УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №5.1

по предмету

Основы алгоритмизации и программирования

Вариант 14

Выполнил:

Крутько А.А.

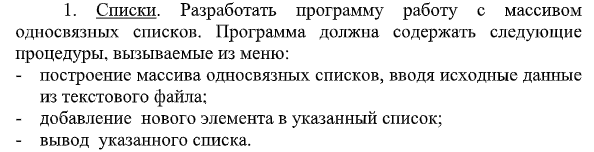
Проверила:

Данилова Г.В.

Группа 251004

Минск 2022

Задание:



Код программы на **Delphi**:

unit MainUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.CustomizeDlg, Vcl.ComCtrls,

Vcl.StdCtrls, Vcl.CheckLst, Vcl.Mask, Vcl.Buttons, Vcl.ExtCtrls, Vcl.Menus, ClipBrd,

Vcl.Grids;

type

TMainForm = class(TForm)

ChooseArr: TListBox;

AddButton: TBitBtn;

AddCount: TLabeledEdit;

PopupMenu1: TPopupMenu;

ListGrid: TStringGrid;

RadioButtonLeft: TRadioButton;

RadioButtonRight: TRadioButton;

InsertElem: TLabeledEdit;

MainMenu: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

OpenFile: TOpenDialog;

SaveFile: TSaveDialog;

procedure FormCreate(Sender: TObject);

procedure AddButtonClick(Sender: TObject);

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

Shift: TShiftState);

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

procedure AddCountChange(Sender: TObject);

procedure ChooseArrClick(Sender: TObject);

procedure ChooseArrDblClick(Sender: TObject);

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

Shift: TShiftState);

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

procedure N4Click(Sender: TObject);

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

Shift: TShiftState);

procedure N5Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure N2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

Type

PList = ^TList;

TList = Record

Data: Integer;

Next: PList;

End;

TArrCount = Array of Integer;

TArrList = Array of PList;

TIPEGrid = Class(TStringGrid);

Var

ArrCount: TArrCount;

ArrList: TArrList;

TempCol: Integer;

var

MainForm: TMainForm;

implementation

Uses InputUnit;

{$R \*.dfm}

procedure TMainForm.AddCountChange(Sender: TObject);

Var

Str: String;

Numb: Integer;

IsCorrect: Boolean;

begin

AddButton.Enabled := Not(''.IsNullOrWhiteSpace(AddCount.Text));

ChooseArr.Visible := False;

ListGrid.Visible := False;

N5.Enabled := False;

end;

Procedure CheckBuffer(Sender: TObject; Var Key: Word; Shift: TShiftState; LowerNumb, UpperNumb, MaxLen: Integer);

Var

TempBeforeCursor, TempAfterCursor: String;

Begin

If (Shift = [ssShift]) and (Key = VK\_INSERT) then

If Clipboard.HasFormat(CF\_TEXT) then //?

begin

Try

StrToInt(ClipBoard.AsText);

TempBeforeCursor := Copy(TEdit(Sender).Text, 1, TEdit(Sender).SelStart);

TempAfterCursor := Copy(TEdit(Sender).Text, TEdit(Sender).SelStart + 1, TEdit(Sender).GetTextLen);

If (TEdit(Sender).SelStart = 0) and (StrToInt(ClipBoard.AsText + TempAfterCursor) < LowerNumb) or (StrToInt(ClipBoard.AsText + TempAfterCursor) > UpperNumb) Then

raise Exception.Create('Error Message');

If ((TEdit(Sender).SelStart > 0) and (TEdit(Sender).SelStart < MaxLen - 1)) and (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) < LowerNumb) or (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) > UpperNumb) Then

raise Exception.Create('Error Message');

If (TEdit(Sender).SelStart = MaxLen - 1) and (StrToInt(TempBeforeCursor + ClipBoard.AsText) < LowerNumb) or (StrToInt(TempBeforeCursor + ClipBoard.AsText) > UpperNumb) Then

raise Exception.Create('Error Message');

If (Length(ClipBoard.AsText) > 0) and (ClipBoard.AsText[1] = '0') Then

raise Exception.Create('Error Message');

If (TEdit(Sender).GetTextLen > 0) and ((TEdit(Sender).SelStart = 0) or (TEdit(Sender).Text[1] = '0')) Then

raise Exception.Create('Error Message');

Except

Key := 0;

End;

end;

End;

procedure TMainForm.AddCountKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

CheckBuffer(Sender, Key, Shift, 1, 9, 1);

end;

procedure TMainForm.AddCountKeyPress(Sender: TObject; var Key: Char);

Var

Count: Integer;

begin

Count := AddCount.SelStart;

If (TEdit(Sender).SelLength > 0) and (Key In ['1'..'9', #08]) Then

TEdit(Sender).SelText := Key;

//Temp := Copy(AddCount.Text, 1, AddCount.SelStart);

If Not(Key In ['0'..'9', #08]) Then

Key := #0;

If (Key <> #0) and (Key <> #08) and ((StrToInt(AddCount.Text + Key) < 1) or (StrToInt(AddCount.Text + Key) > 9)) Then

Key := #0;

end;

Procedure ShowInfo();

Var

I: Integer;

TempPt: PList;

Begin

I := 0;

TempPt := ArrList[MainForm.ChooseArr.ItemIndex];

While TempPt <> Nil do

Begin

MainForm.ListGrid.Cells[I, 0] := IntToStr(TempPt^.Data);

Inc(I);

TempPt := TempPt^.Next;

End;

End;

procedure TMainForm.ChooseArrClick(Sender: TObject);

begin

If ArrCount[ChooseArr.ItemIndex] = 0 Then

Begin

InputForm.BitBtn2.Caption := 'Добавить';

InputForm.InBound.EditLabel.Caption := 'Размер ' + IntToStr(ChooseArr.ItemIndex + 1) + ' списка:';

InputForm.InBound.Text := '';

InputForm.CreateGrid.Visible := False;

InputForm.BitBtn2.Visible := False;

InputForm.CreateGrid.Rows[0].Clear;

InputForm.ShowModal;

ListGrid.ColCount := ArrCount[ChooseArr.ItemIndex];

ShowInfo;

End

Else

Begin

ListGrid.ColCount := ArrCount[ChooseArr.ItemIndex];

ShowInfo;

MainForm.ListGrid.Visible := True;

MainForm.RadioButtonLeft.Visible := True;

MainForm.RadioButtonRight.Visible := True;

MainForm.InsertElem.Visible := True;

End;

end;

procedure TMainForm.ChooseArrDblClick(Sender: TObject);

begin

InputForm.InBound.Text := IntToStr(ListGrid.ColCount);

InputForm.CreateGrid.ColCount := ListGrid.ColCount;

InputForm.BitBtn2.Caption := 'Изменить';

For Var I := 0 to ListGrid.ColCount - 1 do

InputForm.CreateGrid.Cells[I,0] := ListGrid.Cells[I,0];

InputForm.BitBtn2.Visible := True;

InputForm.CreateGrid.Visible := True;

InputForm.ShowModal;

ChooseArr.OnClick(Sender);

end;

procedure TMainForm.AddButtonClick(Sender: TObject);

begin

ChooseArr.Items.Clear;

SetLength(ArrCount, StrToInt(AddCount.Text));

SetLength(ArrList, StrToInt(AddCount.Text));

For Var I := 0 to High(ArrCount) do

ArrCount[I] := 0;

For Var I := 1 to StrToInt(AddCount.Text) do

ChooseArr.Items.Add(IntToStr(I));

AddButton.Enabled := False;

ChooseArr.Height := ChooseArr.ItemHeight \* StrToInt(AddCount.Text) + 4;

ChooseArr.Visible := True;

end;

procedure AdjustResolution(oForm:TForm);

var

iPercentage:integer;

begin

if Screen.Width > 640 then

begin

iPercentage:=Round(((Screen.Width-640)/640)\*100)+100;

oForm.ScaleBy(iPercentage,100);

end;

end;

procedure TMainForm.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

AddButton.Enabled := False;

//AdjustResolution(Self);

end;

procedure TMainForm.InsertElemKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

CheckBuffer(Sender, Key, Shift, 0, 99999, 5);

end;

procedure TMainForm.InsertElemKeyPress(Sender: TObject; var Key: Char);

begin

If (InsertElem.GetTextLen = 5) and (InsertElem.SelLength > 0) and (Key In ['0'..'9', #08]) Then

Begin

If Key = #08 Then

InsertElem.SelText := ''

Else

InsertElem.SelText := Key;

Key := #0;

End;

If (InsertElem.Text = '0') and (InsertElem.SelLength > 0) and (Key In ['0'..'9', #08]) Then

Begin

If Key = #08 Then

InsertElem.SelText := ''

Else

InsertElem.SelText := Key;

Key := #0;

End;

If Not(Key In ['0'..'9', #08]) Then

Key := #0;

If (InsertElem.GetTextLen = 5) and (Key <> #08) Then

Key := #0;

If (InsertElem.Text = '0') and (Key <> #08) Then

Key := #0;

end;

Procedure InsertData(Head: PList; Temp: Integer);

StdCall;

External 'lib\_5\_1.dll';

Procedure InsertIntoListGridLeft();

Var

I: Integer;

TempPt: PList;

Begin

I := 0;

TempPt := ArrList[MainForm.ChooseArr.ItemIndex];

If MainForm.ListGrid.Col <> 0 Then

Begin

While I <> MainForm.ListGrid.Col - 1 do

Begin

Inc(I);

TempPt := TempPt^.Next;

End;

InsertData(TempPt, StrToInt(MainForm.InsertElem.Text));

Inc(ArrCount[MainForm.ChooseArr.ItemIndex]);

MainForm.ListGrid.ColCount := ArrCount[MainForm.ChooseArr.ItemIndex];

ShowInfo();

MainForm.InsertElem.Text := '';

End

Else

Begin

New(TempPt);

TempPt^.Next := ArrList[MainForm.ChooseArr.ItemIndex];

TempPt^.Data := StrToInt(MainForm.InsertElem.Text);

ArrList[MainForm.ChooseArr.ItemIndex] := TempPt;

Inc(ArrCount[MainForm.ChooseArr.ItemIndex]);

MainForm.ListGrid.ColCount := ArrCount[MainForm.ChooseArr.ItemIndex];

ShowInfo();

MainForm.InsertElem.Text := '';

End;

End;

Procedure InsertIntoListGridRight();

Var

I: Integer;

TempPt: PList;

Begin

I := 0;

TempPt := ArrList[MainForm.ChooseArr.ItemIndex];

While I <> MainForm.ListGrid.Col do

Begin

Inc(I);

TempPt := TempPt^.Next;

End;

InsertData(TempPt, StrToInt(MainForm.InsertElem.Text));

Inc(ArrCount[MainForm.ChooseArr.ItemIndex]);

MainForm.ListGrid.ColCount := ArrCount[MainForm.ChooseArr.ItemIndex];

ShowInfo();

MainForm.InsertElem.Text := '';

End;

procedure TMainForm.ListGridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

If (ListGrid.ColCount < 20) and (Length(InsertElem.Text) > 0) and (((ssCtrl in Shift) and (Key = Ord('V'))) or ((ssShift In Shift) and (Key = VK\_Insert))) Then

Begin

If RadioButtonLeft.Checked Then

InsertIntoListGridLeft();

If RadioButtonRight.Checked Then

InsertIntoListGridRight();

End;

{Try

TIPEGrid(CreateGrid).InplaceEditor.ReadOnly := ((Shift = [ssShift]) and (Key = VK\_Insert)) or (Shift = [ssCtrl]) or (Shift = [ssAlt]);

Except

End;

{If (Shift = [ssShift]) or (Shift = [ssCtrl]) Then

CreateGrid.Options := CreateGrid.Options - [goEditing]; }

end;

Function CreateList(Count: Integer; ArrCount: TArrCount): PList;

stdCall;

External 'lib\_5\_1.dll';

procedure Split(Delimiter: Char; Str: string; ListOfStrings: TStringList);

stdCall;

External 'lib\_5\_1.dll';

Function CheckSymbolInTextFile(Temp: String): String;

stdCall;

External 'lib\_5\_1.dll';

procedure TMainForm.N2Click(Sender: TObject);

begin

MessageBox(MainForm.Handle,

'Основное:'#13#10 +

'1. Сначала занести кол-во списков'#13#10 +

'2. Затем, нажав на список в первый раз, будет открыто окно по созданию списка'#13#10 +

'3. Дальше один клик – его просмотр, два клика – редактирование'#13#10 +

'4. Вставка элемента производится при введенном числе и выбранном способе вставки: слева/справа', 'Помощь', MB\_OK);

end;

procedure TMainForm.N3Click(Sender: TObject);

begin

ShowMessage('Студент группы 251004, Крутько Андрей Алексеевич');

end;

Procedure ReadSizeOfList(Var FileInput: TextFile; Var Size: Integer);

stdCall;

External 'lib\_5\_1.dll';

Procedure SetLenForRecords(Var ArrCount: TArrCount; Var ArrList: TArrList; Size: Integer);

stdCall;

External 'lib\_5\_1.dll';

Procedure GetNeedableInfo(Var FileInput: TextFile; Var Size: Integer; Var Count: Integer; Var ListTemp: TStringList; Var Pt: PList; Var ArrCount: TArrCount; Var ArrList: TArrList);

stdCall;

External 'lib\_5\_1.dll';

procedure TMainForm.N4Click(Sender: TObject);

Var

FileInput: TextFile;

Temp: String;

Size: Integer;

Count: Integer;

ListTemp: TStringList;

Pt: PList;

begin

If OpenFile.Execute Then

Begin

Try

Try

AssignFile(FileInput, OpenFile.FileName);

Reset(FileInput);

ReadSizeOfList(FileInput, Size);

If (Size < 1) or (Size > 9) Then

raise Exception.Create('Большой размер для количества массивов');

SetLenForRecords(ArrCount, ArrList, Size);

ChooseArr.Items.Clear;

For Var I := 1 to Length(ArrList) do

ChooseArr.Items.Add(IntToStr(I));

Count := 0;

ListTemp := TStringList.Create;

While Not Eof(FileInput) or (Count < Length(ArrCount)) do

Begin

GetNeedableInfo(FileInput, Size, Count, ListTemp, Pt, ArrCount, ArrList);

If ListTemp.Count = ArrCount[Count] Then

Begin

For Var I := 0 to ListTemp.Count - 1 do

Begin

Pt^.Data := StrToInt(CheckSymbolInTextFile(ListTemp[I]));

Pt := Pt^.Next;

End;

End

Else

raise Exception.Create('Несовпадение элементов с размерностью списка');

Inc(Count);

End;

AddCount.Text := IntToStr(Length(ArrCount));

ChooseArr.Visible := True;

ChooseArr.Height := ChooseArr.ItemHeight \* StrToInt(AddCount.Text) + 4;

AddButton.Enabled := False;

MainForm.N5.Enabled := True;

Except

on E : Exception do

Begin

ShowMessage('Поднята ошибка, с сообщением : ' + E.Message);

SetLength(ArrCount, 0);

For Var I := 0 to High(ArrList) do

ArrList[I] := Nil;

AddCount.Text := '';

ChooseArr.Visible := False;

MainForm.ListGrid.Visible := False;

MainForm.RadioButtonLeft.Visible := False;

MainForm.RadioButtonRight.Visible := False;

MainForm.InsertElem.Visible := False;

MainForm.N5.Enabled := False;

End;

End;

Finally

CloseFile(FileInput);

End;

End;

end;

Procedure WriteInFile(Var FileOutput: TextFile);

Begin

For Var I := 0 to High(ArrList) do

Begin

If ArrList[I] = Nil Then

Continue;

Var Temp := ArrList[I];

Writeln(FileOutput, 'Длина ' + IntToStr(I + 1) + ' списка = ' + IntToStr(ArrCount[I]));

While Temp <> Nil do

Begin

Write(FileOutput, IntToStr(Temp^.Data) + ' ');

Temp := Temp^.Next;

End;

Writeln(FileOutput);

End;

End;

procedure TMainForm.N5Click(Sender: TObject);

Var

FileOutput: TextFile;

begin

If SaveFile.Execute Then

Begin

Try

Try

AssignFile(FileOutput, SaveFile.FileName);

ReWrite(FileOutput);

WriteInFile(FileOutput);

Except

On E: Exception do

ShowMessage('Проверьте файл, ошибка ' + E.Message);

End;

Finally

CloseFile(FileOutput);

End;

End;

end;

end.

unit InputUnit;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls, Vcl.Buttons, ClipBrd,

Vcl.Grids, Vcl.Menus;

type

TInputForm = class(TForm)

InBound: TLabeledEdit;

BitBtn2: TBitBtn;

CreateGrid: TStringGrid;

Button1: TButton;

PopupMenu1: TPopupMenu;

procedure FormCreate(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

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

procedure CreateGridSetEditText(Sender: TObject; ACol, ARow: Integer;

const Value: string);

procedure FormShow(Sender: TObject);

procedure InBoundChange(Sender: TObject);

procedure Button1Click(Sender: TObject);

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

Shift: TShiftState);

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

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

Shift: TShiftState);

private

{ Private declarations }

public

{ Public declarations }

end;

var

InputForm: TInputForm;

implementation

Uses MainUnit;

Type

TIPEGrid = Class(TStringGrid);

{$R \*.dfm}

Function CreateList(): PList;

Var

Head, X, Y: PList;

I: Integer;

Begin

New(Y);

I := 0;

Head := Y;

While I <> ArrCount[MainForm.ChooseArr.ItemIndex] Do

Begin

X := Y;

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

New(Y);

X^.Next := Y;

Inc(I);

End;

X^.Next := Nil;

Result := Head;

End;

procedure TInputForm.BitBtn2Click(Sender: TObject);

begin

ArrCount[MainForm.ChooseArr.ItemIndex] := CreateGrid.ColCount;

ArrList[MainForm.ChooseArr.ItemIndex] := CreateList;

MainForm.ListGrid.ColCount := ArrCount[MainForm.ChooseArr.ItemIndex];

MainForm.ListGrid.Visible := True;

MainForm.RadioButtonLeft.Visible := True;

MainForm.RadioButtonRight.Visible := True;

MainForm.InsertElem.Visible := True;

MainForm.N5.Enabled := True;

InputForm.Close;

end;

procedure TInputForm.Button1Click(Sender: TObject);

begin

CreateGrid.ColCount := StrToInt(InBound.Text);

CreateGrid.Visible := True;

BitBtn2.Visible := True;

Button1.Enabled := False;

end;

Procedure CheckBufferForGrid(Sender: TObject; Var Key: Word; Shift: TShiftState; LowerNumb, UpperNumb, MaxLen: Integer);

Var

Edit: TInplaceEdit;

TempBeforeCursor, TempAfterCursor: String;

Begin

Edit := TIPEGrid(TStringGrid(Sender)).InplaceEditor;

If (Shift = [ssShift]) and (Key = VK\_INSERT) then

If Clipboard.HasFormat(CF\_TEXT) then //?

begin

Try

StrToInt(ClipBoard.AsText);

TempBeforeCursor := Copy(Edit.Text, 1, Edit.SelStart);

TempAfterCursor := Copy(Edit.Text, Edit.SelStart + 1, Edit.GetTextLen);

If (Edit.SelStart = 0) and (StrToInt(ClipBoard.AsText + TempAfterCursor) < LowerNumb) or (StrToInt(ClipBoard.AsText + TempAfterCursor) > UpperNumb) Then

raise Exception.Create('Error Message');

If ((Edit.SelStart > 0) and (Edit.SelStart < MaxLen - 1)) and (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) < LowerNumb) or (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) > UpperNumb) Then

raise Exception.Create('Error Message');

If (Edit.SelStart = MaxLen - 1) and (StrToInt(TempBeforeCursor + ClipBoard.AsText) < LowerNumb) or (StrToInt(TempBeforeCursor + ClipBoard.AsText) > UpperNumb) Then

raise Exception.Create('Error Message');

If (Length(ClipBoard.AsText) > 0) and (ClipBoard.AsText[1] = '0') Then

raise Exception.Create('Error Message');

If (Edit.GetTextLen > 0) and ((Edit.SelStart = 0) or (Edit.Text[1] = '0')) Then

raise Exception.Create('Error Message');

Except

Key := 0;

End;

end;

End;

procedure TInputForm.CreateGridKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

CheckBufferForGrid(Sender, Key, Shift, 0, 99999, 5);

end;

procedure TInputForm.CreateGridKeyPress(Sender: TObject; var Key: Char);

Var

SelLen: Integer;

begin

SelLen := TIPEGrid(CreateGrid).InplaceEditor.SelLength;

If (Length(CreateGrid.Cells[CreateGrid.Col, 0]) = 5) and (SelLen > 0) and (Key In ['0'..'9', #08]) Then

Begin

If Key = #08 Then

TIPEGrid(CreateGrid).InplaceEditor.SelText := ''

Else

TIPEGrid(CreateGrid).InplaceEditor.SelText := Key;

Key := #0;

End;

If (CreateGrid.Cells[CreateGrid.Col, 0] = '0') and (SelLen > 0) and (Key In ['0'..'9', #08]) Then

Begin

If Key = #08 Then

TIPEGrid(CreateGrid).InplaceEditor.SelText := ''

Else

TIPEGrid(CreateGrid).InplaceEditor.SelText := Key;

Key := #0;

End;

If Not(Key In ['0'..'9', #08]) Then

Key := #0;

If (Length(CreateGrid.Cells[CreateGrid.Col, 0]) = 5) and (Key <> #08) Then

Key := #0;

If (CreateGrid.Cells[CreateGrid.Col, 0] = '0') and (Key <> #08) Then

Key := #0;

end;

Function CheckGrid(): Boolean;

Var

I: Integer;

IsCorrect: Boolean;

Begin

I := 0;

IsCorrect := True;

While IsCorrect and (I < InputForm.CreateGrid.ColCount) do

Begin

If (Length(InputForm.CreateGrid.Cells[I, 0]) = 0) or (InputForm.CreateGrid.Cells[I, 0] = '-') Then

IsCorrect := False;

Inc(I);

End;

Result := IsCorrect;

End;

procedure TInputForm.CreateGridSetEditText(Sender: TObject; ACol, ARow: Integer;

const Value: string);

begin

BitBtn2.Enabled := CheckGrid;

end;

procedure TInputForm.FormCreate(Sender: TObject);

begin

BorderIcons := BorderIcons - [biMaximize] + [biHelp];

BorderStyle := BsSingle;

Position := PoDesktopCenter;

end;

procedure TInputForm.FormShow(Sender: TObject);

begin

Button1.Enabled := False;

BitBtn2.Enabled := CheckGrid;

end;

procedure TInputForm.InBoundChange(Sender: TObject);

begin

Button1.Enabled := Not(((Length(InBound.Text) > 0) and (InBound.Text[1] = '0')) xor ''.IsNullOrWhiteSpace(InBound.Text));

CreateGrid.Visible := False;

CreateGrid.Rows[0].Clear;

BitBtn2.Visible := False;

end;

Procedure CheckBuffer(Sender: TObject; Var Key: Word; Shift: TShiftState; LowerNumb, UpperNumb, MaxLen: Integer);

Var

TempBeforeCursor, TempAfterCursor: String;

Begin

If (Shift = [ssShift]) and (Key = VK\_INSERT) then

If Clipboard.HasFormat(CF\_TEXT) then //?

begin

Try

StrToInt(ClipBoard.AsText);

TempBeforeCursor := Copy(TEdit(Sender).Text, 1, TEdit(Sender).SelStart);

TempAfterCursor := Copy(TEdit(Sender).Text, TEdit(Sender).SelStart + 1, TEdit(Sender).GetTextLen);

If (TEdit(Sender).SelStart = 0) and (StrToInt(ClipBoard.AsText + TempAfterCursor) < LowerNumb) or (StrToInt(ClipBoard.AsText + TempAfterCursor) > UpperNumb) Then

raise Exception.Create('Error Message');

If ((TEdit(Sender).SelStart > 0) and (TEdit(Sender).SelStart < MaxLen - 1)) and (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) < LowerNumb) or (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) > UpperNumb) Then

raise Exception.Create('Error Message');

If (TEdit(Sender).SelStart = MaxLen - 1) and (StrToInt(TempBeforeCursor + ClipBoard.AsText) < LowerNumb) or (StrToInt(TempBeforeCursor + ClipBoard.AsText) > UpperNumb) Then

raise Exception.Create('Error Message');

//If (Length(ClipBoard.AsText) + TEdit(Sender).GetTextLen > MaxLen) or ( (TEdit(Sender).GetTextLen <> 0) and (TEdit(Sender).Text[1] = '0') ) Then

//raise Exception.Create('Error Message');

If (Length(ClipBoard.AsText) > 0) and (ClipBoard.AsText[1] = '0') Then

raise Exception.Create('Error Message');

If (TEdit(Sender).GetTextLen > 0) and ((TEdit(Sender).SelStart = 0) or (TEdit(Sender).Text[1] = '0')) Then

raise Exception.Create('Error Message');

Except

Key := 0;

End;

end;

End;

procedure TInputForm.InBoundKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

CheckBuffer(Sender, Key, Shift, 1, 20, 2);

end;

procedure TInputForm.InBoundKeyPress(Sender: TObject; var Key: Char);

Var

TempBeforeCursor, TempAfterCursor: String;

Edit: TEdit;

CursorPosition: Integer;

NewCursorPosition: Integer;

TempAll, TempSelected: String;

begin

If Not(Key In ['0'..'9', #08]) Then

Key := #0;

NewCursorPosition := Length(TEdit(Sender).Text) - Length(TEdit(Sender).SelText);

TempSelected := TEdit(Sender).SelText;

TempAll := TEdit(Sender).Text;

CursorPosition := TEdit(Sender).SelStart;

If (TEdit(Sender).SelStart = 0) and (TEdit(Sender).SelLength < TEdit(Sender).GetTextLen) and (Key = '0') Then

Key := #0;

If (Key <> #0) and (TempSelected <> '') Then

Begin

Try

Delete(TempAll, CursorPosition + 1, Length(TempSelected));

Insert(Key, TempAll, CursorPosition + 1);

If (StrToInt(TempAll) < 1) or (StrToInt(TempAll) > 20) Then

Key := #0

Else

Begin

TEdit(Sender).Text := TempAll;

TEdit(Sender).SelStart := NewCursorPosition + 1;

Key := #0;

End;

Except

Key := #0;

End;

End;

TempBeforeCursor := Copy(TEdit(Sender).Text, 1, TEdit(Sender).SelStart);

TempAfterCursor := Copy(TEdit(Sender).Text, TEdit(Sender).SelStart + 1, TEdit(Sender).GetTextLen);

If (TEdit(Sender).SelStart = 0) and (Key <> #0) and (Key <> #08) and ((StrToInt(Key + TempAfterCursor) < 1) or (StrToInt(Key + TempAfterCursor) > 20)) Then

Key := #0;

//SelStart идет с 0

If ((TEdit(Sender).SelStart > 0) and (TEdit(Sender).SelStart < 4)) and (Key <> #0) and (Key <> #08) and ((StrToInt(TempBeforeCursor + Key + TempAfterCursor) < 1) or (StrToInt(TempBeforeCursor + Key + TempAfterCursor) > 20)) Then

Key := #0;

If (TEdit(Sender).SelStart = 5) and (Key <> #0) and (Key <> #08) and ((StrToInt(TempBeforeCursor + Key) < 1) or (StrToInt(TempBeforeCursor + Key) > 20)) Then

Key := #0;

If (TEdit(Sender).GetTextLen > 0) and (Key <> #0) and (Key <> #08) and (TEdit(Sender).Text[1] = '0') Then

Key := #0;

end;

end.

library lib\_5\_1;

uses

System.SysUtils,

System.Classes;

{$R \*.res}

Type

PList = ^TList;

TList = Record

Data: Integer;

Next: PList;

End;

TArrCount = Array of Integer;

TArrList = Array of PList;

procedure Split(Delimiter: Char; Str: string; ListOfStrings: TStringList); StdCall;

begin

ListOfStrings.Clear;

ListOfStrings.Delimiter := Delimiter;

ListOfStrings.StrictDelimiter := True; // Requires D2006 or newer.

ListOfStrings.DelimitedText := Str;

end;

Function CheckSymbolInTextFile(Temp: String): String; StdCall;

Begin

If (Temp[1] = '0') and (Length(Temp) <> 1) Then

Temp := '';

For Var I := 1 to Length(Temp) do

Begin

If (Temp[I] > #57) or (Temp[I] < #48) Then

Begin

Temp := '';

Break;

End;

End;

Result := Temp;

End;

Function CreateList(Count: Integer; ArrCount: TArrCount): PList; stdCall;

Var

Head, X, Y: PList;

I: Integer;

Begin

New(Y);

I := 0;

Head := Y;

While I <> ArrCount[Count] Do

Begin

X := Y;

New(Y);

X^.Next := Y;

Inc(I);

End;

X^.Next := Nil;

Result := Head;

End;

Procedure SetLenForRecords(Var ArrCount: TArrCount; Var ArrList: TArrList; Size: Integer); StdCall;

Begin

SetLength(ArrCount, Size);

SetLength(ArrList, Size);

For Var I := 0 to High(ArrCount) do

ArrCount[I] := 0;

End;

Procedure ReadSizeOfList(Var FileInput: TextFile; Var Size: Integer); StdCall;

Var

Temp: String;

Begin

Readln(FileInput, Temp);

Size := StrToInt(CheckSymbolInTextFile(Temp));

End;

Procedure GetNeedableInfo(Var FileInput: TextFile; Var Size: Integer; Var Count: Integer; Var ListTemp: TStringList; Var Pt: PList; Var ArrCount: TArrCount; Var ArrList: TArrList); stdCall;

Var

Temp: String;

Begin

Readln(FileInput, Temp);

Size := StrToInt(CheckSymbolInTextFile(Temp));

ArrCount[Count] := Size;

ArrList[Count] := CreateList(Count, ArrCount);

Readln(FileInput, Temp);

Split(' ', Temp, ListTemp);

Pt := ArrList[Count];

End;

Procedure InsertData(Head: PList; Temp: Integer); stdCall;

Var

TempPt, NewPt: PList;

Begin

TempPt := Head;

New(NewPt);

NewPt^.Next := TempPt^.Next;

NewPt^.Data := Temp;

TempPt^.Next := NewPt;

End;

Exports InsertData, GetNeedableInfo, ReadSizeOfList, SetLenForRecords, CreateList, CheckSymbolInTextFile, Split;

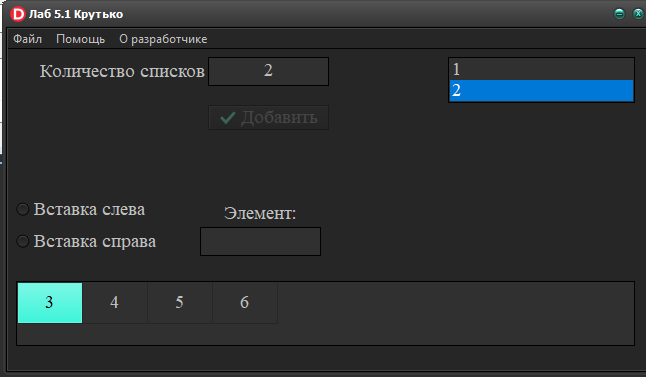
begin

end.

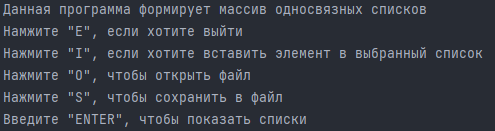
Код программы на **C#**:

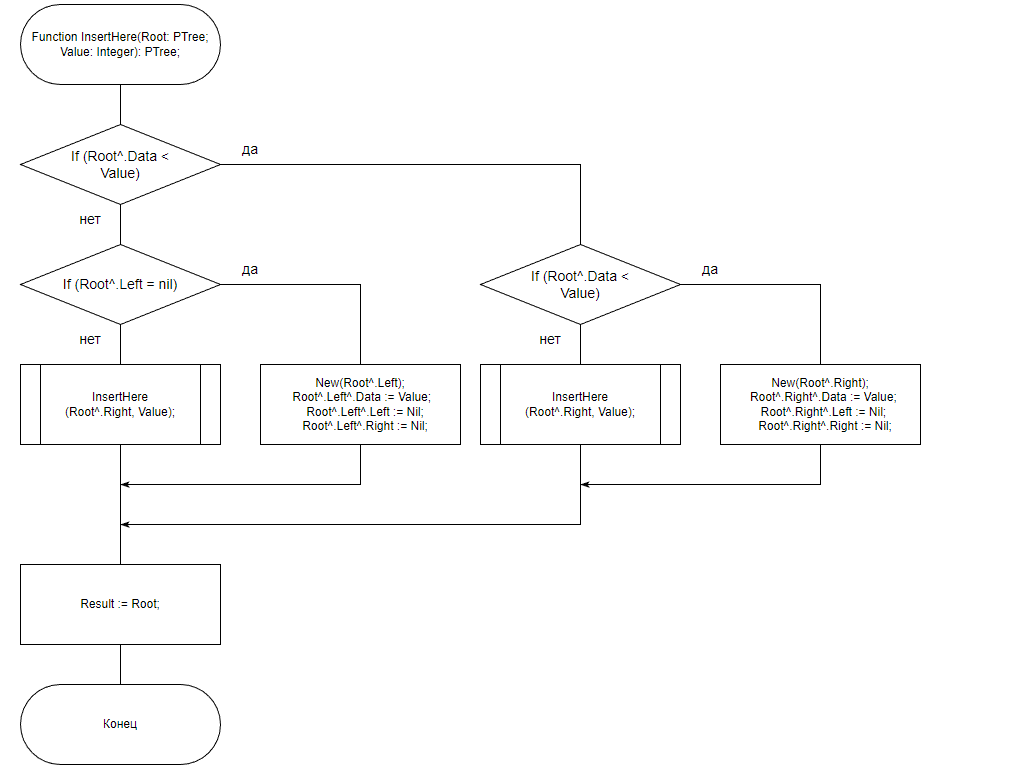
// See https://aka.ms/new-console-template for more information  
  
void InfoInMainMenu()  
{  
 Console.WriteLine("Данная программа формирует массив односвязных списков\n" +  
 "Намжите \"E\", если хотите выйти\nНажмите \"I\", если хотите вставить элемент в выбранный список\nНажмите \"O\", чтобы открыть файл\nНажмите \"S\", чтобы сохранить в файл\nВведите \"ENTER\", чтобы показать списки");  
}  
  
void InfoForHelp()  
{  
 Console.WriteLine("\nВведите \"H\", если хотите вывести помощь по меню");  
}  
  
void MessageAfterAction()  
{  
 Console.WriteLine("Вы вернулись в главное меню. Ожидается ваше следующее действие");  
}  
  
void InputInfo(ref NodeList current, int len, int i)  
{  
 Console.WriteLine("Введите элементы в строку через один пробел:");  
 bool isInCorrect;  
 do {  
 isInCorrect = true;  
 var strArr = Console.ReadLine().Split(' ');  
 if (strArr.Length != len)  
 {  
 isInCorrect = false;  
 Console.WriteLine("Проверьте ввод");  
 }  
 for (int j = 0; j < strArr.Length && isInCorrect; j++)  
 {  
 isInCorrect = int.TryParse(strArr[j], out var temp);  
 if (isInCorrect)  
 current.AddToEnd(temp);  
 else  
 {  
 Console.WriteLine("Проверьте ввод");  
 current.Clear();  
 }  
  
 }  
 } while (!isInCorrect);  
}  
  
int GetNumber(int min, int max)  
{  
 var input = -1;  
 do {  
 try  
 {  
 input = Convert.ToInt32(Console.ReadLine());  
 if (input < min || input > max)   
 Console.WriteLine("Неправильный диапазон данных");  
 }  
 catch (Exception)  
 {  
 Console.Error.WriteLine("Некорректный ввод");   
 }  
 } while (input < min || input > max);  
 return input;  
}  
  
int GetInfo(int min, int max)  
{  
 var input = -1;  
 bool isIncorrect;  
 do  
 {  
 isIncorrect = false;  
 try  
 {  
 input = Convert.ToInt32(Console.ReadLine());  
 }  
 catch (Exception)  
 {  
 isIncorrect = true;  
 Console.Error.WriteLine("Некорректный ввод");   
 }  
 } while (isIncorrect);  
 return input;  
}  
  
void InsertInfoInList (ref NodeList[] massivList)   
{  
 Console.WriteLine("Введите индекс массива:");  
 int a = GetNumber(1, massivList.Length);  
 Console.WriteLine("Введите позицию в списке:");  
 int b = GetNumber(0, massivList[--a].TotalAmountOfLinks());  
 Console.WriteLine("Введите значение:");  
 massivList[a].InsertData(b, GetInfo(int.MinValue, int.MaxValue));   
}  
  
string? TakeFilePath()  
{  
 string? path;  
 bool isIncorrect;  
 do {  
 isIncorrect = false;  
 Console.WriteLine("\nВведите путь к файлу");  
 path = Console.ReadLine();  
 if (!File.Exists(path) || !path.EndsWith(".txt"))  
 {  
 isIncorrect = true;  
 Console.WriteLine("Проверьте параметры файла");  
 }  
 } while (isIncorrect);  
 return path;  
}  
  
void OpenFile(ref NodeList[] massivList)  
{  
 massivList = Array.Empty<NodeList>();  
 bool isIncorrect;  
 do {  
 isIncorrect = false;  
 try  
 {  
 string? path = TakeFilePath();  
 using (var st = new StreamReader(path))  
 {  
 int temp = Convert.ToInt32(st.ReadLine());  
 if (temp > 5 || temp < 1)  
 throw new Exception ("Длина массива не соответствует валидной");  
 massivList = new NodeList[temp];  
 for (int i = 0; i < massivList.GetLength(0); i++)  
 {  
 massivList[i] = new NodeList();  
 temp = Convert.ToInt32(st.ReadLine());  
 if (temp < 1 || temp > 20)  
 throw new Exception ($"Длина массива {i + 1} не соответствует валидной");  
 String[] strArr = st.ReadLine().Split(' ');  
 if (strArr.Length != temp)   
 throw new Exception ($"Введенные числа для {i + 1} списка не валидны");   
 for (int j = 0; j < strArr.Length; j++)  
 {  
 int.TryParse(strArr[j], out var check);  
 massivList[i].AddToEnd(check);  
 }  
 }  
 if (st.ReadLine() != null)  
 throw new Exception("В файле присутствуют остаточные данные");  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 massivList = Array.Empty<NodeList>();  
 isIncorrect = true;  
 }  
 } while (isIncorrect);  
}  
  
void SaveFile(NodeList[] massivList)  
{  
 string? path = TakeFilePath();  
 try  
 {  
 using (StreamWriter sw = new StreamWriter(path, false))  
 {  
 for (int i = 0; i < massivList.Length; i++)  
 massivList[i].ShowInfo(sw.Write);  
 //sw.WriteLineAsync()??  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine(e.Message);  
 }  
}  
  
Console.Write("Введите количество списков: ");  
var massivList = new NodeList[GetNumber(1, 5)];  
for (int i = 0; i < massivList.GetLength(0); i++)  
{  
 massivList[i] = new NodeList();  
 Console.Write($"Введите количество элементов {i+1} списка: ");  
 InputInfo(ref massivList[i], GetNumber(1, 20), i);  
}  
  
InfoInMainMenu();  
ConsoleKeyInfo key;  
do {  
 key = Console.ReadKey();  
 switch (key.Key)  
 {  
 case ConsoleKey.I: //Insert  
 Console.Clear();  
 InsertInfoInList(ref massivList);   
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.O: //OpenFile  
 Console.Clear();  
 OpenFile(ref massivList);  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.S: //SaveFile  
 Console.Clear();  
 SaveFile(massivList);  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.Enter: //ShowList  
 Console.Clear();  
 for (int i = 0; i < massivList.Length; i++)  
 massivList[i].ShowInfo(Console.Write);  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.H: //Help  
 Console.Clear();  
 InfoInMainMenu();  
 break;  
 }  
} while (key.Key != ConsoleKey.E); //Exit

Результат на **Delphi**:



Результат на **C#**:



**Блок-cхема**:

