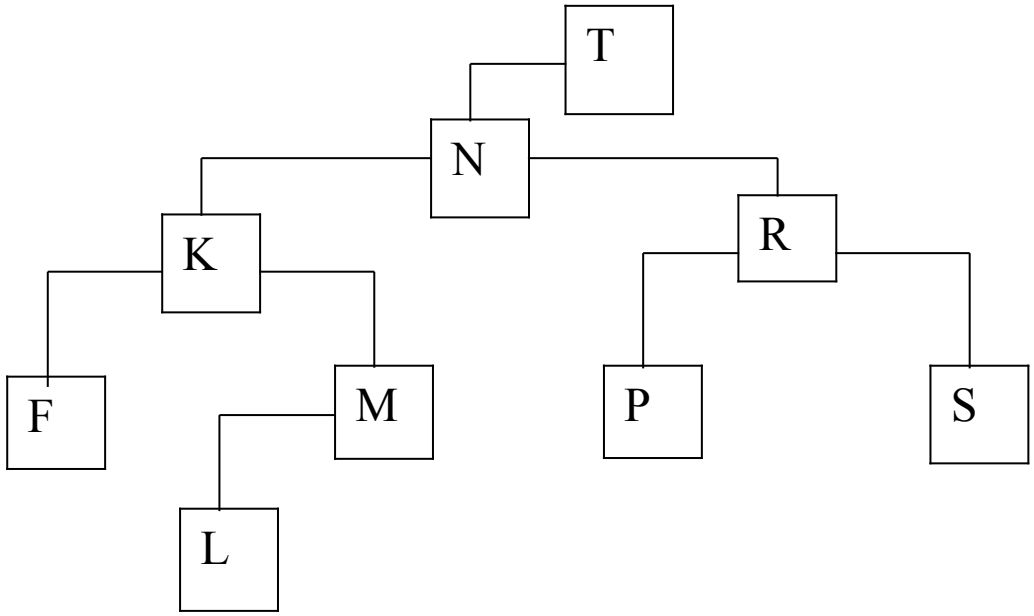


Рис . 3

Например, если просто удалить узел с ключем N, то левый указатель узла с ключем T должен указывать одновременно на K и R что не возможно. В этом случае удаляемый узел нужно заменить на другой узел из дерева. Возникает вопрос, каким же узлом его заменить? Этот узел должен обладать двумя свойствами: во-первых, он должен иметь не более одного потомка; во-вторых, для сохранения упорядоченности ключей, он должен иметь ключ либо не меньший, чем любой ключ левого поддерева удаляемого узла, либо не больший, чем любой ключ правого поддерева удаляемого узла. Таким свойствам обладают два узла, самый правый узел левого поддерева удаляемого узла и самый левый узел его правого поддерева. Любым из этих узлов им можно заменить удаляемый узел. Например, на рис.3 это узлы M и P.

Необходимо различать три случая:

1. Узла с ключем, равным x, нет.
2. Узел с ключем, равным x, имеет не более одного потомка.
3. Узел с ключем, равным x, имеет двух потомков

```
{удаление из дерева}
Procedure Delete (Var p: point; x: char);
Var  q:  point;
Procedure Del (Var r: point);
{процедура удаляет узел имеющий двух потомков, заменяя его на самый правый узел левого поддерева}
Begin
If  r^. Right <> nil then {обойти дерево справа}
Del (r^. Right)
Else {дошли до самого правого узла}
Begin {заменить этим узлом удаляемый}
q^. Data: = r^. Data; q: = r; r: = r^. Left;
            end;
   End;  {del}
Begin {delete}
If p<> nil then {искать удаляемый узел}
If x<p^. Data then {искать удаляемый узел}
```

```
            Delete (p^. Left, x)
            Else  if x>p^. Data   {искать в правом поддереве}
            Delete (p^. Right, x)
            Else {узел найден, надо его удалить}
            Begin {сохранить ссылку на удаленный узел}
                  q: = p;
                  if  q^. Right = nil  then
```
{узел имеет не более одного потомка (слева) и ссылку на узел надо заменить ссылкой на этого потомка}
```
            p: = q^. Left
            else  if  q^. Left = nil  thent
```
{узел имеет не более одного потомка (справа) и ссылку на узел надо заменить ссылкой на этого потомка}
```
            p: = p^. Right
            else {узел имеет двух потомков}
            del ( q^. Left);
            dispose (q);
      End.
```

Вспомогательная рекурсивная процедура del вызывается только в случае, когда удаляемый узел имеет двух потомков. Она “спускается вдоль” самой правой ветви левого поддерева удаляемого узла q^ (при вызове процедуры ей передается в качестве параметра указатель на левое поддерево) и, затем, заменяет существенную информацию (в нашем случае ключ data) в q^ соответствующим значением самого правого узла r^ этого левого поддерева, после чего от r^ можно освободиться.

Процедура disposi (q) освобождает память занимаемую удаляемую узлом. Если узел требуется удалить только из дерева, но оставить в памяти процедура disposi не выполняется, а переменную q надо объявить глобальной.

10.4. Стеки, очереди.

Стек - это линейный список с определенной дисциплиной обслуживания, которая заключается в том, что элементы списка всегда включаются, выбираются и удаляются с одного конца, называемого вершиной стека. Доступ к элементам здесь происходит по принципу “ последним пришел - первым ушел”(LIFO - last in first out), т. е. последний включенный в стек элемент первым из него удаляется. Стеки моделируются на основе линейного списка. Включение элемента вершины стека называется операцией проталкивания в стек, а удаление элемента из вершины стека называется операцией выталкивания из стека.

Для работы со стеками необходимо иметь один основной указатель на вершину стека (Top) и один дополнительный временный указатель (P), который используется для выделения и освобождения памяти элементов стека.

Создание стека:

Исходное состояние

Ton : = nil

2. Выделение памяти под первый элемент стека и внесение в него информации

```
New (P)
P^. Inf : = S
P^. Link: = nil
```

Установка вершины стека Top на созданый элемент

Top: = P

Добавление элемента стека:

1. Выделение памяти под новый элемент

New (P)

2. Внесение значения в информационное поле нового элемента и установка связи между ним и старой вершиной стека Top

P^. Inf: = Val (Val=10)

P^. Link: = Top.

Помещение вершины стека Top на новый элемент

Top: = P

Удаление элемента стека

Извлечение информации из информационного поля вершины стека Top в переменную Val и установка на вершину стека вспомогательного указателя P.

Val: = Top^. Inf;

P: = Top;

Перемещение указателя вершины стека Top на следующий элемент и освобождение памяти, занимаемой «старой» вершиной стека

Top: - P^. Lin K;

Pispose (P)

В качестве примера приведем программу создания и удаления стека из десяти элементов:

```
Pvogram Stack;
Uses Crt
Tupe
        TP tr = ^Telem;
        Telem = record
                Inf: Real;
                Link: TPtr
        End;
Var
        Top: TPfr;
        Value: Real;
        Value: Byte;
Procedure Push (Val: Real)
Var
        P: TPfr;
Begin
        New (P);
        P^.Inf: = Val;
        P^.Link: = Top
        Top: = P
End;
Procedure Top (Var  Val:Real);
        Var
                P: TPtr;
        Begin
                Val: = Top^. Inf;
                P: = Top;
                Top: = P^. Link;
                Pispose (P)
        End;
Begin
        ClrSer;
```

```
            {начальные установки указателей}
        Top: = nil
            {создание стека из десяти элементов}
        for i: = 1 to 10 do  Push (i);
{удаление стека с распечаткой значений его элементов}
        whise Top <> nil do
        begin
            Pop (Value);
            Writeln ('Value= ' , Value = 5:2)
        End;
End.
```

Очередь - это линейный список, в котором элементы включаются с одного конца, называемого хвостом, а выбираются и удаляются с другого конца, называемого вершиной. Дисциплина обслуживания очереди- "первым пришел - первым вышел" (FIFO - first in first out), т. е. первый включенный в очередь элемент первым из нее и удаляется.

Очередь – это частный случай линейного односвязующего списка для которого разрешены только два действия: добавление элемента в конец очереди и удаление элемента из начала очереди.

Для создания очереди и работы с ней необходимо иметь как минимум два указателя:

на начало очереди (возьмем идентификатор Beg Q)

на конец очереди (возьмем идентификатор End Q)

Кроме того, для освобождения памяти удаляемых элементов требуется дополнительный временный указатель (P).

Создание очереди:

Исходное состояние:

```
Beg Q: = nil
Eng Q: = nil
```

Выделение памяти под первый элемент

```
New (P)
```

Занесение информации в первый элемент очереди

```
Beg Q: = P
Eng Q: = P
```

Добавление элемента очереди

Выделение памяти под новый элемент и занесение в него информации:

```
New (P)
P^, Inf: = 5
P^, Link: = hil
```

Установка связи между последним элементом очереди и новым, а также перемещение указателя конца очереди End Q на новый элемент

```
End^. Link: = P
End Q: = P
```

Удаление элемента очереди

Извлечение информации из удаляемого элемента в переменную Val и установка на него вспомогательного указателя P

```
Val: = Beg Q^. Inf
   P: = Beg Q
```

Перестановка указателя начала очередт Beg Q на следующий элемент используя значение поля Link, которое хранится в первом элементе. После этого освобождается память начального элемента очереди, используя дополнительный указатель P:

```
Beg Q: = P^. Link
Dispose (P)
```

В качестве примера приведем программу создания и удаления очереди из десяти элементов:

```
    Prodrsm Queue;
    Uses Crt;
    Type
        TPtr = ^Telem;
        TFlem = record
            Inf: Real;
            Link: Tptr
        End;
  Var
        Beg Q, End Q : TP tr;
        Value        : Real;
        i            : Byte;
  Procedure  AddEl  (Val:Real)
  {создает первый и дабавляет очередной элемент в конец очереди}
        Var
            P: TPtr
        Begin
            New (P);
            P^. Inf: = Val;
            P^. Link: = nil;
            If   End Q = n:l ;   {если создается первый элемент очереди}
            Then Beg Q: = P {если создается очередной элемент очереди}
            Else Eng Q^. Link: = P;
            End Q: = P;
  End;
    Procedure  bet Del E1 (vav Val: Real)
  {извлечение информации из начального элемента очереди с последующим освобождением его памяти}
        Var
            P: TPtr;
        Begin
            Val: = Reg Q^. Inf;
            P: = Beg Q;
            Beg Q: = P^. Link
            If Beg Q = nil {если удаляется последний элемент очереди}
            Then Eng Q: = nil;

            Dispose (P);
    End;

  Begin
    ClrScr;
    {начальные установки указателей}
    Beg Q: = nil;
    Eng Q: = nil;
    {создание очереди из 10 элементов}
        for I: = 1 to 10 to Add E1 (i)
    {удаление очереди с распечаткой значений её элементов}
            while. Beg Q <> nil do
            begin
```

```
                    Get Del E1 (Value);
                    Writeln ('Value=' ,  Value  : 5: 2)
          End;
End.
```

Образец выполнения работы.

Лабораторная работа № 16.

Работа со стеками и очередями.

Часть I

Используя очередь или стек (считать уже описанными их типы и операции над ними) описать процедуру или функцию, которая :

Находит в непустом дереве T длину пути от корня до ближайшей вершины с элементом E, если E не входит в T , то за ответ принять -1.

Текст программы:

```
Program T854a;

Uses CRT;
Const
   E='C';
Type
   Tree=^Root;
   Root=Record
       Element:Char;
       Left,Right:Tree;
       End;

  Stack=^Description;
  Description=Record
       BTree:Tree;
       Process:boolean;
       Next:Stack;
       End;

Var
  T:Tree;
  S:Stack;
{создает поддерево с двумя листьями}
Procedure SubTreeBuilding(var P:Tree);
Var
 TLeft,TRight:Tree;
```

```
Begin

  New(P);
  New(TLeft);
  New(TRight);
  TLeft^.Left:=nil;
  TLeft^.Right:=nil;
  TRight^.Left:=nil;
  TRight^.Right:=nil;
  TLeft^.Element:=Chr(64+Random(28));
  TRight^.Element:=Chr(64+Random(28));

  P^.Element:=Chr(64+Random(28));
  P^.Left:=TLeft;
  P^.Right:=TRight;

End;


{создает дерево}
Procedure TreeBuild;
Begin
 Randomize;
 SubTreeBuilding(T);
 SubTreeBuilding(T^.Left);
 SubTreeBuilding(T^.Right);

 SubTreeBuilding(T^.Left^.Left);
 SubTreeBuilding(T^.Right^.Left);
 SubTreeBuilding(T^.Left^.Right);
 SubTreeBuilding(T^.Right^.Right);

 SubTreeBuilding(T^.Left^.Left^.Left);
 SubTreeBuilding(T^.Right^.Left^.Left);
 SubTreeBuilding(T^.Left^.Right^.Left);
 SubTreeBuilding(T^.Right^.Right^.Left);
 SubTreeBuilding(T^.Left^.Left^.Right);
 SubTreeBuilding(T^.Right^.Left^.Right);
 SubTreeBuilding(T^.Left^.Right^.Right);
 SubTreeBuilding(T^.Right^.Right^.Right);

 SubTreeBuilding(T^.Left^.Left^.Right^.Right);
 SubTreeBuilding(T^.Right^.Left^.Right^.Right);

 {T^.Right^.Left^.Right^.Right^.Right^.Element:='E';}
End;


 {*************   Функции и процедуры работы со стеком   *****************}


{** Процедура создания и очистки стека **}
```

```
Procedure InitStack(var S1:Stack);
var
  P1,P2:Stack;
Begin
    While S1<>nil Do Begin
      P1:=S1^.Next;
      Dispose(S1);
      S1:=P1;
    End;
End;

{** Процедура проталкивания элемента в стек **}
Procedure Shoot(E:Tree);
var
   P:Stack;
Begin
New(P);
P^.BTree:=E;
P^.Process:=false;
P^.Next:=S;
S:=P
End;


{ **Функция выталкивания элементов из стека **}
Procedure Pull(var S1:Stack);
var
   P:Stack;
Begin
 If S1<>nil Then Begin
   P:=S1;
   S1:=S1^.Next;
   Dispose(P);
   P:=nil;
   End
 Else
   S1:=nil;
End;


{Функция определения размера стека}
Function Detect_Stack_Size(S1:Stack):Integer;
var
   i:Integer;
Begin
  i:=0;
  While S1<>nil Do begin
     inc(i);
     S1:=S1^.Next;
  End;
  Detect_Stack_Size:=i;
End;
```

```
{находит длину пути от корня до ближайшей вершины с элементом E}
Procedure Count_Waypoints(T1:Tree);
var
  i:Integer;
  P:Tree;
Begin
 InitStack(S);
 Shoot(T1); {проталкиваем в стек корень дерева}
 i:=0;

                {****** Цикл обхода дерева ********}
 Repeat
 inc(i);
 P:=S^.Btree; {P-узел на верхушке стека}

   {** Если текущий элемент стека - лист **}
   If ((P^.Left=nil) and (P^.Right=nil) and (Detect_Stack_Size(S)>0)) Then Begin
     Pull(S);
     S^.Process:=true;
       {Если данный лист - не правый сын своего отца}
       If P<>S^.BTree^.Right Then Begin
          S^.Process:=true;
          Shoot(S^.BTree^.Right); {проталкиваем в стек его правого брата}
       End
   End
   Else Begin {** элемент на верхушке стека-промежуточный узел **}
       If (not S^.Process) Then Begin {данное дерево еще не обработано}
          S^.Process:=false;
          Shoot(P^.Left); {протолкнуть в стек левого сына}
       End
       Else Begin {данное дерево уже обработано}
          Pull(S);
          If ((Detect_Stack_Size(S)>0) And (P<>S^.Btree^.Right)) {элемент-не корень}
          Then Begin                    {и не правый сын своего отца}
            S^.Process:=true;
            Shoot(S^.Btree^.Right);  {проталкиваем правого брата}
          End;
       End;
   End;
 Until ((Detect_Stack_Size(S)=0) or (S^.Btree^.Element=E)) ;

 If ((Detect_Stack_Size(S)=0) And (S^.Btree^.Element<>E)) Then
   WriteLn('Указанное значение "',E,'" в дереве не найдено, результат равен "1"')
 Else
 WriteLn('длина пути к элементу со значением "',E,'" - ',Detect_Stack_Size(S)-1,'
узла(ов)');

 WriteLn('Всего произведено ',i,' перемещений по узлам дерева');
 End;
```

```
Begin
TreeBuild;
WriteLn;
WriteLn('Результат обхода дерева:');
Count_Waypoints(T);
WriteLn;
WriteLn('Press any key ...');
Repeat Until Keypressed;
End.
```

Результат работы программы:

```
Результат обхода дерева:
длина пути к элементу со значением "C" - 2 узла(ов)
Всего произведено 15 перемещений по узлам дерева

Press any key ...
```

Часть II

Написать рекурсивную функцию или процедуру, которая:

Определяет максимальную глубину непустого дерева T, т. е. число ветвей в самом длинном из путей от корня дерева до листьев;

Текст программы T854b:

```
Program Task854b;

Uses CRT;

Type
   Tree=^Root;
   Root=Record
        Element:Char;
        Left,Right:Tree;
        End;

Var
  T:Tree;
  Depth,n:Integer;

{создает поддерево с двумя листьями}
Procedure SubTreeBuilding(var P:Tree);
Var
 TLeft,TRight:Tree;
```

```
Begin
  New(P);
  New(TLeft);
  New(TRight);

  P^.Element:=Chr(64+Random(28));
  TLeft^.Left:=nil;
  TLeft^.Right:=nil;
  TRight^.Left:=nil;
  TRight^.Right:=nil;
  TLeft^.Element:=Chr(64+Random(28));
  TRight^.Element:=Chr(64+Random(28));

  P^.Left:=TLeft;
  P^.Right:=TRight;
End;


Procedure Tree_Build;
Begin
 Randomize;

 SubTreeBuilding(T);

 SubTreeBuilding(T^.Left);
 SubTreeBuilding(T^.Right);

 SubTreeBuilding(T^.Left^.Left);
 SubTreeBuilding(T^.Right^.Left);
 SubTreeBuilding(T^.Left^.Right);
 SubTreeBuilding(T^.Right^.Right);

 SubTreeBuilding(T^.Right^.Left^.Left);
 SubTreeBuilding(T^.Right^.Left^.Left);
 SubTreeBuilding(T^.Left^.Right^.Left);
 SubTreeBuilding(T^.Right^.Right^.Left);
 SubTreeBuilding(T^.Left^.Left^.Right);
 SubTreeBuilding(T^.Right^.Left^.Right);
 SubTreeBuilding(T^.Left^.Right^.Right);
 SubTreeBuilding(T^.Right^.Right^.Right);

 SubTreeBuilding(T^.Left^.Left^.Right^.Right);
 SubTreeBuilding(T^.Right^.Left^.Right^.Right);

 SubTreeBuilding(T^.Right^.Left^.Right^.Right^.Left);

 SubTreeBuilding(T^.Right^.Left^.Right^.Right^.Left^.Right);

End;


procedure Detect_Tree_Depth(r: tree; Layer: Integer);
```

```
begin
   if r<>nil then
   begin
     Detect_Tree_Depth(r^.left, Layer+1);
     If Layer>Depth Then Depth:=Layer;
     Detect_Tree_Depth(r^.right, Layer+1);
     If Layer>Depth Then Depth:=Layer;
     inc(n);
   end;
end;


Begin
Tree_Build;
Depth:=0;
n:=0;
Detect_Tree_Depth(T,0);
WriteLn;
WriteLn('Максимальная глубина заданного дерева ',Depth,' узла(ов)');
WriteLn('Всего дерево содержит ',n,' узла(ов)');
WriteLn;
WriteLn('Press any key...');
Repeat until Keypressed;
End.
```

Результат работы программы:

```
Максимальная глубина заданного дерева 7 узла(ов)
Всего дерево содержит 37 узла(ов)

Press any key...
```

Лабораторная работа № 16.

Работа со стеками и очередями.

Варианты заданий.

Как упоминалось ранее, для работы с очередью нужны следующие операции:

- создать пустую очередь (очистить очередь);
- проверить, является ли очередь пустой;
- добавить в конец очереди элемент;
- удалить из очереди первый элемент.

1. Необходимо для каждого из указанных ниже представлений очереди описать соответствующий тип очередь, считая, что все элементы очереди имеют некоторый тип, и реализовать в виде процедур и функций перечисленные операции над очередью.

Представление очереди (n - целая константа больше 1):

1) для каждой очереди отводится свой массив из n компонентов некоторого типа, в котором элементы очереди занимают группу соседних компонентов, индексы первой и последней из которых запоминаются: при этом, когда очередь достигает правого края массива, все элементы сдвигаются к левому краю (рис 4,а);

2) аналогичное представление. но массив как бы склеивается в кольцо, поэтому, если очередь достигает правого края массива, то новые элементы записываются в начало массива (рис 4, б);

3) для каждой очереди создается сво однонаправленный список из элементов некоторого типа, при этом запоминаются ссылки на первое и последнее звенья списка (рис 4,в).

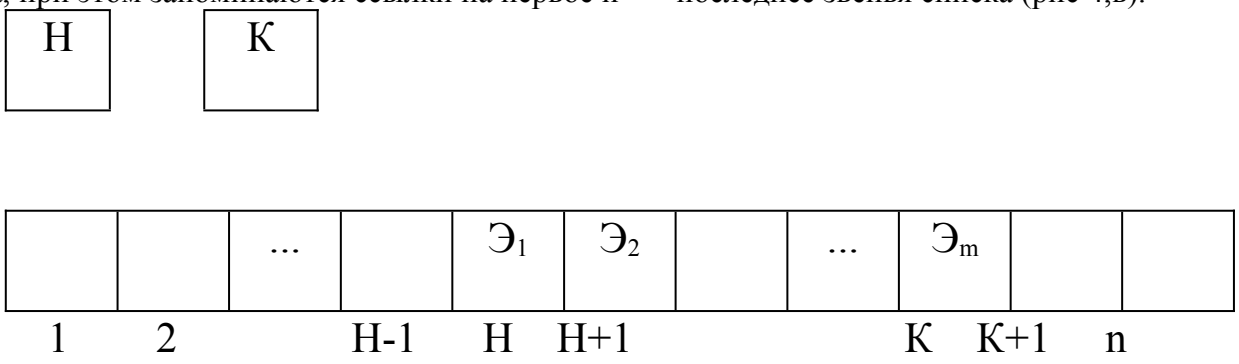


Рис. 4, а

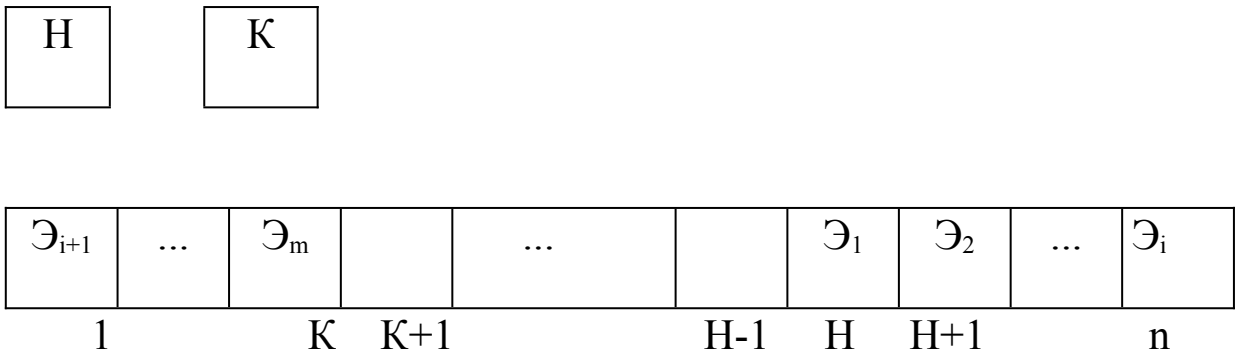


Рис. 4, б

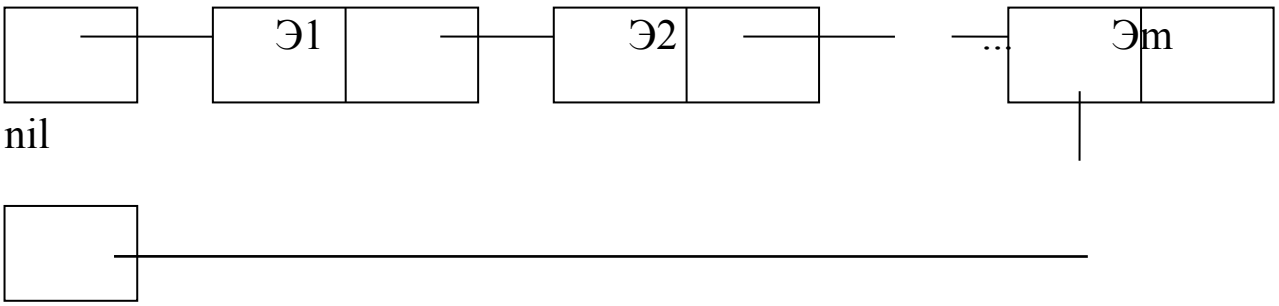


Рис. 4, в

2. Используя очередь (считать уже описанным тип очередь при подходящем типе и всех описанных ранее операций для работы с очередью) решить следующую задачу (решение записать в виде процедуры):

4) Type FR=file of real;

За один просмотр файла f типа FR и без использования дополнительных файлов напечатать элементы файла f в следующем порядке: сначала все числа меньше a, затем все числа из отрезка [a,b], и наконец все остальные числа, сохраняя исходный взаимный порядок в каждой из этих трех групп чисел (a и b заданные числа, a<b).

5) содержимое текстового файла f, разделенное на строки, переписать в текстовый файл g, перенося при этом в конец каждой строки все входящие в нее цифры (с сохранением исходного взаимного порядка как среди цифр, так и среди остальных литер строки).

6) type имя=(Анна,..., Яков);
дети=array[имя,имя]of boolean;

потомки=file of имя;

Считая заданным имя И и массив Д типа дети (Д[x,y]=true, если человек по имени y является ребенком человека по имени x), записать в файл П типа потомки имена всех потомков человека с именем И в следующем порядке:

- сначала имена всех его детей;
- всех его внуков;
- всех правнуков и т. д.

Как было сказано ранее, для работы со стеком обычно нужны следующие операции:

- создать пустой стек (очистить стек);
- проверить является ли стек пустым;
- добавить в конец стека элемент;
- удалить из стека последний элемент.

3. Требуется для каждого из указанных ниже представлений стека описать соответствующий тип стек , считая , что все элементы стека имеют некоторый тип, и реализовать в виде процедур и функций перечисленные операции над стеком.

Представление стека (n - целая константа больше 1):

7) для стека отводится свой массив из n компонентов некоторого типа, в начале которого располагаются элементы стека, при этом запоминается индекс компонента массива, занятый последним элементом стека.

8) для каждого стека создается свой однонаправленный список, в котором элементы стека располагаются в обратном порядке.

4. Используя стек (считать уже описанным тип стек с элементами типа char, функцию проверки пустоты стека, процедуры создания, добавления и удаления) решить следующую задачу (решение записать в виде процедуры или функции).

9) напечатать содержимое текстового файла t, выписывая литеры каждой его строки в обратном порядке.

10) проверить, является ли содержимое текстового файла t правильной записью формулы следующего вида:

<формула>::=<терм>|<терм>+<формула>|
<терм>-<формула>
<терм>::=<имя>|(<формула>)|[<формула>]
<имя>::=x|y|z

11) в текстовом файле f записана без ошибок формула следующего вида:

<формула>::=<цифра>|M<формула>,<формула>|
m<формула>,<формула>
<цифра>::=0|1|2|3|4|5|6|7|8|9,

где M обозначает функцию max, m - min.

Вычислить (как целое число) значение данной формулы (например, M(5,m(6,8))=6).

12) в текстовом файле записано без ошибок логическое выражение (ЛВ) в следующем виде:

<ЛВ>::=true| false |(\<ЛВ>)| (<ЛВ>^<ЛВ>)| (<ЛВ>V<ЛВ>),

где знаки \ , ^ , V обозначают соответственно отрицание, конъюнкцию, дизъюнкцию.

Вычислить (как boolean) значение этого выражения.

5. Используя очередь и/или стек (считать уже описанными их типы и операции над ними) решение описать в виде процедуры).

В текстовом файле записан текст, сбалансированный по круглым скобкам :

<текст>::=<пусто>|<элемент><текст>
<элемент>::=<буква>|<текст>

Требуется для каждой пары соответствующих открывающей и закрывающей скобок напечатать номера их позиций в тексте , упорядочив пары номеров в порядке возрастания номеров позиций

13)закрывающих скобок;

14)открывающих скобок.

Например, для текста A+(45-F(X)*(B+C)) надо напечатать:

13) 8 10; 12 16; 3 17;

14) 3 17; 8 10; 12 16;

Под <выражением> будем понимать конструкцию следующего вида

<выражение>::=<терм>|<терм><знак+-><выражение>

<знак+->::=+|-

<терм>::=<множитель>|<множитель>*<терм>

<множитель>::=<число>|<переменная>|(<выражение>)|

<множитель>^<число>

<число>::=<цифра>

<переменная>::=<буква>,

где знак ^ обозначает возведение в степень.

Постфиксной формой записи выражения a^b называется запись, в которой знак операции размещен за операндами: ab^.

Например:

a+b-c это ab+c-

a*b+c это ab*c+

15) описать функцию value(postfix), которая вычисляет как целое число значение выражения (без переменных), записанного в постфиксной форме в текстовом файле postfix.

Использовать следующий алгоритм вычисления:

Выражение просматривается слева направо. Если встречается операнд (число), то его значение (целое) заносится в стек, а если встречается знак операции , то из стека исключаются два последних элемента (это операнды данной операции) над ними выполняется операция и ее результат записывается в стек. В конце концов в стеке остается только одно число - значение всего выражения.

16) описать процедуру translate (infix, postfix), которая переводит выражение, записанное в обычной (инфиксной) форме в текстовом файле infix в постфиксную форму и в таком виде записывает его в текстовый файл postfix. Если встречается операнд (число или переменная), то он сразу переносится в файл postfix. Если встречается открывающая скобка, то она заносится в стек, а если встречается закрывающая скобка, то из стека извлекаются находящиеся там знаки операций до ближайшей открывающей скобки, которая тоже удаляется из стека, и все эти знаки в (порядке извлечения) записываются в файл postfix. Когда же встречается знак операции, то из конца стека извлекаются (до ближайшей скобки, которая сохраняется в стеке) знаки операций, старшинство которых больше или равно старшинству данной операции, и они записываются в файл postfix, после чего рассматриваемый знак заносится в стек. В заключение выполняются такие же действия, как если бы встретилась закрывающая скобка.

17) описать (нерекурсивную) процедуру infixprint(postfix), которая печатает в обычной (инфиксной) форме выражение, записанное в постфиксной форме в текстовом файле postfix.

18) описать (нерекурсивную) процедуру postfixprint(infix), которая печатает в постфиксной форме выражение, записанное в инфиксной форме в текстовом файле infix.

Для решения следующих задач использовать двоичные деревья при следующем их описании:

type тип= некоторый тип элементов дерева;

дерево=^вершина;
вершина=record элем: тип элементов дерева;
лев, прав: дерево end;

В задачах T, T1, T2 обозначают деревья, E - величину некоторого типа элементов дерева.

6. Используя очередь или стек (считать уже описанными их типы и операции над ними) описать процедуру или функцию, которая :

1) присваивает параметру E элемент из самого левого листа непустого дерева T;

2) определяет число вхождений элемента E в дерево T;

3) вычисляет среднее арифметическое всех элементов непустого дерева T (тип элементов дерева real);

4) заменяет в дереве T все отрицательные элементы на их абсолютные значения (тип элементов дерева real);

5) меняет местами максимальный и минимальный элементы непустого дерева T, все элементы которого различны (тип элементов дерева real);

6)печатает все элементы из всех листьев дерева T (тип элементов дерева char);

7) печатает все элементы дерева T по уровням: сначала - из корня дерева, затем (слева направо) - из вершин дочерних по отношению к корню, затем (слева направо) - из вершин дочерних по отношению к этим вершинам и т. д. (тип элементов дерева integer);

8) находит в непустом дереве T длину пути от корня до ближайшей вершины с элементом E, если E не входит в T , то за ответ принять -1.

9) подсчитывает число вершин на n-ом уровне непустого дерева T (корень считать вершиной нулевого уровня).

7. Написать рекурсивную функцию или процедуру, которая:

10) определяет, входит ли элемент E в дерево T;

11) определяет число вхождений элемента E в дерево T;

12) вычисляет сумму всех элементов непустого дерева T (тип элементов дерева real);

13) находит величину наибольшего элемента непустого дерева T (тип элементов дерева real);

14)печатает все элементы из всех листьев дерева T (тип элементов дерева char);

15) определяет максимальную глубину непустого дерева T, т. е. число ветвей в самом длинном из путей от корня дерева до листьев;

16) подсчитывает число вершин на n-ом уровне непустого дерева T (корень считать вершиной нулевого уровня);

17) Рекурсивно и нерекурсивно описать логическую функцию equal(T1, T2), проверяющую на равенство деревья T1 и T2;

18) описать процедуру copy(T, T1) , которая строит дерево T1 - копию дерева T;

19) описать логическую функцию same (T), которая определяет , есть ли в дереве T хотя бы два одинаковых элемента;

Опишем следующий вид дерева, который можно назвать “деревом-формулой”. Формулу вида:

<формула>::=<терминал>|<формула><знак><формула>

<знак>::=+|-|*

<терминал>::=0|1|2|3|4|5|6|7|8|9

можно представить в виде двоичного дерева с типом элементов дерева = char согласно следующим правилам: формула из одного терминала (цифры) представляется деревом из одной вершины с этим терминалом, а формула вида (f_1 s f_2) - деревом, в котором корень это знак s, а левое и правое поддеревья - это соответствующие представления формул f_1 и f_2. На рис. 5 показано дерево-формула, соответствующее формуле(5*(3+8))).

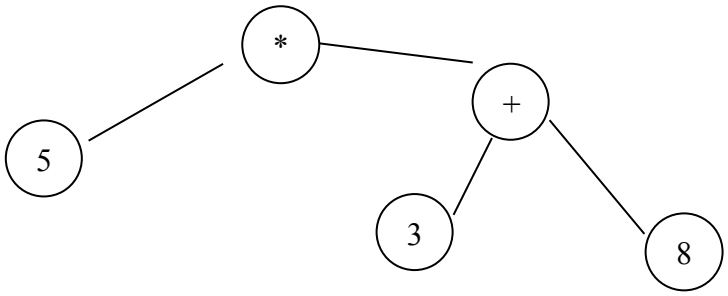


Рис. 5

Описать рекурсивную функцию или процедуру, которая:

20) вычисляет (как целое число) значение дерева-формулы T;
21) по формуле из текстового файла F строит соответствующее дерево- формулу T;
22) печатает дерево- формулу T в виде соответствующей формулы;
23) проверяет, является ли двоичное дерево T деревом формулой.

11. Организация меню с использованием средств среды Turbo Pascal

Turbo Pascal предоставляет возможность организации меню с помощью встроенного стандартного модуля **CRT**.

Инициализация модуля **CRT**:

```
Program name;
Uses CRT, {через запятую перечисляются другие нужные в данной программе модули};
Type  ……
      ……
var   ……
      ……
begin
end.
```

В модуле CRT сосредоточены процедуры и функции, обеспечивающие управление текстовым режимом работы экрана. С помощью входящих в модуль программ можно перемещать курсор в произвольную позицию экрана, менять цвет выводимых символов и окружающего их фона, создавать окна. Кроме того, в модуль включены также процедуры “слепого” чтения клавиатуры и управления звуком.

В режиме текстового вывода используются следующие координаты экрана: левый верхний угол экрана имеет координаты 1,1; горизонтальная координата возрастает слева направо, вертикальная -сверху вниз. Если на экране определено окно, все координаты определяются относительно границ окна.

Для чтения клавиатуры используются две функции –**KeyPressed** и **ReadKey**. Функция **KeyPressed** определяет факт нажатия на любую клавишу и не приостанавливает дальнейшего исполнения программы.

Функция **ReadKey** читает расширенный код нажатой клавиши. Если к моменту обращения к функции не была нажата ни одна клавиша, то программа приостанавливает работу, ожидая действия пользователя.

Управления звуковым генератором строится по схеме: **Sound** –**Delay** –**NoSound**. Процедура **Sound(n),** (где **n**-параметр в Герцах **type n:Word**), включает звуковой

генератор и заставляет его непрерывно генерировать звук нужного тона. Процедура **Delay(n)**, (где **n**-параметр в миллисекундах **type n:word**), приостанавливает роботу программы на заданное число миллисекунд реального времени. Процедура **NoSound** отключает звуковой генератор.

Определение текстового окна на экране выполняется с помощью процедуры **Window(x1,y1,x2,y2: byte)**, где x1,y1 –координаты левого верхнего угла, x2,y2 –правого нижнего угла.

Clrscr; - процедура без параметра, очищает текущее окно и придает ему цветовые параметры заданные прежде.

Определение цвета текстового окна на экране выполняется с помощью процедур:
Textcolor(код цвета), цвет символов,
Textbackground(код цвета), цвет фона.

0	Чёрный
1	Синий
2	Зелёный
3	Голубой
4	Красный
5	Фиолетовый
6	Коричневый
7	Тёмно-серый
8	Светло-серый
9	Ярко-голубой
10	Ярко-зелёный
11	Ярко-голубой
12	Розовый
13	Ярко-фиолетовый
14	Жёлтый
15	Белый

Лабораторная работа №17.

Составления меню.

Цель работы:

1. Закрепить полученные теоретические знания и практические навыки.
2. Изучить стандартный модуль TurboPascal CRT.
3. Познакомится с работой с окнами.
4. Освоить некоторые методы сортировки записей.

Постановка задачи:

1. Написать программу, которая формирует меню, состоящее из 9 пунктов
   1. Открыть.
   2. Создать.
   3. Поиск.
   4. Добавить.
   5. Удалить.
   6. Корректировка.
   7. Просмотр.
   8. Сортировка.
   9. Выход.

Каждый пункт должен быть оформлен в виде отдельной процедуры.

2. В каждой процедуре должна быть проверка существования файла.

3. Программа должна работать с данными, приведёнными в конкретном варианте.

Содержание отчёта:

1. Постановка задачи для конкретного варианта.
2. Распечатать исходные данные.
3. Текст программы.
4. Распечатка результатов работы программы после выполнения пунктов меню 4,5,6 и 8.

Образец выполнения работы.

Лабораторная работа № 17.

Составления меню.

Постановка задачи:

Написать программу, которая формирует меню, состоящее из 9 пунктов

1. Открыть.
2. Создать.
3. Поиск.
4. Добавить.
5. Удалить.
6. Корректировка.
7. Просмотр.
8. Сортировка.
9. Выход.

Анкетные данные на абитуриентов в конце методического пособия.

Текст программы:

```
program menu;
uses crt,dos;
const text1='Год рождения.....';
      text2='Год окончания школы...';
      text3='      Оценки в атестате';
      text4='Метематика.....';
      text5='Физика.....';
      text6='Русский язык.....';
      text7='    Оценки на вступительных экзаменах';
      text8='Нуждается в общежитии';
      text9='Не нуждается в общежитии';
type data=record
         fio:string[30];
         godr,godo:integer;
         ates:record
              mat,fiz,rus:integer;
              end;
         haus:boolean;
         ekz:record
            mat,fiz,rus:integer;
            end;
         end;
```

```
     filetype=file of data;
var stu:data;
    files,filetemp:file of data;
    name,path,namefile:string;
    keys:char;
    menuunit:byte;
    rec,position,freeunit:integer;
    quit,openfile:boolean;
procedure sortirovka;
var a,b,c:filetype;
    z:integer;   {для подсчета числа серий}
    eor:boolean; {индикатор конца серии}
procedure copys(var x,y:filetype);
var buf,buf1:data;
begin
 read(x,buf);write(y,buf);
 if eof(x) then eor:=true
          else begin
               {заглядываем вперед}
               read(x,buf1);
               {возвращаемся на исходную запись}
               seek(x,filepos(x)-1);
               if keys='1' then eor:=copy(buf1.fio,1,1)<copy(buf.fio,1,1);
               if keys='2' then eor:=buf1.godr<buf.godr;
               if keys='3' then eor:=buf1.godo<buf.godo;
              end;
end;
procedure copyrun(var x,y:filetype);
 {переписать серии из x в y}
begin
 repeat
  copys(x,y);
 until eor;
end;
procedure mergerun;
{слияние серий из A и B в C}
var bufa,bufb:data;
begin
 repeat
  read(a,bufa);seek(a,filepos(a)-1);
  read(b,bufb);seek(b,filepos(b)-1);
  if keys='1'then if copy(bufa.fio,1,1)<copy(bufb.fio,1,1)
                  then begin;copys(a,c);if eor then copyrun(b,c);end
                  else begin;copys(b,c);if eor then copyrun(a,c);end;
  if keys='2'then if bufa.godr<bufb.godr
                  then begin;copys(a,c);if eor then copyrun(b,c);end
                  else begin;copys(b,c);if eor then copyrun(a,c);end;
  if keys='3'then if bufa.godo<bufb.godo
                  then begin;copys(a,c);if eor then copyrun(b,c);end
                  else begin;copys(b,c);if eor then copyrun(a,c);end;
 until eor
end;
```

```
procedure distribute; {из C в A и B}
begin
 reset(c);rewrite(a);rewrite(b);
 repeat
  copyrun(c,a);
  if not eof(c) then copyrun(c,b);
 until eof(c);
 close(a);close(b);close(c);
end;
procedure merge;
begin
 reset(a);reset(b);rewrite(c);
 while (not eof(a))and(not eof(b)) do
  begin
   mergerun;z:=z+1;
  end;
 while not eof(a) do begin;copyrun(a,c);z:=z+1;end;
 while not eof(b) do begin;copyrun(b,c);z:=z+1;end;
 close(a);close(b);close(c);
end;
begin {main}
 close(files);
 assign(c,namefile);
 assign(b,'c:\file2');
 assign(a,'c:\file3');
 repeat
  distribute;z:=0;
  merge;
 until z=1;
 erase(a);erase(b);
 reset(files)
end;
procedure intemp;
begin
 reset(files);rewrite(filetemp);
 repeat read(files,stu);write(filetemp,stu);until eof(files);
 reset(filetemp);rewrite(files);
end;
procedure outtextmenu(nmenu:byte;invert:boolean);
begin
 window(30,nmenu+7,43,nmenu+7);
 if invert then begin textbackground(2);textcolor(0);end
          else begin textbackground(0);textcolor(15);end;
 case nmenu of
  1:write('Открыть      ');
  2:write('Создать      ');
  3:write('Поиск        ');
  4:write('Добавить     ');
  5:write('Удалить      ');
  6:write('Корректировка');
  7:write('Просмотр     ');
  8:write('Сортировка   ');
```

```
  9:write('Выход        ');
 end;
end;
procedure reader;
begin
  writeln('      Вводим данные об абитуриентах');
  writeln('      Фамилия Имя Отчество');readln(stu.fio);
  write(text1);readln(stu.godr);
  write(text2);readln(stu.godo);
  writeln(text3);
  write(text4);readln(stu.ates.mat);
  write(text5);readln(stu.ates.fiz);
  write(text6);readln(stu.ates.rus);
  writeln('Нуждается ли в общежитии (1-да/2-нет)');
  keys:=readkey;if keys='1' then stu.haus:=true
                          else stu.haus:=false;
  writeln(text7);
  write(text4);readln(stu.ekz.mat);
  write(text5);readln(stu.ekz.fiz);
  write(text6);readln(stu.ekz.rus);
end;
function outfile:boolean;
begin
 if fsearch(namefile,'')='' then begin;
                                  outfile:=true;
                                  namefile:='**********';
                                 end
                           else outfile:=false;
end;
procedure cls;
begin
 window(1,1,80,25);textcolor(15);textbackground(0);clrscr;
 nosound;
end;
procedure sort;
begin
 window(1,1,39,6);textcolor(0);textbackground(14);clrscr;
 nosound;
 writeln('     Сортировка');
 writeln('1-по Фамилие');
 writeln('2-по Году рождения');
 writeln('3-по Году поступления');
 writeln('4-Выход');
 repeat
  keys:=readkey;
  if keys='1' then begin sortirovka;keys:='4';end;
  if keys='2' then begin sortirovka;keys:='4';end;
  if keys='3' then begin sortirovka;keys:='4';end;
 until keys='4';
 cls;writeln('Фамилия Имя Отчество');
end;
procedure startmenu;
```

```
label ex;
begin
 if not openfile then goto ex;
 reset(files);
 rec:=0;
 while not eof(files) do begin read(files,stu);rec:=rec+1;end;
 ex:window(1,1,80,25);textcolor(15);textbackground(0);clrscr;
 writeln('Рабочий файл :',namefile,', записей в файле:',rec);
 for menuunit:=1 to 9 do outtextmenu(menuunit,false);
end;
procedure koreckt;
var zamena:data;
begin
 window(1,1,39,22);textcolor(0);textbackground(14);clrscr;
 nosound;
 reader;
 zamena:=stu;
 intemp;
 freeunit:=1;
 while not eof(filetemp) do
 begin
  read(filetemp,stu);
  if freeunit=position then write(files,zamena)
                       else write(files,stu);
  freeunit:=freeunit+1;
 end;
 cls;writeln('Фамилия Имя Отчество');
end;
procedure delet;
begin
 intemp;
 freeunit:=1;
 while not eof(filetemp) do
 begin
  read(filetemp,stu);
  if position=freeunit then rec:=rec-1
                else write(files,stu);
  freeunit:=freeunit+1;
 end;
 if position=rec+1 then begin position:=position-1;if position=0 then position:=1;end;
end;
procedure lookmenu;
label ex;
const fiounit=20;
begin
 if not(openfile) or (rec=0)  then goto ex;
 if menuunit=3 then position:=freeunit
               else position:=1;
 writeln('Фамилия Имя Отчество');
 repeat
  freeunit:=1;
  reset(files);while not(freeunit=position) do begin read(files,stu);freeunit:=freeunit+1;end;
```

```
 window(40,7,80,20);textcolor(3);textbackground(0);clrscr;
 read(files,stu);
 writeln('      ',stu.fio);
 writeln(text1,stu.godr);
 writeln(text2,stu.godo);
 writeln(text3);
 writeln(text4,stu.ates.mat);
 writeln(text5,stu.ates.fiz);
 writeln(text6,stu.ates.rus);
 writeln(text7);
 writeln(text4,stu.ekz.mat);
 writeln(text5,stu.ekz.fiz);
 writeln(text6,stu.ekz.rus);
 if stu.haus then writeln(text8)
           else writeln(text9);
 window(1,3,39,3+fiounit);clrscr;
 freeunit:=1;
 reset(files);while not(freeunit=position) do begin read(files,stu);freeunit:=freeunit+1;end;
 while not eof(files)and not(freeunit=position+fiounit) do
 begin
  read(files,stu);
  if (position)=freeunit then
                       begin
                        textcolor(0);
                        textbackground(4);
                        writeln(freeunit,'.',stu.fio);
                        textcolor(15);
                        textbackground(0);
                       end
                      else writeln(freeunit,'.',stu.fio);
   freeunit:=freeunit+1;
 end;
 while not(freeunit=position+fiounit) do
 begin
  writeln(freeunit,'.....................');
  freeunit:=freeunit+1;
 end;
 window(1,25,80,25);textcolor(0);textbackground(2);
 write('| Esc-выход | Delete-удалить | Enter-изменить | End-сортировка |');
 keys:=readkey;
 sound(2000);
 quit:=false;
 if keys=#79 then begin sort;end;
 if keys=#72 then begin position:=position-1;if position=0 then position:=1;end;
 if keys=#80 then begin position:=position+1;if position=(rec+1) then position:=rec;end;
 if keys=#83 then delet;
 if keys=#13 then koreckt;
 nosound;
until (keys=#27)or(rec=0);
ex:end;
procedure creat;
begin
```

```
 cls;
 path:=path+name+'\';
 write('Введите имя создаваемого файла ',path);
 readln(name);
 namefile:=path+name+'.dat';
 if openfile then close(files);
 assign(files,namefile);
 rewrite(files);
 close(files);
end;
procedure lookdir;
var f:searchrec;
    recdir:integer;
    attr:byte;
begin
 cls;position:=1;path:='c:\';
 repeat
  freeunit:=1;recdir:=0;
  window(30,1,60,21);textcolor(15);textbackground(0);clrscr;
  writeln(path);writeln;
  FindFirst(path+'*',Directory,f);
  while doserror=0 do
  begin
   recdir:=recdir+1;
   if position+20>=recdir then if position=recdir then begin
                        textcolor(0);textbackground(15);
                        writeln(f.name);
                        textcolor(15);textbackground(0);
                        name:=f.name;attr:=f.attr;
                       end
                 else if position <recdir then writeln(f.name);
   findnext(f);
  end;
  FindFirst(path+'*.dat',AnyFile-Directory,f);
  while doserror=0 do
  begin
   recdir:=recdir+1;
   if position+20>=recdir then if position=recdir then begin
                        textcolor(0);textbackground(15);
                        writeln(f.name);
                        textcolor(15);textbackground(0);
                        name:=f.name;attr:=f.attr;
                       end
                 else if position <recdir then writeln(f.name);
   findnext(f);
  end;
  window(1,25,80,25);textcolor(0);textbackground(2);
  write('| Esc-Выход | Insert-Выбрать каталог |');
  keys:=readkey;
  sound(2000);
  quit:=false;
```

```
      if keys=#82 then if ((attr=48)or(attr=16))and not((name='.')or(name='..'))then begin
creat;quit:=true;end;
      if keys=#72 then begin position:=position-1;if position=0 then position:=1;end;
      if keys=#80 then begin position:=position+1;if position=(recdir+1) then position:=recdir;end;
      if keys=#13 then if (attr=48)or(attr=16) then if (name='..')or(name='.') then begin
                                                                  repeat
                                                                   delete(path,length(path),1);
                                                                  until '\'=copy(path,length(path),1);
                                                                  position:=1;
                                                                 end
                                                           else begin
path:=path+name+'\';position:=1;end
                                      else begin quit:=true;namefile:=path+name;end;
      if keys=#27 then quit:=true;
      nosound;
     until quit;
     if not(keys=#27) then
                    begin
                     openfile:=true;
                     assign(files,namefile);
                    end;
     quit:=false;
    end;
    procedure find;
    label ex;
    var finfio:string[30];
    begin
     if not openfile then goto ex;
     write('Введите Ф.И.О. :');
     readln(finfio);
     reset(files);
     freeunit:=1;
     while not eof(files) do
     begin
      read(files,stu);if stu.fio=finfio then begin clrscr;lookmenu;goto ex;end;
      freeunit:=freeunit+1;
     end;
    ex:end;
    procedure add;
    label ex;
    begin
     if not openfile then goto ex;
     reset(files);rewrite(filetemp);
     while not eof(files) do begin read(files,stu);write(filetemp,stu);end;
     reset(filetemp);rewrite(files);
     while not eof(filetemp) do begin read(filetemp,stu);write(files,stu);end;
     repeat;
      reader;
      write(files,stu);
      clrscr;
      writeln('      Данные об абитуриенте введены');
      writeln('1-Добавить данные');
```

```
      writeln('2-Выход');
      writeln('      (нажмите 1 или 2)');
      keys:=readkey;
     until keys='2';
    ex:end;
    begin
     namefile:='**********';rec:=0;openfile:=false;
     assign(filetemp,'c:\temp.dat');
     startmenu;
     repeat
     outtextmenu(menuunit,true);
     keys:=readkey;
     sound(2000);
     quit:=false;
     if keys=#72 then begin outtextmenu(menuunit,false);menuunit:=menuunit-1;if menuunit=0
then menuunit:=9;end;
     if keys=#80 then begin outtextmenu(menuunit,false);menuunit:=menuunit+1;if menuunit=10
then menuunit:=1;end;
     nosound;
     if keys=#13 then begin
                     cls;
                     case menuunit of
                      1:lookdir;
                      2:lookdir;
                      3:find;
                      4:add;
                      5:lookmenu;
                      6:lookmenu;
                      7:lookmenu;
                      8:lookmenu;
                      9:quit:=true;
                     end;
                     startmenu;
                    end;
     until quit;
     if openfile then close(files);
     rewrite(filetemp);close(filetemp);erase(filetemp);
    end.
```

Распечатка результатов работы программы после выполнения пунктов меню 4,5,6 и 8:

```
Фамилия Имя Отчество

1.Бажин Никита Андреевич
2.Гаспер Валентина Александровна
3.Демидов Иван Сергеевич
4.Егорова Пелагея Луповна
5.Еговцев Иван Артурович                Бажин Никита Андреевич
6.Кирьянов Антон Алексеевич             Год рождения.....……..…1983
7.Корягина Нина Плахова                 Год окончания школы…..2000
8.Маслова Нина Михайловна                 Оценки в атестате
9.Молчановский Ильнар Ирекович          Метематика....…….....5
10.Мельникова Лариса Анатольевна        Физика.....……..5
11.Михайлов Артём Егорович               Русский язык.....……4
12.Малышев Василий Владимирович             Оценки на вступительных экзаменах
13.Ползунова Елена Андреевна             Метематика.....…….5
14.Смирнов Никита Владимирович          Физика.....……...4
15.Созинов Алексей Петрович              Русский язык.....…….4
16.Синилов Сергей Иванович                  Нуждается в общежитии
17.Теплоухов Юрий Леонидович
18.Токарева Надежда Александровна
19.Тихонов Сергей Геннадьевич
20.Шарапов Евгений Владимирович


| Esc-выход | Delete-удалить | Enter-изменить | End-сортировка |
```

Варианты заданий.

1) Книги в библиотеке. Содержится следующая информация: Название книги, автор, год издания, тираж, инвентарный номер.
2) Автостоянка. Сведения о прибывающих машинах: Марка, номер, ФИО владельца, дата прибытия на стоянку, время нахождения на стоянке.
3) Фирма. Анкетные данные сотрудников: ФИО, год поступления в фирму, дата рождения, оклад, адрес.
4) Аптека. Номенклатура товаров: Название лекарства, внутреннее/наружное, дата изготовления, срок годности, ФИО кассира (продавца).
5) Фильмы. Список содержит следующую информацию: название фильма, режиссёр, год выпуска, продолжительность в минутах, студия, где снимался фильм.
6) Отдел “Соки-воды”. Информация об имеющихся в продаже напитках: название, фирма-изготовитель, калорийность, ёмкость посуды, цена.
7) Фотография. Журнал записей содержит информацию: номер заказа, дата приёма заказа, размер фотографий, ФИО фотографа, ФИО заказчика, цена заказа.
8) Гостиница. Содержится следующая информация о проживающих в гостинице: ФИО клиента, номер комнаты, дата въезда, количество дней проживания, стоимость суточного проживания (зависит от категории номера).
9) Продажа видео-аудио кассет. Хранится следующая информация: Марка кассеты, фирма-изготовитель, название альбома, время записи (например: 60, 90 минут).
10) Магазин “Мебель”. Номенклатура товаров: Наименование изделия, дата изготовления, цена продажи, завод-изготовитель, цвет, название материала из которого изготовлено изделие.
11) Магазин “Продукты”. Ведётся учёт товаров: Наименование товара, поступившее количество (штук), количество проданного (штук), цена продажи, дата изготовления, дата реализации.

12) Магазин “Одежда”. Ведётся учёт товаров: Наименование изделия, цена за штуку, количество полученное, продано, размер, цвет, вид ткани.

13) Детский сад. Информация о дошкольниках: ФИО ребёнка, дата рождения, адрес проживания, уровень подготовки (значение 1-5).

14) Отдел “Бытовая техника”. Содержится следующая информация: Наименование товара, фирма изготовитель, серийный номер, цвет, срок гарантии.

15) Киоск “Цветы”. Ведётся учёт проданных цветов: Название цветка, цена за штуку, цвет, комнатное/садовое.

16) Ремонт часов. Имеется следующая информация: Марка часов, ФИО часовщика, дата приёмки, дата выхода из ремонта, стоимость ремонта.

17) Картинная галерея. Ведётся учёт экспонатов галереи: Наименование картины, художник, инвентарный номер, год создания картины, реставратор, цена.

18) Магазин “Ткани”. Ведётся учёт товаров: Наименования ткани, фабрика - изготовитель, полученное количество (в метрах), цена за метр, ширина ткани.

19) Телефонная станция. Имеется информация: ФИО абонента, номер телефона, адрес, тариф со звонка.

20) Речное пароходство. Информация о судах: Название парохода, маршрут, водоизмещение, количество пассажирских мест, класс судна (пароход, теплоход, трамвайчик).

21) Ателье. Имеется следующая информация: номер заказа, наименование изделия, количество ткани для изделия, стоимость заказа, дата приёма заказа, дата изготовления.

22) Дворец творчества юных. Хранится следующая информация: Название кружка, год создания кружка, ФИО преподавателя, количество человек в кружке, средний возраст занимающихся.

23) Почта. Ведётся учёт предоставляемых услуг: Категория отправления (посылка, бандероль, телеграмма, письмо, перевод), вес или количество букв, стоимость услуги, дата отправления, время отправления.

24) Стройматериалы. Ведётся учёт продажи материалов: Наименование материала, завод-изготовитель, цвет, тонкость помола, цена за единицу.

Анкетные данные абитуриентов:

Анисимов Петр Иванович Год рождения: 1982 Год окончания школы: 1999 Аттестат: Математика: 4 Физика: 4 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 5 Физика: 3 Русский язык: 4	Синилов Сергей Иванович Год рождения: 1983 Год окончания школы: 2000 Аттестат: Математика: 4 Физика: 4 Русский язык: 4 Не нуждается в общежитии. Экзамены: Математика: 4 Физика: 5 Русский язык: 3	Шарапов Евгений Владимирович Год рождения: 1984 Год окончания школы: 2001 Аттестат: Математика: 4 Физика: 5 Русский язык: 4 Не нуждается в общежитии. Экзамены: Математика: 4 Физика: 5 Русский язык: 3
Бажин Никита Андреевич Год рождения: 1983 Год окончания школы: 2000 Аттестат: Математика: 5 Физика: 5 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 5 Физика: 4 Русский язык: 4	Созинов Алексей Петрович Год рождения: 1984 Год окончания школы: 2001 Аттестат: Математика: 5 Физика: 5 Русский язык: 5 Не нуждается в общежитии. Экзамены: Математика: 5 Физика: 5 Русский язык: 4	Малышев Василий Владимирович Год рождения: 1982 Год окончания школы: 1999 Аттестат: Математика: 4 Физика: 4 Русский язык: 5 Не нуждается в общежитии. Экзамены: Математика: 4 Физика: 5 Русский язык: 4
Мельникова Лариса Анатольевна Год рождения: 1984 Год окончания школы: 2001 Аттестат: Математика: 5 Физика: 5 Русский язык: 4 Не нуждается в общежитии. Экзамены: Математика: 5 Физика: 5 Русский язык: 5	Тихонов Сергей Геннадьевич Год рождения: 1982 Год окончания школы: 1999 Аттестат: Математика: 4 Физика: 5 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 5 Физика: 4 Русский язык: 3	Еговцев Иван Артурович Год рождения: 1983 Год окончания школы: 2000 Аттестат: Математика: 5 Физика: 4 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 4 Физика: 4 Русский язык: 4
Ползунова Елена Андреевна Год рождения: 1982 Год окончания школы: 1999 Аттестат: Математика: 5 Физика: 4 Русский язык: 4 Не нуждается в общежитии. Экзамены: Математика: 4	Михайлов Артем Егорович Год рождения: 1983 Год окончания школы: 2000 Аттестат: Математика: 4 Физика: 5 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 4	Смирнов Никита Владимирович Год рождения: 1984 Год окончания школы: 2001 Аттестат: Математика: 4 Физика: 4 Русский язык: 5 Не нуждается в общежитии. Экзамены: Математика: 4

Физика: 4 Русский язык: 3	Физика: 4 Русский язык: 4	Физика: 4 Русский язык: 4
Токарева Надежда Александровна Год рождения: 1983 Год окончания школы: 2000 Аттестат: Математика: 4 Физика: 5 Русский язык: 5 Не нуждается в общежитии. Экзамены: Математика: 5 Физика: 5 Русский язык: 5	Маслова Нина Михайловна Год рождения: 1984 Год окончания школы: 2001 Аттестат: Математика: 4 Физика: 5 Русский язык: 3 Не нуждается в общежитии. Экзамены: Математика: 4 Физика: 4 Русский язык: 3	Молчановский Ильнар Ирекович Год рождения: 1982 Год окончания школы: 1999 Аттестат: Математика: 4 Физика: 4 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 4 Физика: 4 Русский язык: 3
Корягина Нина Плахова Год рождения: 1984 Год окончания школы: 2001 Аттестат: Математика: 4 Физика: 4 Русский язык: 5 Не нуждается в общежитии. Экзамены: Математика: 4 Физика: 5 Русский язык: 4	Егорова Пелагея Луповна Год рождения: 1983 Год окончания школы: 2000 Аттестат: Математика: 5 Физика: 5 Русский язык: 4 Не нуждается в общежитии. Экзамены: Математика: 5 Физика: 4 Русский язык: 5	Гаспер Валентина Александровна Год рождения: 1982 Год окончания школы: 1999 Аттестат: Математика: 4 Физика: 4 Русский язык: 3 Нуждается в общежитии. Экзамены: Математика: 4 Физика: 5 Русский язык: 3
Теплоухов Юрий Леонидович Год рождения: 1982 Год окончания школы: 1999 Аттестат: Математика: 4 Физика: 5 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 5 Физика: 4 Русский язык: 4	Кирьянов Антон Алексеевич Год рождения: 1983 Год окончания школы: 2000 Аттестат: Математика: 4 Физика: 4 Русский язык: 4 Нуждается в общежитии. Экзамены: Математика: 4 Физика: 4 Русский язык: 3	