1. Лабораторная работа № 10

«Программирование с использованием объектов и модулей»

* 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.

## Постановка задачи

Получить навыки создания программ, использующих модули и объекты.

* 1. Ход выполнения работы

Вариант выполнения задания – 18.

* + 1. Текст программы в основном файле

Program LW\_10;

Uses Crt;

Uses TreeABC;

Uses IOFile;

var answer: integer; answerChar: char;

fileName: string[128];

i: Data; isDataChanged: boolean;

BEGIN

var c := new BinTree();

isDataChanged := false;

fileName := 'database-v9.18.db';

repeat

clrscr;

alert(110, 'Интерактивное меню программы');

alert(010, concat('1 -> Загрузить дерево из файла (по умолчанию "', fileName, '").'));

alert(010, concat('2 -> Сохранить дерево в файл (по умолчанию "', fileName, '").'));

alert(010, '3 -> Удалить старое и создать новое дерево.');

alert(010, '4 -> Добавить элемент в дерево.');

alert(010, '5 -> Печать дерева на экран (слева направо).');

alert(010, '6 -> \* my \* Найти элемент и вывести длину пути.');

alert(010, '7 -> Выход из программы.');

alert(120, 'номер действия');

readln(answer);

clrscr;

case answer of

1:

begin

alert(130, 'Дерево в памяти будет удалено.');

alert(011, 'Запущен процесс загрузки нового дерева...');

Tree\_LoadFromFile(c, fileName);

alert(020, 'какой-то символ для продолжения...');

isDataChanged := false;

end;

2:

begin

alert(130, 'Сохранённое в файле дерево будет удалено.');

alert(011, 'Запущен процесс сохранения дерева...');

Tree\_SaveToFile(c, fileName);

alert(020, 'какой-то символ для продолжения...');

isDataChanged := false;

end;

3:

begin

alert(130, 'Старое дерево удалено.');

alert(011, 'Запущен процесс создания нового дерева...');

Tree\_Create(c);

alert(110, 'Новые элементы были успешно добавлены.');

alert(020, 'какой-то символ для продолжения...');

isDataChanged := true;

end;

4:

begin

alert(111, 'Запуск процесса добавления элемента в дерево...');

alert(020, 'ФИО (выйти нельзя)'); readln(i.name);

alert(020, 'год рождения'); readln(i.year);

alert(020, 'пол товарища'); readln(i.sex);

alert(020, 'семейное положение'); readln(i.familyPos);

alert(020, 'количество детей'); readln(i.childrenCnt);

alert(020, 'оклад'); readln(i.salary);

c.AddElem(i);

alert(110, 'Новый элемент был успешно добавлен.');

alert(020, 'какой-то символ для продолжения...');

isDataChanged := true;

end;

5:

begin

alert(111, 'Запуск процесса печати дерева на экран...');

if c.GetRoot() = nil then

alert(040, 'Дерево в оперативной памяти не существует.')

else

begin

Tree\_Print(c.GetRoot(), 1);

alert(110, 'Всё дерево расположилось на ЖК экране.');

end;

alert(020, 'какой-то символ для продолжения...');

end;

6:

begin

alert(111, 'Запуск процесса поиска элемента в дереве...');

if c.GetRoot() <> nil then

begin

alert(020, 'ФИО (выйти нельзя)'); readln(i.name);

{alert(020, 'год рождения'); readln(i.year);

alert(020, 'пол товарища'); readln(i.sex);

alert(020, 'семейное положение'); readln(i.familyPos);

alert(020, 'количество детей'); readln(i.childrenCnt);

alert(020, 'оклад'); readln(i.salary);

}

answer := c.GetPathToElem(i);

if answer = -1 then

alert(141, 'Увы, но введённый элемент не найден...')

else

alert(111, concat('Найдено. Длина пути: ', intToStr(answer - 1), ' у.е.'));

end

else

alert(041, 'Для начала заполните дерево.');

alert(020, 'какой-то символ для продолжения...')

end;

7:

begin

alert(110, 'Поступило предложение выйти из программы.');

if isDataChanged then

begin

alert(130, 'Есть несохранённые данные.');

alert(020, 'символ для выбора действия (Y,Д/n,н)');

readln(answerChar);

if not ( (lowerCase(answerChar) = 'n') or (lowerCase(answerChar) = 'н') ) then

Tree\_SaveToFile(c, fileName);

end;

alert(020, 'какой-то символ для выхода...');

end;

end;

readkey;

until answer = 7;

END.

* + 1. Текст программы в модуле TreeABC

unit TreeABC;

interface

Uses Crt;

Type Data = record

name: string[32];

year: integer;

sex: string[10];

familyPos: string[16];

childrenCnt: word;

salary: integer;

end;

Tree = record

info: Data;

pL, pR: ^Tree;

end;

BinTree = class

private

rootElem: ^Tree;

procedure \_\_DeleteElem(var p: ^Tree);

begin

if p <> nil then

begin

self.\_\_DeleteElem(p^.pL);

self.\_\_DeleteElem(p^.pR);

dispose(p);

p := nil;

end;

end;

procedure \_\_AddElem(var p: ^Tree; const elem: Data);

begin

if p = nil then

begin

new(p);

p^.pL := nil; p^.pR := nil;

p^.info := elem

end

else

if p^.info.name > elem.name then

self.\_\_AddElem(p^.pL, elem)

else

self.\_\_AddElem(p^.pR, elem);

end;

function \_\_SearchElem(const root:^Tree; const elem: Data; const level: integer): integer;

begin

Result := -1;

if root = nil then exit;

if root^.info.name = elem.name then

Result := level

else

if root^.info.name < elem.name then

Result := self.\_\_SearchElem(root^.pR, elem, level + 1)

else

Result := self.\_\_SearchElem(root^.pL, elem, level + 1);

end;

public

constructor();

begin

self.rootElem := nil;

end;

procedure DeleteAll;

begin

self.\_\_DeleteElem(self.rootElem);

end;

procedure AddElem(const elem: Data);

begin

self.\_\_AddElem(self.rootElem, elem);

end;

function GetRoot() := self.rootElem;

function GetPathToElem(const elem: Data) := self.\_\_SearchElem(self.rootElem, elem, 1);

end;

// Процедура создания нового дерева или нового поддерева

procedure Tree\_Create (var c: BinTree);

// Процедура отображения дерева или его подчасти

procedure Tree\_Print(const root: ^Tree; const level: integer);

// Процедура интерактивного вывода сообщений

procedure alert(const messageType: word; const message: string);

implementation

procedure Tree\_Create (var c: BinTree);

var i: Data;

begin

c.DeleteAll();

alert(020, 'ФИО товарища (выйти нельзя)'); readln(i.name);

alert(020, 'год рождения'); readln(i.year);

alert(020, 'пол товарища'); readln(i.sex);

alert(020, 'семейное положение'); readln(i.familyPos);

alert(020, 'количество детей'); readln(i.childrenCnt);

alert(020, 'оклад'); readln(i.salary);

c.AddElem(i);

alert(110, 'Подготовка к вводу второго товарища (ввод можно будет отменить).');

alert(120, 'любую кнопку для ввода товарища...');

readkey;

while true do

begin

clrscr;

alert(110, 'Переход к вводу следующего товарища...');

alert(020, 'ФИО (\* для выхода)'); readln(i.name);

if i.name = '\*' then break;

alert(020, 'год рождения'); readln(i.year);

alert(020, 'пол товарища'); readln(i.sex);

alert(020, 'семейное положение'); readln(i.familyPos);

alert(020, 'количество детей'); readln(i.childrenCnt);

alert(020, 'оклад'); readln(i.salary);

c.AddElem(i);

alert(110, 'Подготовка к вводу следующего товарища.');

alert(120, 'любую кнопку для ввода следующего товарища...');

readkey;

end;

end;

procedure Tree\_Print(const root: ^Tree; const level: integer);

var m: integer;

begin

if root = nil then

exit;

m := round( ( log(level) + 0.2 ) \* 8 );

writeln();

Tree\_Print(root^.pR, level + 1);

write('':(m\*(level-1))); alert(050, concat(' Уровень: ', intToStr(level)));

write('':(m\*(level-1))); alert(050, concat(' ФИО: ', root^.info.name));

write('':(m\*(level-1))); alert(050, concat('Год рождения: ', intToStr(root^.info.year)));

write('':(m\*(level-1))); alert(050, concat('Пол товарища: ', root^.info.sex));

write('':(m\*(level-1))); alert(050, concat('Семейное положение: ', root^.info.familyPos));

write('':(m\*(level-1))); alert(050, concat('Количество детей: ', intToStr(root^.info.childrenCnt)));

write('':(m\*(level-1))); alert(050, concat(' Оклад: ', intToStr(root^.info.salary)));

writeln();

Tree\_Print(root^.pL, level + 1);

end;

procedure alert(const messageType: word; const message: string);

const maxWidth = 68;

begin

case (messageType div 100) of

1:

writeln('<===================>', ' ':(maxWidth-40), '<===================>');

end;

case ((messageType mod 100) div 10) of

1:

begin

{ Обычное сообщение }

writeln('| ', message, ' |':(maxWidth-length(message)));

end;

2:

begin

{ Сообщение с приглашением для ввода}

writeln('| Введите ', message);

write ('| пользователь$: ')

end;

3:

begin

{ Сообщение с предупреждением }

writeln('| ', ' ( ! ) Внимание ( ! )', ' |':(maxWidth-22));

writeln('| ', message, ' |':(maxWidth-length(message)));

end;

4:

begin

{ Сообщение с ошибкой }

writeln('| ', ' ( x ) Ошибка ( x )', ' |':(maxWidth-20));

writeln('| ', message, ' |':(maxWidth-length(message)));

end;

5:

begin

{ Обычное сообщение (только левая грань) }

writeln('| ', message);

end;

end;

case (messageType mod 10) of

1:

writeln('<===================>', ' ':(maxWidth-40), '<===================>');

end;

end;

end.

* + 1. Текст программы в модуле IOFile

unit IOFile;

interface

Uses TreeABC;

// Процедура сохранение одного элемента

procedure Tree\_SaveToFile\_ForEachElem(const elem: ^Tree; var fileP: file of Data);

// Процедура сохранения дерева в файл

procedure Tree\_SaveToFile(const c: BinTree; const fileName: string);

// Процедура загрузки дерева из файла

procedure Tree\_LoadFromFile(var c: BinTree; const fileName: string);

implementation

procedure Tree\_SaveToFile\_ForEachElem(const elem: ^Tree; var fileP: file of Data);

begin

if elem = nil then exit;

write(fileP, elem^.info);

Tree\_SaveToFile\_ForEachElem(elem^.pL, fileP);

Tree\_SaveToFile\_ForEachElem(elem^.pR, fileP);

end;

procedure Tree\_SaveToFile(const c: BinTree; const fileName: string);

var fileP: file of Data;

begin

if c.GetRoot() = nil then

begin

alert(040, 'Нечего сохранять в файл, а значит данные в файле сохранены.');

exit

end;

assign(fileP, fileName);

rewrite(fileP);

Tree\_SaveToFile\_ForEachElem(c.GetRoot(), fileP);

close(fileP);

alert(010, 'Данные успешно загружены в файл.');

end;

procedure Tree\_LoadFromFile(var c: BinTree; const fileName: string);

var fileP: file of Data;

counter: integer;

i: Data;

begin

assign(fileP, fileName);

if not fileExists(fileName) then

begin

alert(040, 'Файл сохранений не доступен для чтения. Будет создан новый файл.');

rewrite(fileP);

end

else

begin

reset(fileP);

if fileSize(fileP) < 1 then

alert(040, 'Данные в файле отсутствуют. Ничего не загруженено, всё сохранено.')

else

begin

c.DeleteAll();

counter := 0;

reset(fileP);

while not eof(fileP) do

begin

read(fileP, i); c.AddElem(i);

counter := counter + 1;

end;

alert(010, concat('Успешно импортировано записей: ', intToStr(counter), '.'));

end

end;

close(fileP)

end;

end.

* + 1. Описание тестовых примеров

На Рисунках 10.1 – 10.10 продемонстрированы тестовые примеры, показывающие работоспособность программы.

На Рисунке 10.1 показано меню программы.

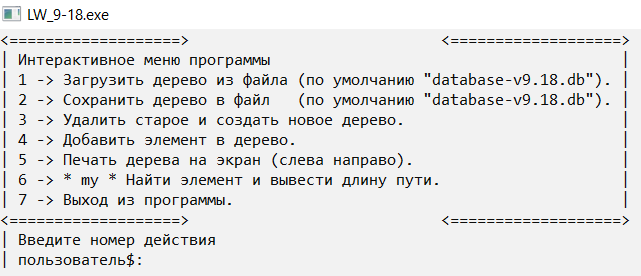


Рисунок 10.1 – Меню программы

На Рисунке 10.2 показана работоспособность процедуры, импортирующей данные из файла. Файл уже хранит какие-то данные.

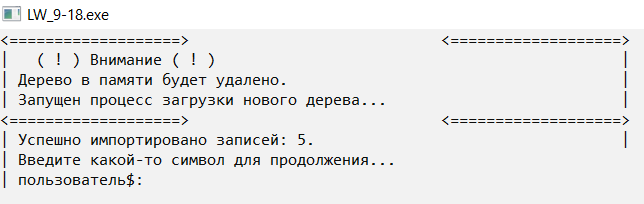


Рисунок 10.2 – Импорт данных в программу (данные в файле присутствуют)

На Рисунке 10.3 показана работоспособность процедуры, импортирующей данные из файла. Файл уже хранит какие-то данные.

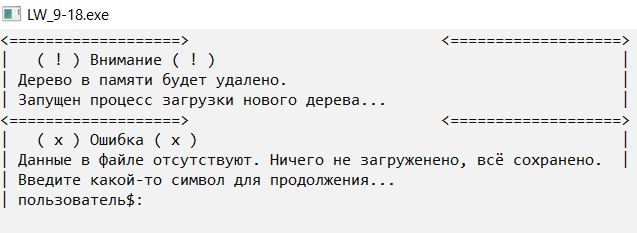


Рисунок 10.3 – Импорт данных в программу (данные в файле отсутствуют)

На Рисунке 10.4 показан процесс организации дерева на примере ввода одного элемента. Ввод будет осуществляться до тех пор, пока пользователь вместо ФИО не введёт звёздочку.

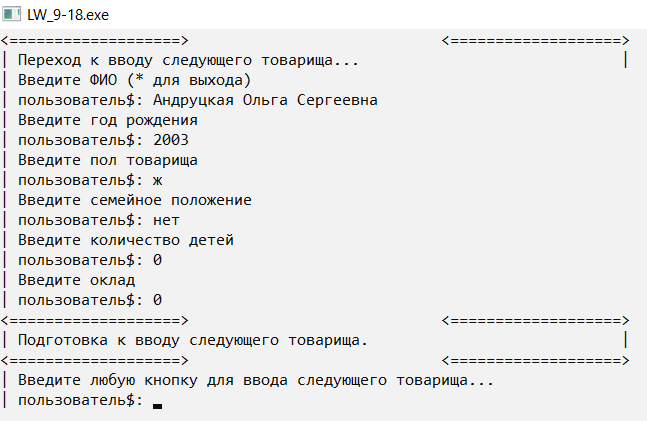


Рисунок 10.4 – Процесс организации дерева

На Рисунке 10.5 показан процесс добавления одного элемента в дерево.

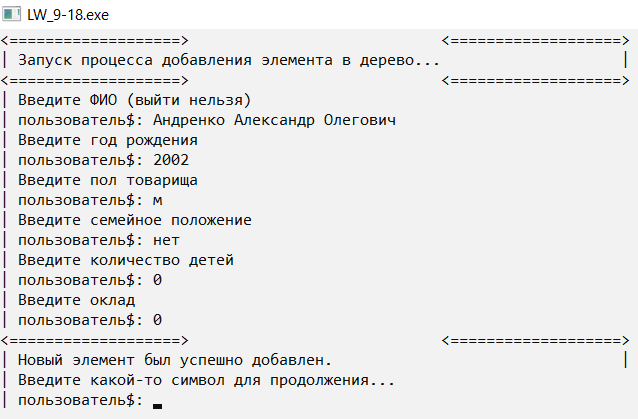


Рисунок 10.5 – Добавление элемента в дерево

На Рисунке 10.6 показано, как программа отображает дерево на экране.

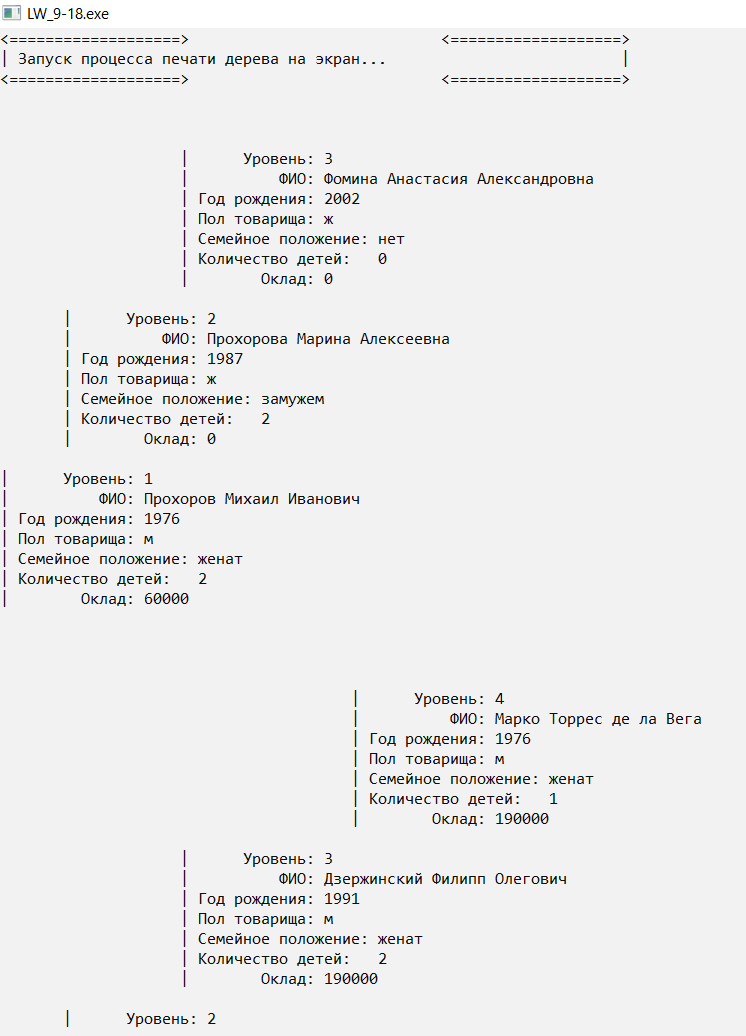


Рисунок 10.6 – Отображение дерева на экране

На Рисунках 10.7 и 10.8 продемонстрирована работоспособность процедуры поиска элемента в дереве и, если этот элемент существует, программа также выводит длину пути. В первом случае элемент был найден, и поэтому программа вывела длину пути. Во втором случае найти введённый элемент не удалось, поэтому программа вывела соответствующую ошибку.

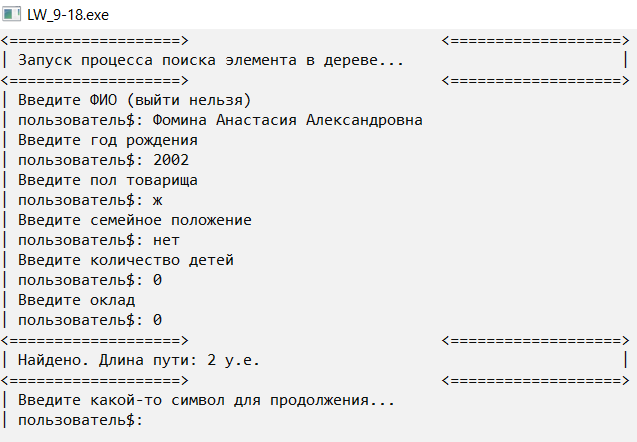


Рисунок 10.7 – Поиск элемента в дереве (элемент существует)

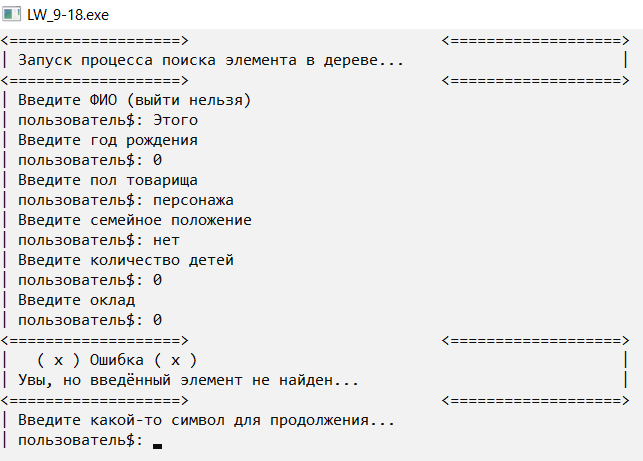


Рисунок 10.8 – Поиск элемента в дереве (элемент не существует)

На Рисунке 10.9 показан процесс сохранения данных в файл. Данные, содержащиеся файле, автоматически удаляются.

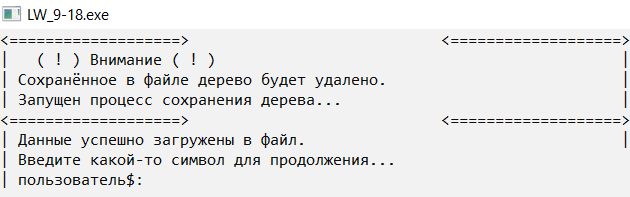


Рисунок 10.9 – Сохранение данных в файл

Дополнительная функция программы продемонстрирована на Рисунке 10.10. Она заключается в следующем: если после совершения каких-либо действий над деревом пользователь забыл сохранить данные, программа предупредит об этом и предложит сохраниться.

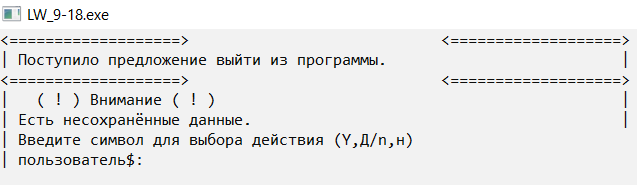


Рисунок 10.10 – Дополнительная функция

## Вывод

При выполнении данной лабораторной работы были получены навыки разработки приложений, работающих с объектами (классами) 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, более эффективно использовать основные принципы ООП.