УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №5.1

по предмету «Основы алгоритмизации и программирования»

Вариант 18

Выполнил:

Машевский Д.В

Гр. 351003

Проверил:

Данилова Г. В.

Минск 2024

**Задание:**

Разработать программу слияния двух односвязных упорядоченных по неубыванию линейных списков в один упорядоченный список, изменив только указатели в исходных списках.

**Код программы Delphi:**

**unit Unit4Laba51;**

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Clipbrd,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Vcl.StdCtrls, Vcl.Grids;

Const

CORRECT\_SYMBOLS: Set of Char = ['0' .. '9', #8];

CORRECT\_FILE\_SYMBOLS: Set of Char = ['0' .. '9', #10, #13, ' '];

MAX\_VALUE = 9999;

MIN\_VALUE = 0;

MAX\_LENGTH\_VALUE = 20;

type

PNode = ^TNode;

TNode = record

Data: Integer;

Next: PNode;

end;

Pt = ^TList;

TArr = Array of Integer;

TList = Record

Data: Integer;

Next: Pt;

Prev: Pt;

End;

TFormMainMenu = class(TForm)

EditNumListFirst: TEdit;

ButtonEnterSizeFirstList: TButton;

EditNumListSecond: TEdit;

ButtonEnterSizeSecondList: TButton;

ButtonMakeFinalList: TButton;

MainMenuFirst: TMainMenu;

PopupMenuMain: TPopupMenu;

NFile: TMenuItem;

NDeveloper: TMenuItem;

NOpenFile: TMenuItem;

NSaveFile: TMenuItem;

LabelFirstList: TLabel;

StringGridFirstList: TStringGrid;

StringGridSecondList: TStringGrid;

LabelSecondList: TLabel;

NInstruction: TMenuItem;

ButtonSorted: TButton;

StringGridThirdList: TStringGrid;

LabelGiveNaskInfo: TLabel;

procedure FormCreate(Sender: TObject);

procedure ButtonEnterSizeFirstListClick(Sender: TObject);

procedure ButtonEnterSizeSecondListClick(Sender: TObject);

procedure ButtonSortedClick(Sender: TObject);

Procedure MakeListFirst(Var X: Pt);

Procedure MakeListSecond(Var X: Pt);

procedure CheckAndEnableSortButton();

procedure ButtonMakeFinalListClick(Sender: TObject);

procedure EditNumListFirstChange(Sender: TObject);

procedure EditNumListFirstKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure EditNumListFirstKeyPress(Sender: TObject; var Key: Char);

procedure EditNumListSecondChange(Sender: TObject);

procedure NOpenFileClick(Sender: TObject);

procedure NSaveFileClick(Sender: TObject);

procedure NInstructionClick(Sender: TObject);

procedure NDeveloperClick(Sender: TObject);

procedure EditNumListSecondKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure EditNumListSecondKeyPress(Sender: TObject; var Key: Char);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

procedure StringGridFirstListMouseActivate(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y, HitTest: Integer;

var MouseActivate: TMouseActivate);

procedure StringGridSecondListSetEditText(Sender: TObject;

ACol, ARow: Integer; const Value: string);

procedure StringGridFirstListSetEditText(Sender: TObject;

ACol, ARow: Integer; const Value: string);

procedure StringGridFirstListKeyPress(Sender: TObject; var Key: Char);

procedure StringGridFirstListKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure StringGridSecondListKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FormMainMenu: TFormMainMenu;

First, Last: Pt;

implementation

uses

Unit4ДляСдачи;

{$R \*.dfm}

Function CheckZeroStr(Str: String): Boolean; External 'C:\Users\Library51.dll';

procedure MergeLists(var List1Head, List2Head: PNode); External 'C:\Users\Library51.dll';

Procedure TFormMainMenu.MakeListFirst(Var X: Pt);

Var

I, NumCount: Integer;

Y: Pt;

Begin

NumCount := StrToInt(EditNumListFirst.Text);

New(X);

First := X;

X^.Prev := Nil;

For I := 1 To NumCount Do

Begin

Y := X;

Y^.Data := StrToInt(StringGridFirstList.Cells[I, 0]);

If I <> NumCount Then

Begin

New(X);

Y^.Next := X;

X^.Prev := Y;

End

Else

Y^.Next := Nil;

End;

Last := X;

End;

Procedure TFormMainMenu.MakeListSecond(Var X: Pt);

Var

I, NumCount: Integer;

Y: Pt;

Begin

NumCount := StrToInt(EditNumListSecond.Text);

New(X);

First := X;

X^.Prev := Nil;

For I := 1 To NumCount Do

Begin

Y := X;

Y^.Data := StrToInt(StringGridSecondList.Cells[I, 0]);

If I <> NumCount Then

Begin

New(X);

Y^.Next := X;

X^.Prev := Y;

End

Else

Y^.Next := Nil;

End;

Last := X;

End;

procedure TFormMainMenu.CheckAndEnableSortButton();

Var

I, J: Integer;

AllFilled: Boolean;

Begin

AllFilled := True;

For I := 0 to StringGridFirstList.ColCount - 1 do

for J := 0 to StringGridFirstList.RowCount - 1 do

If (StringGridFirstList.Cells[I, J] = '') Then

AllFilled := False;

For I := 0 to StringGridSecondList.ColCount - 1 do

for J := 0 to StringGridSecondList.RowCount - 1 do

If (StringGridSecondList.Cells[I, J] = '') Then

AllFilled := False;

ButtonSorted.Enabled := AllFilled;

end;

procedure TFormMainMenu.EditNumListFirstChange(Sender: TObject);

Var

I, J, ArrLen: Integer;

begin

ButtonEnterSizeFirstList.Enabled := False;

ButtonEnterSizeSecondList.Enabled := False;

StringGridFirstList.Enabled := False;

MainMenuFirst.Items[0].Items[1].Enabled := False;

ArrLen := StringGridFirstList.ColCount + 1;

ArrLen := StringGridFirstList.RowCount;

If (EditNumListFirst.Text = '0') then

EditNumListFirst.Text := '';

If (EditNumListFirst.Text <> '') then

Begin

ButtonEnterSizeFirstList.Enabled := True;

StringGridFirstList.Enabled := True;

End;

If (EditNumListFirst.Text = '') then

Begin

ButtonEnterSizeFirstList.Enabled := False;

For I := 1 to ArrLen + 1 do

Begin

For J := 1 to ArrLen do

StringGridFirstList.Cells[I, J] := '';

End;

StringGridFirstList.Enabled := False;

End;

For I := 1 to ArrLen do

Begin

StringGridFirstList.Cells[I, 0] := '';

End;

end;

procedure TFormMainMenu.EditNumListFirstKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

If (Key = VK\_INSERT) and (SsShift in Shift) then

Key := 0

end;

procedure TFormMainMenu.EditNumListFirstKeyPress(Sender: TObject;

var Key: Char);

Var

Value: Integer;

chars: set of Char;

editText: String;

begin

editText := (Sender as TEdit).Text;

if (Length(editText) = 0) then

begin

chars := ['0' .. '9', #8, #31];

end

else if (editText[1] = '-') then

begin

if (Length(editText) = 1) then

chars := ['1' .. '9', #8, #31]

else

chars := ['0' .. '9', #8, #31];

end

else

chars := ['0' .. '9', #8, #31];

if EditNumListFirst.Text = '0' then

Begin

EditNumListFirst.Text := '';

End;

if not(Key in chars) then

Key := #0;

If TryStrToInt(EditNumListFirst.Text + Key, Value) Then

If (Value > MAX\_LENGTH\_VALUE) Then

Key := #0;

end;

procedure TFormMainMenu.EditNumListSecondChange(Sender: TObject);

Var

I, J, ArrLen: Integer;

begin

ArrLen := StringGridSecondList.ColCount + 1;

If (EditNumListSecond.Text = '0') then

EditNumListSecond.Text := '';

If (EditNumListSecond.Text <> '') then

Begin

ButtonEnterSizeSecondList.Enabled := True;

StringGridSecondList.Enabled := True;

End;

If (EditNumListSecond.Text = '') then

Begin

ButtonEnterSizeSecondList.Enabled := False;

For I := 1 to ArrLen + 1 do

Begin

For J := 1 to ArrLen do

StringGridSecondList.Cells[I, J] := '';

End;

StringGridSecondList.Enabled := False;

End;

For I := 1 to ArrLen do

Begin

StringGridSecondList.Cells[I, 0] := '';

End;

end;

procedure TFormMainMenu.EditNumListSecondKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

If (Key = VK\_INSERT) and (SsShift in Shift) then

Key := 0

end;

procedure TFormMainMenu.EditNumListSecondKeyPress(Sender: TObject;

var Key: Char);

Var

Value: Integer;

chars: set of Char;

editText: String;

begin

editText := (Sender as TEdit).Text;

if (Length(editText) = 0) then

begin

chars := ['0' .. '9', #8, #31];

end

else if (editText[1] = '-') then

begin

if (Length(editText) = 1) then

chars := ['1' .. '9', #8, #31]

else

chars := ['0' .. '9', #8, #31];

end

else

chars := ['0' .. '9', #8, #31];

if EditNumListSecond.Text = '0' then

Begin

EditNumListSecond.Text := '';

End;

if not(Key in chars) then

Key := #0;

If TryStrToInt(EditNumListSecond.Text + Key, Value) Then

If (Value > MAX\_LENGTH\_VALUE) Then

Key := #0;

end;

procedure TFormMainMenu.StringGridFirstListKeyDown(Sender: TObject;

var Key: Word; Shift: TShiftState);

begin

If (Key = VK\_INSERT) and (SsShift in Shift) then

Key := 0

end;

procedure TFormMainMenu.StringGridFirstListKeyPress(Sender: TObject;

var Key: Char);

Var

Value: Integer;

CellValue: String;

begin

CellValue := StringGridFirstList.Cells[StringGridFirstList.Col,

StringGridFirstList.Row];

If (Key = #22) and (GetKeyState(VK\_CONTROL) < 0) then

Key := #0;

If not(Char(Key) in CORRECT\_SYMBOLS) Then

Key := #0;

If Length(CellValue + Key) > 4 then

Key := #0;

If TryStrToInt(CellValue + Key, Value) Then

If (Value > MAX\_VALUE) Then

Key := #0;

end;

procedure TFormMainMenu.StringGridFirstListMouseActivate(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y, HitTest: Integer;

var MouseActivate: TMouseActivate);

begin

Clipboard.Clear; // Очищает содержимое буфера обмена

end;

procedure TFormMainMenu.StringGridFirstListSetEditText(Sender: TObject;

ACol, ARow: Integer; const Value: string);

Var

CellValue: Integer;

Begin

If not TryStrToInt(Value, CellValue) then

Begin

StringGridFirstList.Cells[ACol, ARow] := '';

End;

CheckAndEnableSortButton;

ButtonMakeFinalList.Enabled := False;

end;

procedure TFormMainMenu.StringGridSecondListKeyDown(Sender: TObject;

var Key: Word; Shift: TShiftState);

begin

If (Key = VK\_INSERT) and (SsShift in Shift) then

Key := 0

end;

procedure TFormMainMenu.StringGridSecondListSetEditText(Sender: TObject;

ACol, ARow: Integer; const Value: string);

Var

CellValue: Integer;

Begin

If not TryStrToInt(Value, CellValue) then

Begin

StringGridSecondList.Cells[ACol, ARow] := '';

End;

CheckAndEnableSortButton;

ButtonMakeFinalList.Enabled := False;

end;

Procedure Sort(Var X: Pt);

Var

Temp: Integer;

Swapped: Boolean;

Current, Next: Pt;

Begin

X := First;

If X <> Nil Then

Repeat

Swapped := False;

Current := X;

While Current^.Next <> Nil Do

Begin

Next := Current^.Next;

If Current^.Data > Next^.Data Then

Begin

Temp := Current^.Data;

Current^.Data := Next^.Data;

Next^.Data := Temp;

Swapped := True;

End;

Current := Next;

End;

Until Not Swapped;

End;

Procedure TFormMainMenu.ButtonSortedClick(Sender: TObject);

Var

X, Y: Pt;

I: Integer;

Begin

MakeListFirst(X);

Sort(X);

I := 1;

While X <> Nil Do

Begin

StringGridFirstList.Cells[I, 0] := IntToStr(X^.Data);

X := X^.Next;

Inc(I);

End;

MakeListSecond(Y);

Sort(Y);

I := 1;

While Y <> Nil Do

Begin

StringGridSecondList.Cells[I, 0] := IntToStr(Y^.Data);

Y := Y^.Next;

Inc(I);

End;

ButtonMakeFinalList.Enabled := True;

End;

procedure TFormMainMenu.ButtonEnterSizeFirstListClick(Sender: TObject);

Var

Number, I, NumCount: Integer;

Y: Pt;

begin

StringGridFirstList.Enabled := True;

NumCount := StrToInt(EditNumListFirst.Text);

StringGridFirstList.ColCount := NumCount + 1;

For I := 1 to NumCount do

Begin

StringGridFirstList.Cells[I, 0] := '';

End;

StringGridFirstList.Options := StringGridFirstList.Options + [goEditing];

EditNumListSecond.Enabled := True;

ButtonSorted.Enabled := False;

ButtonMakeFinalList.Enabled := False;

end;

procedure TFormMainMenu.ButtonEnterSizeSecondListClick(Sender: TObject);

Var

Number, I, NumCount: Integer;

Y: Pt;

begin

StringGridSecondList.Enabled := True;

NumCount := StrToInt(EditNumListSecond.Text);

StringGridSecondList.ColCount := NumCount + 1;

For I := 1 to NumCount do

Begin

StringGridSecondList.Cells[I, 0] := '';

End;

StringGridSecondList.Options := StringGridSecondList.Options + [goEditing];

ButtonSorted.Enabled := False;

ButtonMakeFinalList.Enabled := False;

end;

Function CheckFile(Path: String): Boolean;

Var

Fl: Boolean;

InputFile: TextFile;

Begin

Fl := True;

Try

AssignFile(InputFile, Path);

Reset(InputFile);

Except

ShowMessage('Произошла ошибка:файл занят другим процессом');

Fl := False;

End;

CloseFile(InputFile);

CheckFile := Fl;

End;

Function CheckFileMatrixValues(FileInputPath: String): Boolean;

Var

InputFile: TextFile;

Value: Char;

IsChecked, Fl: Boolean;

Counter: Integer;

IsHigher: Boolean;

Begin

IsChecked := False;

IsHigher := False;

Counter := 0;

AssignFile(InputFile, FileInputPath);

Reset(InputFile);

While (not(Eof(InputFile)) and (IsChecked = False)) do

Begin

Read(InputFile, Value);

If IsInt(Value) > 0 then

Begin

IsChecked := True;

Inc(Counter);

End;

End;

If (Counter > 0) then

Begin

ShowMessage

('В файле некорректные данные:'#10'Был найден символ,не являющийся

числом');

CheckFileMatrixValues := False;

End

Else

CheckFileMatrixValues := True;

CloseFile(InputFile);

End;

function CheckMaxValueInMatrices(const FilePath: string): Boolean;

var

F: TextFile;

Value: Integer;

Line: string;

FoundMaxValue: Boolean;

begin

Result := True;

FoundMaxValue := False;

AssignFile(F, FilePath);

Reset(F);

try

while not Eof(F) do

begin

ReadLn(F, Line);

if Line <> '' then

begin

var

Elements := Line.Split([' ']);

for var Element in Elements do

begin

if TryStrToInt(Element, Value) then

begin

if Value > MAX\_VALUE then

begin

Result := False;

FoundMaxValue := True;

end;

end;

end;

end;

if FoundMaxValue then

Break;

end;

finally

CloseFile(F);

end;

if not Result then

begin

ShowMessage

('В файле некорректные данные:'#10'Было найдено значение, превышающее

максимальное.');

CheckMaxValueInMatrices := False;

end

else

CheckMaxValueInMatrices := True;

end;

procedure TFormMainMenu.NDeveloperClick(Sender: TObject);

Var

MessageBoxCaption: String;

MessageBoxText: String;

IsMessageShow: Boolean;

Begin

IsMessageShow := False;

if not IsMessageShow then

Begin

IsMessageShow := True;

MessageBoxCaption := 'О разработчике';

MessageBoxText :=

'Машевский Даниил Витальевич, группа 351003, Лабораторная №

2.4.'#10''#10'Задача:'#10'Разработать программу слияния двух односвязных

упорядоченных по неубыванию линейных списков в один упорядоченный список,

изменив только указатели в исходных списках.';

MessageBox(Handle, PChar(MessageBoxText),

PChar(MessageBoxCaption), MB\_OK);

End;

IsMessageShow := False;

end;

procedure TFormMainMenu.NInstructionClick(Sender: TObject);

Var

MessageBoxCaption: String;

MessageBoxText: String;

IsMessageShow: Boolean;

Begin

IsMessageShow := False;

if not IsMessageShow then

Begin

IsMessageShow := True;

MessageBoxCaption := 'Инструкция';

MessageBoxText :=

'1)Размер массивов не должен превышать 20'#10'2)Размер значений в массивах

не должен превышать 9999'#10'3)Файл должен быть формата .txt'#10'4)В

файле должны быть записаны только числа'#10'5)Пример массивов в

файле:'#10'7'#10'34 2 12 34 32 21 2'#10'4'#10'3 2 3 7';

MessageBox(Handle, PChar(MessageBoxText),

PChar(MessageBoxCaption), MB\_OK);

End;

IsMessageShow := False;

end;

procedure TFormMainMenu.NOpenFileClick(Sender: TObject);

var

OpenDialog: TOpenDialog;

InputFile: TextFile;

NumArrFirst, NumArrSecond: TArr;

LenFirst, LenSecond, I: Integer;

begin

OpenDialog := TOpenDialog.Create(nil);

OpenDialog.Title := 'Выберите файл';

OpenDialog.Filter := 'Текстовые файлы (\*.txt)|\*.txt';

if OpenDialog.Execute then

begin

If CheckMaxValueInMatrices(OpenDialog.FileName) = True then

Begin

if CheckFileMatrixValues(OpenDialog.FileName) then

Begin

AssignFile(InputFile, OpenDialog.FileName);

Reset(InputFile);

ReadLn(InputFile, LenFirst);

EditNumListFirst.Text := IntToStr(LenFirst);

StringGridFirstList.ColCount := LenFirst + 1;

SetLength(NumArrFirst, LenFirst);

for I := 0 to LenFirst - 1 do

begin

Read(InputFile, NumArrFirst[I]);

StringGridFirstList.Cells[I + 1, 0] :=

IntToStr(NumArrFirst[I]);

end;

ReadLn(InputFile, LenSecond);

EditNumListSecond.Text := IntToStr(LenSecond);

StringGridSecondList.ColCount := LenSecond + 1;

SetLength(NumArrSecond, LenSecond);

for I := 0 to LenSecond - 1 do

begin

Read(InputFile, NumArrSecond[I]);

StringGridSecondList.Cells[I + 1, 0] :=

IntToStr(NumArrSecond[I]);

end;

CloseFile(InputFile);

ButtonSorted.Enabled := True;

EditNumListSecond.Enabled := True;

StringGridFirstList.Options := StringGridFirstList.Options +

[goEditing];

StringGridSecondList.Options := StringGridSecondList.Options +

[goEditing];

End;

End;

end;

end;

procedure TFormMainMenu.NSaveFileClick(Sender: TObject);

var

SaveDialog: TSaveDialog;

MText, S1, S2: String;

FileName, Path: String;

MessageBoxCaption: String;

MessageBoxText: String;

IsMessageShow: Boolean;

InputFile: TextFile;

Massiv: TArr;

I, J: Integer;

begin

// Initialize the Massiv array

J := 1;

SetLength(Massiv, StringGridThirdList.ColCount - 1);

for I := 0 to High(Massiv) do

Begin

Massiv[I] := StrToInt(StringGridThirdList.Cells[J, 0]);

Inc(J);

End;

SaveDialog := TSaveDialog.Create(nil);

SaveDialog.Title := 'Сохранить файл';

SaveDialog.Filter := 'Текстовые файлы (\*.txt)|\*.txt';

SaveDialog.DefaultExt := 'txt';

if SaveDialog.Execute then

begin

FileName := SaveDialog.FileName;

if not FileName.EndsWith('.txt') then

FileName := FileName + '.txt';

Path := FileName;

WriteStrIntoFile(Path, Massiv);

SaveDialog.Free;

IsMessageShow := False;

if not IsMessageShow then

Begin

IsMessageShow := True;

MessageBoxCaption := 'Работа с файлом:';

MessageBoxText := 'Запись в файл прошла успешно!';

MessageBox(Handle, PChar(MessageBoxText),

PChar(MessageBoxCaption), MB\_OK);

End;

IsMessageShow := False;

end;

end;

procedure TFormMainMenu.ButtonMakeFinalListClick(Sender: TObject);

var

XHead, YHead, Z: PNode;

X, Y, Temp: PNode;

I, Row: Integer;

begin

StringGridThirdList.ColCount := StringGridSecondList.ColCount +

StringGridFirstList.ColCount - 1;

New(XHead);

X := XHead;

X^.Data := StrToInt(StringGridFirstList.Cells[1, 0]);

X^.Next := nil;

for I := 2 to StringGridFirstList.ColCount - 1 do

begin

New(X^.Next);

X := X^.Next;

X^.Data := StrToInt(StringGridFirstList.Cells[I, 0]);

X^.Next := nil;

end;

New(YHead);

Y := YHead;

Y^.Data := StrToInt(StringGridSecondList.Cells[1, 0]);

Y^.Next := nil;

for I := 2 to StringGridSecondList.ColCount - 1 do

begin

New(Y^.Next);

Y := Y^.Next;

Y^.Data := StrToInt(StringGridSecondList.Cells[I, 0]);

Y^.Next := nil;

end;

// Find the last element in lists X and Y to pass to the merge function

Temp := X;

while X^.Next <> nil do

X := X^.Next;

while Y^.Next <> nil do

Y := Y^.Next;

MergeLists(XHead, YHead);

// Display the result on the screen

Z := XHead;

Row := 1;

while Z <> nil do

begin

StringGridThirdList.Cells[Row, 0] := IntToStr(Z^.Data);

Inc(Row);

Z := Z^.Next;

end;

MainMenuFirst.Items[0].Items[1].Enabled := True;

StringGridThirdList.Enabled := True;

end;

procedure TFormMainMenu.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

var

UserChoice: Integer;

Begin

CanClose := False;

UserChoice := MessageBox(Handle, 'Вы уверены, что хотите выйти?',

'Подтверждение', MB\_ICONQUESTION or MB\_YESNO);

If UserChoice = IDYES then

CanClose := True;

end;

procedure TFormMainMenu.FormCreate(Sender: TObject);

begin

StringGridFirstList.Cells[0, 0] := ' 1-й список';

StringGridSecondList.Cells[0, 0] := ' 2-й список';

StringGridThirdList.Cells[0, 0] := 'Уп.список:';

StringGridFirstList.ColWidths[0] := 100;

StringGridSecondList.ColWidths[0] := 100;

StringGridSecondList.Enabled := False;

StringGridFirstList.Enabled := False;

ButtonMakeFinalList.Enabled := False;

ButtonSorted.Enabled := False;

ButtonEnterSizeSecondList.Enabled := False;

EditNumListSecond.Enabled := False;

ButtonEnterSizeFirstList.Enabled := False;

MainMenuFirst.Items[0].Items[1].Enabled := False;

StringGridThirdList.Enabled := False;

end;

end.

**unit Unit4ДляСдачи;**

interface

uses Unit4Laba51;

type

PNode = ^TNode;

TNode = record

Data: Integer;

Next: PNode;

end;

Pt = ^TList;

TArr = Unit4Laba51.TArr;

TList = Record

Data: Integer;

Next: Pt;

Prev: Pt;

End;

Function IsInt(Value: Char): Integer;

Function ReadFirstLen(Path: String): Integer;

Procedure FillArr(Var Arr: TArr; Path: String);

Procedure WriteStrIntoFile(Path: String; Massiv: TArr);

Function CheckZeroStr(Str: String): Boolean;

var

First, Last: Pt;

implementation

Function IsInt(Value: Char): Integer;

Var

Counter: Integer;

Begin

Counter := 0;

If not(Value in CORRECT\_FILE\_SYMBOLS) Then

Inc(Counter);

IsInt := Counter;

End;

Function CheckZeroStr(Str: String): Boolean;

Var

I, Len: Integer;

Fl: Boolean;

Begin

Fl := False;

Len := Length(Str) - 1;

For I := 1 to Len do

Begin

If (Str[I] = '0') and (I <> High(Str)) then

Begin

If (Str[I + 1] <> '') then

Fl := True;

End;

End;

CheckZeroStr := Fl;

End;

Procedure WriteStrIntoFile(Path: String; Massiv: TArr);

Var

F: TextFile;

I, J: Integer;

Begin

AssignFile(F, Path);

Rewrite(F);

For I := 0 to High(Massiv) do

Begin

Write(F, Massiv[I], ' ');

End;

CloseFile(F);

End;

Procedure FillArr(Var Arr: TArr; Path: String);

Var

Value: Integer;

I: Integer;

InputFile: TextFile;

Begin

AssignFile(InputFile, Path);

Reset(InputFile);

I := 0;

While not(Eof(InputFile)) do

Begin

Read(InputFile, Value);

Arr[I] := Value;

Inc(I);

End;

CloseFile(InputFile);

End;

Function ReadFirstLen(Path: String): Integer;

Var

Value: Integer;

I: Integer;

InputFile: TextFile;

Begin

AssignFile(InputFile, Path);

Reset(InputFile);

ReadLn(InputFile, Value);

ReadFirstLen := Value;

CloseFile(InputFile);

End;

end.

**DLL:**

library Library51;

uses

System.SysUtils,

System.Classes;

type

PNode = ^TNode;

TNode = record

Data: Integer;

Next: PNode;

end;

Pt = ^TList;

TList = Record

Data: Integer;

Next: Pt;

Prev: Pt;

End;

var

First, Last: Pt;

{$R \*.res}

Function CheckZeroStr(Str: String): Boolean;

Var

I, Len: Integer;

Fl: Boolean;

Begin

Fl := False;

Len := Length(Str) - 1;

For I := 1 to Len do

Begin

If (Str[I] = '0') and (I <> High(Str)) then

Begin

If (Str[I + 1] <> '') then

Fl := True;

End;

End;

CheckZeroStr := Fl;

End;

procedure MergeLists(var List1Head, List2Head: PNode);

var

Current, Temp: PNode;

begin

if List1Head = nil then

begin

List1Head := List2Head;

Exit;

end

else if List2Head = nil then

Exit;

if List1Head^.Data <= List2Head^.Data then

begin

Current := List1Head;

List1Head := List1Head^.Next;

end

else

begin

Current := List2Head;

List2Head := List2Head^.Next;

end;

Temp := Current;

while (List1Head <> nil) and (List2Head <> nil) do

begin

if List1Head^.Data <= List2Head^.Data then

begin

Current^.Next := List1Head;

List1Head := List1Head^.Next;

end

else

begin

Current^.Next := List2Head;

List2Head := List2Head^.Next;

end;

Current := Current^.Next;

end;

if List1Head = nil then

Current^.Next := List2Head

else

Current^.Next := List1Head;

List1Head := Temp;

end;

Exports

CheckZeroStr,

MergeLists;

begin

end.

**Код программы на Java:**

import java.io.File;

import java.util.Scanner;

import java.io.FileNotFoundException;

import java.io.\*;

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;

}

}

public class Main {

static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

int vibFirst;

int vibSecond;

giveInfo();

Node listFirst;

Node listSecond;

vibFirst = inputChoice();

giveInfoAboutFirstList();

listFirst = spisokRead(vibFirst);

giveInfoAboutSecondList();

listSecond = spisokRead(vibFirst);

listFirst = insertionSortList(listFirst);

listSecond = insertionSortList(listSecond);

Node mergedList = mergeLists(listFirst, listSecond);

vibSecond = outputChoice();

vivodSum(vibSecond, mergedList);

scanner.close();

}

public static void giveInfo() {

System.out.println("Данная программа производит ввод двух списков, их

сортировку по неубыванию, а также их непосредственное

слияние.");

}

public static void giveInfoAboutFirstList() {

System.out.println("Работа с первым списком:");

}

public static void giveInfoAboutSecondList() {

System.out.println("Работа со вторым списком:");

}

public static void vivodSum(int vib, Node mergedList) {

String path;

if (vib == 0) {

printList(mergedList);

} else {

path = pTF();

writeListToFile(path, mergedList);

}

}

public static void writeListToFile(String path, Node mergedList) {

try {

PrintWriter writer = new PrintWriter(new File(path));

Node current = mergedList;

while (current != null) {

writer.print(current.data + " ");

current = current.next;

}

writer.close();

} catch (IOException e) {

System.out.println("Не удалось записать данные в файл.");

}

System.out.println("Запись в файл прошла успешно");

}

public static int outputChoice() {

int vib;

boolean isIncorrect;

vib = 0;

System.out.println("Введите 0, если хотите сделать вывод в консоль; 1, если

хотите вывод в файл. ");

do {

isIncorrect = false;

System.out.print("Выбор: ");

try {

vib = Integer.parseInt(scanner.nextLine());

} catch (Exception e) {

isIncorrect = true;

System.out.println("\nОшибка ввода. ");

}

if (!isIncorrect && (vib != 1) && (vib != 0)) {

isIncorrect = true;

System.out.println("Число должно быть либо 0, либо 1. ");

}

} while (isIncorrect);

return vib;

}

public static int enterN() {

int n;

boolean isIncorrect;

n = 0;

do {

isIncorrect = false;

System.out.print("Введите размер списка: ");

try {

n = Integer.parseInt(scanner.nextLine());

} catch (Exception e) {

isIncorrect = true;

System.out.println("Ошибка ввода. ");

}

if (!isIncorrect && (n == 0)) {

isIncorrect = true;

System.out.println("Размер списка не должен ровняться 0, повторите

ввод.");

}

} while (isIncorrect);

return n;

}

public static int readSizeFile(String path) {

int n;

boolean isIncorrect;

n = 0;

isIncorrect = false;

try {

Scanner scannerFile = new Scanner(new File(path));

n = scannerFile.nextInt();

scannerFile.close();

} catch (Exception e) {

isIncorrect = true;

System.out.println("Ошибка при считывании размера списка ");

}

if (!isIncorrect && n == 0) {

isIncorrect = true;

System.out.println("Размер списка в файле не должен ровняться 0.");

}

if (isIncorrect) {

System.out.println("Введите размер списка с консоли. ");

n = enterN();

}

return n;

}

public static Node spisokRead(int vib) {

Node listFirst = null;

int n;

String path;

if (vib == 0) {

n = enterN();

listFirst = createListFromInput(n);

}

else{

path = pTF();

n = readSizeFile(path);

listFirst = createListFromFile(path, n);

}

return listFirst;

}

public static Node createListFromFile(String filePath, int n) {

Node head = null;

Node tail = null;

int i = 0;

File file = new File(filePath);

try (Scanner scanner = new Scanner(file)) {

System.out.println("Чтение чисел из файла...");

for (i = 0; i < n; i++){

if (scanner.hasNextInt()) {

int number = scanner.nextInt();

Node newNode = new Node(number);

if (head == null) {

head = newNode;

tail = newNode;

} else {

tail.next = newNode;

tail = newNode;

}

} else {

System.out.println("Некорректное значение пропущено: " + scanner.next());

}

}

} catch (FileNotFoundException e) {

System.out.println("Файл не найден: " + filePath);

}

return head;

}

public static int inputChoice() {

int vib;

boolean isIncorrect;

vib = 0;

System.out.println("Введите 0, если хотите сделать ввод с консоли; 1, если

хотите ввод из файла. ");

do {

isIncorrect = false;

System.out.print("Выбор: ");

try {

vib = Integer.parseInt(scanner.nextLine());

} catch (Exception e) {

isIncorrect = true;

System.out.println("Ошибка ввода. ");

}

if (!isIncorrect && (vib != 1) && (vib != 0)) {

isIncorrect = true;

System.out.println("Число должно быть либо 0, либо 1. ");

}

} while (isIncorrect);

return vib;

}

public static String pTF() {

String path;

boolean isIncorrect;

do {

isIncorrect = false;

System.out.println("Укажите путь к файлу: ");

path = scanner.nextLine();

File file = new File(path);

if (!file.exists()) {

isIncorrect = true;

System.out.println("Такой файл не найден. ");

}

if (!path.endsWith(".txt")) {

isIncorrect = true;

System.out.println("У файла должно быть расширение txt. ");

}

} while (isIncorrect);

return path;

}

public static Node createListFromInput(int n) {

Node head = null;

Node tail = null;

int number = 0;

Scanner scanner = new Scanner(System.in);

for (int i = 0; i < n; i++) {

boolean isIncorrect;

do {

System.out.print("Введите число: ");

isIncorrect = false;

try {

number = Integer.parseInt(scanner.nextLine());

} catch (Exception e) {

isIncorrect = true;

System.out.println("Ошибка ввода. Пожалуйста, введите целое

число.");

}

} while (isIncorrect);

Node newNode = new Node(number);

if (head == null) {

head = newNode;

tail = newNode;

} else {

tail.next = newNode;

tail = newNode;

}

}

return head;

}

public static void printList(Node head) {

System.out.print("Список: ");

while (head != null) {

System.out.print(head.data + " ");

head = head.next;

}

System.out.println();

}

public static Node insertionSortList(Node head) {

if (head == null || head.next == null) {

return head;

}

Node dummy = new Node(0);

Node curr = head;

Node prev = dummy;

while (curr != null) {

Node next = curr.next;

while (prev.next != null && prev.next.data < curr.data) {

prev = prev.next;

}

curr.next = prev.next;

prev.next = curr;

prev = dummy;

curr = next;

}

return dummy.next;

}

public static Node mergeLists(Node list1, Node list2) {

if (list1 == null) {

return list2;

}

if (list2 == null) {

return list1;

}

if (list1.data < list2.data) {

list1.next = mergeLists(list1.next, list2);

return list1;

} else {

list2.next = mergeLists(list1, list2.next);

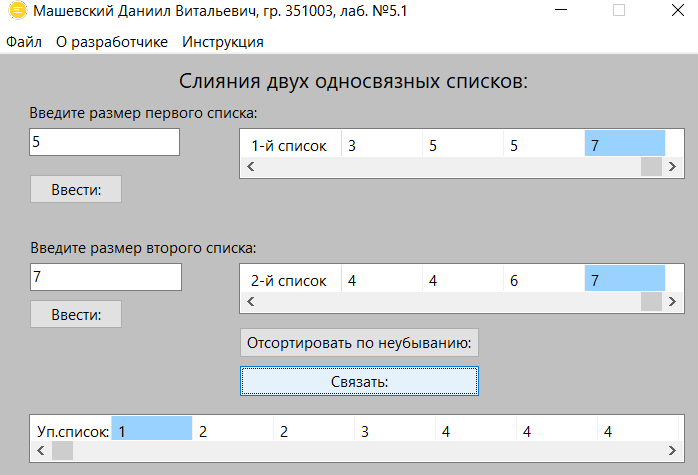
return list2;

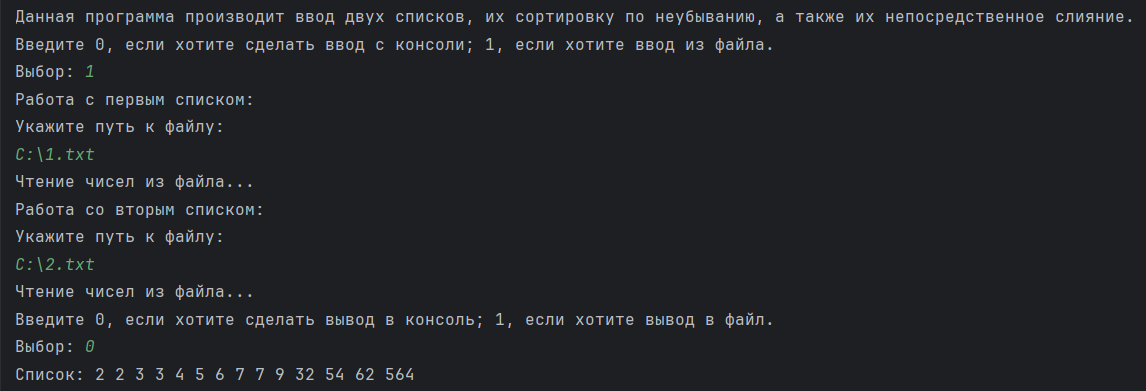
}

}

}

**Скриншоты:**





**Блок-схема:**

