**УО «Белорусский государственный университет информатики и**

**радиоэлектроники»**

**Кафедра ПОИТ**

**Отчет по лабораторной работе №3.3**

**по предмету**

**Основы Алгоритмизации и Программирования**

**Вариант 1**

**Выполнил**

**Андросов И.С.**

**Проверила**

**Данилова Г.В.**

Группа:

**8**51001

**Минск 2018**

**Задание**

Сортировка двухпутевыми вставками.

Метод простых вставок

Согласно методу простых вставок элементы последовательности, начиная со второго вставляются в уже упорядоченную подпоследовательность. Для того, чтобы не проверять сложное условие в цикле-пока, очередной вставляемый элемент ставится на место нулевого и служит барьером, когда вставляемый элемент оказывается меньшим элементом подпоследовательности. Одновременно с поиском места вставки большие элементы сдвигаются на одну позицию к концу подпоследовательности. Одним из лучших методов сортировки является метод Шелла, который основан на методе простых вставок. Этот метод подобен методу Бэтчера, только подмассивы сортируются не метом обмена, а методом простых вставок.

Метод двухпутевых вставок

Считается, что слева и справа от первого элемента имеется n-1

свободное место и вставляемые элементы сравниваются либо с последним

вставленным элементом справа или слева. Этот выбор зависит от того, были

ли сдвиги элементов на предыдущем шаге, если были, то направление

вставки меняется на противоположный.

**Код программы**

**(Delphi)**

**program** laba\_3\_3;

{$APPTYPE CONSOLE}

**uses**

SysUtils;

**type**

TShortIntArray = **array of** ShortInt;

**const**

MinNumber: ShortInt = -128;

MaxNumber: ShortInt = 127;

MinSize: ShortInt = 2;

MaxSize: ShortInt = 127;

**procedure** WriteArrayToConsole(MyArray: TShortIntArray);

**var**

i: ShortInt;

**begin**

Write('The sequence has been ordered: ');

**for** i := 0 **to** High(MyArray) **do**

Write(MyArray[i], ' ');

WriteLn;

**end**;

**function** GetYesNoChoice(): Char;

**var**

Answer: Char;

IsCorrect: Boolean;

**begin**

**repeat**

ReadLn(Answer);

Answer := UpCase(Answer);

**if** (Answer = 'Y') **or** (Answer = 'N') **then**

IsCorrect := true

**else**

**begin**

IsCorrect := false;

WriteLn('Incorrect input. Enter Y(Yes) or N(No):');

**end**;

**until** IsCorrect;

GetYesNoChoice := Answer;

**end**;

**procedure** GetFile(**var** ThatFile: TextFile);

**var**

IsCorrect: Boolean;

NameOfFile: String;

**begin**

WriteLn('Enter file name(Name.txt):');

IsCorrect := false;

**repeat**

ReadLn(NameOfFile);

**if** (**not** FileExists(NameOfFile)) **then**

WriteLn('File does not exist. Try again: ')

**else**

**try**

AssignFile(ThatFile, NameOfFile);

IsCorrect := true;

**except**

WriteLn('Access is not alowed. Try Again: ');

**end**;

**until** IsCorrect;

Reset(ThatFile);

**end**;

**procedure** GetOutPutFile(**var** NewFile: TextFile);

**var**

IsCorrect: Boolean;

**begin**

IsCorrect := false;

**repeat**

GetFile(NewFile);

WriteLn('Would you like to rewrite the file? Press Y(Yes) or N(No):');

**try**

**if** GetYesNoChoice = 'Y' **then**

Rewrite(NewFile)

**else**

**begin**

Append(NewFile);

WriteLn(NewFile);

**end**;

IsCorrect := true;

**except**

Write('This file can not be edited. Try again. ');

**end**;

**until** IsCorrect;

**end**;

**procedure** WriteArrayToFile(MyArray: TShortIntArray);

**var**

NewFile: TextFile;

i: ShortInt;

**begin**

GetOutputFile(NewFile);

Write(NewFile, 'The sequence has been ordered: ');

**for** i := 0 **to** High(MyArray) **do**

Write(NewFile, MyArray[i], ' ');

CloseFile(NewFile);

**end**;

**function** OrderArray(MyArray: TShortIntArray): TShortIntArray;

**var**

ThatArray: TShortIntArray;

i, j, Left, Right, Size, Temp: ShortInt;

**begin**

Size := High(MyArray);

Left := Size;

Right := Size;

SetLength(ThatArray, Size \* 2 + 1);

ThatArray[Size] := MyArray[0];

**for** i := 1 **to** Size **do**

**begin**

Temp := MyArray[i];

**if** Temp >= MyArray[0] **then**

**begin**

Inc(Right);

j := Right;

**while** Temp < ThatArray[j - 1] **do**

**begin**

ThatArray[j] := ThatArray[j-1];

Dec(j);

**end**;

ThatArray[j] := Temp;

**end**

**else**

**begin**

Dec(Left);

j := Left;

**while** Temp > ThatArray[j + 1] **do**

**begin**

ThatArray[j] := ThatArray[j + 1];

Inc(j);

**end**;

**end**;

ThatArray[j] := Temp;

**end**;

**for** j := 0 **to** Size **do**

MyArray[j] := ThatArray[j + Left];

OrderArray := MyArray;

**end**;

**function** CheckInput(Min, Max: Integer): ShortInt;

**var**

IsCorrect: Boolean;

Number: ShortInt;

**begin**

IsCorrect := false;

**repeat**

**try**

ReadLn(Number);

**if** (Number >= Min) **and** (Number <= Max) **then**

IsCorrect := true

**else**

WriteLn('Enter number from interval ', Min, '..', Max, ':');

**except**

WriteLn('Check entered data. Enter number from interval ', Min, '..', Max,

':');

**end**;

**until** IsCorrect;

CheckInput := Number;

**end**;

**function** GetArrayFromConsole(): TShortIntArray;

**var**

i, Size: ShortInt;

MyArray: TShortIntArray;

**begin**

WriteLn('Enter size of array ', MinSize, '..', MaxSize, ':');

Size := CheckInput(MinSize, MaxSize);

SetLength(MyArray, Size);

Dec(Size);

**for** i := 0 **to** Size **do**

**begin**

WriteLn('Enter [', i + 1, '] number of array ', MinNumber, '..', MaxNumber,

':');

MyArray[i] := CheckInput(MinNumber, MaxNumber);

**end**;

GetArrayFromConsole := MyArray;

**end**;

**function** GetArray(**var** MyFile: TextFile; **const** Size: ShortInt): TShortIntArray;

**var**

i: ShortInt;

MyArray: TShortIntArray;

**begin**

SetLength(MyArray, Size);

**for** i := 0 **to** High(MyArray) **do**

Read(MyFile, MyArray[i]);

GetArray := MyArray;

**end**;

**function** GetSizeOfArray(**var** MyFile: TextFile): ShortInt;

**var**

Size, Number: ShortInt;

**begin**

Size := 0;

**repeat**

Read(MyFile, Number);

Inc(Size);

**until** Eof(MyFile);

Reset(MyFile);

GetSizeOfArray := Size;

**end**;

**function** CheckInputFile(**var** MyFile: TextFile): Boolean;

**var**

IsCorrect: boolean;

Number, Size: ShortInt;

**begin**

IsCorrect := true;

Size := 0;

**repeat**

**try**

Read(MyFile, Number);

**if** (Number <= MinNumber) **or** (Number >= MaxNumber) **then**

IsCorrect := false;

Inc(Size);

**except**

IsCorrect := false;

**end**;

**until** (Eoln(MyFile)) **or not**(IsCorrect);

**if** Size = 1 **then**

IsCorrect := false;

CheckInputFile := IsCorrect;

**end**;

**function** GetArrayFromFile(**var** MyFile: TextFile): TShortIntArray;

**var**

MyArray: TShortIntArray;

IsCorrect: Boolean;

Size, i: ShortInt;

**begin**

IsCorrect := false;

**repeat**

GetFile(MyFile);

**if not**(SeekEof(MyFile)) **then**

**if** CheckInputFile(MyFile) **then**

**if not**(Eof(MyFile)) **then**

**begin**

Write('There must be only one sequence. Try again. ');

CloseFile(MyFile);

**end**

**else**

**begin**

Reset(MyFile);

IsCorrect := true;

Size := GetSizeOfArray(MyFile);

MyArray := GetArray(MyFile, Size);

**end**

**else**

**begin**

Write('Check entered data. Enter numbers from interval ', MinNumber,

'..', MaxNumber, '. And the number of elemens must be in interval

', MinSize, '..', MaxSize, '. Try Again. ');

CloseFile(MyFile);

**end**

**else**

**begin**

Write('File is empty. Try again. ');

IsCorrect := false;

CloseFile(MyFile);

**end**;

**until** IsCorrect;

Write('The sequence is: ');

**for** i := 0 **to** High(MyArray) **do**

Write(MyArray[i], ' ');

WriteLn;

GetArrayFromFile := MyArray;

**end**;

**procedure** Main();

**var**

MyFile: TextFile;

MyArray: TShortIntArray;

**begin**

WriteLn('This program sorts a sequence of numbers using double-path inserts.', #10,

'Would you like to open the file or work in console? Press Y(Yes) or

N(No):');

**if** GetYesNoChoice() = 'Y' **then**

MyArray := GetArrayFromFile(MyFile)

**else**

MyArray := GetArrayFromConsole();

MyArray := OrderArray(MyArray);

WriteLn('Would you like to write down the answer to file? Press Y(Yes) or N(No):');

**if** GetYesNoChoice = 'Y' **then**

WriteArrayToFile(MyArray);

WriteArrayToConsole(MyArray);

WriteLn('Press "Enter" to exit the console.');

ReadLn;

**end**;

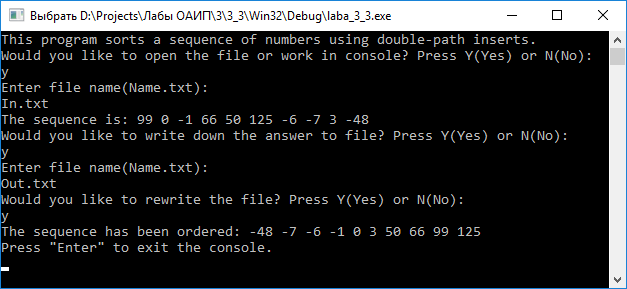
**begin**

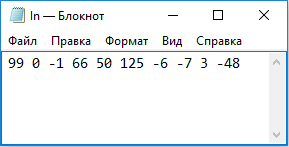
Main();

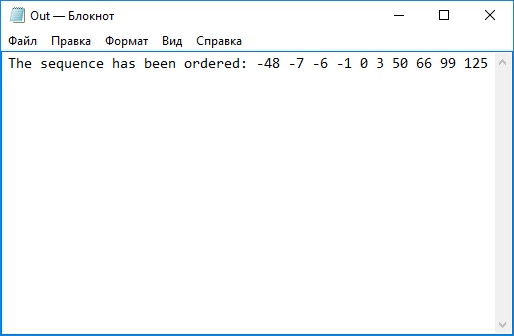
**end**.

**Скриншоты**

**(Delphi)**

****

****

****

**Схема алгоритма**





