СОДЕРЖАНИЕ

ВВЕДЕНИЕ

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++ для различных платформ.

Имя языка C++ происходит от оператора унарного постфиксного инкремента C ++ (увеличение значения переменной на единицу). Имя C+ не было использовано потому, что является синтаксической ошибкой в C и, кроме того, это имя было занято другим языком. Язык также не был назван D, поскольку «является расширением C и не пытается устранять проблемы путём удаления элементов C».

Язык C++ непосредственно и всесторонне поддерживает множество стилей программирования, в том числе процедурное программирование, абстракцию данных, объектно-ориентированное программирование и обобщённое программирование. Также он избегает особенностей, которые зависят от платформы или не являются универсальными. C++ не требует слишком усложнённой среды программирования.

Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности.

C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

1 ПОСТАНОВКА ЗАДАЧИ

Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. Информация должна храниться в различных файлах, связанных определенным образом. Реализовать функции: добавление, редактирование, удаление данных, вывод на экран.

Разработать иерархию классов с использованием наследования. Производить обработку исключительных ситуаций. Использовать инкапсуляцию, перегрузку методов, переопределение методов, абстрактные типы данных (интерфейсы, абстрактные классы), передачу параметров по ссылке и по значению, статические методы.

Для реализации игры используется объектно-ориентированный язык программирования C++, среда разработки Microsoft Visual Studio 2017. Ограничения на использование других операционных систем нет. Приложение написано на ОС Windows 10.

2 ОБЗОР МЕТОДОВ

Существует много методов реализации программы. Данную программу можно реализовать на языках программирования как: Java, C#, Python,. У каждого языка программирования есть свои особенности.

Java строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle.

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины.

C# программирования общего назначения, разработанный в 1969 — 1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би. Первоначально был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств — от суперкомпьютеров до встраиваемых систем. Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C.

Python программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой набор полезных функций.

Python поддерживает структурное, обобщенное, объектно-ориентированное, функциональное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

3 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ

В программе используется 6 текстовых файла:

-Файл clients.txt. Хранит данные о клиентов (логин, пароль, ид клиента, Ф.И.О, номер пасспорта, e-mail, мобильный номер).

string	string	string	string	string	string	string
Логин	Пароль	ID	ФИО	Номер паспорта	e-mail	Мобильный номер
User	12345	AA11	Ysmayyl Atayew	A12345	ismayyl@mail.com	252341

-Файл landtours.txt. Хранит данные о сухопутных турах (наименования тура, ID, вид, дата, длительность, цена, страна, город, вид транспорта).

string	string	string	string	int	float	string	string	string
Наимено-вания тура	ID	Вид	Дата	Длитель-ность	цена	страна	город	вид транс-порта
Aura	F111	Excursion	11.11.20	14	1000	Germany	Berlin	Bus

-Файл orders.txt. Хранит данные о заказах (ID клиента, ID тура, длительность, дата).

string	string	int	string
ID клиента	ID тура	длительность	дата
F111	B222	15	11.11.20

-Файл seatours.txt. Хранит данные о морских турах (наименования тура, ID, вид, дата, длительность, цена, страна, город, имя корабля, названия море).

string	string	string	string	int	float	string	string	string	string
наименования тура	ID	вид	дата	длительность	цена	страна	город	имя корабля	названия море
Aura	F1	Excursion	11.11.11	14	1000	India	Bombey	PacificEye	Atlantic

-Файл localtours.txt. Хранит данные о местных турах (наименования тура, ID, вид, дата, длительность, цена, город).

string	string	string	string	int	float	string
Наимено-вания тура	ID	Вид	Дата	Длитель-ность	цена	город
Aura	F111	Excursion	11.11.20	14	1000	Berlin

- Файл tickets.txt. Хранит данные о билетах (пункт прибытие, пункт отправление, дата отправление, ID билета, вид транспорта, ID клиента).

string	string	string	string	string	String
пункт прибытие	пункт отправление	дата отправление	ID билета	вид транспорта	ID клиента
India	China	11.11.20	F111	Boat	A111

Также в ходе работы программы для хранения промежуточных данных используются контейнер Очередь, написанный вручную.

При изменении данных в программе, происходит изменение в файле после закрытия программы.

4 ОПИСАНИЕ АЛГОРИТМОВ

4 Схема Алгоритмов

4.1.1 Алгоритм функции enqueue(T data).

Позволяет добавить элемент в очередь. Представлен в Приложении А.

4.1.2 Алгоритм функции T dequeue().

Позволяет удалять. Представлен в Приложении А.

4.2.1 Функция enqueuer(T data) позволяет добавить данные в очередь.

Шаг 1. Начало.

Шаг 2. Входные данные data новый объект, который хотим добавить в очередь Head - указатель на первый элемент в очереди. Tail – указатель на последний элемент в очереди. Size – текущий размер очереди

Шаг 3. Если указатель head равен nullptr переходим к шагу 9.

Шаг 4. Информационной части temp присваиваем значение data.

Шаг 5. Указатель предыдущего элемента temp->prev присваиваем nullptr.

Шаг 6. На указатель следующего элемента temp->next присваиваем указатель head
на начало очереди.

Шаг 7. На указатель предыдущего элемента head->prev присваиваем указатель temp.

Шаг 8. Увеличиваем текущий размер очереди на единицу и переходим к шагу 13.

Шаг 9. Информационной части первого элемента очереди присваиваем значение data.

Шаг 10. На указатель следующего элемента head присваиваем значение nullptr.

Шаг 11. На указатель предыдущего элемента head присваиваем значение nullptr.

Шаг 12. Увеличиваем текущий размер очереди на единицу и переходим.

Шаг 13. Конец.

4.2.2 Функция T enterInt (T min, T max) позволяет вводить числовое значение.

Шаг 1. Начало.

Шаг 2. Объявляем переменной number типа T.

Шаг 3. Объявляем переменной flag типа bool.

Шаг 5. Создаем цикл работающий пока значение flag не равно true.

Шаг 4. Присваиваем переменной flag значение true.

Шаг 5. Вводим информацию в number.

Шаг 6. Если ввели не число присваиваем в flag значение false и переходим к шагу 8.

Шаг 7. Если введенное значение меньше 0 или больше 999999, присваиваем в flag значение.

Шаг 8. Если flag не равно false, то очищаем буфер.

Шаг 9. Возвращаем переменной num.

Шаг 10. Конец.

5 ДИАГРАММА КЛАССОВ

Предоставляется в Приложении Б.

6 ОПИСАНИЕ КЛАССОВ

1. Базовый абстрактный класс Information

class Information // Базовый абстрактный класс Information

{

public:

virtual void Edit() = 0; - виртуальный метод для редактирование данных

virtual void PutData() = 0; - виртуальный метод для заполнение данных

virtual void Show() = 0; - виртуальный метод для вывода информации

};

2. Производный класс Client

```
class Client : public Information
{
private:
string clientCode;
string FIO;
string passportId;
string mail;
string mobileNumber;
string login;
string password;

public:
Client();
Client(string _l, string _p);
~Client();
Client(string _cCode, string _fio, string _passId, string _mail, string _mNumber, string _login,
string _pass);
Client(const Client& obj);

Client& operator=(const Client& obj);

friend fstream& operator<<(fstream& f, Client& obj); // перегруженный оператор вывода для
записи данных в файл
friend fstream& operator>>(fstream& f, Client& obj); // перегруженный оператор ввода для
чтение данных с файла
// Гетторы
string getClientCode();
string getFio();
string getPassportId();
string getMail();
```

```
string getMobileNumber();
// Сетторы
void setClientCode(string s);
void setFio(string s);
void setPasswordId(string s);
void setMail(string s);
void setMobile(string s);

string getLogin() { return login; }
string getPassword() { return password; }

// перегруженные методы
void Edit() override;
void PutData() override;
void Show() override;
};
```

3. Производный класс Ticket

```
#pragma once
#include "Header.h"
#include "Information.h"

class Ticket : public Information
{
private:
string ticketCode; //ID билета
string userCode; // ID клиента
string transportType; //вид транспорта
string departurePoint;
string arrivalPoint;
string deprtureData;
string arrivalData;
public:
Ticket();
~Ticket(); //деструктор
Ticket(string _tCode, string _uCode, string _tType, string _dPoint, string _aPoint,
string _dData);
Ticket(const Ticket& obj); //конструктор копирование

Ticket& operator=(const Ticket& obj); //перегруженный оператор
присваивания

friend fstream& operator<<(fstream& f, Ticket& obj); //перегруженный
оператор вывода для записи данных в файл
friend fstream& operator>>(fstream& f, Ticket& obj); //перегруженный
оператор ввода для чтение данных с файла
```

```
 // гетторы
 string getTicketCode();
 string getUserCode();
 string getTransportType();
 string getDeparturePoint();
 string getArrivalPoint();
 string getDepartureData();
  // сетторы
 void setTicketCode(string _s);
 void setUserCode(string _s);
 void setTransportType(string _s);
 void setDeparturePoint(string _s);
 void setArrivalPoint(string _s);
 void setDepartureData(string _s);

 void Edit() override;
 void PutData() override;
 void Show() override;

 static void header();
 static void headerLine();
 };
```

4. Производный класс Order

```
class Order : public Information
{
 private:
string clientCode;
string tourCode;
 public:
Order(); - конструктор
~Order();- деструктор
Order(const Order& obj); - конструктор копирование

Order& operator=(const Order& obj); - перегруженный оператор присваивания

friend fstream& operator<<(fstream& f, Order& obj); - перегруженный оператор вывода для записи данных в файл

friend fstream& operator>>(fstream& f, Order& obj); - перегруженный оператор ввода для чтение данных с файла

string getClientCode();-метод возвращающий ID клиента
void setClientCode(string clientCode); - метод присваивания ID клиента
```

string getTourCode();-метод возвращающий ID тура
void setTourCode(string _t); - метод присваивания ID тура

void Edit() override;- перегруженный метод
void PutData() override;- перегруженный метод
void Show() override; -перегруженный метод

static void header(); - метод загаловок
static void headerLine(); -метод загаловок
};

5. Производный класс Tour

class Tour : public Information

{

protected:

string tourName; - наименования тура

string tourCode;- ID тура

string tourType; - вид тура

string tourDate; - дата тура

int duration; - длительность

float price; - цена

public:

Tour(); -конструктор

~Tour(); - деструктор

Tour(const Tour& obj); - конструктор копирования

string getTourCode();-метод для получение код тура

string getTourName();-метод для получение названия тура

string getTourType();-метод для получение названия тип тура

};

6. Производный класс InternationalTour

class InternationalTour :public Tour

{

protected:

string country; - страна

string city; - город

public:

InternationalTour() - конструктор

~InternationalTour() - деструктор

string getCountry(); - метод для получение название страны

string getCity(); - метод для получение названия города

void setCountry(string s); - метод для присваивания названия страны

void setCity(string s); - метод для присваивания названия города

};

7. Производный класс LocalTour

class LocalTour :public Tour

{

private:

string city; - город

public:

LocalTour();- конструктор;

~LocalTour();- деструктор

LocalTour(const LocalTour& obj); - конструктор копирования

LocalTour& operator=(const LocalTour& obj); - перегруженный оператор присваивания

friend fstream& operator>>(fstream& f, LocalTour& obj); - перегруженный оператор вывода для записи данных в файл

friend fstream& operator<<(fstream& f, LocalTour& obj); - перегруженный оператор ввода для чтение данных с файла

void Edit() override; - перегруженный метод

void PutData() override; - перегруженный метод

void Show() override; - перегруженный метод

static void header(); - метод выводящий загаловок
static void headerLine(); - метод выводящий загаловок конца

};

8. Производный класс LandInterNational

class LandInternational: public InternationalTour

{

private:

string carType; - вид транспорта

public:

LandInternational();- конструктор;

~LandInternational();- деструктор

LandInternational(const LandInternational& obj); - конструктор копирования

LandInternational& operator=(const LandInternational& obj); - перегруженный оператор присваивания

friend fstream& operator<< (fstream& f, LandInternational& obj); - перегруженный оператор вывода для записи данных в файл

friend fstream& operator>> (fstream& f, LandInternational& obj); - перегруженный оператор ввода для чтение данных с файла

string getCarType(); метод для получение вид транспорта

void setCarTpye(string _t); - метод для присваивания вид транспорта

void Edit() override; - перегруженный метод

void PutData() override; - перегруженный метод

void Show() override; - перегруженный метод

static void header(); - метод выводящий загаловок

static void headerLine(); - метод выводящий загаловок конца

};

9. Производный класс SeaInternational

class SeaInternational :public InternationalTour

{

private:

string cargoName; - наименования корабля

string seaName; - названия моря

public:

SeaInternational(); - конструктор

~SeaInternational(); - деструктор

SeaInternational(const SeaInternational& obj); - конструктор копирования

SeaInternational& operator=(const SeaInternational& obj); - перегруженный оператор присваивания

friend fstream& operator<< (fstream& f, SeaInternational& obj); - перегруженный оператор вывода для записи данных в файл

friend fstream& operator>> (fstream& f, SeaInternational& obj); - перегруженный оператор ввода для чтение данных с файла

void setCargoName(string _c);

void setSeaName(string _s);

string getCargoName();- метод для получение названия корабля

string getSeaName(); - - метод для получение названия моря

void Edit() override; - перегруженный метод

void PutData() override; - перегруженный метод

void Show() override; - перегруженный метод

static void header(); - метод выводящий загаловок
static void headerLine(); - метод выводящий загаловок конца

};

10.Шаблонный класс File

template <class T>

class File{

private:

fstream filestream; - поток

char filename[30]; - названия файла

public:

File(); - конструктор

File(char* filename); - конструктор по умолчанию

~File(){ filestream.close(); }деструктор

void WriteData(T& obj); - метод записывающий данные в файл

void ReadData(T& obj); - метод читающий данные с файла

bool REndFile(); - метод для проверки конца файла

void OpenEnd(char* filename); - метод для открытие файла для записи в конец

void closeFile();

};

11.Шаблонный класс Interface

template <class T>

class Interface

{

Queue <T>* queue; - очередь

Queue<Order>* orders;
Queue<Ticket>* tickets;

char filename[30]; - названия файла

public:

Interface(char* f); - конструктор

~Interface(); - деструктор

void add(); - метод для добавление

void del(); - метод для удаление данных

void edit(); - метод для редактирования

void show(); - метод для вывода

void start();

void setFilename(char* filename); - метод для присваивания названия файла

};

12.Класс InterfaceClient

```
class InterfaceClient
{
private:
Client client;
Queue<Order>* orders;
Queue<Ticket>* tickets;
public:
InterfaceClient(Client obj);
~InterfaceClient();
void start();
template<class T>
void makeOrder(Queue<T> *q);
void loadTicket();
void loadOrder();
void writeOrder();
};
```

13.Шаблонный класс Queue

```
template<typename T>
struct Node
{
T data;
Node<T>* next = NULL;
Node<T>* prev = NULL;
};
```

```
template<typename T>

class Queue {
```

private:
int Size;
Node<T>* head; - голова

Node<T>* tail; - конец

public:

Queue() : head(NULL), tail(NULL), Size(0) - конструктор

~Queue(); - деструктор

Queue(const Queue<T>& obj); - конструктор копирования

void Delete(int index); - метод для удаления элемента

void show(); - метод для вывода данных

T& operator[](int index); - перегруженный метод индексаций

void enqueue(T data); - метод добавления объекта в очередь

bool is_Empty(); - метод на проверяющий пустоту

int getSize(); - метод возвращающий размер очереди

T dequeue(); - метод удаление последнего элемента в очереди

void Clear();}; - метод для очищения очереди

};

7 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫк

Рисунок 1 - Окно авторизации.
Рисунок 2 - Главное меню.
Рисунок 3 - Добавление данных.
Рисунок 4 - Отображение информации.
Рисунок 5 - Редактирования данных.
Рисунок 6 - Бронирования тура.
Рисунок 7 - Раздел «Мои заказы».

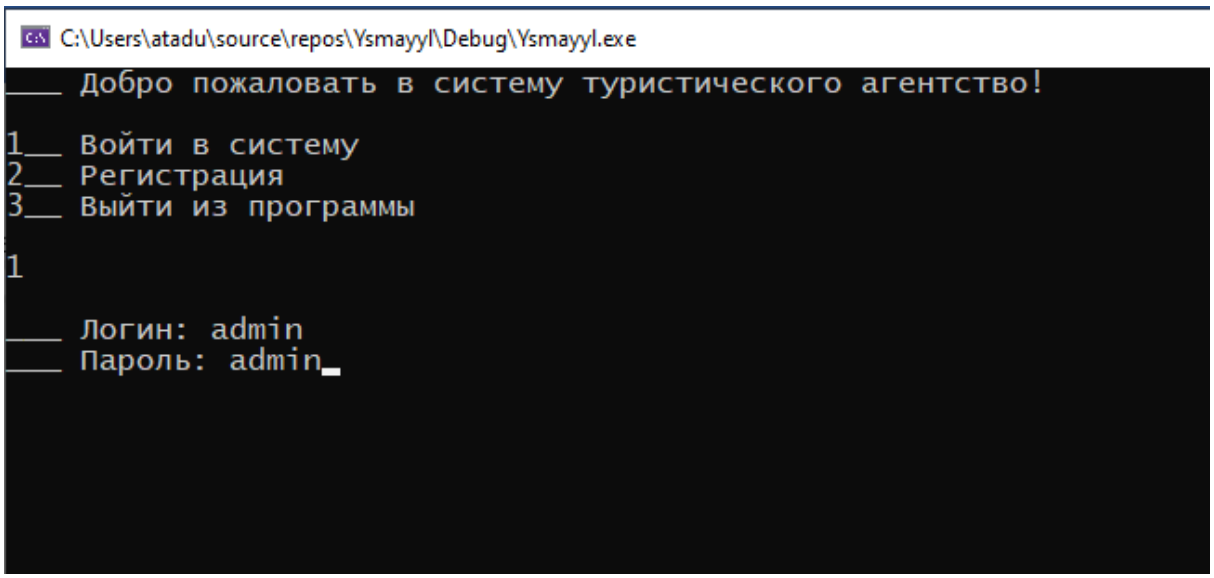

Рисунок 1 – Окно авторизации

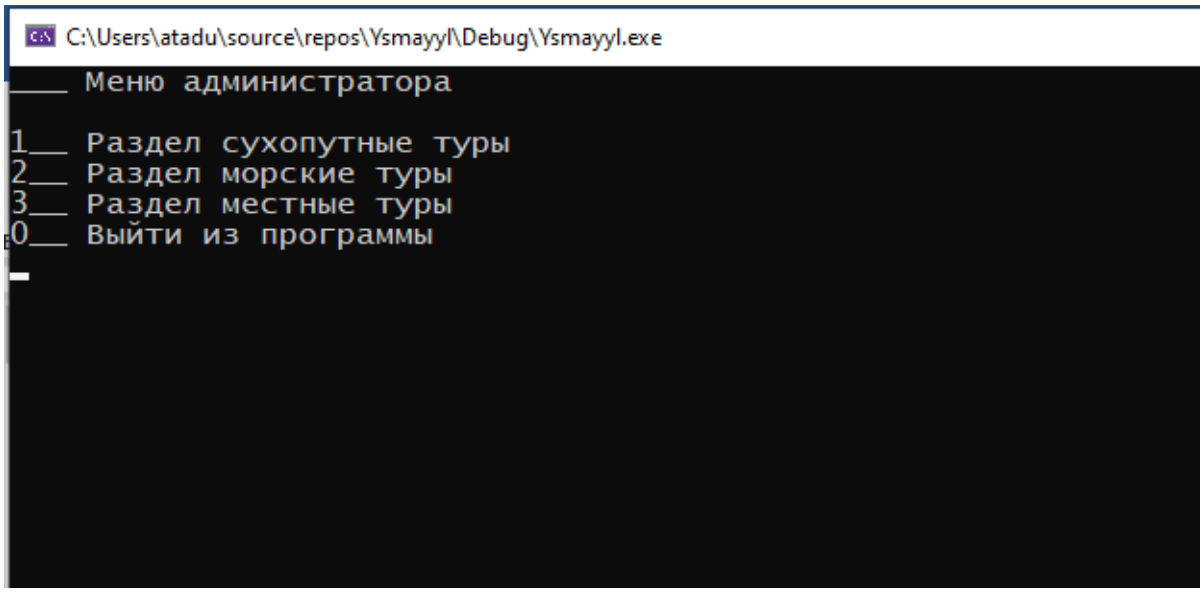

Рисунок 2 – Главное меню

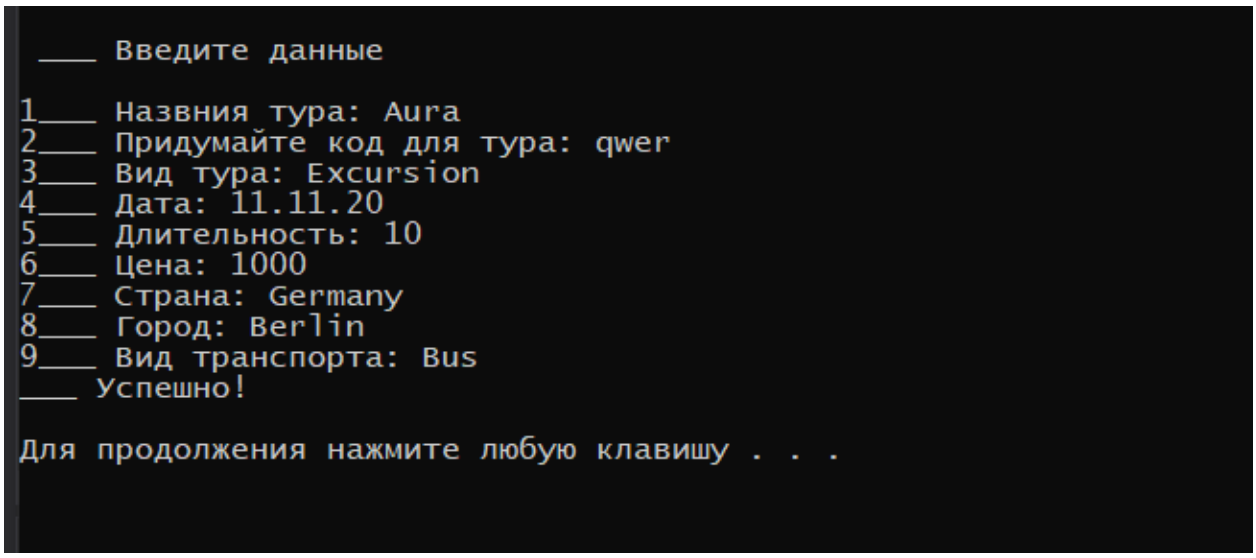

Рисунок 3 - Добавление данных

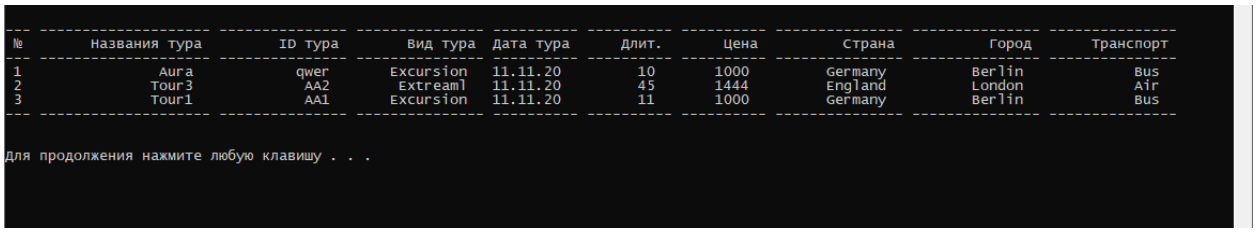

Рисунок 4 - Отображение информации

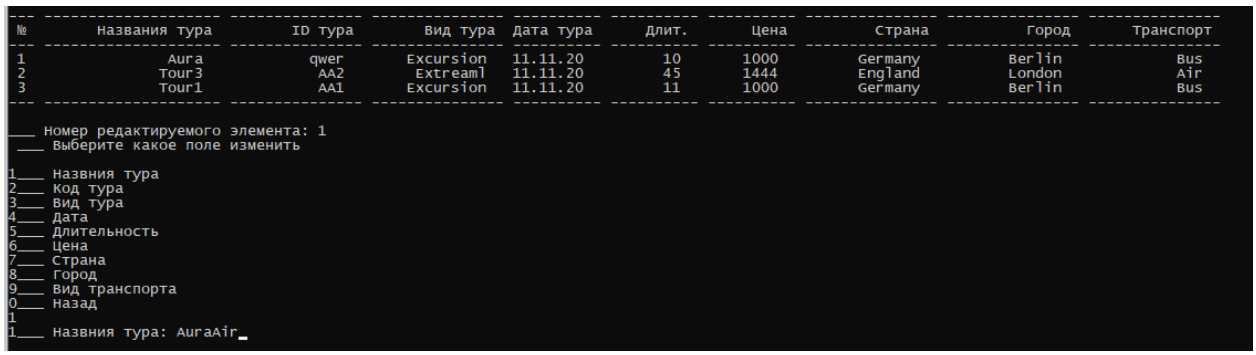

Рисунок 5 – Редактирования данных

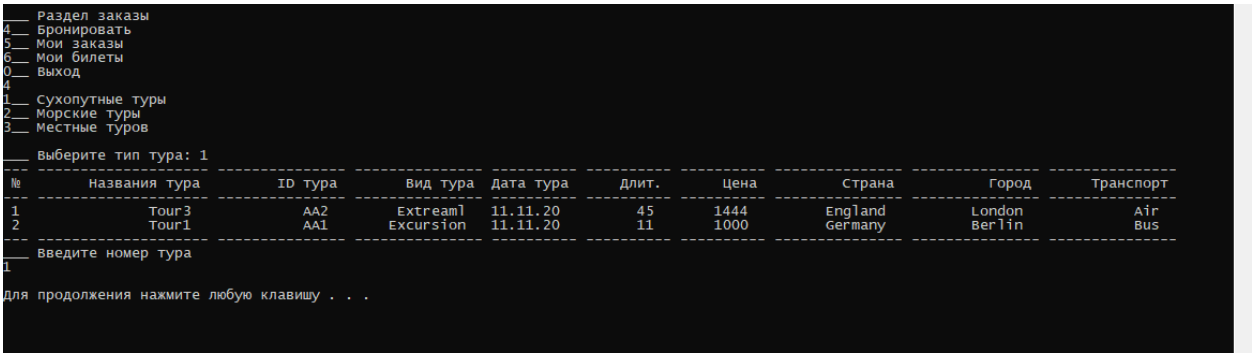

Рисунок 6 – Бронирования тура

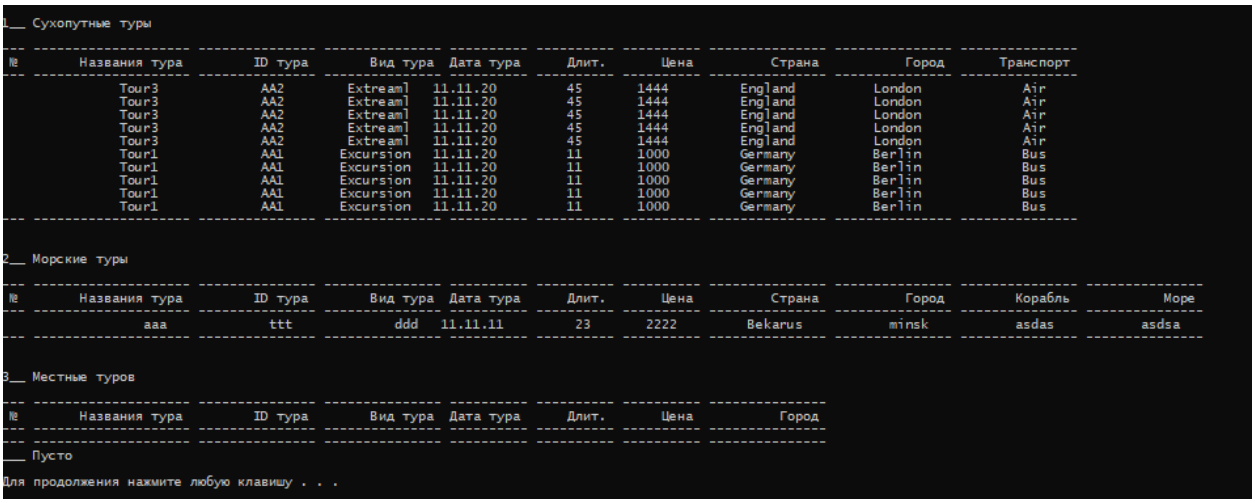

Рисунок 7 – Раздел «Мои заказы»

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано приложение автоматизации системы турагентство, позволяющее пользователю вводить различную информацию, выводить ее на экран, изменять, удалять, добавлять. Были закреплены знания в области ООП. Использовались среда разработки Visual Studio 2019 и операционная система Windows 10. К достоинствам программы можно отнести простой и понятный интерфейс, что в свою очередь обеспечивает удобство пользования для обычных пользователей. В дальнейшем планируется усовершенствование программы, а именно усовершенствование интерфейса и добавление новых функций, поддержка многопользовательности. После освоения в ходе учебного курса основ и принципов работы с базами данных планируется переход с системы хранения информации в текстовых файлах на хранение в базе данных, что предоставит возможность добавления информации нескольким пользователям одновременно и позволит увеличить скорость работы программы.

ПРИЛОЖЕНИЕ А

(схемы алгоритмов)

ПРИЛОЖЕНИЕ Б

(диаграмма классов)

ПРИЛОЖЕНИЕ В

(листинг кода)

ЛИТЕРАТУРА

1. Дейтел, Х.М. Как программировать на C++ / Х.М. Дейтел, П.Д. Дейтел; пер. с англ. – М. : Бином, 2007. – 1152 с.

2. Страуструп, Б. Язык программирования C++ / Б. Страуструп; специальное издание. Пер. с англ. – СПб. : BHV, 2008. – 1098 с.

3. Скляров, В.А. Язык C++ и объектно-ориентированное программирование: справ. пособие / В.А. Скляров. – Мн. : Вышэйшая школа, 1997. – 478 с.

4. Элджер, Дж. C++: библиотека программиста / Дж. Элджер. – СПб. : Питер, 2001. – с.

5. Шилд, Г. Программирование на Borland C++ для профессионалов / Г. Шилд. – Мн. : ООО «Попури», 1998. – 800 с.

6. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон; пер. с англ. – СПб. : ДМК, 2004. – 429 с.

7. Доманов, А.Т. Предварительный стандарт предприятия. Дипломные проекты(работы) : общие требования / А.Т. Доманов, Н.И. Сороко. – Мн.: БГУИР, 2009. – 171 с.