МИНИСТЕРСТВО ОБРАЗОВАНИЯ и НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Коломенский институт (филиал)**

федерального государственного бюджетного образовательного учреждения

высшего образования

**«Московский политехнический университет»**

Профиль «**Программное обеспечение вычислительной техники и автоматизированных систем**»

**Направление подготовки:**

**«Информатика и вычислительная техника»**

Форма обучения: очная

**лабораторная работа № 6**

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /М.C. Хоботов

(подпись) И.О. Фамилия

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ И.Н. Филоненко /

(подпись) И.О. Фамилия

Дата:

Коломна 2020

**Лабораторная работа № 5**

**Деревья , как динамические структуры данных .**

**Цель работы:**

а )Организовать программно двоичное дерево с помощью процедуры поиска и включения ключа (Search).

б )Написать и отладить программу для поэлементного вывода значений узлов построенного дерева обходом дерева слева направо(Inorder).

Замечание: при правильной организации бинарного дерева с целочисленными ключами в результате отображения ключей при обходе в порядке Inorder , должна получиться отсортированная в порядке возрастания последовательность ключей.

***Код программы:***

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;

type

TForm1 = class(TForm)

TreeView1: TTreeView;

Button1: TButton;

Memo1: TMemo;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

procedure Button1Click(Sender: TObject);

procedure FormShow(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

TData = Integer;

TPNode = ^TNode;

TNode = record

Data: TData;

PLeft, PRight: TPNode;

end;

var

Form1: TForm1;

InputData: string;

PTree : TPNode;

implementation

{$R \*.dfm}

procedure 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;

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 TreeWrite(const aPNode : TPNode);

begin

if aPNode = nil then

exit;

TreeWrite(aPNode^.PLeft);

InputData := InputData + inttostr(aPNode^.Data);

Form1.Label2.Caption := InputData;

TreeWrite(aPNode^.PRight);

end;

procedure TForm1.Button1Click(Sender: TObject);

var

aPNode: TPNode ;

Data: TData;

tryAdd: Integer;

i: integer;

begin

for i:= 0 to Memo1.Lines.Count - 1 do

begin

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

AddNode(PTree, Data);

end;

TreeWrite(PTree);

PrintTree(nil, PTree);

end;

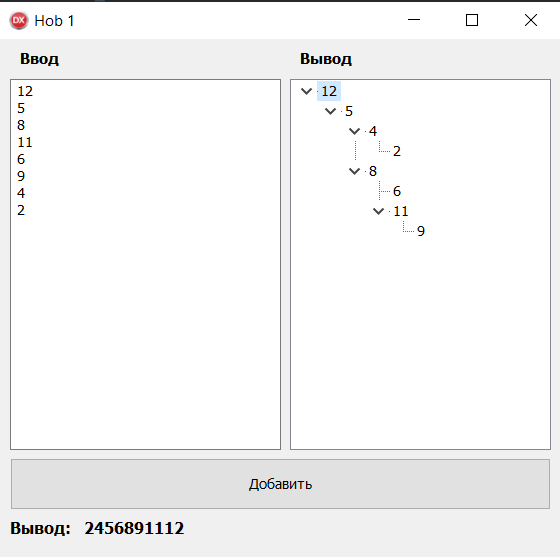
procedure TForm1.FormShow(Sender: TObject);

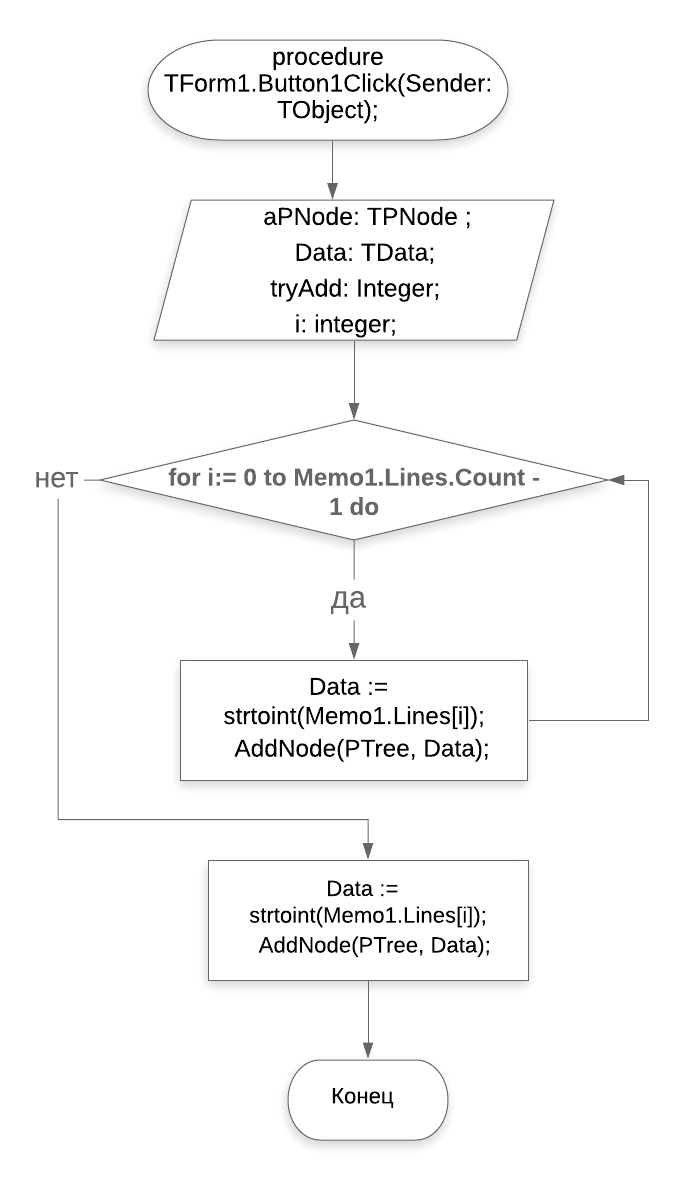
begin

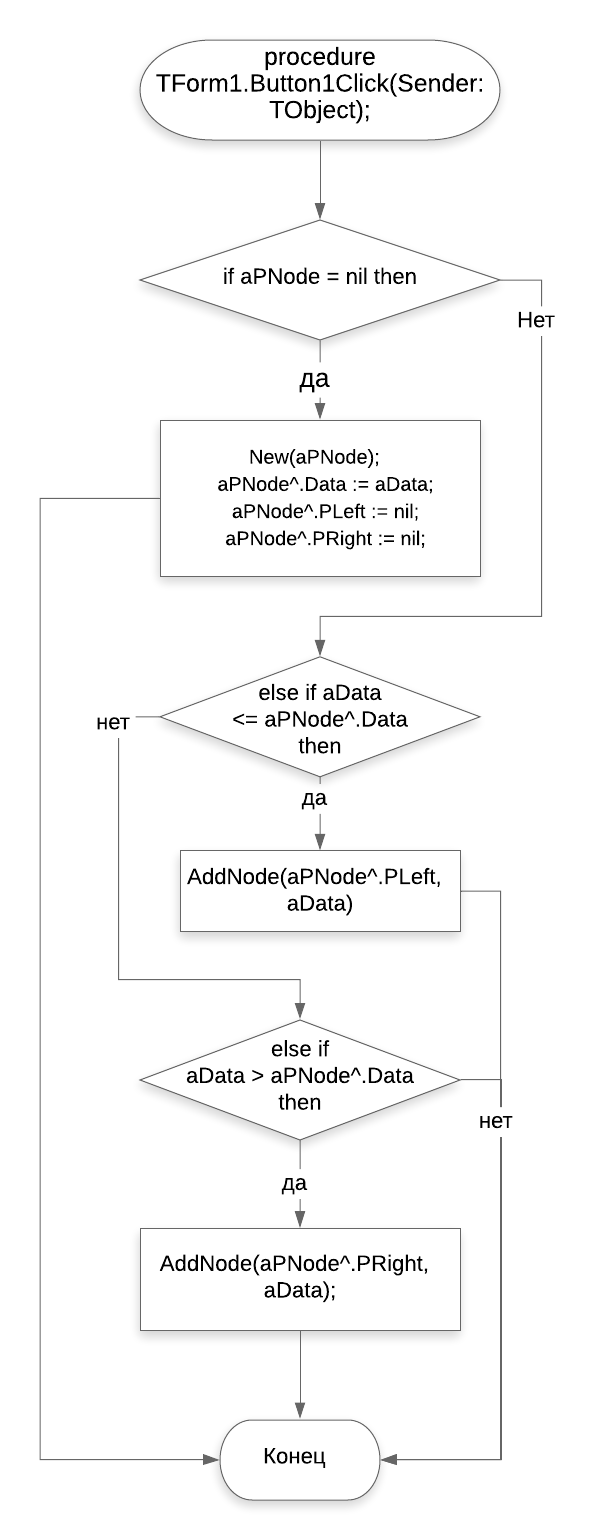
PTree := nil;

end;

end.

******

Button1Click

AddNode

TreeWrite

