УО «Белорусский государственный университет информатики и

радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №6.2

по предмету

Основы Алгоритмизации и Программирования

Вариант 15

**Выполнил**

**Коловайтис Н. А.**

**Проверила**

**Данилова Г.В.**

Группа:

**8**51001

Минск 2019

**Задание**

Деревья. Вывести номера вершин, у которых количество потомков в левом поддереве не равно количеству потомков в правом поддереве.

Дерево визуализировать!

Реализовать задания №1 и №2 в виде:

* внешнего файла;
* в виде библиотечного модуля.

Написать программы и подключить модуль и внешний файл.

**Код программы**

**(Delphi)**

**unit** Main;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants,

FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Objects,

FMX.Menus, Vcl.Dialogs, Winapi.Windows, FMX.Platform.Win,

FMX.Controls.Presentation, FMX.StdCtrls, ElementView;

**type**

TPointer = ^TTree;

TTree = **Record**

Sons: **array** [**Boolean**] **of** TPointer;

Value: TElementView;

**End**;

TFormMain = **class**(TForm)

MainMenu: TMainMenu;

MIToStart: TMenuItem;

MIterm: TMenuItem;

BGetAnswer: TButton;

**procedure** **MIToStartClick**(Sender: **TObject**);

**procedure** **MItermClick**(Sender: **TObject**);

**procedure** **BGetAnswerClick**(Sender: **TObject**);

**private**

Root: TPointer;

**public**

Answer: **String**;

**procedure** **AddElement**(Pos: TTrack; Value: TViewElementPointer);

**procedure** **DeleteElement**(Element: TPointer);

**procedure** **DeleteElementByPos**(Pos: TTrack);

**Function** **FindAnswer**(Element: TPointer): **Integer**;

**end**;

**var**

FormMain: TFormMain;

**implementation**

{$R \*.fmx}

**procedure** **TFormMain**.**AddElement**(Pos: TTrack; Value: TViewElementPointer);

**var**

NewElement, PastElement: TPointer;

Navigation: **Boolean**;

**begin**

PastElement := **nil**;

NewElement := Root;

**for** Navigation **in** Pos **do**

**begin**

PastElement := NewElement;

NewElement := NewElement.Sons[Navigation];

**end**;

**new**(NewElement);

NewElement.Sons[**true**] := **nil**;

NewElement.Sons[**false**] := **nil**;

NewElement.Value := Value^;

**if** (PastElement <> **nil**) **then**

PastElement.Sons[Pos[High(Pos)]] := NewElement

**else**

Root := NewElement;

**end**;

**procedure** **TFormMain**.**BGetAnswerClick**(Sender: **TObject**);

**begin**

Answer := '';

FindAnswer(Root);

**if** (Length(Answer) > **0**) **then**

**begin**

SetLength(Answer, Length(Answer) - **2**);

MessageBox(GetDesktopWindow,

**PChar**('Номера вершин, у которых не совпадает количество потомков:' +

#13#10 + Answer), **PChar**('Ответ'), MB\_OK)

**end**

**else**

MessageBox(GetDesktopWindow,

**PChar**('Вершин с несовпадающим количеством потомков не обнаружено.'),

**PChar**('Ответ'), MB\_OK)

**end**;

**procedure** **TFormMain**.**DeleteElement**(Element: TPointer);

**begin**

**if** (Element <> **nil**) **then**

**begin**

DeleteElement(Element.Sons[**true**]);

DeleteElement(Element.Sons[**false**]);

Element.Value.Delete;

FreeMem(Element);

**end**;

**end**;

**procedure** **TFormMain**.**DeleteElementByPos**(Pos: TTrack);

**var**

FindElement, Father: TPointer;

Direction: **Boolean**;

**begin**

Father := **nil**;

FindElement := Root;

**for** Direction **in** Pos **do**

**begin**

Father := FindElement;

FindElement := FindElement.Sons[Direction];

**end**;

DeleteElement(FindElement);

**if** (Father <> **nil**) **then**

Father.Sons[Pos[High(Pos)]] := **nil**

**else**

Root := **nil**;

**end**;

**function** **TFormMain**.**FindAnswer**(Element: TPointer): **Integer**;

**var**

Values: **array** [**false** .. **true**] **of** **Integer**;

i: **Boolean**;

**begin**

Result := **0**;

**if** (Element <> **nil**) **then**

**begin**

**for** i := **false** **to** **true** **do**

**begin**

Values[i] := FindAnswer(Element.Sons[i]);

**end**;

**if** (Values[**true**] <> Values[**false**]) **then**

Answer := Answer + IntToStr(Element.Value.Number) + ', ';

Result := Values[**true**] + Values[**false**] + **1**;

**end**;

**end**;

**procedure** **TFormMain**.**MItermClick**(Sender: **TObject**);

**begin**

MessageBox(GetDesktopWindow,

**PChar**('Вывести номера вершин, у которых количество потомков в левом поддереве не

равно количеству потомков в правом поддереве. Дерево визуализировать!'),

**PChar**('Условие'), MB\_OK)

**end**;

**procedure** **TFormMain**.**MIToStartClick**(Sender: **TObject**);

**var**

Pos: TTrack;

ElementRoot: TElementView;

**begin**

**if** (Root <> **nil**) **then**

**begin**

DeleteElement(Root);

**end**;

SetLength(Pos, **0**);

ElementRoot := TElementView.Create(**Self**, Pos, AddElement,

DeleteElementByPos);

**end**;

**end**.

**unit** ElementView;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants,

FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Objects,

FMX.Controls.Presentation, FMX.StdCtrls;

**type**

TTrack = **array** **of** **Boolean**;

TViewElementPointer = ^TElementView;

TAddProcedure = **procedure**(Pos: TTrack; Value: TViewElementPointer) **of** **object**;

TDeleteProcedure = **procedure**(Pos: TTrack) **of** **object**;

TElementView = **class**(TCircle)

**public**

AddElement: TAddProcedure;

DeleteByPos: TDeleteProcedure;

Number: **Integer**;

**Constructor** **Create**(AOwner: TFmxObject; **const** Position: TTrack;

AddElement: TAddProcedure; DeleteByPos: TDeleteProcedure); overload;

**Function** **GetTrack**(): TTrack;

**procedure** **AddLeftClick**(Sender: **TObject**);

**procedure** **AddRightClick**(Sender: **TObject**);

**procedure** **Delete**();

**function** **TrackToNumber**(Track: TTrack): **Integer**;

**const**

WidthOfField = **640**;

DeltaHeight = **50**;

Diametr = **20**;

StartY = **10**;

MaxLevel = **5**;

TextMarginXOne = **6**;

TextMarginXTwo = **4**;

TextMarginY = **2**;

NewColor = TAlphaColors.Green;

**private**

LineLeft, LineRight: TArc;

AddLeft, AddRight: TCircle;

Text: TLabel;

Track: TTrack;

**procedure** **DblClick**(Sender: **TObject**);

**end**;

**implementation**

{ TElementView }

**procedure** **TElementView**.**AddLeftClick**(Sender: **TObject**);

**var**

NewTrack: TTrack;

LastIndex: **Integer**;

NewElement: TElementView;

**begin**

LastIndex := Length(Track);

NewTrack := Copy(Track);

SetLength(NewTrack, LastIndex + **1**);

NewTrack[LastIndex] := **false**;

NewElement := TElementView.Create(**Self**.Parent, NewTrack, AddElement,

DeleteByPos);

NewElement.AddElement := AddElement;

**end**;

**procedure** **TElementView**.**AddRightClick**(Sender: **TObject**);

**var**

NewTrack: TTrack;

LastIndex: **Integer**;

NewElement: TElementView;

**begin**

LastIndex := Length(Track);

NewTrack := Copy(Track);

SetLength(NewTrack, LastIndex + **1**);

NewTrack[LastIndex] := **true**;

NewElement := TElementView.Create(**Self**.Parent, NewTrack, AddElement,

DeleteByPos);

**end**;

**constructor** **TElementView**.**Create**(AOwner: TFmxObject; **const** Position: TTrack;

AddElement: TAddProcedure; DeleteByPos: TDeleteProcedure);

**var**

XPos, YPos, CurWidth: **Integer**;

Track: **Boolean**;

**begin**

**inherited** Create(AOwner);

**Self**.AddElement := AddElement;

**Self**.DeleteByPos := DeleteByPos;

AddElement(Position, @**Self**);

**Self**.Track := Copy(Position);

**Self**.Width := Diametr;

**Self**.Height := Diametr;

YPos := StartY;

XPos := WidthOfField **div** **2**;

CurWidth := XPos;

**for** Track **in** Position **do**

**begin**

Inc(YPos, DeltaHeight);

CurWidth := CurWidth **div** **2**;

**if** (Track) **then**

Inc(XPos, CurWidth)

**else**

Dec(XPos, CurWidth);

**end**;

**if** (Length(Position) < MaxLevel) **then**

**begin**

LineLeft := TArc.Create(**Self**);

LineLeft.Parent := AOwner;

LineLeft.Width := CurWidth;

LineLeft.Height := DeltaHeight \* **2**;

LineLeft.Position.X := XPos - CurWidth **div** **2**;

LineLeft.Position.Y := YPos + **Self**.Height / **2**;

LineLeft.StartAngle := -**90**;

LineRight := TArc.Create(**Self**);

LineRight.Parent := AOwner;

LineRight.Width := CurWidth;

LineRight.Height := DeltaHeight \* **2**;

LineRight.Position.X := LineLeft.Position.X;

LineRight.Position.Y := LineLeft.Position.Y;

AddLeft := TCircle.Create(**Self**);

AddLeft.Parent := AOwner;

AddLeft.Width := Diametr;

AddLeft.Height := Diametr;

AddLeft.Position.X := XPos - CurWidth **div** **2** - Diametr / **2**;

AddLeft.Position.Y := YPos + DeltaHeight;

AddLeft.Fill.Color := NewColor;

AddLeft.OnClick := AddLeftClick;

AddRight := TCircle.Create(**Self**);

AddRight.Parent := AOwner;

AddRight.Width := Diametr;

AddRight.Height := Diametr;

AddRight.Position.X := XPos + CurWidth **div** **2** - Diametr / **2**;

AddRight.Position.Y := AddLeft.Position.Y;

AddRight.Fill.Color := NewColor;

AddRight.OnClick := AddRightClick;

**end**;

**Self**.Position.X := XPos - **Self**.Width / **2**;

**Self**.Position.Y := YPos;

**Self**.Parent := AOwner;

**Self**.OnDblClick := DblClick;

Number := TrackToNumber(Position);

Text := TLabel.Create(**Self**);

Text.Parent := **Self**;

Text.Text := IntToStr(Number);

**if** (Number > **9**) **then**

Text.Position.X := TextMarginXTwo

**else**

Text.Position.X := TextMarginXOne;

Text.Position.Y := TextMarginY;

**end**;

**procedure** **TElementView**.**Delete**;

**begin**

**Self**.Destroy;

**end**;

**function** **TElementView**.**GetTrack**: TTrack;

**begin**

GetTrack := Track;

**end**;

**procedure** **TElementView**.**DblClick**(Sender: **TObject**);

**begin**

DeleteByPos(Track);

**end**;

**function** **TElementView**.**TrackToNumber**(Track: TTrack): **Integer**;

**var**

Direction: **Boolean**;

**begin**

Result := **1**;

**for** Direction **in** Track **do**

**begin**

Result := Result \* **2**;

**if** (Direction) **then**

Inc(Result);

**end**;

**end**;

**end**.

**С помощью Unit**

**Сам проект**

**(Delphi)**

**unit** UnitVariant;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants,

FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, MakeAllClass,

FMX.Controls.Presentation, FMX.StdCtrls, FMX.Menus, Vcl.Dialogs,

Winapi.Windows, FMX.Platform.Win;

**type**

TFormMain = **class**(TForm)

**procedure** **BGetAnswerClick**(Sender: **TObject**);

**procedure** **FormCreate**(Sender: **TObject**);

**procedure** **MIToStartClick**(Sender: **TObject**);

**procedure** **MItermClick**(Sender: **TObject**);

**private**

MakeAll: TMakeAll;

**end**;

**var**

FormMain: TFormMain;

**implementation**

{$R \*.fmx}

**procedure** **TFormMain**.**BGetAnswerClick**(Sender: **TObject**);

**var**

Answer: **String**;

**begin**

Answer := MakeAll.GetAnswer;

**if** (Length(Answer) > **0**) **then**

**begin**

MessageBox(GetDesktopWindow,

**PChar**('Номера вершин, у которых не совпадает количество потомков:' +

#13#10 + Answer), **PChar**('Ответ'), MB\_OK)

**end**

**else**

MessageBox(GetDesktopWindow,

**PChar**('Вершин с несовпадающим количеством потомков не обнаружено.'),

**PChar**('Ответ'), MB\_OK)

**end**;

**procedure** **TFormMain**.**FormCreate**(Sender: **TObject**);

**begin**

MakeAll := TMakeAll.Create;

**end**;

**procedure** **TFormMain**.**MItermClick**(Sender: **TObject**);

**begin**

MessageBox(GetDesktopWindow,

**PChar**('Вывести номера вершин, у которых количество потомков в левом поддереве не равно количеству потомков в правом поддереве. Дерево визуализировать!'),

**PChar**('Условие'), MB\_OK);

**end**;

**procedure** **TFormMain**.**MIToStartClick**(Sender: **TObject**);

**begin**

MakeAll.ToStart(TForm(**Self**));

**end**;

**end**.

**MakeAllClass.pas**

**(Delphi)**

**unit** MakeAllClass;

**interface**

**uses**

System.SysUtils,

System.Classes,

Fmx.Types,

ElementView, Fmx.Forms;

**type**

TPointer = ^TTree;

TTree = **Record**

Sons: **array** [**Boolean**] **of** TPointer;

Value: TElementView;

**End**;

TMakeAll = **class**

**public**

**var**

Root: TPointer;

Answer: **String**;

**procedure** **AddElement**(Pos: TTrack; Value: TViewElementPointer);

**procedure** **DeleteElement**(Element: TPointer);

**procedure** **DeleteElementByPos**(Pos: TTrack);

**function** **FindAnswer**(Element: TPointer): **Integer**;

**function** **GetAnswer**(): **String**;

**procedure** **ToStart**(**var** Parent: TForm);

**end**;

**implementation**

**procedure** **TMakeAll**.**AddElement**(Pos: TTrack; Value: TViewElementPointer);

**var**

NewElement, PastElement: TPointer;

Navigation: **Boolean**;

**begin**

PastElement := **nil**;

NewElement := Root;

**for** Navigation **in** Pos **do**

**begin**

PastElement := NewElement;

NewElement := NewElement.Sons[Navigation];

**end**;

**new**(NewElement);

NewElement.Sons[**true**] := **nil**;

NewElement.Sons[**false**] := **nil**;

NewElement.Value := Value^;

**if** (PastElement <> **nil**) **then**

PastElement.Sons[Pos[High(Pos)]] := NewElement

**else**

Root := NewElement;

**end**;

**procedure** **TMakeAll**.**DeleteElement**(Element: TPointer);

**begin**

**if** (Element <> **nil**) **then**

**begin**

DeleteElement(Element.Sons[**true**]);

DeleteElement(Element.Sons[**false**]);

Element.Value.Delete;

FreeMem(Element);

**end**;

**end**;

**procedure** **TMakeAll**.**DeleteElementByPos**(Pos: TTrack);

**var**

FindElement, Father: TPointer;

Direction: **Boolean**;

**begin**

Father := **nil**;

FindElement := Root;

**for** Direction **in** Pos **do**

**begin**

Father := FindElement;

FindElement := FindElement.Sons[Direction];

**end**;

DeleteElement(FindElement);

**if** (Father <> **nil**) **then**

Father.Sons[Pos[High(Pos)]] := **nil**

**else**

Root := **nil**;

**end**;

**function** **TMakeAll**.**FindAnswer**(Element: TPointer): **Integer**;

**var**

Values: **array** [**false** .. **true**] **of** **Integer**;

i: **Boolean**;

**begin**

Result := **0**;

**if** (Element <> **nil**) **then**

**begin**

**for** i := **false** **to** **true** **do**

**begin**

Values[i] := FindAnswer(Element.Sons[i]);

**end**;

**if** (Values[**true**] <> Values[**false**]) **then**

Answer := Answer + IntToStr(Element.Value.Number) + ', ';

Result := Values[**true**] + Values[**false**] + **1**;

**end**;

**end**;

**function** **TMakeAll**.**GetAnswer**(): **String**;

**begin**

Answer := '';

FindAnswer(Root);

**if** (Length(Answer) > **0**) **then**

SetLength(Answer, Length(Answer) - **2**);

Result := Answer;

**end**;

**procedure** **TMakeAll**.**ToStart**(**var** Parent: TForm);

**var**

Pos: ElementView.TTrack;

AddProc: TAddProcedure;

DeleteProc: TDeleteProcedure;

StartElement: TElementView;

**begin**

**if** (Root <> **nil**) **then**

**begin**

DeleteElement(Root);

**end**;

SetLength(Pos, **0**);

StartElement := TElementView.Create(Parent, Pos, AddElement,

DeleteElementByPos);

**end**;

**end**.

**С помощью Dll**

**Сам проект**

**(Delphi)**

**unit** MainWithDll;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants,

FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Objects,

FMX.Menus, Vcl.Dialogs, Winapi.Windows, FMX.Platform.Win,

FMX.Controls.Presentation, FMX.StdCtrls, ElementView;

**procedure** **AddElement**(Pos: TTrack; Number: **Integer**); **external** 'Lib\_6\_2.dll';

**function** **DeleteElementByPos**(Pos: TTrack): TElements; **external** 'Lib\_6\_2.dll';

**function** **GetAnswer**(): **String**; **external** 'Lib\_6\_2.dll';

**type**

TFormMain = **class**(TForm)

MainMenu: TMainMenu;

MIToStart: TMenuItem;

MIterm: TMenuItem;

BGetAnswer: TButton;

**procedure** **MIToStartClick**(Sender: **TObject**);

**procedure** **MItermClick**(Sender: **TObject**);

**procedure** **BGetAnswerClick**(Sender: **TObject**);

**procedure** **FormCreate**(Sender: **TObject**);

**private**

AllElements: **array** [**1** .. **63**] **of** TElementView;

**public**

**procedure** **AddElementMethod**(Pos: TTrack; Value: TViewElementPointer);

**procedure** **DeleteElementByPosMethod**(Pos: TTrack);

**end**;

**var**

FormMain: TFormMain;

**implementation**

{$R \*.fmx}

**procedure** **TFormMain**.**AddElementMethod**(Pos: TTrack; Value: TViewElementPointer);

**var**

Number: **Integer**;

**begin**

AllElements[Value.Number] := Value^;

AddElement(Pos, Value.Number);

**end**;

**procedure** **TFormMain**.**BGetAnswerClick**(Sender: **TObject**);

**var**

Answer: **String**;

**begin**

Answer := GetAnswer;

**if** (Length(Answer) > **0**) **then**

**begin**

MessageBox(GetDesktopWindow,

**PChar**('Номера вершин, у которых не совпадает количество потомков:' +

#13#10 + Answer), **PChar**('Ответ'), MB\_OK)

**end**

**else**

MessageBox(GetDesktopWindow,

**PChar**('Вершин с несовпадающим количеством потомков не обнаружено.'),

**PChar**('Ответ'), MB\_OK)

**end**;

**procedure** **TFormMain**.**DeleteElementByPosMethod**(Pos: TTrack);

**var**

DeletedElement: **Integer**;

**begin**

**for** DeletedElement **in** DeleteElementByPos(Pos) **do**

**begin**

AllElements[DeletedElement].Delete;

**end**;

**end**;

**procedure** **TFormMain**.**MItermClick**(Sender: **TObject**);

**begin**

MessageBox(GetDesktopWindow,

**PChar**('Вывести номера вершин, у которых количество потомков в левом поддереве не

равно количеству потомков в правом поддереве. Дерево визуализировать!'),

**PChar**('Условие'), MB\_OK)

**end**;

**procedure** **TFormMain**.**MIToStartClick**(Sender: **TObject**);

**var**

Path: TTrack;

StartElement: TElementView;

**begin**

SetLength(Path, **0**);

DeleteElementByPosMethod(Path);

StartElement := TElementView.Create(**Self**, Path, AddElementMethod,

DeleteElementByPosMethod);

**end**;

**end**.

**Lib\_6\_2.dll**

**(Delphi)**

**library** Lib\_6\_2;

**uses**

System.SysUtils,

System.Classes,

Fmx.Types,

ElementView;

**type**

TPointer = ^TTree;

TTree = **Record**

Sons: **array** [**Boolean**] **of** TPointer;

Number: **Integer**;

**End**;

**var**

Root: TPointer;

Answer: **String**;

CurrentIndex: **Integer**;

**const**

MaxElementCount = **63**;

**procedure** **DeleteElement**(Element: TPointer; **var** DeletedElements: TElements);

**begin**

**if** (Element <> **nil**) **then**

**begin**

DeletedElements[CurrentIndex] := Element.Number;

Inc(CurrentIndex);

DeleteElement(Element.Sons[**true**], DeletedElements);

DeleteElement(Element.Sons[**false**], DeletedElements);

FreeMem(Element);

**end**;

**end**;

**function** **FindAnswer**(Element: TPointer): **Integer**;

**var**

Values: **array** [**false** .. **true**] **of** **Integer**;

i: **Boolean**;

**begin**

Result := **0**;

**if** (Element <> **nil**) **then**

**begin**

**for** i := **false** **to** **true** **do**

**begin**

Values[i] := FindAnswer(Element.Sons[i]);

**end**;

**if** (Values[**true**] <> Values[**false**]) **then**

Answer := Answer + IntToStr(Element.Number) + ', ';

Result := Values[**true**] + Values[**false**] + **1**;

**end**;

**end**;

**procedure** **AddElement**(Pos: TTrack; Number: **Integer**); **export**;

**var**

NewElement, PastElement: TPointer;

Navigation: **Boolean**;

**begin**

PastElement := **nil**;

NewElement := Root;

**for** Navigation **in** Pos **do**

**begin**

PastElement := NewElement;

NewElement := NewElement.Sons[Navigation];

**end**;

**new**(NewElement);

NewElement.Sons[**true**] := **nil**;

NewElement.Sons[**false**] := **nil**;

NewElement.Number := Number;

**if** (PastElement <> **nil**) **then**

PastElement.Sons[Pos[High(Pos)]] := NewElement

**else**

Root := NewElement;

**end**;

**function** **DeleteElementByPos**(Pos: TTrack): TElements; **export**;

**var**

FindElement, Father: TPointer;

Direction: **Boolean**;

DeletedElements: TElements;

**begin**

SetLength(DeletedElements, MaxElementCount);

CurrentIndex := **0**;

Father := **nil**;

FindElement := Root;

**for** Direction **in** Pos **do**

**begin**

Father := FindElement;

FindElement := FindElement.Sons[Direction];

**end**;

DeleteElement(FindElement, DeletedElements);

**if** (Father <> **nil**) **then**

Father.Sons[Pos[High(Pos)]] := **nil**

**else**

Root := **nil**;

SetLength(DeletedElements, CurrentIndex);

DeleteElementByPos := DeletedElements;

**end**;

**function** **GetAnswer**(): **String**; **export**;

**begin**

Answer := '';

FindAnswer(Root);

**if** (Length(Answer) > **0**) **then**

SetLength(Answer, Length(Answer) - **2**);

Result := Answer;

**end**;

**exports**

AddElement, DeleteElementByPos, GetAnswer;

**begin**

Root := **nil**;

**end**.

**Схема алгоритма**



**Скриншоты**



