**МИНИСТЕРСТВО НАУКИ и ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Коломенский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Московский политехнический университет»**

**Направление:** Информатика и вычислительная техника

**Профиль:** Программное обеспечение вычислительной техники

и автоматизированных систем

**Форма обучения:** очная

**КУРСОВОЙ ПРОЕКТ**

По дисциплине «Структуры данных и алгоритмы их обработки»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_ / М.А. Ковалёв

(подпись) И.О. Фамилия

Руководитель

курсового проекта

старший преподаватель

кафедры АП и ИТ \_\_\_\_\_\_\_\_\_\_\_\_\_ / И.Н.Филоненко

(подпись) И.О. Фамилия

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Коломна

2021

– организация иерархической структуры данных: бинарное дерево поиска(БД);

– создание БД, поиск и вставка элемента в БД, удаления элемента из БД, обход дерева тремя стандартными способами.

Написать программу, иллюстрирующую работу с организованной библиотекой.

**Код библиотеки (Unit):**

unit TreeLib;

interface

uses

System.SysUtils,

Vcl.Dialogs;

type

TData = Integer;

TPNode = ^TNode;

TNode = record

Data: TData;

PLeft, PRight: TPNode;

end;

TreeArrT = array[0..100] of TData;

TTree = class

TreePointer: TPNode;

Data: TData;

OutData: string;

Private

LeftNode: TPNode;

DataSaver: integer;

Procedure FindLeft(Pointer: TPNode);

Public

procedure AddNode(var aPNode: TPNode; const aData: TData);

function TreeWrite(const aPNode : TPNode; WriteType:char): string;

procedure FillTree(ArraySize: integer; MainArray:TreeArrT; var Pointer: TPNode);

procedure Find(const aPNode : TPNode; ForSearch: integer);

procedure Add(var Pointer : TPNode; ForAdd: integer);

Procedure Del(Pointer: TPNode; ForDel: Integer);

constructor Create();

end;

Var

Way: string;

NodeSaver: TPNode;

implementation

constructor TTree.Create();

begin

TreePointer := nil;

end;

Procedure TTree.FindLeft(Pointer: TPNode);

begin

if Pointer^.PLeft = nil then

begin

LeftNode := Pointer;

exit

end

else

FindLeft(Pointer^.PLeft);

end;

procedure TTree.Del(Pointer: TPNode; ForDel: Integer);

Var

LSave, RSave: TPNode;

begin

if Pointer = nil then

exit;

if ForDel > Pointer^.Data then

begin

NodeSaver := Pointer;

Del(Pointer^.PRight, ForDel);

end

else if ForDel < Pointer^.Data then

begin

NodeSaver := Pointer;

Del(Pointer^.PLeft, ForDel);

end

else if ForDel = Pointer^.Data then

begin

if (Pointer^.PLeft = nil) and (Pointer^.PRight = nil) then

begin

if Pointer^.Data > NodeSaver^.Data then

NodeSaver^.PRight := nil

else if Pointer^.Data < NodeSaver^.Data then

NodeSaver^.PLeft := nil;

Dispose(Pointer);

end

else if (Pointer^.PLeft <> nil) and (Pointer^.PRight = nil) then

begin

if Pointer^.Data > NodeSaver^.Data then

begin

NodeSaver^.PRight := Pointer^.PLeft;

Pointer := nil;

Dispose(Pointer);

end

else if Pointer^.Data < NodeSaver^.Data then

begin

NodeSaver^.PLeft := Pointer^.PLeft;

Pointer := nil;

Dispose(Pointer);

end;

end

else if (Pointer^.PLeft = nil) and (Pointer^.PRight <> nil) then

begin

if Pointer^.Data > NodeSaver^.Data then

begin

NodeSaver^.PRight := Pointer^.PRight;

Pointer := nil;

Dispose(Pointer);

end

else if Pointer^.Data < NodeSaver^.Data then

begin

NodeSaver^.PLeft := Pointer^.PRight;

Pointer := nil;

Dispose(Pointer);

end;

end

else if (Pointer^.PLeft <> nil) and (Pointer^.PRight <> nil) then

begin

FindLeft(Pointer^.PRight);

DataSaver := LeftNode^.Data;

Del(Pointer, DataSaver);

Pointer^.Data := DataSaver;

end;

end;

end;

procedure TTree.Add(var Pointer: TPNode; ForAdd: integer);

var

AddPointer: TPNode;

begin

if Pointer = nil then

begin

AddNode(Pointer, ForAdd);

ShowMessage('Добавлено: ' + inttostr(ForAdd));

exit;

end;

if ForAdd > Pointer^.Data then

Add(Pointer^.PRight, ForAdd)

else if ForAdd < Pointer^.Data then

Add(Pointer^.PLeft, ForAdd)

else if ForAdd = Pointer^.Data then

ShowMessage(('Такой потомок уже существует!'));

end;

procedure TTree.Find(const aPNode: TPNode; ForSearch: integer);

begin

if aPNode = nil then

begin

ShowMessage('Совпадений нет!');

exit;

end;

if ForSearch > aPNode^.Data then

begin

Way := Way + ' -> ' + inttostr(aPNode^.Data);

Find(aPNode^.PRight, ForSearch);

end

else if ForSearch < aPNode^.Data then

begin

Way := Way + ' -> ' + inttostr(aPNode^.Data);

Find(aPNode^.PLeft, ForSearch);

end

else if ForSearch = aPNode^.Data then

ShowMessage(('Найдено: ' + inttostr(ForSearch) + #13 + 'Tree' + Way + ' -> ' + inttostr(ForSearch)));

end;

procedure TTree.FillTree(ArraySize: integer; MainArray:TreeArrT; var Pointer: TPNode);

var

i: integer;

begin

for i := 0 to ArraySize - 1 do

AddNode(Pointer, MainArray[i]);

end;

procedure TTree.AddNode(var aPNode: TPNode; const aData: TData);

begin

if aPNode = nil then

begin

New(aPNode);

aPNode^.Data := aData;

aPNode^.PLeft := nil;

aPNode^.PRight := nil;

end

else if aData <= aPNode^.Data then

AddNode(aPNode^.PLeft, aData)

else if aData > aPNode^.Data then

AddNode(aPNode^.PRight, aData);

end;

function TTree.TreeWrite(const aPNode : TPNode; WriteType:char): string;

begin

if aPNode = nil then

exit;

if WriteType = 's' then

begin

TreeWrite(aPNode^.PLeft, WriteType);

OutData := OutData + ' ' + inttostr(aPNode^.Data);

TreeWrite(aPNode^.PRight, WriteType);

end

else if WriteType = 'd' then

begin

OutData := OutData + ' ' + inttostr(aPNode^.Data);

TreeWrite(aPNode^.PLeft, WriteType);

TreeWrite(aPNode^.PRight, WriteType);

end

else if WriteType = 'r' then

begin

TreeWrite(aPNode^.PLeft, WriteType);

TreeWrite(aPNode^.PRight, WriteType);

OutData := OutData + ' ' + inttostr(aPNode^.Data);

end

else

result := 'ERROR';

result := OutData;

end;

end**.**

**Код формы:**

unit Unit1;

interface

uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ComCtrls, Vcl.ExtCtrls, TreeLib;

type

TForm1 = class(TForm)

TreeView1: TTreeView;

Button1: TButton;

Memo1: TMemo;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Edit1: TEdit;

Label5: TLabel;

Button2: TButton;

RadioGroup1: TRadioGroup;

Edit2: TEdit;

Button3: TButton;

Button4: TButton;

Button5: TButton;

Label1: TLabel;

Label6: TLabel;

procedure Button1Click(Sender: TObject);

procedure RadioGroup1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

MTree: TTree;

Form1: TForm1;

InputArray: TreeArrT;

implementation

{$R \*.dfm}

procedure PrintTree(treenode:TTreeNode; root:TPNode);

var newnode : TTreeNode;

begin

if Assigned(root) then

begin

newnode := Form1.TreeView1.Items.AddChild(treenode, IntToStr(root^.Data));

PrintTree(newnode, root^.PLeft);

PrintTree(newnode, root^.PRight);

end;

end;

procedure TForm1.Button1Click(Sender: TObject);

var

i, TreeArrSize: integer;

begin

if Memo1.Lines.Count = 0 then

ShowMessage('Введите данные для построения дерева!')

else

begin

TreeArrSize := Memo1.Lines.Count - 1;

for i := 0 to TreeArrSize do

InputArray[i] := strtoint(Memo1.Lines[i]);

MTree := TTree.Create();

MTree.FillTree(TreeArrSize + 1, InputArray, MTree.TreePointer);

PrintTree(nil, MTree.TreePointer);

RadioGroup1.Enabled := True;

Button3.Enabled := True;

Button4.Enabled := True;

Button5.Enabled := True;

end;

end;

procedure TForm1.Button2Click(Sender: TObject);

var

NTreeVal: integer;

i: Integer;

begin

Memo1.Clear;

NTreeVal := strtoint(Edit1.Text);

for i := 1 to NTreeVal do

Memo1.Lines.Add(inttostr(1 + random(99)));

end;

procedure TForm1.Button3Click(Sender: TObject);

Var

FindVal: integer;

begin

TreeLib.Way := '';

FindVal := strtoint(Edit2.Text);

MTree.Find(MTree.TreePointer, FindVal);

end;

procedure TForm1.Button4Click(Sender: TObject);

Var

AddVal: integer;

begin

TreeView1.Items.Clear;

AddVal := strtoint(Edit2.Text);

MTree.Add(MTree.TreePointer, AddVal);

PrintTree(nil, MTree.TreePointer);

end;

procedure TForm1.Button5Click(Sender: TObject);

Var

DelVal: integer;

begin

TreeView1.Items.Clear;

DelVal := strtoint(Edit2.Text);

MTree.Del(MTree.TreePointer, DelVal);

PrintTree(nil, MTree.TreePointer);

end;

procedure TForm1.RadioGroup1Click(Sender: TObject);

var

OutMode: char;

begin

if RadioGroup1.ItemIndex = 0 then

OutMode := 'd'

else if RadioGroup1.ItemIndex = 1 then

OutMode := 'r'

else if RadioGroup1.ItemIndex = 2 then

OutMode := 's';

MTree.OutData := '';

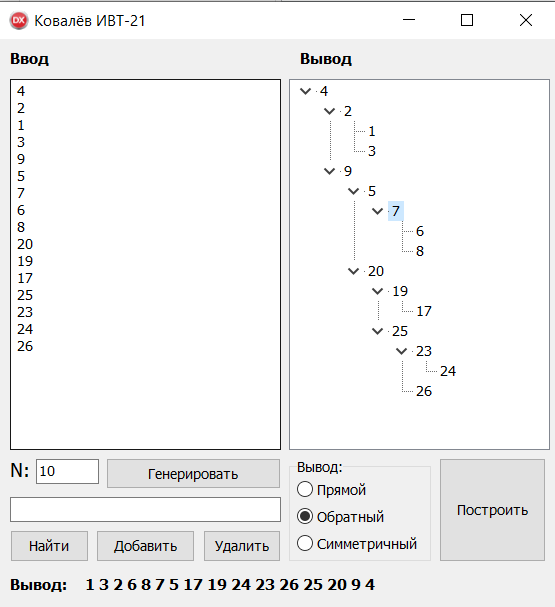
MTree.TreeWrite(MTree.TreePointer, OutMode);

Label6.Caption := MTree.OutData;

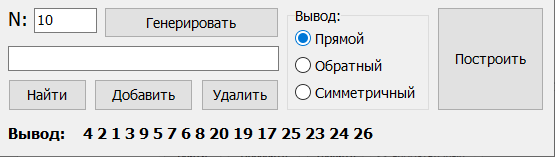
end;

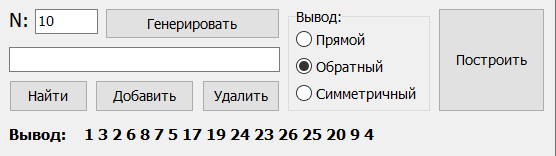
end.

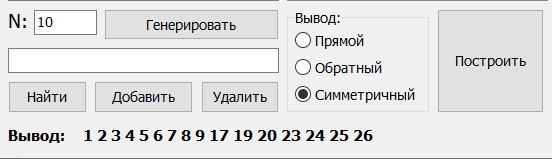
**Работа программы:**

****

**Обход дерева:**

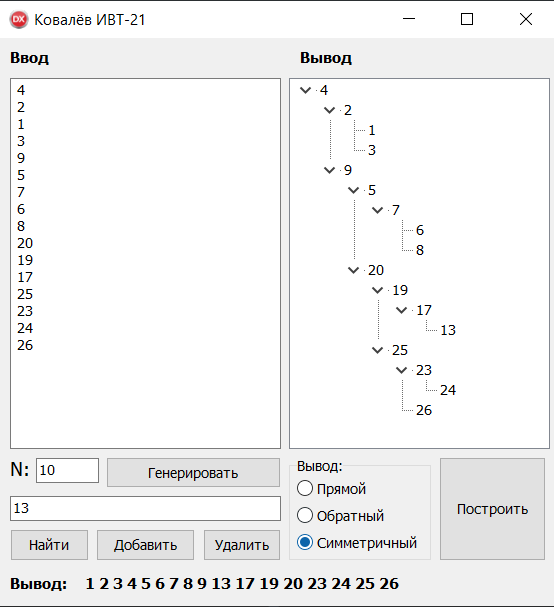
****

****

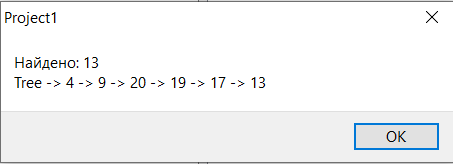


**Найти, добавить, удалить:**

**Добавление:**

****

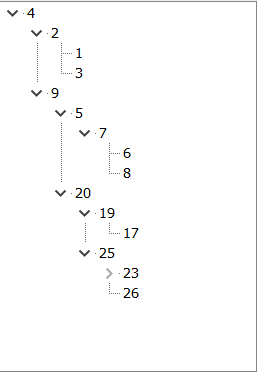
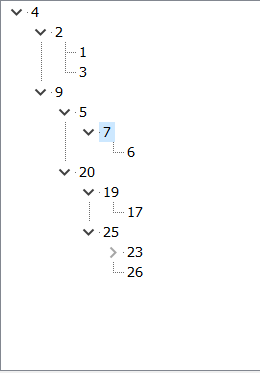
**Поиск:**

****

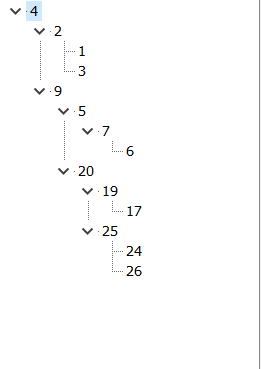
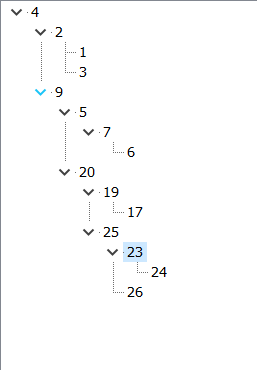
**Удаление:**

*\*См. сл. стр.*

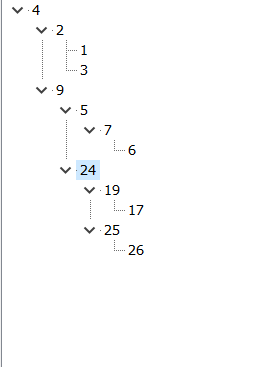
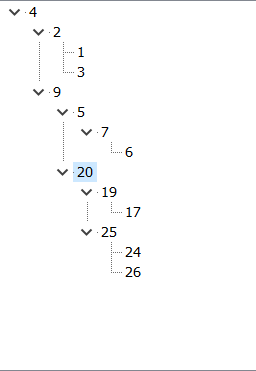
**8 (без потомков)**

****

**23 (Один потомок - 24)**



**20 (2 потомка)**

****