**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**КУРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра “Комплексная Защита Информационных Систем”**

**Специальность 090104 “Комплексная Защита Объектов Информатизации”**

**КУРСОВАЯ РАБОТА**

**по Методам Программирования и Прикладным Алгоритмам**

**(дисциплина)**

**"Сортировка методом сцепления очередей. Поиск методом составных атрибутов"**

**(тема КР)**

**Допустить к защите**

**И.О. Зав. кафедрой д.ф-м.н. Добрица В.П.**

**Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(Ф.И.О. студента)**

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(курс, группа)**

**Работа защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(дата)**

**Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Руководитель к.т.н. С.С.Шевелев**

**(подпись, Ф.И.О. преподавателя)**

**Курск 2009 г.**

**Содержание**

**1 Введение ………………………………….…………………………………………..3**

**2 Теоретическая часть………………………………….…………………………….5**

**3. Практическая часть………………………………….………………………..…...7**

**3.1. Листинг………………………………….……………………………..…...7**

**3.2. Блок-схема………………………………….……………………………..10**

**3.3. Описание переменных………………………………….………………..17**

**3.4. Внешний вид формы…………………………………………………….17**

**3.5. Результаты выполнения программы…………………………………18**

**4. Заключение……………………………………..………………………………….21**

**5. Список используемой литературы……………………………………………..22**

**1. Введение**

Наиболее частый вопрос, который возникает в программировании: переразмещение элементов в возрастающем или убывающем порядке. Представьте, насколько трудно было бы пользоваться словарем, если бы слова в нем не располагались в алфавитном порядке. Точно так же от порядка, в котором хранятся элементы в памяти ЭВМ, во многом зависит скорость и простота алгоритмов, предназначенных для их обработки.

Хотя в словарях слово "сортировка" (sorting) определяется как "распределение, отбор по сортам; деление на категории, сорта, разряды", программисты традиционно используют это слово в гораздо более узком смысле, обозначая им сортировку предметов в возрастающем или убывающем порядке. Этот процесс, пожалуй, следовало бы назвать не сортировкой, а упорядочением (ordering), но использование этого слова привело бы к путанице из-за перегруженности значениями слова "порядок". В математической терминологии это слово также изобилует значениями (порядок группы, порядок перестановки, порядок точки ветвления, отношение порядка и т. п.). Итак, слово "порядок" приводит к хаосу.

В качестве обозначения для процесса упорядочения предлагалось также слово "ранжирование", но оно во многих случаях, по-видимому, не вполне отражает суть дела, особенно если присутствуют равные элементы, и, кроме того, иногда не согласуется с другими терминами. Конечно, слово "сортировка" и само имеет довольно много значений, но оно прочно вошло в программистский жаргон. Поэтому мы без дальнейших извинений будем использовать слово "сортировка" в узком смысле "сортировка по порядку".

Вот некоторые из наиболее важных применений сортировки:

Решение задачи "группировки", когда нужно собрать вместе все элементы с одинаковым значением некоторого признака. Допустим, имеется 10000 элементов, расположенных в случайном порядке, причем значения многих из них равны; и предположим, нам нужно переупорядочить файл так, чтобы элементы с равными значениями занимали соседние позиции в файле. Это, по существу, задача "сортировки" в широком смысле слова, и она легко может быть решена путем сортировки файла в узком смысле слова, а именно расположением элементов в неубывающем порядке v1 v2 ... v10000. Эффективностью, которая может быть достигнута в этой процедуре, и объясняется изменение первоначального смысла слова "сортировка".

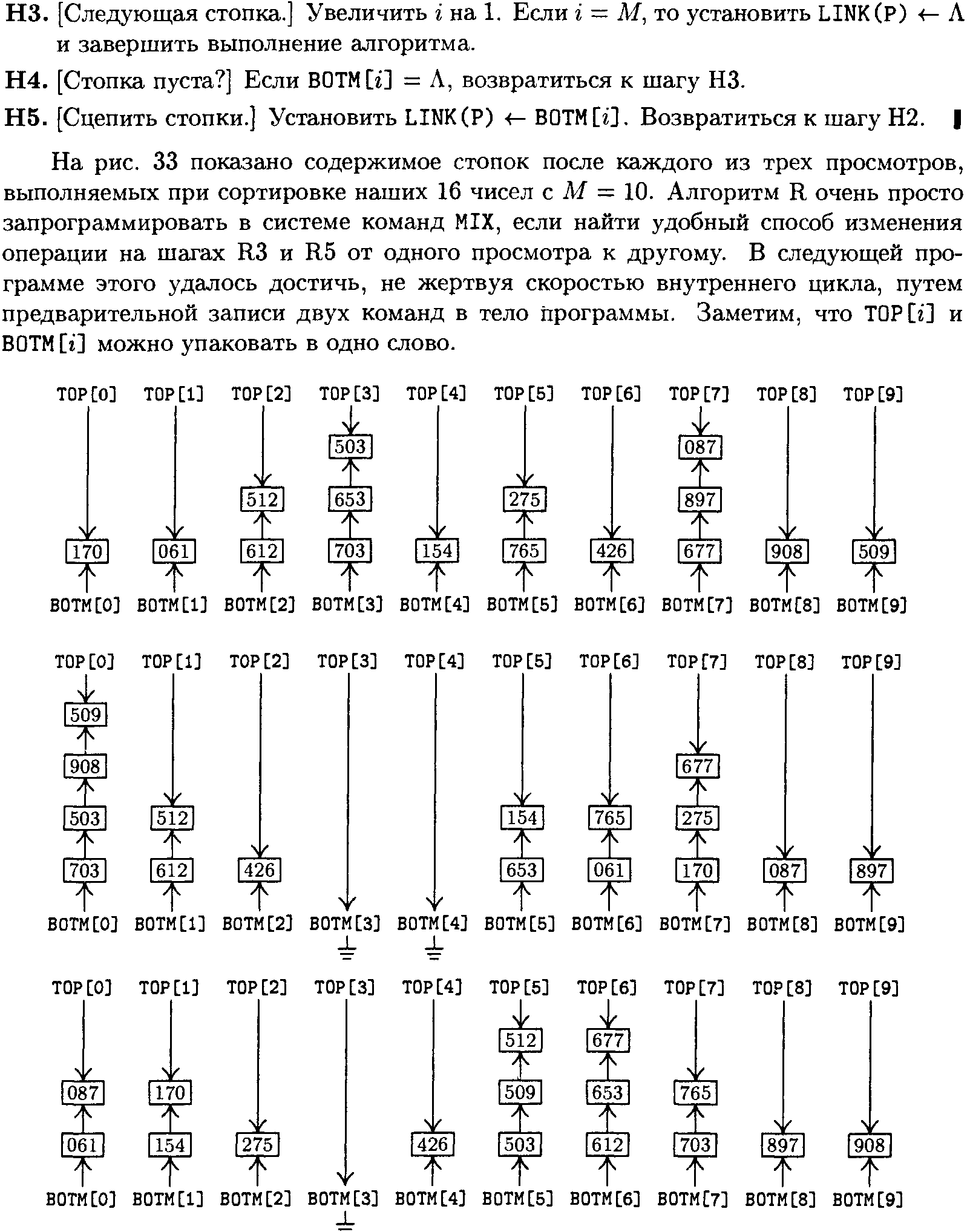
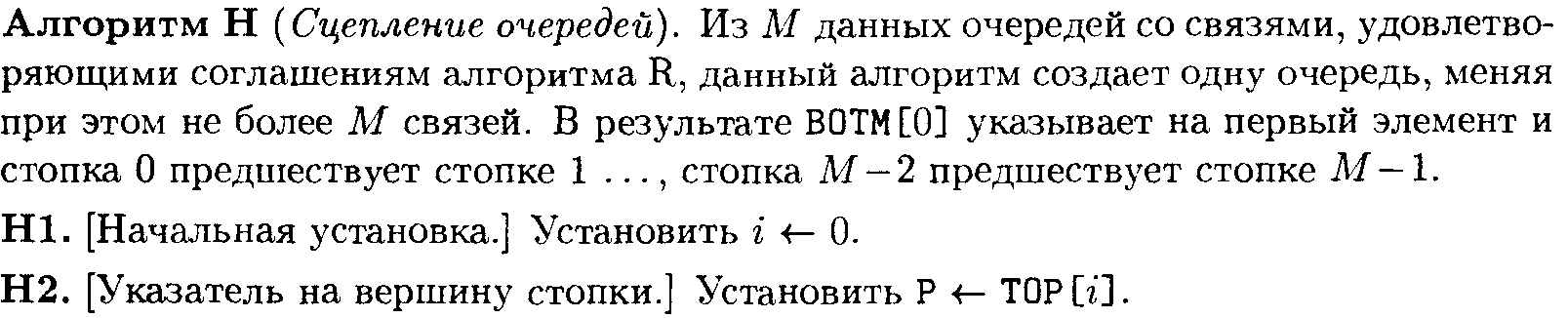
Если два или более файла отсортировать в одном и том же порядке, то можно отыскать в них все общие элементы за один последовательный просмотр всех файлов, без возвратов. Оказывается, что, как правило, гораздо экономнее просматривать список последовательно, а не перескакивая с места на место случайным образом, если только список не настолько мал, что он целиком помещается в оперативной памяти. Сортировка позволяет использовать последовательный доступ к большим файлам в качестве приемлемой замены прямой адресации.

Сортировка помогает и при поиске, с ее помощью можно сделать выдачи ЭВМ более удобными для человеческого восприятия. В самом деле, листинг (напечатанный машиной документ), отсортированный в алфавитном порядке, зачастую выглядит весьма внушительно, даже если соответствующие числовые данные были рассчитаны неверно.

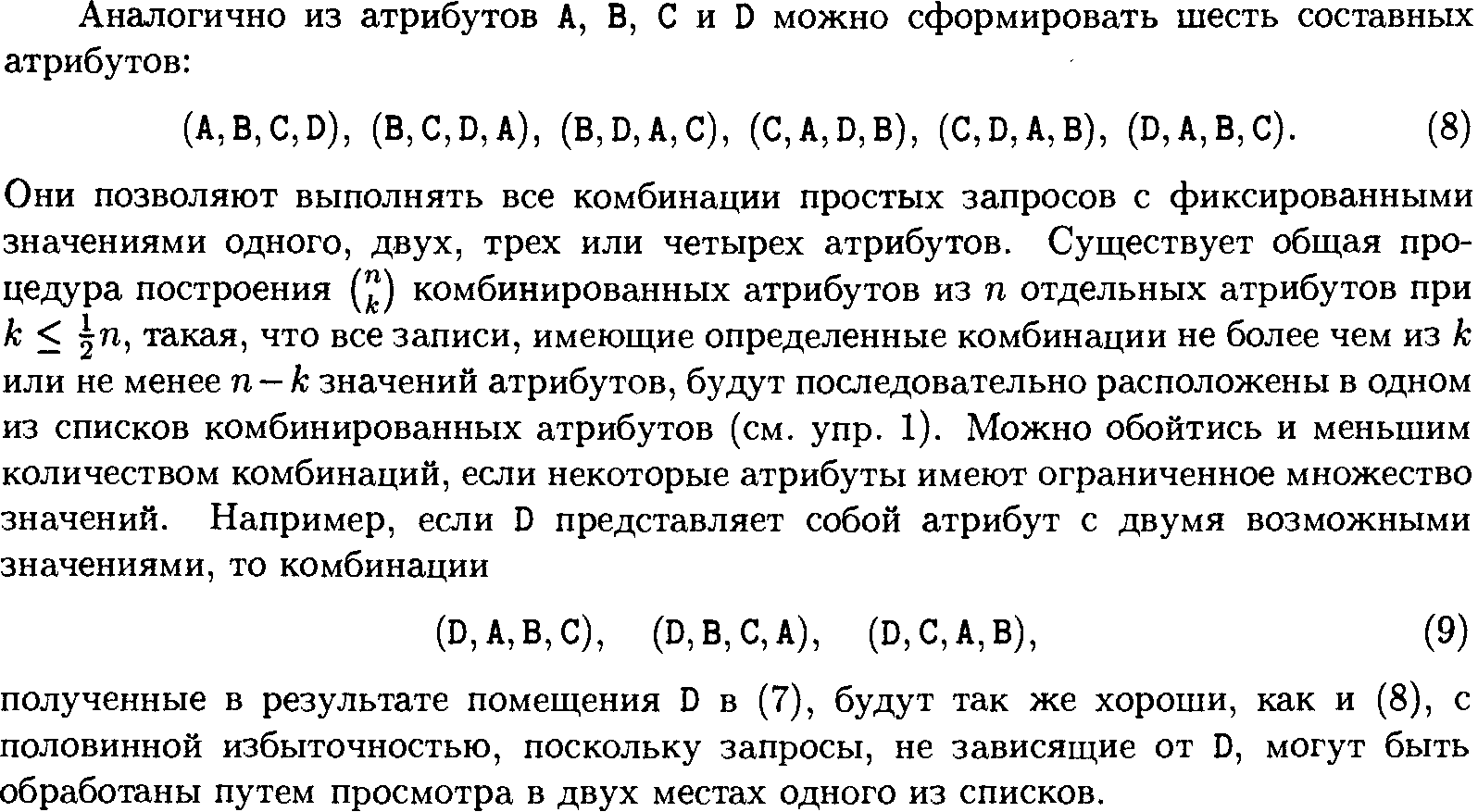
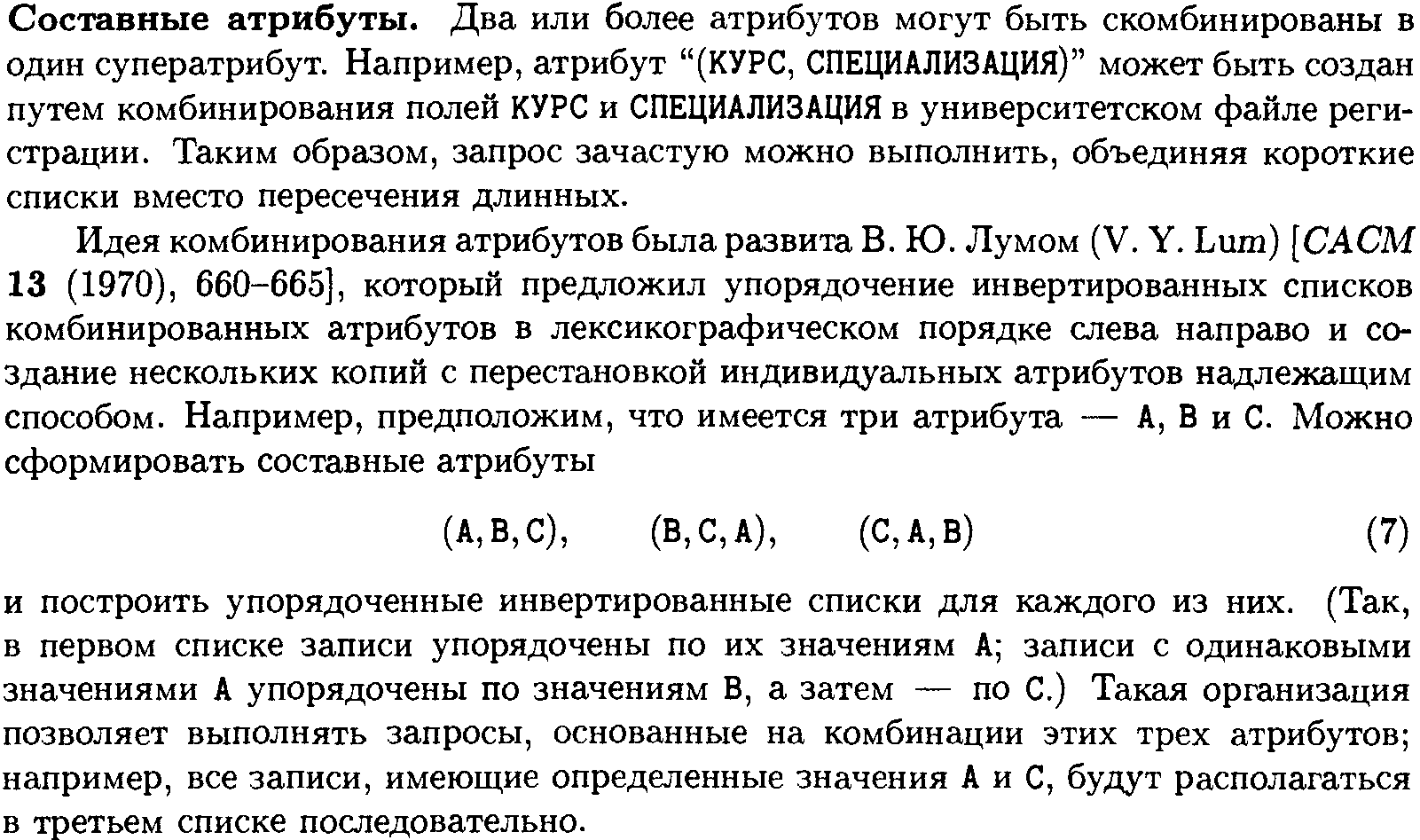
Хотя сортировка традиционно и большей частью использовалась для обработки коммерческих данных, на самом деле она является инструментом, полезным в самых разных ситуациях, и поэтому о нем не следует забывать.

**2. Теоретическая часть**

**Сортировка**



**Поиск**



**3. Практическая часть**

**3.1. Листинг**

unit Unit1;

interface

uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Grids;

type

FindRec = record

count:Integer;

dats:array of integer;

end;

put = record

ind:array of integer;

col\_vo:Integer; end;

TForm1 = class(TForm)

StringGrid1: TStringGrid;

StringGrid2: TStringGrid;

StringGrid3: TStringGrid;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Edit4: TEdit;

Edit5: TEdit;

Edit6: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Edit7: TEdit;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

a,b:array of integer;

implementation

uses Math, DateUtils;

{$R \*.dfm}

function obmen(i:integer;j:integer):Boolean;

var

p:integer;

begin

p:=a[i];

a[i]:=a[j];

a[j]:=p;

Result:=true;

end;

function Zameni(x:integer;y:integer):Integer;

var

i,j:Integer;

pr:boolean;

begin

i:=x;

j:=y;

pr:=false;

while i<j do begin

if a[i]>a[j] then

begin

if pr= false then

i:=i+1

else

j:=j-1;

end;

end;

end;

procedure ScepOch(l, t: integer);

var i: integer;

begin

if l<t then begin

i:=Zameni(l, t);

ScepOch(l,i-1);

ScepOch(i+1,t);

end;

end;

function SostAtr(obr:Integer;kol:Integer):FindRec;

var

sr,j:integer;

n,v:Integer;

founded:Boolean;

rez:FindRec;

tree:put;

begin

n:=0;

v:=kol-1;

founded:=false;

rez.count:=0;

SetLength(rez.dats,0);

tree.col\_vo:=0;

while (n<=v) and (founded=false) do

begin

sr:=(n+v) div 2;

SetLength(tree.ind,tree.col\_vo+1);

tree.ind[tree.col\_vo]:=sr;

inc(tree.col\_vo);

if a[sr]=obr then

begin

for j:=n to v do begin

if a[j]=obr then begin

inc(rez.count);

SetLength(rez.dats,rez.count);

rez.dats[rez.count-1]:=j;

end;

end;

end;

Result:=rez;

end;

procedure TForm1.Button1Click(Sender: TObject);

var

i,k:Integer;

t1,t2:integer;

r:TTimeStamp;

begin

t1:=MilliSecondOfTheMinute(time);

k:=0;

while k<=100 do

begin

a:=b;

ScepOch(0,StrToInt(Edit2.Text)-1);

k:=k+1;

end;

t2:=MilliSecondOfTheMinute(Time);

for i:=0 to StrToInt(Edit2.Text)-1 do

begin

StringGrid2.Cells[i,0]:=IntToStr(a[i]);

end;

Edit3.Text:=FloatToStr(abs(t1-t2)/1000);

end;

procedure TForm1.Button2Click(Sender: TObject);

var i:Integer;

begin

SetLength(b,StrToInt(Edit2.Text));

for i:=0 to StrToInt(Edit2.Text)-1 do begin

b[i]:=Random(StrToInt(Edit1.Text));

StringGrid1.Cells[i,0]:=IntToStr(b[i]);

end;

StringGrid1.ColCount:=StrToInt(Edit2.Text);

StringGrid2.ColCount:=StrToInt(Edit2.Text);

Button1Click(Sender);

StringGrid3.ColCount:=StrToInt(Edit2.Text);

for i:=0 to StrToInt(Edit2.Text)-1 do

StringGrid3.Cells[i,0]:=StringGrid2.Cells[StrToInt(Edit2.Text)-1-i,0];

end;

procedure TForm1.Button3Click(Sender: TObject);

var obr,i,k:integer;

f:FindRec;

t1,t2:Integer;

begin

obr:=StrToInt(Edit4.Text);

t1:=MilliSecondOfTheMinute(time); k:=0;

while k<=10000 do begin

f:=SostAtr(obr,StrToInt(Edit2.Text));

inc(k); end;

t2:=MilliSecondOfTheMinute(time);

if f.count<>0 then begin for i:=0 to f.count-1 do begin

Edit5.Text:=Edit5.Text+IntToStr(f.dats[i]+1)+';';

Edit7.Text:=Edit7.Text+IntToStr(StrToInt(Edit2.Text) - f.dats[i])+';';

end; end

else begin

Edit5.Text:='нет такого элемента';

Edit7.Text:='нет такого элемента'; end;

Edit6.Text:=FloatToStr(abs(t1-t2)/10000);

end; end.

**3.2. Блок-схема:**

начало

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

конец

1

2

3

4

5

**procedure ScepOch (l, t: integer);**

начало

i

l<t

i:=Zameni(l, t);

ScepOch t(l,i-1);

ScepOch (i+1,t);

конец

да

нет

1

2

3

4

5

6

7

**function Zameni(x:integer;y:integer):Integer;**

начало

i,j, pr

i:=x;

j:=y;

pr:=false;

i<j

a[i]<=a[j]

pr=false

j:=j-1

i:=i+1

конец

да

да

да

нет

нет

нет

1

2

3

4

5

6

7

8

9

10

11

**function Obmen(i:integer;j:integer):Boolean;**

начало

p

p:=a[i];

a[i]:=a[j];

a[j]:=p;

конец

1

2

3

4

5

6

**procedure TForm1.Button1Click(Sender: TObject);**

начало

i

a:=b;

ScepOch (0,StrToInt(Edit2.Text)-1);

i:=0 to StrToInt(Edit2.Text)-1

StringGrid2.Cells[i,0]:=IntToStr(a[i]);

конец

1

2

3

4

5

6

7

**procedure TForm1.Button2Click(Sender: TObject);**

начало

i

SetLength(b,StrToInt(Edit2.Text));

i:=0 to StrToInt(Edit2.Text)-1

b[i]:=Random(StrToInt(Edit1.Text));

StringGrid1.Cells[i,0]:=IntToStr(b[i]);

StringGrid1.ColCount:=StrToInt(Edit2.Text);

StringGrid2.ColCount:=StrToInt(Edit2.Text);

Button1Click(Sender);

конец

1

2

3

4

5

6

7

8

9

10

**procedure TForm1.Button3Click(Sender: TObject);**

начало

obr,i, f

obr:=StrToInt(Edit4.Text);

f:= SostAtr (obr,StrToInt(Edit2.Text));

i:=0 to f.count-1

Edit5.Text:=Edit5.Text+IntToStr(f.dats[i]+1)+';';

конец

1

2

3

4

5

6

7

**function SostAtr (obr:Integer;kol:Integer):FindRec;**

начало

sr,j, n,v, founded,

rez, tree

n:=0;

v:=kol-1;

founded:=false;

rez.count:=0;

SetLength(rez.dats,0);

tree.col\_vo:=0;

sr:=(n+v) div 2;

SetLength(tree.ind,tree.col\_vo+1);

1

2

3

4

5

6

7

8

9

10

a[sr]=obr

a[j]=obr

inc(rez.count);

SetLength(rez.dats,rez.count);

Result:=rez;

конец

да

да

нет

нет

11

12

13

14

15

16

**3.3. Описание переменных:**

**function Obmen(i:integer;j:integer):Boolean;**

p – переменная для перестановки двух элементов массива

**function Zameni(x:integer;y:integer):Integer;**

i,j – счетчики

**procedure ScepOch (l, t: integer);**

i – элемент массива для сортировки

**procedure TForm1.Button1Click(Sender: TObject);**

i – счетчик

**procedure TForm1.Button2Click(Sender: TObject);**

i – счетчик

**procedure TForm1.Button3Click(Sender: TObject);**

obr – образец для поиска

i - счетчик

f – номер данного элемента в массиве

**function SostAtr (obr:Integer;kol:Integer):FindRec;**

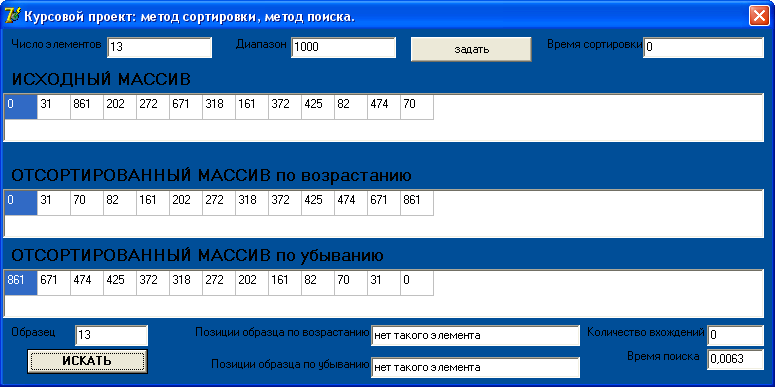
j - счетчик

n – количество элементов

v – промежуточная переменная для работы

rez – номер образца в массиве

**3.4. Внешний вид формы:**



**3.5. Результаты выполнения программы:**

|  |  |
| --- | --- |
| Количество элементов | Среднее время поиска (мс) |
| **100** | **5** |
| **1000** | **7,60** |
| **3000** | **8** |
| **5000** | **10** |
| **10000** | **12** |

|  |  |
| --- | --- |
| Количество  элементов | время сортировки (мс) |
| **10** | **14** |
| **50** | **16** |
| **100** | **17** |
| **200** | **18** |
| **300** | **20** |
| **500** | **25** |
| **1000** | **49** |
| **2000** | **127** |
| **3000** | **263** |
| **5000** | **405** |
| **6000** | **504** |
| **7000** | **559** |
| **8000** | **775** |
| **9000** | **965** |
| **10000** | **1068** |



**4. Заключение**

Поставщики вычислительных машин считают, что в среднем более 25% машинного времени систематически тратится на сортировку. Во многих вычислительных системах на нее уходит больше половины машинного времени. Из этой статистики можно заключить, что либо сортировка имеет много важных применений, либо ею часто пользуются без нужды, либо применяются в основном неэффективные алгоритмы сортировки. По-видимому, каждое из трех предположений содержит долю истины. Во всяком случае ясно, что сортировка заслуживает серьезного изучения с точки зрения ее практического использования.

Но даже если бы сортировка была почти бесполезна, нашлась бы масса других причин заняться ею! Изобретательные алгоритмы сортировки говорят о том, что она и сама по себе интересна как объект исследования. В этой области существует множество увлекательных нерешенных задач наряду с весьма немногими уже решенными.

Делая вывод после изучения сортировки и поиска, можем сказать, что сортировка не достаточно эффективна при значительных размерах массива, т.к. значительное количество ненужных операций негативно влияет на время сортировки. Поиск при исследовании показал хорошие результаты по времени и прост в реализации.

Таким образом, метод сортировки можно рекомендовать при работе с массивами малой и средней величины, т.к. время, необходимое для его написания, не велико, и при работе с большими массивами лучше использовать более эффективные алгоритмы. Метод поиска можно рекомендовать при работе с массивами любого размера, т.к. он прост в реализации и эффективен по времени.

**5. Список использованной литературы**

1. Д. Кнут «Искусство программирования» 3 том

2. Н. Вирт «Алгоритмы+структуры данных=программы».-М.:Мир.1977

3. Белова Т.М.,Старков Ф.А. «Программирование в Delphi».-КурскГТУ.Курск.2002

4. Фаронов В.В. «Delphi.Программирование на языке высокого уровня».-СПб.:Питер,2006

5. А.Ахо,Дж.Хопкрофт,Дж.Ульман «Построение и анализ вычислительных алгоритмов».-

М.:Мир,1979

6. Фаронов В. «Turbo Pascal.Наиболее полное руководство».-СПб.:БХВ-Петербург,2004

7. Иванов Б.Н. «Дискретная матеметика.Алгоритмы и программы».-М.:Лаборатория базовых знаний,2003

8. Липский В. «Комбинаторика для программистов».-М.:Мир,1988

9. Д.Грин , Д.Кнут «Комбинаторная теория».- М.:Мир,1987

10. Д.Грин , Д.Кнут «математические методы анализа алгоритмов».М.:Мир,1987

11. Дж.Макконнелл «Основы современных алгоритмов».М.:Техносфера,2004

12. Аветисян Р.Д.,Аветисян Д.О. «Теоретические основы информатики»