Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №9 по курсу «Программирование»

**Исследование алгоритмов сортировки данных на внешней памяти**

Выполнил студент группы ИВТ-12 /Крючков И. С/ Проверил /Чистяков Г. А./

Киров 2021

**Цель работы:**

# познакомиться с организацией стандартных диалоговых окон, изучить принципы работы с данными на внешней памяти, получить навыки работы с типизированными файлами.

# Задание:

1. Разработать программу для сортировки данных, находящихся на жестком диске в типизированном файле.

2. Размер соритруемых данных должен быть не менее 1 Гб. Объем потребляемой оперативной памяти не должен превышать 10% от размера исходного файла.

3. Исходный и целевой файлы должны выбираться с помощью стандартных диалоговых окон.

4. Структуру сортируемого файла и ключ сортировки согласовать с преподавателем.

5. Помимо основной программы, требуется реализовать две вспомогательные утилиты - для генерации массива данных и проверки упорядоченности.

**Структура:**

rec = record

id: longint;

tittle: string[100];

description: string[255];

size: integer;

cost: integer;

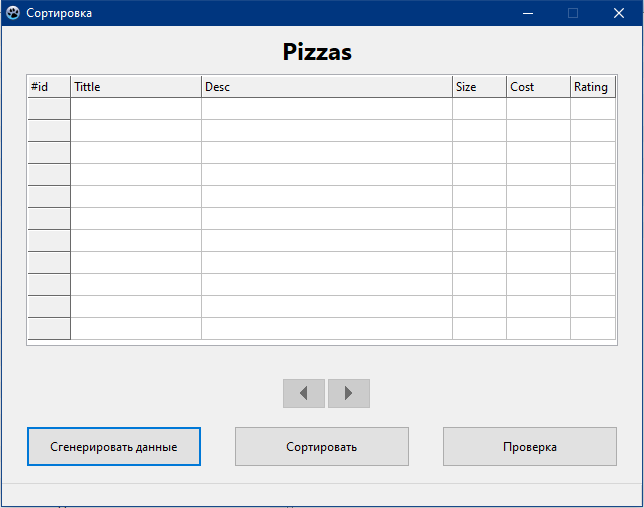
feedback: 1..5;

end;

**Схема алгоритма:**

****

**Экранная форма:**



**Листинг кода:**

**Основная программа:**

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ComCtrls,

Menus, EditBtn, DBCtrls, Grids, Buttons, ExtCtrls, RTTIGrids;

type

rec = record

id: longint;

tittle: string[100];

description: string[255];

size: integer;

cost: integer;

feedback: 1..5;

end;

ft = file of rec;

{ TForm1 }

TForm1 = class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

OpenDialog1: TOpenDialog;

OpenDialog2: TOpenDialog;

SaveDialog1: TSaveDialog;

SaveDialog2: TSaveDialog;

StaticText1: TStaticText;

StaticText2: TStaticText;

StatusBar1: TStatusBar;

StringGrid1: TStringGrid;

UpDown1: TUpDown;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);

procedure toggleButtons(val: boolean);

procedure ClearTable;

private

public

end;

var

Form1: TForm1;

page: longint = 1;

perPage: longint = 11;

maxpage: longint;

sortfile: string;

pizzaFile: ft;

pizza: rec;

implementation

uses sorting, generator, checking;

{$R \*.lfm}

{ TForm1 }

// Сгенерировать

procedure TForm1.Button1Click(Sender: TObject);

begin

generate(Form1);

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

sort(Form1);

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

check(Form1);

end;

procedure TForm1.UpDown1Click(Sender: TObject; Button: TUDBtnType);

var

n: longint;

fs: longint;

begin

if opened then

begin

AssignFile(pizzaFile, sortfile);

reset(pizzaFile);

fs := FileSize(pizzaFile);

maxpage := fs div perPage;

if fs mod perPage <> 0 then inc(maxpage);

case Button of

btNext:

begin

if page + 1 <= maxpage then

inc(page)

else

begin

CloseFile(pizzaFile);

exit;

end;

end;

btPrev:

begin

if page - 1 >= 1 then

dec(page)

else

begin

CloseFile(pizzaFile);

exit;

end;

end;

end;

Form1.StaticText2.Caption := inttostr(page) + '/' + inttostr(maxpage);

seek(pizzaFile, (page-1)\*perPage);

Form1.ClearTable;

n := 0;

while (n < perPage) and ((page-1)\*perPage+n < fs) do

begin

read(pizzaFile,pizza);

Form1.StringGrid1.Cells[0, n+1] := inttostr(pizza.id);

Form1.StringGrid1.Cells[1, n+1] := pizza.tittle;

Form1.StringGrid1.Cells[2, n+1] := pizza.description;

Form1.StringGrid1.Cells[3, n+1] := inttostr(pizza.size);

Form1.StringGrid1.Cells[4, n+1] := inttostr(pizza.cost);

Form1.StringGrid1.Cells[5, n+1] := inttostr(pizza.feedback);

Form1.StringGrid1.update;

inc(n);

end;

CloseFile(pizzaFile);

end;

end;

procedure TForm1.toggleButtons(val: boolean);

begin

Form1.Button1.Enabled := val;

Form1.Button2.Enabled := val;

Form1.Button3.Enabled := val;

Form1.UpDown1.Enabled := val;

Form1.Button1.Update();

Form1.Button2.Update();

Form1.Button3.Update();

Form1.UpDown1.Update();

end;

procedure TForm1.ClearTable;

var i: integer;

begin

with StringGrid1 do

for i:=1 to RowCount-1 do

Rows[i].Clear;

end;

end.

**Модуль sorting**

unit sorting;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Unit1, Dialogs;

procedure sort(Form1: TForm1);

implementation

var

f: ft;

helpfile: file of rec;

pizzaRec: array of rec;

function comparator(rec1,rec2:rec):boolean;

begin

if rec1.size>rec2.size then comparator:=true

else if rec1.size<rec2.size then comparator:=false

else if rec1.cost>rec2.cost then comparator:=true

else if rec1.cost<rec2.cost then comparator:=false

else comparator:=false;

end;

procedure swap(i,j:longint);

var

tmp: rec;

begin

tmp:= pizzaRec[i];

pizzaRec[i]:=pizzaRec[j];

pizzaRec[j]:=tmp;

end;

procedure heapsort(j,heapsize:longint);

var

lc, rc, max:integer;

begin

lc := j \* 2;

rc := j \*2 + 1;

max := j;

if rc <= heapsize then

begin

if comparator(pizzaRec[lc], pizzaRec[max]) then max := lc;

if comparator(pizzaRec[rc], pizzaRec[max]) then max := rc;

swap(j, max);

if max = j then max := lc;

end;

if lc = heapsize then

begin

if comparator(pizzaRec[lc], pizzaRec[max]) then swap(lc, max);

max := lc;

end;

if max <= heapsize div 2 then heapsort(max,heapsize);

end;

procedure SortSer(heapsize: longint);

begin

while HeapSize > 0 do

begin

swap(1, heapsize);

dec(heapsize);

heapsort(1,heapsize);

end;

end;

procedure mergeSort(count: integer);

var

mergfile, f, helpfile: file of rec;

f1,f2: rec;

i,l1,l2: integer;

begin

for i := 1 to count-1 do

begin

case i of

1: begin

AssignFile(f, 'tmp1.txt');

AssignFile(helpfile, 'tmp2.txt');

AssignFile(mergfile, 'sl1.txt');

end;

2: begin

AssignFile(f, 'tmp3.txt');

AssignFile(helpfile, 'tmp4.txt');

AssignFile(mergfile, 'sl2.txt');

end;

3: begin

AssignFile(f, 'tmp5.txt');

AssignFile(helpfile, 'tmp6.txt');

AssignFile(mergfile, 'sl3.txt');

end;

4: begin

AssignFile(f, 'tmp7.txt');

AssignFile(helpfile, 'tmp8.txt');

AssignFile(mergfile, 'sl4.txt');

end;

5: begin

AssignFile(f, 'tmp9.txt');

AssignFile(helpfile, 'tmp10.txt');

AssignFile(mergfile, 'sl5.txt');

end;

6: begin

AssignFile(f, 'tmp11.txt');

AssignFile(helpfile, 'tmp12.txt');

AssignFile(mergfile, 'sl6.txt');

end;

7: begin

AssignFile(f, 'sl1.txt');

AssignFile(helpfile, 'sl2.txt');

AssignFile(mergfile, 'tmp1.txt');

end;

8: begin

AssignFile(f, 'sl3.txt');

AssignFile(helpfile, 'sl4.txt');

AssignFile(mergfile, 'tmp2.txt');

end;

9: begin

AssignFile(f, 'sl5.txt');

AssignFile(helpfile, 'sl6.txt');

AssignFile(mergfile, 'tmp3.txt');

end;

10: begin

AssignFile(f, 'tmp1.txt');

AssignFile(helpfile, 'tmp2.txt');

AssignFile(mergfile, 'sl1.txt');

end;

11: begin

AssignFile(f, 'sl1.txt');

AssignFile(helpfile, 'tmp3.txt');

AssignFile(mergfile, 'tmp1.txt');

end;

12: begin

AssignFile(f, 'tmp1.txt');

AssignFile(helpfile, 'tmp13.txt');

AssignFile(mergfile, 'sl1.txt');

end;

end;

Reset(f);

Reset(helpfile);

Rewrite(mergfile);

read(f, f1);

read(helpfile, f2);

l1 := 1;

l2 := 1;

while (l1 <= filesize(f)) and (l2 <= filesize(helpfile)) do

begin

if comparator(f1, f2)=false then

begin

write(mergfile, f1);

inc(l1);

if l1<=filesize(f) then read(f, f1);

end

else

begin

write(mergfile, f2);

inc(l2);

if l2<=filesize(helpfile) then read(helpfile, f2);

end;

end;

if l1 > filesize(f) then

while l2 <= filesize(helpfile) do

begin

write(mergfile, f2);

inc(l2);

if l2 <= filesize(helpfile) then read(helpfile, f2);

end

else

while l1 <= filesize(f) do

begin

write(mergfile, f1);

inc(l1);

if l1 <= filesize(f) then read(f, f1);

end;

CloseFile(f);

CloseFile(helpfile);

Erase(f);

Erase(helpfile);

CloseFile(mergfile);

end;

end;

procedure sort(Form1: TForm1);

var n, i, j, fz, heapsize, count, fs:longint;

begin

Form1.Statusbar1.SimpleText := 'Выберите сортируемый файл';

if Form1.OpenDialog1.Execute() then

begin

Form1.Statusbar1.SimpleText := 'Выберите конечный файл';

if Form1.SaveDialog2.Execute() then

begin

AssignFile(f,Form1.OpenDialog1.FileName);

Reset(f);

fz := FileSize(f);

n := fz div 12;

SetLength(pizzaRec,n+1);

if fz mod 12 > 0 then count:=13

else count := 12;

Form1.Statusbar1.SimpleText := 'Сортировка данных...';

Form1.Statusbar1.Update;

Form1.toggleButtons(False);

for i := 1 to count do

begin

if i = 13 then

begin

n := fz mod 12;

SetLength(pizzaRec, n+1);

end;

heapsize := n;

for j:=1 to n do read(f, pizzaRec[j]);

for j:=n div 2 downto 1 do heapsort(j, heapsize);

SortSer(heapsize);

case i of

1: AssignFile(helpfile, 'tmp1.txt');

2: AssignFile(helpfile, 'tmp2.txt');

3: AssignFile(helpfile, 'tmp3.txt');

4: AssignFile(helpfile, 'tmp4.txt');

5: AssignFile(helpfile, 'tmp5.txt');

6: AssignFile(helpfile, 'tmp6.txt');

7: AssignFile(helpfile, 'tmp7.txt');

8: AssignFile(helpfile, 'tmp8.txt');

9: AssignFile(helpfile, 'tmp9.txt');

10: AssignFile(helpfile, 'tmp10.txt');

11: AssignFile(helpfile, 'tmp11.txt');

12: AssignFile(helpfile, 'tmp12.txt');

13: AssignFile(helpfile, 'tmp13.txt');

end;

Rewrite(helpfile);

for j := 1 to n do write(helpfile, pizzaRec[j]);

CloseFile(helpfile);

end;

CloseFile(f);

SetLength(pizzaRec,0);

mergeSort(count);

AssignFile(helpfile, Form1.SaveDialog2.FileName);

if fileexists(Form1.SaveDialog2.FileName) then erase(helpfile);

if count=12 then AssignFile(f, 'tmp1.txt')

else AssignFile(f, 'sl1.txt');

Rename(f, Form1.SaveDialog2.FileName);

reset(f);

maxpage := FileSize(f) div perPage;

if FileSize(f) mod perPage <> 0 then inc(maxpage);

Form1.ClearTable;

fs := FileSize(f);

n := 0;

while (n < perPage) and (n < fs) do

begin

read(f,pizza);

Form1.StringGrid1.Cells[0, n+1] := inttostr(pizza.id);

Form1.StringGrid1.Cells[1, n+1] := pizza.tittle;

Form1.StringGrid1.Cells[2, n+1] := pizza.description;

Form1.StringGrid1.Cells[3, n+1] := inttostr(pizza.size);

Form1.StringGrid1.Cells[4, n+1] := inttostr(pizza.cost);

Form1.StringGrid1.Cells[5, n+1] := inttostr(pizza.feedback);

Form1.StringGrid1.update;

inc(n);

end;

sortfile := Form1.SaveDialog2.FileName;

CloseFile(f);

Form1.toggleButtons(True);

page := 1;

Form1.StaticText2.Caption := '1/' + inttostr(maxpage);

Form1.Statusbar1.SimpleText := 'Сортировка данных завершена.';

end;

end;

end;

end.

**Модуль generator**

unit generator;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Dialogs, Unit1, Math;

type

pt = array[0..200] of string[100];

var

ptittle: TextFile;

ptittles: pt;

gd: longint;

lp, pp: integer;

opened: boolean = False;

//Генерация данных

procedure generate(Form1: TForm1);

implementation

var

lorem: string = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Etiam dignissim diam quis enim lobortis scelerisque fermentum dui faucibus. Dapibus ultrices in iaculis nunc sed augue lacus';

i, id, n, fs: longint;

procedure generate(Form1:TForm1);

begin

if Form1.SaveDialog1.Execute() then

begin

Assignfile(pizzaFile, Form1.SaveDialog1.FileName);

AssignFile(ptittle, 'ptittle.txt');

rewrite(pizzaFile);

opened := True;

try

reset(ptittle);

except

on eInOutError do

begin

ShowMessage('Файл ptittle.txt не найден!');

CloseFile(pizzaFile);

exit;

end

else

begin

ShowMessage('Не удалось открыть файл ptittle.txt');

CloseFile(pizzaFile);

exit;

end;

end;

i := 0;

while not eof(ptittle) do

begin

readln(ptittle, ptittles[i]);

inc(i);

end;

CloseFile(ptittle);

randomize;

id := 0;

gd := 3500000;

n := 0;

Form1.toggleButtons(False);

Form1.ClearTable;

Form1.Statusbar1.SimpleText := 'Идет генерация данных - 0%';

while FileSize(pizzaFile) < gd do

begin

pizza.id := id;

pizza.tittle := ptittles[randomrange(0, i)];

pizza.description := lorem;

pizza.size := randomrange(20, 150);

pizza.cost := randomrange(50, 100000);

pizza.feedback := randomrange(1, 5);

inc(id);

write(pizzaFile,pizza);

fs := FileSize(pizzaFile);

if (n < perPage) and (n < fs) then

begin

Form1.StringGrid1.Cells[0, n+1] := inttostr(pizza.id);

Form1.StringGrid1.Cells[1, n+1] := pizza.tittle;

Form1.StringGrid1.Cells[2, n+1] := pizza.description;

Form1.StringGrid1.Cells[3, n+1] := inttostr(pizza.size);

Form1.StringGrid1.Cells[4, n+1] := inttostr(pizza.cost);

Form1.StringGrid1.Cells[5, n+1] := inttostr(pizza.feedback);

Form1.StringGrid1.update;

inc(n);

end;

pp := round((FileSize(pizzaFile)/gd)\*100);

Form1.Statusbar1.SimpleText := 'Идет генерация данных - ' + inttostr(pp) + '%';

if (pp mod 5 = 0) AND (lp <> pp) then

begin

Form1.Statusbar1.Update;

lp := pp;

end;

end;

sortfile := Form1.SaveDialog1.FileName;

CloseFile(pizzaFile);

maxpage := fs div perPage;

if fs mod perPage <> 0 then inc(maxpage);

Form1.toggleButtons(True);

page := 1;

Form1.StaticText2.Caption := '1/' + inttostr(maxpage);

Form1.Statusbar1.SimpleText := 'Генерация данных завершена';

end;

end;

end.

**Модуль checking**

unit checking;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Unit1;

procedure check(Form1: TForm1);

implementation

function comparator(rec1,rec2:rec):boolean;

begin

if rec1.size>rec2.size then comparator:=true

else if rec1.size<rec2.size then comparator:=false

else if rec1.cost>rec2.cost then comparator:=true

else if rec1.cost<rec2.cost then comparator:=false

else comparator:=false;

end;

procedure check(Form1: TForm1);

var

rec1, rec2: rec;

i:longint;

begin

Form1.Statusbar1.SimpleText := 'Выберите проверяемый файл';

if Form1.OpenDialog2.Execute then

begin

Form1.toggleButtons(False);

AssignFile(pizzaFile, Form1.OpenDialog2.FileName);

Reset(pizzaFile);

read(pizzaFile, rec1);

Form1.Statusbar1.SimpleText := 'Проверка...';

Form1.Statusbar1.Update;

for i:=2 to filesize(pizzaFile) do

begin

read(pizzaFile, rec2);

if comparator(rec1,rec2)=true then

begin

Form1.Statusbar1.SimpleText := 'Файл не отсортирован';

Form1.toggleButtons(True);

CloseFile(pizzaFile);

exit;

end

else rec1:=rec2;

end;

Form1.Statusbar1.SimpleText := 'Файл отсортирован';

Form1.toggleButtons(True);

CloseFile(pizzaFile);

end;

end;

end.

**Вывод:** в ходе выполнения лабораторной работы были получены навыки работы с данными на внешней памяти, получен опыт работа с типизированными файлами и организацией стандартных диалоговых окон.