УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №5.2

по предмету

Основы алгоритмизации и программирования

Вариант 14

Выполнил:

Крутько А.А.

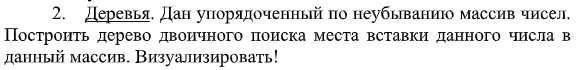
Проверила:

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.StdCtrls, Vcl.ExtCtrls, Vcl.Menus, ClipBrd,

Vcl.Buttons, Vcl.Grids;

type

PTree = ^TTree;

TTree = record

Data: Integer;

Left: PTree;

Right: PTree;

end;

TLabeledEdit = Class(Vcl.ExtCtrls.TLabeledEdit)

Public

Procedure WMPaste(Var Msg:TMessage); Message WM\_PASTE;

End;

TIPEGrid = Class(TStringGrid);

TMainForm = class(TForm)

SetArr: TLabeledEdit;

SetButton: TBitBtn;

ArrGrid: TStringGrid;

SortButton: TBitBtn;

TreeButton: TBitBtn;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

OpenFile: TOpenDialog;

procedure FormCreate(Sender: TObject);

procedure SetArrChange(Sender: TObject);

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

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

Shift: TShiftState);

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

const Value: string);

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

procedure SetButtonClick(Sender: TObject);

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

Shift: TShiftState);

procedure SortButtonClick(Sender: TObject);

procedure TreeButtonClick(Sender: TObject);

procedure N1Click(Sender: TObject);

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

var Value: string);

procedure SetArrContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

private

{ Private declarations }

public

{ Public declarations }

end;

var

MainForm: TMainForm;

Tree: PTree;

IsAlreadySubclassing: Boolean = False;

implementation

{$R \*.dfm}

uses ShowTree;

Var

TempBeforeCursor, TempAfterCursor: String; //Были не глобальными, а созданными локально

Procedure TLabeledEdit.WMPaste(Var Msg: TMessage);

Const

MIN\_VALUE = 1;

MAX\_VALUE = 20;

Begin

If Clipboard.HasFormat(CF\_TEXT) Then // TryStrToInt(ClipBoard.AsText, TempVar) тоже подойдет

Begin

Try

Var TempBeforeCursor := Copy(MainForm.SetArr.Text, 1, MainForm.SetArr.SelStart);

Var TempAfterCursor := Copy(MainForm.SetArr.Text, MainForm.SetArr.SelStart + 1, MainForm.SetArr.GetTextLen);

If (StrToInt(TempBeforeCursor + Clipboard.AsText + TempAfterCursor) < MIN\_VALUE) Or (StrToInt(TempBeforeCursor + Clipboard.AsText + TempAfterCursor) > MAX\_VALUE) Then

Begin

Application.MessageBox(Pchar('В буфере обмена содержится неподходящее значение!'), 'Ошибка', MB\_ICONWARNING);

Exit; //ReadOnly := True мб тоже подойдет, не тестил

End;

Except

Application.MessageBox(Pchar('При чтении из буфера произошла ошибка!'), 'Ошибка', MB\_ICONWARNING);

Exit;

End;

End

Else

Begin

Application.MessageBox(Pchar('В буфере обмена содержатся некорректные данные!'), 'Ошибка', MB\_ICONWARNING);

Exit;

End;

inherited;

End;

Function CheckGrid(): Boolean;

Var

Edit: TInplaceEdit;

Begin

Var I := 0;

Var IsCorrect := True;

While IsCorrect and (I < MainForm.ArrGrid.ColCount) do

Begin

If (Length(MainForm.ArrGrid.Cells[I, 0]) = 0) or ((Length(MainForm.ArrGrid.Cells[I,0]) <> 1) and (MainForm.ArrGrid.Cells[I,0][1] = '0')) Then

IsCorrect := False;

Inc(I);

End;

Result := IsCorrect;

End;

Function NewEditProc(Window: HWND; uMsg: UINT; WindowParametr: WPARAM; lParam:LPARAM): Integer; Stdcall;

Const

MIN\_VALUE = 0;

MAX\_VALUE = 99999;

Var

Col, Row: Integer;

Begin

Col := MainForm.ArrGrid.Col;

Row := MainForm.ArrGrid.Row;

If uMsg = WM\_PASTE Then

Begin

Try

If (StrToInt(Clipboard.AsText) < MIN\_VALUE) Or (StrToInt(Clipboard.AsText) > MAX\_VALUE) Then

Begin

Application.MessageBox(Pchar('В буфере обмена содержится неподходящее значение!'), 'Ошибка', MB\_ICONWARNING);

Exit;

End

Else

Begin

MainForm.ArrGrid.Cells[Col, Row] := Clipboard.AsText; //отобразится в ячейке только когда сменишь фокус.

//Т.е.кликнуть куда-то надо после вставки

Exit;

End;

Except

Application.MessageBox(Pchar('В буфере обмена находятся неподходящие данные!'), 'Ошибка', MB\_ICONWARNING);

Exit;

End;

End

Else

Begin

Result := CallWindowProc(Pointer(GetWindowLong(Window,GWL\_USERDATA)), Window, uMsg, WindowParametr, lParam);

MainForm.SortButton.Enabled := CheckGrid();

MainForm.TreeButton.Visible := False; //? ломает все Ибо закрывается вкладка при выделении текста в гриде

End;

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;

//AdjustResolution(Self);

end;

Function CheckSymbolInTextFile(Temp: String): String;

StdCall;

External 'lab\_5\_2.dll';

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

StdCall;

External 'lab\_5\_2.dll';

procedure TMainForm.N1Click(Sender: TObject);

Var

FileInput: TextFile;

Temp: String;

Size: Integer;

Count: Integer;

ListTemp: TStringList;

begin

If OpenFile.Execute Then

Begin

Try

Try

AssignFile(FileInput, OpenFile.FileName);

Reset(FileInput);

Readln(FileInput, Temp);

Size := StrToInt(CheckSymbolInTextFile(Temp));

If (Size < 1) or (Size > 20) Then

raise Exception.Create('Проверьте правильность размера массива');

SetArr.Text := IntToStr(Size);

SetButton.Click;

ArrGrid.Visible := False;

SortButton.Visible := False;

ListTemp := TStringList.Create;

While Not Eof(FileInput) do

Begin

Readln(FileInput, Temp);

Split(' ', Temp, ListTemp);

If ListTemp.Count = Size Then

Begin

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

ArrGrid.Cells[I,0] := IntToStr(StrToInt(CheckSymbolInTextFile(ListTemp[I])));

End

Else

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

SortButton.Enabled := True;

End;

ArrGrid.Visible := True;

SortButton.Visible := True;

Except

on E : Exception do

Begin

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

ArrGrid.Rows[0].Clear;

SortButton.Enabled := False;

ArrGrid.Visible := False;

SetArr.Text := '';

End;

End;

Finally

CloseFile(FileInput);

End;

End;

end;

procedure TMainForm.SetArrChange(Sender: TObject);

begin

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

ArrGrid.Visible := False;

SortButton.Visible := False;

SortButton.Enabled := False;

ArrGrid.Rows[0].Clear;

TreeButton.Visible := False;

end;

procedure TMainForm.SetArrContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

begin

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

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

end;

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

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 (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.SetArrKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

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

end;

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

Var

Edit: TEdit;

CursorPosition: Integer;

NewCursorPosition: Integer;

TempAll, TempSelected: String;

begin

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

Key := #0;

{If (TEdit(Sender).SelStart <> TEdit(Sender).GetTextLen) and ((Length(TEdit(Sender).SelText) < 1) or (Length(TEdit(Sender).Text) = 0)) Then

Begin

Key := #0;

Exit;

End;} //Проверка, чтобы нельзя было удалить из середины

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);

//SelStart идет с 0

If (Key <> #0) and (Key <> #08) and ((StrToInt(TempBeforeCursor + Key + TempAfterCursor) < 1) or (StrToInt(TempBeforeCursor + Key + TempAfterCursor) > 20)) Then

Key := #0;

end;

procedure TMainForm.SetButtonClick(Sender: TObject);

begin

SortButton.Visible := True;

ArrGrid.ColCount := StrToInt(SetArr.Text);

ArrGrid.Visible := True;

SetButton.Enabled := False;

end;

Procedure Swap(L, R: Integer);

Begin

Var Temp := MainForm.ArrGrid.Cells[L, 0];

MainForm.ArrGrid.Cells[L, 0] := MainForm.ArrGrid.Cells[R, 0];

MainForm.ArrGrid.Cells[R, 0] := Temp;

End;

Procedure QSort(Left, Right: Integer);

Begin

Var NewLeft := Left;

Var NewRight := Right;

Var Middle := StrToInt(MainForm.ArrGrid.Cells[(Left + Right) div 2, 0]);

While (NewLeft < NewRight) do

Begin

While StrToInt(MainForm.ArrGrid.Cells[NewLeft, 0]) < Middle do

Inc(NewLeft);

While StrToInt(MainForm.ArrGrid.Cells[NewRight, 0]) > Middle do

Dec(NewRight);

If (NewLeft <= NewRight) Then

Begin

Swap(NewLeft, NewRight);

Inc(NewLeft);

Dec(NewRight);

End;

End;

If Left < NewRight Then QSort(Left, NewRight);

If Right > NewLeft Then QSort(NewLeft, Right);

End;

procedure TMainForm.SortButtonClick(Sender: TObject);

begin

QSort(0, ArrGrid.ColCount - 1);

TreeButton.Visible := True;

end;

Function GetNode(Node: PTree; Left, Right: Integer): PTree;

Var

Mid: Integer;

Begin

If (Right < Left) Then Node := Nil

Else

Begin

Mid := (Right + Left) div 2; //start + (end - start) / 2

New(Node);

Node^.Data := StrToInt(MainForm.ArrGrid.Cells[Mid, 0]);

Node^.Left := GetNode(Node^.Left, Left, Mid - 1);

Node^.Right := GetNode(Node^.Right, Mid + 1, Right);

Result := Node;

End;

Result := Node;

End;

procedure TMainForm.TreeButtonClick(Sender: TObject);

begin

Tree := Nil;

Tree := GetNode(Tree, 0, ArrGrid.ColCount - 1);

TreeForm.ShowModal;

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 (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) < LowerNumb) or (StrToInt(TempBeforeCursor + ClipBoard.AsText + TempAfterCursor) > UpperNumb) Then

raise Exception.Create('Error Message');

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

raise Exception.Create('Error Message');

//Ниже чтобы была вставка перед нулем, но не после 0

If Not(Edit.SelStart = 0) and (Edit.GetTextLen > 0) and (Edit.Text[1] = '0') Then

raise Exception.Create('Error Message');

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

raise Exception.Create('Error Message');

Except

Key := 0;

End;

end;

End;

procedure TMainForm.ArrGridGetEditText(Sender: TObject; ACol, ARow: Integer;

var Value: string);

Var

Window: HWND;

begin

If Not IsAlreadySubclassing Then // Если сабклассинга не было, то сабклассим. Иначе - ничего,т.к. уже сделано

Begin

Window := GetWindow(ArrGrid.Handle, GW\_Child);

If IsWindow(Window) Then

SetWindowLong(Window,GWL\_USERDATA,SetWindowLong(Window, GWL\_WNDPROC, LPARAM(@NewEditProc)));

IsAlreadySubclassing := True;

End;

end;

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

Shift: TShiftState);

begin

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

end;

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

Var

SelLen: Integer;

begin

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

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

Begin

If Key = #08 Then

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

Else

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

Key := #0;

End;

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

Begin

If Key = #08 Then

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

Else

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

Key := #0;

End;

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

Key := #0;

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

Key := #0;

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

Key := #0;

end;

procedure TMainForm.ArrGridSetEditText(Sender: TObject; ACol, ARow: Integer;

const Value: string);

begin

SortButton.Enabled := CheckGrid();

TreeButton.Visible := False;

end;

end.

unit ShowTree;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Menus, Vcl.ExtDlgs,

JPEG, Pngimage, Vcl.StdCtrls, ClipBRD, Vcl.StdActns, System.Actions,

Vcl.ActnList;

type

TTreeForm = class(TForm)

MainMenu1: TMainMenu;

N1: TMenuItem;

SavePicture: TSavePictureDialog;

Image1: TImage;

InsertNumberEdit: TLabeledEdit;

InsertButton: TButton;

PopupMenu1: TPopupMenu;

procedure FormCreate(Sender: TObject);

procedure FormPaint(Sender: TObject);

procedure N1Click(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure InsertNumberEditChange(Sender: TObject);

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

Shift: TShiftState);

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

procedure InsertButtonClick(Sender: TObject);

procedure FormShow(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

Const

Radius = 40;

var

TreeForm: TTreeForm;

implementation

uses MainUnit;

{$R \*.dfm}

Var

Temp: PTree;

procedure TTreeForm.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Image1.Picture := Nil;

N1.Enabled := False;

InsertNumberEdit.Visible := True;

InsertButton.Visible := True;

Temp := Nil;

InsertNumberEdit.Text := '';

end;

procedure TTreeForm.FormCreate(Sender: TObject);

begin

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

BorderStyle := BsSingle;

Position := PoDesktopCenter;

TreeForm.Height := Screen.Height;

TreeForm.Width := Screen.Width;

Image1.Height := TreeForm.Height - TreeForm.Height div 5;

Image1.Width := TreeForm.Width;

Image1.Top := 0;

Image1.Left := 0;

InsertNumberEdit.Left := TreeForm.Width div 2;

InsertNumberEdit.Top := Image1.Height - Image1.Height div 6;

InsertButton.Top := InsertNumberEdit.Top;

InsertButton.Left := InsertNumberEdit.Left + InsertNumberEdit.Width + 10;

end;

procedure Circle(x, y: integer);

var

x1, y1, x2, y2: integer;

begin

x1 := x - Radius;

x2 := x + Radius;

y1 := y - Radius;

y2 := y + Radius;

TreeForm.Image1.Canvas.Ellipse(x1, y1, x2, y2);

TreeForm.Canvas.Ellipse(x1, y1, x2, y2);

end;

Procedure Preoder(Tree: PTree; X, Y: Integer; Level: Integer);

begin

If Tree = NIL then exit;

With TreeForm do

Begin

Circle(X, Y);

Var Temp := Y + Radius;

Image1.Canvas.TextOut(X - Canvas.Font.Size, Y - Canvas.Font.Size, IntToStr(Tree^.Data));

Canvas.TextOut(X - Canvas.Font.Size, Y - Canvas.Font.Size, IntToStr(Tree^.Data));

If Tree^.Left <> Nil Then

Begin

Image1.Canvas.MoveTo(X, Temp);

Image1.Canvas.LineTo(X - Radius - Level \* Radius, Temp + Radius);

Canvas.MoveTo(X, Temp);

Canvas.LineTo(X - Radius - Level \* Radius, Temp + Radius);

End;

If Tree^.Right <> Nil Then

Begin

Image1.Canvas.MoveTo(X, Temp);

Image1.Canvas.LineTo(X + Radius + Level \* Radius, Temp + Radius);

Canvas.MoveTo(X, Temp);

Canvas.LineTo(X + Radius + Level \* Radius, Temp + Radius);

End;

Preoder(Tree^.Left, X - Radius - Level \* Radius, Temp + Radius + (Radius div 2), Level div 3); //div 2? И вносить 7 изначально

PreOder(Tree^.Right, X + Radius + Level \* Radius, Temp + Radius + (Radius div 2), Level div 3); //div 2?

End;

end;

procedure TTreeForm.FormPaint(Sender: TObject);

begin

Canvas.Pen.Width := 3;

Canvas.Pen.Color := clBlack;

Canvas.Font.Size := 14;

{Var Count := 0;

Var I := MainForm.ArrGrid.ColCount;

While I > 0 do

Begin

I := I div 2;

Inc(Count, 2);

End;}

//Canvas.Font.Color := clBlack;

Var X := TreeForm.Width div 2;

Var Y := Radius;

PreOder(Temp, X, Y, 9);

N1.Enabled := True;

//InsertNumberEdit.Visible := True;

end;

procedure TTreeForm.FormShow(Sender: TObject);

begin

Temp := Tree;

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//or ((Shift = [ssCtrl]) and (Key = Ord('V'))) 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'); //Надо ли?

//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;

Function InsertHere(Root: PTree; Value: Integer): PTree;

StdCall;

External 'lab\_5\_2.dll';

Procedure Insert(Root: PTree; Value: Integer);

Begin

Root := InsertHere(Root, Value);

End;

procedure TTreeForm.InsertButtonClick(Sender: TObject);

Var

Meow: PTree;

begin

InsertButton.Visible := False;

InsertNumberEdit.Visible := False;

Insert(Temp, StrToInt(InsertNumberEdit.Text));

TreeForm.OnPaint(Sender);

end;

procedure TTreeForm.InsertNumberEditChange(Sender: TObject);

begin

InsertButton.Enabled := Not(((TEdit(Sender).GetTextLen > 1) and (TEdit(Sender).Text[1] = '0')) xor ''.IsNullOrWhiteSpace(TEdit(Sender).Text));

end; //Len > 1, чтобы проходил 0 только, но если число и перед ним 0, то не работает. (иногда нужен > 0)

procedure TTreeForm.InsertNumberEditKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

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

end;

procedure TTreeForm.InsertNumberEditKeyPress(Sender: TObject; var Key: Char);

Var

SelLen: Integer;

begin

SelLen := TEdit(Sender).SelLength;

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

Begin

If Key = #08 Then

TEdit(Sender).SelText := ''

Else

TEdit(Sender).SelText := Key;

Key := #0;

End;

If (TEdit(Sender).Text = '0') and (SelLen > 0) and (Key In ['0'..'9', #08]) Then

Begin

If Key = #08 Then

TEdit(Sender).SelText := ''

Else

TEdit(Sender).SelText := Key;

Key := #0;

End;

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

Key := #0;

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

Key := #0;

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

Key := #0;

end;

procedure TTreeForm.N1Click(Sender: TObject);

Var

Im: TPNGImage;

begin

Im := TPNGImage.Create;

If SavePicture.Execute Then

Begin

try

//Image1.Picture.(?Graphic?)SaveToFile(SavePicture.FileName + '.jpg');

Im.Assign(Image1.Picture.Graphic);

Im.SaveToFile(SavePicture.FileName + '.png');

finally

FreeAndNil(Im);

end;

End;

end;

end.

library lab\_5\_2;

uses

System.SysUtils,

System.Classes;

type

PTree = ^TTree;

TTree = record

Data: Integer;

Left: PTree;

Right: PTree;

end;

{$R \*.res}

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

begin

ListOfStrings.Clear;

ListOfStrings.Delimiter := Delimiter;

ListOfStrings.StrictDelimiter := True;

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 InsertHere(Root: PTree; Value: Integer): PTree; StdCall;

Begin

If (Root^.Data < Value) Then

Begin

If (Root^.Right = nil) Then

Begin

New(Root^.Right);

Root^.Right^.Data := Value;

Root^.Right^.Left := Nil;

Root^.Right^.Right := Nil;

End

Else

InsertHere(Root^.Right, Value);

End

Else

Begin

If (Root^.Left = nil) Then

Begin

New(Root^.Left);

Root^.Left^.Data := Value;

Root^.Left^.Left := Nil;

Root^.Left^.Right := Nil;

End

Else

InsertHere(Root^.Left, Value);

End;

Result := Root;

End;

Exports Split, CheckSymbolInTextFile, InsertHere;

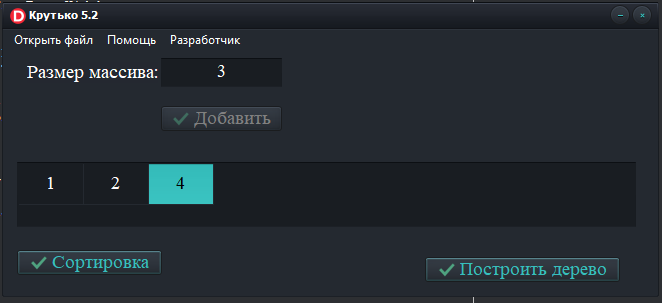
begin

end.

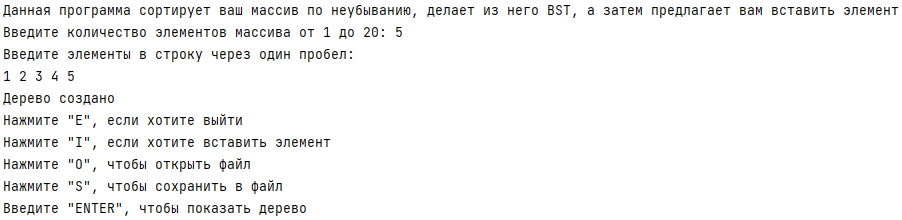
Код программы на **C#**:

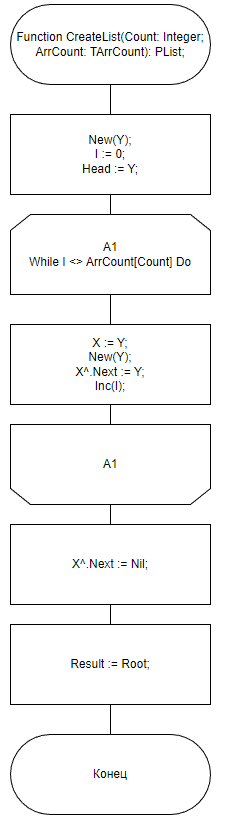
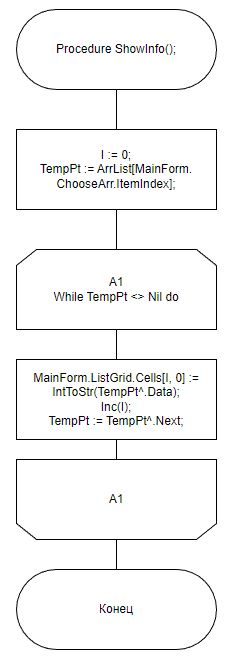
void InfoInMainMenu()  
{  
 Console.*WriteLine*("Нажмите \"E\", если хотите выйти\nНажмите \"I\", если хотите вставить элемент\nНажмите \"O\", чтобы открыть файл\nНажмите \"S\", чтобы сохранить в файл\nВведите \"ENTER\", чтобы показать дерево");  
}  
  
void InfoForHelp()  
{  
 Console.*WriteLine*("\nВведите \"H\", если хотите вывести помощь по меню");  
}  
  
void MessageAfterAction()  
{  
 Console.*WriteLine*("Вы вернулись в главное меню. Ожидается ваше следующее действие");  
}  
  
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;  
}  
  
void InputInfo(ref int[] current)  
{  
 Console.*WriteLine*("Введите элементы в строку через один пробел:");  
 bool isInCorrect;  
 do {  
 isInCorrect = true;  
 var strArr = Console.*ReadLine*().Split(' ');  
 if (strArr.Length != current.Length)  
 {  
 isInCorrect = false;  
 Console.*WriteLine*("Проверьте ввод");  
 }  
 for (int i = 0; i < strArr.Length && isInCorrect; i++)  
 {  
 isInCorrect = (int.*TryParse*(strArr[i], out current[i]));  
 if (!isInCorrect)  
 Console.*WriteLine*("Проверьте ввод");  
 }  
 } while (!isInCorrect);  
 QSort(ref current, current.GetLowerBound(0), current.GetUpperBound(0));  
}  
  
Node? CreateBst(in int[] arr, int start, int end) {  
 if (end < start) return null;  
 int mid = (start + end) / 2; *//start + (end - start) / 2* Node node = new Node(arr[mid]);  
 node.Left = CreateBst(arr, start, mid - 1);  
 node.Right = CreateBst(arr, mid + 1, end);  
 return node;  
}  
  
Node? SortedArrayToBst(in int[] arr)  
{  
 if (arr.Length == 0) return null;  
 return CreateBst(arr, arr.GetLowerBound(0), arr.GetUpperBound(0));  
}  
  
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 Node? node)  
{  
 node = null;  
 string? path = TakeFilePath();  
 bool isIncorrect;  
 do {  
 isIncorrect = true;  
 try  
 {  
 using (var st = new StreamReader(path))  
 {  
 int temp = Convert.*ToInt32*(st.ReadLine());  
 if (temp > 20 || temp < 1)  
 throw new Exception ("Длина массива не соответствует валидной");  
 int[] massiv = new int[temp];  
 var strArr = st.ReadLine().Split(' ');  
 if (strArr.Length != massiv.Length)  
 {  
 isIncorrect = false;  
 Console.*WriteLine*("Проверьте ввод");  
 }  
 for (int i = 0; i < strArr.Length && isIncorrect; i++)  
 {  
 isIncorrect = int.*TryParse*(strArr[i], out massiv[i]);  
 if (!isIncorrect)  
 Console.*WriteLine*("Проверьте ввод в файле");  
 }  
 if (st.ReadLine() != null)  
 throw new Exception("В файле присутствуют остаточные данные");  
 QSort(ref massiv, massiv.GetLowerBound(0), massiv.GetUpperBound(0));  
 node = SortedArrayToBst(massiv);  
 }  
 }  
 catch (Exception e)  
 {  
 Console.*WriteLine*(e.Message);  
 node = null;  
 isIncorrect = false;  
 }  
 } while (!isIncorrect);  
}  
  
void SaveFile(Node? node)  
{  
 string? path = TakeFilePath();  
 try  
 {  
 using (StreamWriter sw = new StreamWriter(path, false))  
 {  
 node.Print(sw.Write, node, 10);  
 *//sw.WriteLineAsync()??* }  
 }  
 catch (Exception e)  
 {  
 Console.*WriteLine*(e.Message);  
 }  
}  
  
void QSort(ref int[] arr, int left, int right)  
{  
 var newleft = left;  
 var newright = right;  
 var middle = arr[(left + right) / 2];  
 while (newleft < newright)  
 {  
 while (arr[newleft] < middle)  
 newleft++;  
 while (arr[newright] > middle)  
 newright--;  
 if (newleft <= newright)  
 {  
 (arr[newleft], arr[newright]) = (arr[newright], arr[newleft]);  
 newleft++;  
 newright--;  
 }  
 }  
 if (left < newright) QSort(ref arr, left, newright);  
 if (right > newleft) QSort(ref arr, newleft, right);  
}  
  
Console.*WriteLine*("Данная программа сортирует ваш массив по неубыванию, делает из него BST, а затем предлагает вам вставить элемент");  
Console.*Write*("Введите количество элементов массива от 1 до 20: ");  
int[] current = new int[GetNumber(1, 20)];  
InputInfo(ref current);  
Node? tree = SortedArrayToBst(current);  
Console.*WriteLine*("Дерево создано");  
InfoInMainMenu();  
ConsoleKeyInfo key;  
do {  
 key = Console.*ReadKey*();  
 switch (key.Key)  
 {  
 case ConsoleKey.*I*: *//Insert* Console.*Clear*();  
 tree.Insert(tree, GetNumber(-500000, 500000));  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.*O*: *//OpenFile* Console.*Clear*();  
 OpenFile(ref tree);  
 Console.*WriteLine*("Дерево создано");  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.*S*: *//SaveFile* Console.*Clear*();  
 SaveFile(tree);  
 MessageAfterAction();  
 InfoForHelp();  
 break;  
 case ConsoleKey.*Enter*: *//ShowTree* Console.*Clear*();  
 tree.Print(Console.*Write*, tree,10);  
 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**:

