Федеральное агентство по образованию РФ

Государственное образовательное учреждение

высшего профессионального образования

“Тверской государственный университет”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 Факультет прикладной математики и кибернетики

Направление: 010300.62 Прикладная математика и информатика

**Отчет по учебно-вычислительной практике**

 Выполнила:

 студентка 14 группы

Киут Л.Л.

Научный руководитель:

Дадеркин Д. О.

Задание на учебно-вычислительную практику

Вариант 18

**Задача 1**.

Разработать программу для работы с БД, которая обеспечивает автоматизированную продажу авиабилетов на месяц вперед. Программа проверяет правильность каждой даты, делает возможным менять расписание.

В файле хранятся входные данные: номера рейсов, маршруты, количество свободных мест, дата вылета, ФИО.

При введении нового пункта расписания не разрешается вводить прошедшую дату (дата на данный момент времени высвечивается).

При покупке билета так же проверяется дата и номер рейса. Не разрешается покупать билеты более чем на месяц вперед.

Содержание

1. Словесное описание алгоритма……………………………………………………………....4

2.Описание классов: переменные и методы. Типы доступа. Объявление методов............. 4

3. Тестовый запуск……………………………………………………………………………….5

4. Код программы ………………………………………………………………….…………....6

5. Список использованной литературы……………………………………………………… 13

***1. Словесное описание алгоритма.***

Программа позволяет оперировать с базой данных авиабилетов. База данных читается из файла в вектор, после чего с ней можно производить различные операции, а именно: добавление нового расписания, проверка дат, изменение имеющегося расписания. Операции выполняются с помощью функций и методов класса MyData.

***2. Описание классов: переменные и методы. Типы доступа. Объявление методов***

struct Date

{

 int Day; //день

 int Month; //месяц

 int Year; //год

 void Read();//ввод даты с консоли

 Date Get\_Date();//вывод текущей даты

 void Print();//вывод даты на консоль

};

//структура билет

struct Ticket

{

 int NumReys; //номер рейса

 string Route; //маршрут

 DateFlight Date; //дата отправления

};

//стурктура рейс

struct Flight

{

 int NumReys; // номер рейса

 int EmptySeats; //количество свободных мест

 string Route; //маршрут

 Date DateRoute; //дата рейса

};

\\ Класс MyData

class MyData

{

public:

 vector<Ticket> Tickets; //массив билетов

 vector<Flight> Flights; //массив рейсов

 MyData(); //конструктор

 vector<int> ArrFlights(Date today); //поис рейсов по дате

 vector<int> FindRoute(string Route, vector<int> Arr); //поиск рейсов по маршруту

 vector<int> FindRoute\_Date(DateFlight D, vector<int> Arr); //поиск рейсов по дате и маршруту

 void PrintFlights(); //вывод всего на консоль

 void WriteDataInFile(Ticket T); //запись в файл

};

***3. Тестовый запуск***





***4. Код программы***

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

#include <stdio.h>

#include <fstream>

using namespace std;

struct Date

{

 int Day;

 int Month;

 int Year;

 //ввод даты с консоли

 void Read()

 {

 Date D = Get\_Date();

 while(1)

 {

y: cout<<"Год: ";

 cin>>Year;

 if(Year < D.Year )

 goto y;

 else

 break;

 }

 while(1)

 {

m: cout<<"Месяц: ";

 cin>>Month;

 if(Month > 12 || Month < 1 )

 goto m;

 else

 if(Year == D.Year && Month < D.Month )

 goto m;

 else

 break;

 }

 while(1)

 {

d: cout<<"День: ";

 cin>>Day;

 if(Day > 31 && Day < 1 )

 goto d;

 else

 if(Year == D.Year && Month == D.Month && Day < D.Day )

 goto d;

 else

 break;

 }

 }

 Date Get\_Date()

 {

 Date D;

 SYSTEMTIME systime;

 GetLocalTime(&systime);

 D.Day = systime.wDay;

 D.Month = systime.wMonth;

 D.Year = systime.wYear;

 return D;

 }

 void Print()

 {

 cout<<Day<<"."<<Month<<"."<<Year<<endl;

 }

};

struct DateFlight

{

 int Day;

 int Month;

 int Year;

 string FIO;

 void Read()

 {

 Date D = Get\_Date();

 while(1)

 {

d: cout<<"День: ";

 cin>>Day;

 if(Day > 31 && Day < 1 )

 goto d;

 else

 break;

 }

 while(1)

 {

m: cout<<"Месяц: ";

 cin>>Month;

 if(Month > 12 || Month < 1 )

 goto m;

 else

 break;

 }

 while(1)

 {

y: cout<<"Год: ";

 cin>>Year;

 if(Year < D.Year)

 goto y;

 else

 if(Year == D.Year && Month < D.Month)

 goto y;

 else

 if(Year == D.Year && Month == D.Month && Day < D.Day)

 goto y;

 else

 if(Year == D.Year && D.Month != 12 && Month >= D.Month +1 && Day > D.Day + 1)

 {

 cout<<"Не разрешается покупать билеты более, чем на месяц вперед"<<endl;

 goto d;

 }

 else

 if(Year == D.Year+1 && D.Month == 12 && Month == 1 && Day > D.Day + 1)

 {

 cout<<"Не разрешается покупать билеты более, чем на месяц вперед"<<endl;

 goto d;

 }

 else

 break;

 }

 }

 Date Get\_Date()

 {

 Date D;

 SYSTEMTIME systime;

 GetLocalTime(&systime);

 D.Day = systime.wDay;

 D.Month = systime.wMonth;

 D.Year = systime.wYear;

 return D;

 }

};

int dig(string s) //перевод числа из char в int

{

 int k=0;

 int d=1;

 for(int i=s.size()-1; i>=0; i--)

 {

 if(!isdigit(s[i]))

 {

 return -1;

 }

 k+= (s[i] - '0') \*d;

 d\*=10;

 }

 return k;

}

struct Ticket

{

 int NumReys;

 string Route;

 DateFlight Date;

};

struct Flight

{

 int NumReys;

 int EmptySeats;

 string Route;

 Date DateRoute;

};

class MyData

{

public:

 vector<Ticket> Tickets;

 vector<Flight> Flights;

 MyData()

 {

 ifstream f("flights.txt");

 if(f==NULL)

 {

 cout<<"Ошибка открытия файла(\"flights.txt\")\n";

 exit(1);

 }

 string str;

 while(f)

 {

 getline(f,str);

 if(str.size() > 0)

 {

 string num;

 int In=0;

 for(; str[In]!= ','; In++)

 num+=str[In];

 Flight Reys;

 Reys.NumReys=dig(num);

 if(Reys.NumReys==-1)

 {

 cout<<"Ошибка.Некорректный номер рейса в файле flights.txt"<<endl;

 f.close();

 exit(1);

 }

 In++;

 for(; str[In]!= ','; In++)

 Reys.Route+=str[In];

 In++;

 num.clear();

 for(; str[In]!= '.'; In++)

 num+=str[In];

 Reys.DateRoute.Day = dig(num);

 In++;

 num.clear();

 for(; str[In]!= '.'; In++)

 num+=str[In];

 Reys.DateRoute.Month = dig(num);

 In++;

 num.clear();

 for(; str[In]!= ','; In++)

 num+=str[In];

 Reys.DateRoute.Year = dig(num);

 In++;

 num.clear();

 for(; In<str.size(); In++)

 num+=str[In];

 Reys.EmptySeats=dig(num);

 if(Reys.EmptySeats==-1)

 {

 cout<<"Ошибка.Некорректное количество свободных мест в файле flights.txt"<<endl;

 f.close();

 exit(1);

 }

 Flights.push\_back(Reys);

 str.clear();

 }

 }

 f.close();

 ifstream g("tickets.txt");

 if(g==NULL)

 {

 cout<<"Ошибка открытия файла(\"tickets.txt\")\n";

 exit(1);

 }

 while(g)

 {

 str.clear();

 getline(g,str); //считывает построчно в str

 if(str.size() > 0)

 {

 string num;

 int In=0;

 for(; str[In]!= ','; In++)

 num+=str[In];

 Ticket bilet;

 bilet.NumReys=dig(num);

 if(bilet.NumReys==-1)

 {

 cout<<"Ошибка.Некорректный номер рейса в файле tickets.txt"<<endl;

 g.close();

 exit(1);

 }

 In++;

 for(; str[In]!= ','; In++)

 bilet.Date.FIO+=str[In];

 In++;

 for(; str[In]!= ','; In++)

 bilet.Route+=str[In];

 In++;

 num.clear();

 for(; str[In]!= '.'; In++)

 num+=str[In];

 bilet.Date.Day = dig(num);

 In++;

 num.clear();

 for(; str[In]!= '.'; In++)

 num+=str[In];

 bilet.Date.Month = dig(num);

 In++;

 num.clear();

 for(; In<str.size(); In++)

 num+=str[In];

 bilet.Date.Year = dig(num);

 Tickets.push\_back(bilet);

 }

 }

 g.close();

 }

 ~MyData()

 {

 Tickets.clear();

 Flights.clear();

 }

 vector<int> ArrFlights(Date today)

 {

 vector<int> Arr;

 int num=1;

 for(int i=0; i<Flights.size(); i++)

 {

 if(Flights[i].DateRoute.Year == today.Year && Flights[i].DateRoute.Month == today.Month && Flights[i].DateRoute.Day >= today.Day && Flights[i].EmptySeats > 0 )

 {

 cout<<num<<") ";

 num++;

 Arr.push\_back(i);

 cout<<Flights[i].NumReys<<","<<Flights[i].Route<<","<<Flights[i].DateRoute.Day<<"."<<Flights[i].DateRoute.Month<<"."<<Flights[i].DateRoute.Year<<","<<Flights[i].EmptySeats<<endl;

 }

 else

 if(Flights[i].DateRoute.Year == today.Year && today.Month !=12 && Flights[i].DateRoute.Month == today.Month +1 && Flights[i].DateRoute.Day <= today.Day && Flights[i].EmptySeats > 0 )

 {

 cout<<num<<") ";

 num++;

 Arr.push\_back(i);

 cout<<Flights[i].NumReys<<","<<Flights[i].Route<<","<<Flights[i].DateRoute.Day<<"."<<Flights[i].DateRoute.Month<<"."<<Flights[i].DateRoute.Year<<","<<Flights[i].EmptySeats<<endl;

 }

 else

 if(Flights[i].DateRoute.Year == today.Year+1 && Flights[i].DateRoute.Month == 1 && today.Month == 12 && Flights[i].DateRoute.Day <= today.Day && Flights[i].EmptySeats > 0 )

 {

 cout<<num<<") ";

 num++;

 Arr.push\_back(i);

 cout<<Flights[i].NumReys<<","<<Flights[i].Route<<","<<Flights[i].DateRoute.Day<<"."<<Flights[i].DateRoute.Month<<"."<<Flights[i].DateRoute.Year<<","<<Flights[i].EmptySeats<<endl;

 }

 }

 return Arr;

 }

 vector<int> FindRoute(string Route, vector<int> Arr)

 {

 vector<int> ArrRoute;

 int num=1;

 for(int i=0; i<Arr.size(); i++)

 {

 if( Flights[Arr[i]].Route == Route )

 {

 cout<<num<<") ";

 num++;

 cout<<Flights[Arr[i]].NumReys<<","<<Flights[Arr[i]].Route<<","<<Flights[Arr[i]].DateRoute.Day<<"."<<Flights[Arr[i]].DateRoute.Month<<"."<<Flights[Arr[i]].DateRoute.Year<<","<<Flights[Arr[i]].EmptySeats<<endl;

 ArrRoute.push\_back(Arr[i]);

 }

 }

 return ArrRoute;

 }

 vector<int> FindRoute\_Date(DateFlight D, vector<int> Arr)

 {

 vector<int> ArrRoute\_Date;

 int num=1;

 for(int i=0; i<Arr.size(); i++)

 {

 if( Flights[Arr[i]].DateRoute.Day == D.Day && Flights[Arr[i]].DateRoute.Month == D.Month && Flights[Arr[i]].DateRoute.Year == D.Year )

 {

 cout<<num<<") ";

 num++;

 cout<<Flights[Arr[i]].NumReys<<","<<Flights[Arr[i]].Route<<","<<Flights[Arr[i]].DateRoute.Day<<"."<<Flights[Arr[i]].DateRoute.Month<<"."<<Flights[Arr[i]].DateRoute.Year<<","<<Flights[Arr[i]].EmptySeats<<endl;

 ArrRoute\_Date.push\_back(Arr[i]);

 }

 }

 return ArrRoute\_Date;

 }

 void PrintFlights()

 {

 for(int i=0; i<Flights.size(); i++)

 {

 cout<<Flights[i].NumReys<<","<<Flights[i].Route<<","<<Flights[i].DateRoute.Day<<"."<<Flights[i].DateRoute.Month<<"."<<Flights[i].DateRoute.Year<<","<<Flights[i].EmptySeats<<endl;

 }

 }

 void PrintTickets()

 {

 for(int i=0; i<Tickets.size(); i++)

 {

 cout<<Tickets[i].NumReys<<","<<Tickets[i].Date.FIO<<","<<Tickets[i].Route<<","<<Tickets[i].Date.Day<<"."<<Tickets[i].Date.Month<<"."<<Tickets[i].Date.Year<<endl;

 }

 }

 void WriteDataInFile(Ticket T)

{

 ofstream f("tickets.txt",ios::app);

 if(f==NULL)

 {

 cout<<"Ошибка открытия файла для записи данных о покупки билета(\"tickets.txt\")\n";

 exit(1);

 }

 char NumReys[10], Day[10], Month[10],Year[10];

 string S;

 S+="\n";

 itoa(T.NumReys,NumReys,10);

 S+=NumReys;

 S+=",";

 S+=T.Date.FIO;

 S+=",";

 S+=T.Route;

 S+=",";

 itoa(T.Date.Day,Day,10);

 S+=Day;

 S+=".";

 itoa(T.Date.Month,Month,10);

 S+=Month;

 S+=".";

 itoa(T.Date.Year,Year,10);

 S+=Year;

 f<<S;

 f.close();

 ofstream g("flights.txt");

 if(g==NULL)

 {

 cout<<"Ошибка открытия файла для записи данных о покупки билета(\"flights.txt\")\n";

 exit(1);

 }

 for(int i=0; i<Flights.size(); i++)

 {

 S.clear();

 char NumReys1[10], Day1[10], Month1[10],Year1[10], EmptySeats1[10];

 itoa(Flights[i].NumReys,NumReys1,10);

 S+=NumReys1;

 S+=",";

 S+=Flights[i].Route;

 S+=",";

 itoa(Flights[i].DateRoute.Day,Day1,10);

 S+=Day1;

 S+=".";

 itoa(Flights[i].DateRoute.Month,Month1,10);

 S+=Month1;

 S+=".";

 itoa(Flights[i].DateRoute.Year,Year1,10);

 S+=Year1;

 S+=",";

 if(T.NumReys == Flights[i].NumReys)

 itoa(Flights[i].EmptySeats - 1,EmptySeats1,10);

 else

 itoa(Flights[i].EmptySeats,EmptySeats1,10);

 S+=EmptySeats1;

 if(i<Flights.size()-1)

 S+="\n";

 g<<S;

 }

 g.close();

}

};

void WriteReysInFile(Flight F)

{

 ofstream f("flights.txt",ios::app);

 if(f==NULL)

 {

 cout<<"Ошибка открытия файла для записи нового расписания(\"flights.txt\")\n";

 exit(1);

 }

 char NumReys[10], Day[10], Month[10],Year[10], EmptySeats[10];

 string S;

 S+="\n";

 itoa(F.NumReys,NumReys,10);

 S+=NumReys;

 S+=",";

 S+=F.Route;

 S+=",";

 itoa(F.DateRoute.Day,Day,10);

 S+=Day;

 S+=".";

 itoa(F.DateRoute.Month,Month,10);

 S+=Month;

 S+=".";

 itoa(F.DateRoute.Year,Year,10);

 S+=Year;

 S+=",";

 itoa(F.EmptySeats,EmptySeats,10);

 S+=EmptySeats;

 f<<S;

 f.close();

}

int main()

{

 setlocale( LC\_ALL,"Russian" );

 Date today = today.Get\_Date();

 cout<<"Сегодня: ";

 today.Print();

 MyData D;

 cout<<"Расписание маршрутов:"<<endl;

 D.PrintFlights();

 cout<<"Выберите режим работы с программой:\n";

 cout<<"1. Изменить расписание.\n";

 cout<<"2. Продать билет.\n";

 int punkt;

 cin>>punkt;

 switch(punkt)

 {

 case 1:{

 Flight Reys;

 cout<<"Введите номер рейса:\n";

 cin>>Reys.NumReys;

 cout<<"Введите маршрут(City-City):\n";

 cin>>Reys.Route;

 cout<<"Введите дату нового маршрута:\n";

 Reys.DateRoute.Read();

 cout<<"Введите общее количество мест:\n";

 cin>>Reys.EmptySeats;

 WriteReysInFile(Reys);

 D.Flights.push\_back(Reys);

 cout<<"Новое расписание успешно добавлено:\n";

 D.PrintFlights();

 break;

 }

 case 2:{

 Ticket T;

 cout<<"Предлагаем вам все имеющиеся рейсы, которые будут в течение этого месяца:\n";

 vector<int> NumArray = D.ArrFlights(today);

 if(NumArray.size() == 0)

 {

 cout<<"К сожалению, на данный момент не имеются рейсы, которые будут в течение месяца."<<endl;

 return 0;

 }

route: cout<<"Введите маршрут(City-City)"<<endl;

 cin>>T.Route;

 vector<int> RouteArray = D.FindRoute(T.Route,NumArray);

 cout<<"Вам показаны все рейсы с введенным маршрутом:"<<endl;

 if(RouteArray.size() == 0)

 {

 cout<<"К сожалению, на данный момент не имеются рейсы с указанным маршрутом."<<endl;

 cout<<"Хотите выбрать другой маршрут?"<<endl;

 cout<<"1. Да"<<endl;

 cout<<"2. Нет"<<endl;

 cin>>punkt;

 if(punkt == 1)

 {

 goto route;

 }

 else

 {

 return 0;

 }

 }

mdate: cout<<"Введите дату поездки"<<endl;

 T.Date.Read();

 NumArray.clear();

 NumArray = D.FindRoute\_Date(T.Date, RouteArray);

 if(NumArray.size() == 0)

 {

 cout<<"К сожалению, на данный момент не имеются рейсы с указанной датой."<<endl;

 cout<<"Хотите выбрать другую дату?"<<endl;

 cout<<"1. Да"<<endl;

 cout<<"2. Нет"<<endl;

 cin>>punkt;

 if(punkt == 1)

 {

 goto mdate;

 }

 else

 {

 return 0;

 }

 }

 cout<<"Выберите номер рейса"<<endl;

 cin>>T.NumReys;

 string str;

 cout<<"Введите ваше ФИО:\n";

 cout<<"Фамилия: ";

 cin>>str;

 T.Date.FIO+=str;

 T.Date.FIO+=" ";

 cout<<"Имя: ";

 cin>>str;

 T.Date.FIO+=str;

 T.Date.FIO+=" ";

 cout<<"Отчество: ";

 cin>>str;

 T.Date.FIO+=str;

metka: cout<<"Оплатить билет?"<<endl;

 cout<<"1. Да\n";

 cout<<"2. Нет\n";

 cin>>punkt;

 if(punkt == 1)

 {

 D.WriteDataInFile(T);

 cout<<"Билет успешно куплен\n";

 return 0;

 }

 else

 {

 cout<<"Вы точно уверены, что хотите все отменить?"<<endl;

 cout<<"1. Да\n";

 cout<<"2. Нет\n";

 cin>>punkt;

 if(punkt == 2)

 {

 goto metka;

 }

 }

 cout<<"Вы вышли из системы!\n";

 break;

 }

 }

 system("pause");

 return 0;

}

Список использованной литературы

1. Камаев В. А. Технологии программирования : [учебник для вузов по направлению "Информатика и вычислительная техника"] / В. А. Камаев, В. В. Костерин. - М., 2006. - 453, [1] с. : ил., табл. - Рекомендовано МО.
2. Крылов Е. В. Техника разработки программ. В 2 кн.. Кн. 1 : [учебник для вузов по направлениям "Информатика и вычислительная техника" и "Техника и технологии"] / Е. В. Крылов, В. А. Острейковский, Н. Г. Типикин. - М., 2007. - 374, [1] с. : ил. - Рекомендовано УМО.
3. Хорев П. Б. Объектно-ориентированное программирование : [учебное пособие по направлению "Информатика и вычислительная техника"] / П. Б. Хорев. - М., 2011. - 446, [1] с. : ил. - Рекомендовано УМО.
4. Шеферд Д. Программирование на Microsoft Visual C++ .NET : мастер-класс [пер. с англ.] / Джордж Шеферд по материалам Дэвида Круглински. - М., 2007. - 892 с. : ил. + 1 CD-ROM.
5. Базы данных: модели, разработка, реализация/ Т.С. Карпова. - СПб.: Питер, 2001г.