**10 ЛАБОРАТОРНАЯ РАБОТА №10**

**«ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОБЪЕКТОВ И МОДУЛЕЙ»**

**10.1 Цель работы**

Изучение базовых понятий объектно-ориентированного программирования, приобретение навыков разработки и отладки программ, использующих модули и объекты. Исследование особенностей поисковых бинарных деревьев, представленных в виде объекта.

**10.2 Вариант задания – 20**

Требуется создать модуль, содержащий описание объекта, который представляет бинарное дерево в соответствии с вариантом из предыдущей лабораторной работы.

**10.3 Порядок выполнения работы**

10.1.1 Разработать модуль на языке Pascal.

10.1.2 Разработать программу на языке Pascal.

10.1.3 Разработать тестовые примеры.

10.1.4 Выполнить отладку программы.

10.1.5 Сделать выводы по проделанной работе.

**10.4 Ход работы**

10.4.1 Текст основной программы

**Program** lab\_10;

**Uses** crt,mod1;

**Var** FileName:string;

num,otstup:integer;

myTree:^tree;

//// Основная программа \\\\

**BEGIN**

FileName:='C:\PABCWork.NET\lab\_9.txt';

Writeln;

Writeln('============================================');

Writeln(' Программа работает с бинарными деревьями ');

Writeln(' (дерево упрорядочено по номерам поездов) ');

Writeln('============================================');

Writeln;

readkey;

clrscr;

**While** true **do**

**begin**

Writeln;

Writeln('1 - Создание нового дерева');

Writeln('2 - Удаление дерева');

Writeln('3 - Просмотр дерева');

Writeln('4 - Узнать входит ли элемент в дерево'); //my

Writeln('5 - Сохранить дерево');

Writeln('6 - Загрузить дерево');

Writeln('7 - Выход из программы');

Writeln;

Write('Ввод -> ');readln(num);

ClrScr;

**case** num **of**

1:

**Begin**

CreateTree(myTree);

clrscr;

**End**;

2:

**Begin**

Writeln('Дерево было удалено');

DeleteTree(myTree);

readkey;

clrscr;

**End**;

3:

**Begin**

Writeln('Выполняется просмотр дерева':50);

Writeln('---------------------------':50);

Writeln('| Номер поезда |':50);

Writeln('| Станция отправления |':50);

Writeln('| Станция прибытия |':50);

Writeln('| Время отправления |':50);

Writeln('| Время прибытия |':50);

Writeln('| Стоимость билета |':50);

Writeln('---------------------------':50);

Writeln;

ShowTree(myTree,otstup);

readkey;

ClrScr;

**end**;

4:

**Begin**

Writeln('Выполняется проверка присутствия элемента в дереве');

Write('Введите номер искомого поезда -> '); readln(num);

SearchInTree(myTree,num);

ClrScr;

**end**;

5:

**Begin**

SaveTree(myTree,FileName);

Writeln('Дерво было успешно сохранено!');

readkey;

ClrScr;

**end**;

6:

**Begin**

DownloadTree(myTree,FileName);

Writeln('Дерво было успешно загружено!');

readkey;

ClrScr;

**end**;

7: **Exit**;

**end**;

**end**;

**END**.

10.4.2 Текст программы в модуле mod1

**Unit** mod1;

**interface**

**Type**

rasp=**record**

n,cost:integer; //номер поезда,стоимость билета

sO,sP:string[20]; //станция отправления/прибытия

vO,vP:string[5]; //время отправления/прибытия

**end**;

tree=**record**

data:rasp;

left,right:^tree;

**end**;

**Procedure** DeleteTree(**var** tmp:^tree);

**Procedure** AddElement(**var** root:^tree; **const** elem:^tree);

**Procedure** CreateTree (**var** root:^tree);

**Procedure** ShowTree (**const** root:^tree; otstup:integer);

**Procedure** SearchInTree(root:^tree; poezd:integer);

**Procedure** SaveElement(**const** elem:^tree; **var** fileP: **file of** rasp);

**Procedure** SaveTree(**const** root:^tree; FileName:string);

**Procedure** DownloadTree(**var** root:^tree; **const** FileName:string);

**implementation**

**uses** crt;

//// Удаление дерева \\\\

**Procedure** DeleteTree(**var** tmp:^tree);

**Begin**

**if** tmp<>nil **then**

**begin**

// Очистка левой и правой ветви

DeleteTree(tmp^.left);

DeleteTree(tmp^.right);

// Удаление корня

dispose(tmp);

tmp:=nil;

**end**;

**End**;

//// Добавить новый элемент \\\\

**Procedure** AddElement(**var** root:^tree; **const** elem:^tree);

**begin**

**if** root = nil **then**

**begin**

root := elem;

**EXIT**;

**end**;

**if** root^.data.n < elem^.data.n **then**

AddElement(root^.right,elem)

**else**

AddElement(root^.left,elem);

**end**;

//// Процедура создания дерева \\\\

**Procedure** CreateTree (**var** root:^tree);

**Var** tmp:^tree; r:rasp;

**Begin**

//удалить предыдущее дерево если таково было

DeleteTree(root);

Writeln('Создаём дерево');

writeln;

write('Введите номер поезда: '); readln(R.n);

write('Введите станцию отправления: '); readln(R.sO);

write('Введите станцию прибытия: '); readln(R.sP);

write('Введите время отправления: '); readln(R.vO);

write('Введите время прибытия: '); readln(R.vP);

write('Введите стоимость билета: '); readln(R.cost);

**new**(tmp);

tmp^.data:=r;

root:=tmp;

readkey;

**while** true **do**

**begin**

clrscr;

write('Введите номер поезда: '); readln(R.n);

write('Введите станцию отправления (\*-выход): '); readln(R.sO);

**if** R.sO = '\*' **then Exit**;

write('Введите станцию прибытия: '); readln(R.sP);

write('Введите время отправления: '); readln(R.vO);

write('Введите время прибытия: '); readln(R.vP);

write('Введите стоимость билета: '); readln(R.cost);

**new**(tmp);

tmp^.data:=r;

AddElement(root,tmp);

readkey;

**end**;

**End**;

//// Процедура демонстрации дерева \\\\

**Procedure** ShowTree (**const** root:^tree; otstup:integer);

**begin**

**if** root<>nil **then**

**Begin**

otstup:=otstup+10;

ShowTree(Root^.right,otstup);

Writeln(' ':otstup, Root^.Data.n);

Writeln(' ':otstup, Root^.Data.sO);

Writeln(' ':otstup, Root^.Data.sP);

Writeln(' ':otstup, Root^.Data.vO);

Writeln(' ':otstup, Root^.Data.vP);

Writeln(' ':otstup, Root^.Data.cost);

ShowTree(Root^.left, otstup);

**End**

**end**;

//// Узнать входит ли элемент в дерево \\\\

**Procedure** SearchInTree(root:^tree; poezd:integer);

**Var** found:boolean; i:integer;

**Begin**

Found:=false;

**While** (root<>nil) **and** (**not** found) **and** (i<>1000) **do**

**begin**

i:=i+1;

**if** root^.data.n=poezd **then**

found:=true

**else**

**if** root^.data.n>poezd **then**

root:=root^.left

**else**

root:=root^.right;

**end**;

**if** found=false **then**

**begin**

Writeln('Я не нашёл такой поезд :(');

readkey;

**exit**;

**end**

**else**

**begin**

writeln('Поезд найден!!! :)');

Writeln('------------------------------------------------------');

Writeln('| Номер поезда: | ',root^.data.n:25,'|');

Writeln('| Станция отправления: | ',root^.data.sO:25,'|');

Writeln('| Станция прибытия: | ',root^.data.sP:25,'|');

Writeln('| Время отправления: | ',root^.data.vO:25,'|');

Writeln('| Время прибытия: | ',root^.data.vP:25,'|');

Writeln('| Стоимость билета: | ',root^.data.cost:25,'|');

Writeln('------------------------------------------------------');

Writeln;

readkey;

**end**;

**end**;

//// Сохранение одного элемента \\\\

**Procedure** SaveElement(**const** elem:^tree; **var** fileP: **file of** rasp);

**begin**

**if** elem = nil **then exit**;

write(fileP, elem^.data);

SaveElement(elem^.left, fileP);

SaveElement(elem^.right, fileP);

**end**;

//// Сохранения дерева в файл \\\\

**Procedure** SaveTree(**const** root:^tree; FileName:string);

**var** fileP: **file of** rasp;

**begin**

assign(fileP, fileName);

rewrite(fileP);

SaveElement(root,fileP);

close(fileP);

writeln;

**end**;

//// Загрузка дерева из файла \\\\

**Procedure** DownloadTree(**var** root:^tree; **const** FileName:string);

**var** fileP: **file of** rasp;

element:^tree;

**begin**

assign(fileP, fileName);

reset(fileP);

DeleteTree(root);

**while not** eof(fileP) **do**

**begin**

**new**(element);

element^.right:=nil;

element^.left:=nil;

read(fileP, element^.data);

AddElement(root,element);

element:=nil;

**end**;

close(fileP);

writeln;

**end**;

**begin**

**end**.

10.4.3 Выполнена отладка программы. Результаты тестирования отображены на рисунках 10.1 – 10.6. На рисунках 10.1 и 10.2 видно, как запускается программа, которая предлагает выбрать одну из семи процедур и мы как сознательные пользователи выбираем первую процедуру и начинаем создавать дерево.

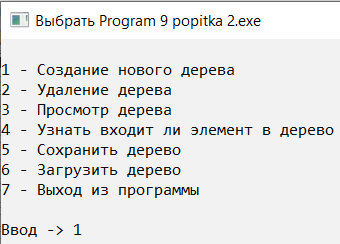


Рисунок 10.1 – Главное меню программы

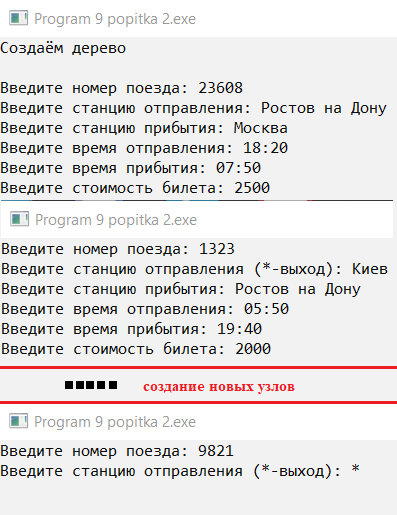


Рисунок 10.2 – Создание дерева

На рисунке 10.3 мы вызываем процедуру демонстрации дерева.

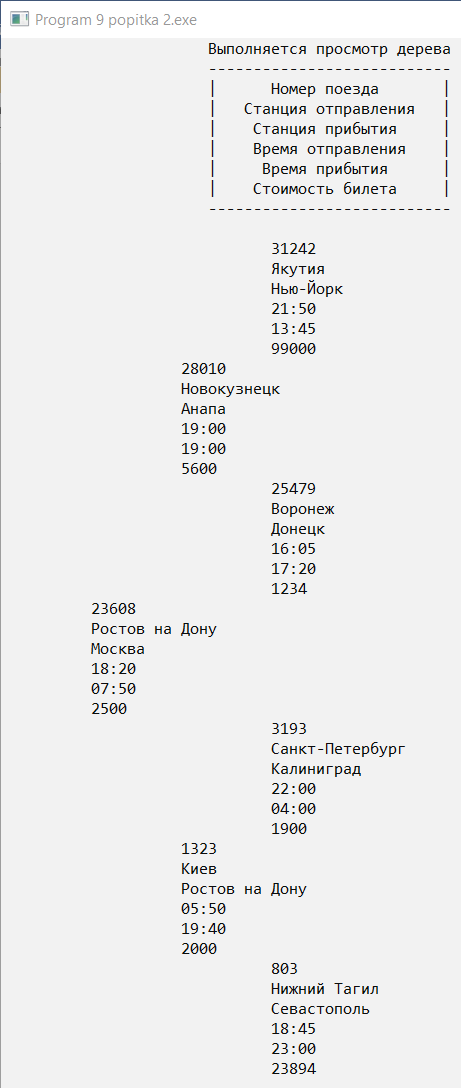


Рисунок 10.3 – Процедура демонстрации дерева

На рисунке 10.4 мы вызываем четвёртую процедуру которая проверяет входит ли элемент в дерево. В первый раз мы вводим несуществующий поезд, а во второй раз существующий под номером «25479».

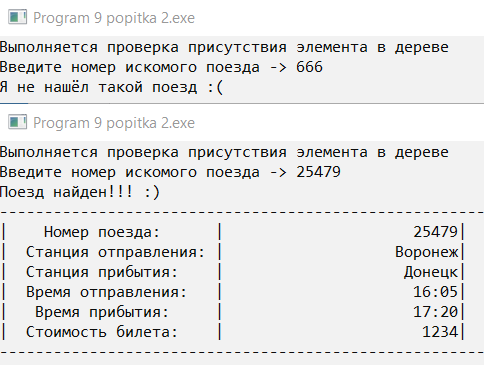


Рисунок 10.4 – Процедура проверки наличия элемента в дереве

На рисунках 10.5 и 10.6 мы сохраняем дерево в файл, затем удаляем наше дерево второй процедурой, в результате чего нашего дерева в программе нет. Затем с помощью процедуры загрузки мы загружаем дерево из файла в программу.

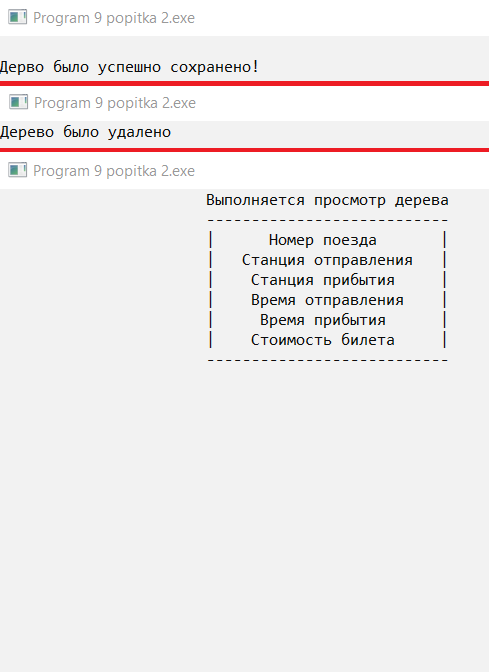


Рисунок 10.5 – Процедуры сохранения, удаления и демонстрации дерева

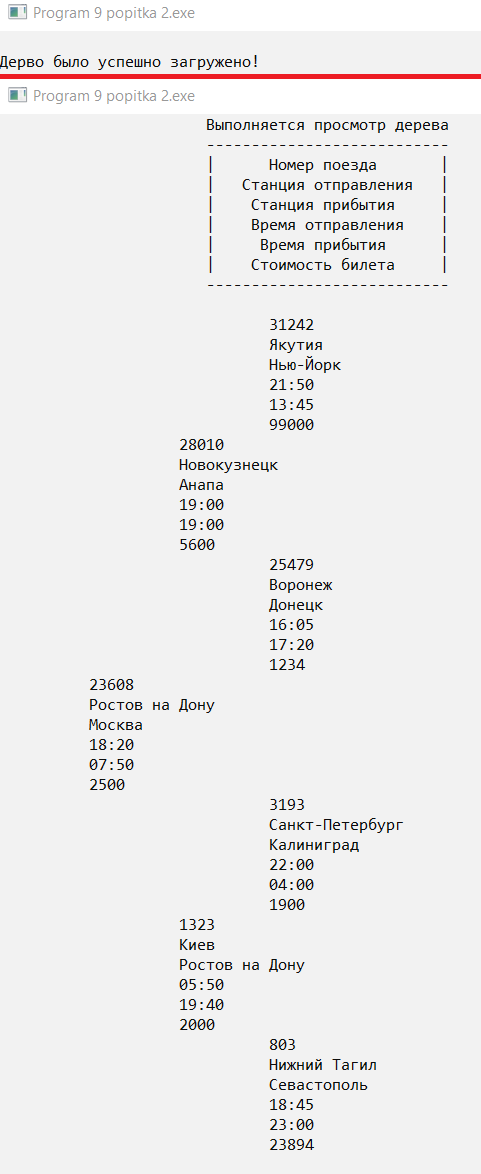


Рисунок 10.6 – Процедуры загрузки и демонстрации дерева

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были получены навыки cоздания программ, работающих с объектами и модулями. Также были повторно закреплены навыки отладки программы, работы с файлами, а именно чтение данных из бинарного типизированного файла и запись в него, работы с ссылками, с бинарными деревьями. Полученные навыки в будущем помогут создавать более сложные объекты.