**Лабораторная работа №4**

**Контейнеры STL**

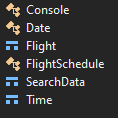
**Цель работы**

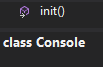
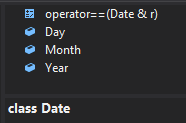
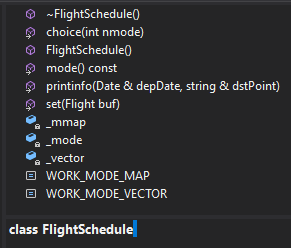
Ознакомление с классами-контейнерами библиотеки STL.

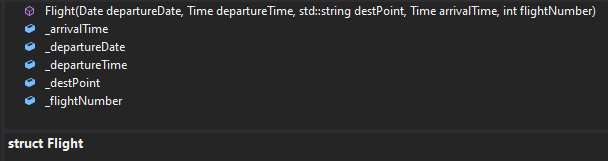
**Формулировка задачи**

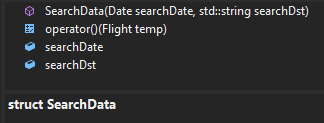
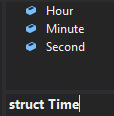
Класс «Расписание полётов» с полями: дата вылета, время вылета, пункт назначения, время прибытия, номер рейса. Вывести номера рейсов, время вылета и время прибытия на задаваемую дату и задаваемый пункт назначения.

**Диаграммы классов**





**Листинг программы на языке С++**

**Time.h**

struct Time

{

int Second;

int Minute;

int Hour;

};

**SearchData.h**

#pragma once

#include"Date.h"

#include "Flight.h"

#include <string>

struct SearchData

{

Date searchDate;

std::string searchDst;

SearchData(Date searchDate, std::string searchDst) : searchDate(searchDate), searchDst(searchDst) {}

bool operator()(Flight temp)

{

return temp.\_departureDate == searchDate && temp.\_destPoint == searchDst;

}

};

**Date.h**

#pragma once

class Date

{

public:

int Day;

int Month;

int Year;

bool operator==(Date& r)

{

long long sum1, sum2;

sum1 = Day + Month \* 31 + Year \* 366;

sum2 = r.Day + r.Month \* 31 + r.Year \* 366;

return sum1 == sum2;

}

};

**Flight.h**

#include <string>

#include "Date.h"

#include "Time.h"

struct Flight

{

Date \_departureDate; // Дата отправления

Time \_departureTime; // Время отправления

Time \_arrivalTime; // Время прибытия

int \_flightNumber; // Номер рейса

std::string \_destPoint; // Пункт назначения

Flight(Date departureDate, Time departureTime, std::string destPoint, Time arrivalTime, int flightNumber) : \_departureDate(departureDate), \_departureTime(departureTime), \_arrivalTime(arrivalTime), \_flightNumber(flightNumber), \_destPoint(destPoint) {}

};

**FlightScgedule.h**

#pragma once

#include <vector>

#include <map>

#include "SearchData.h"

#include "Flight.h"

using namespace std;

class FlightSchedule

{

public:

FlightSchedule() {};

~FlightSchedule() {};

const int WORK\_MODE\_VECTOR = 0;

const int WORK\_MODE\_MAP = 1;

public:

void choice(int nmode);

int mode()const;

public:

/\*

void readingfile(string& namefile); // exception

\*/

void set(Flight buf);

private:

public:

string printinfo(Date& depDate, string& dstPoint);

private:

int \_mode = 0;

vector<Flight> \_vector;

multimap<string, Flight> \_mmap;

};

**FlightScgedule.cpp**

#include "FlightScgedule.h"

using namespace std;

void FlightSchedule::choice(int nmode)

{

\_mode = nmode;

}

int FlightSchedule::mode()const

{

return \_mode;

};

void FlightSchedule::set(Flight buf) {

if (\_mode == WORK\_MODE\_VECTOR)

\_vector.push\_back(buf);

else if (\_mode == WORK\_MODE\_MAP)

{

\_mmap.insert(pair<string, Flight>(buf.\_destPoint, buf));

}

};

string FlightSchedule::printinfo(Date& depDate, string& dstPoint)

{

string s = "";

if (\_mode == WORK\_MODE\_VECTOR)

{

auto it = \_vector.begin();

while (it != \_vector.end())

{

it = find\_if(it, \_vector.end(), SearchData(depDate, dstPoint));

if (it != \_vector.end())

{

//cout << it->flightNo << " " << it->departureTime << " " << it->arrivalTime << endl;

s += to\_string(it->\_departureDate.Day); s += ".";

s += to\_string(it->\_departureDate.Month); s += ".";

s += to\_string(it->\_departureDate.Year); s += " ";

s += to\_string(it->\_departureTime.Hour); s += ":";

s += to\_string(it->\_departureTime.Minute); s += ":";

s += to\_string(it->\_departureTime.Second); s += " ";

s += it->\_destPoint; s += " ";

s += to\_string(it->\_arrivalTime.Hour); s += ":";

s += to\_string(it->\_arrivalTime.Minute); s += ":";

s += to\_string(it->\_arrivalTime.Second); s += " ";

s += to\_string(it->\_flightNumber); s += "\n";

it++;

}

}

}

else if (\_mode == WORK\_MODE\_MAP)

{

auto it = \_mmap.begin();

int counter = \_mmap.count(dstPoint);

it = \_mmap.find(dstPoint);

for (int i = 0; i < counter; ++i, ++it)

{

if (it->second.\_departureDate == depDate)

{

//cout << it->second.departureDate << " " << it->second.departureTime << " " << it->second.destinationPoint<< " " << it->second.arrivalTime << " " << it->second.flightNumber << endl;

s += to\_string(it->second.\_departureDate.Day); s += ".";

s += to\_string(it->second.\_departureDate.Month); s += ".";

s += to\_string(it->second.\_departureDate.Year); s += " ";

s += to\_string(it->second.\_departureTime.Hour); s += ":";

s += to\_string(it->second.\_departureTime.Minute); s += ":";

s += to\_string(it->second.\_departureTime.Second); s += " ";

s += it->second.\_destPoint; s += " ";

s += to\_string(it->second.\_arrivalTime.Hour); s += ":";

s += to\_string(it->second.\_arrivalTime.Minute); s += ":";

s += to\_string(it->second.\_arrivalTime.Second); s += " ";

s += to\_string(it->second.\_flightNumber); s += "\n";

}

}

}

return s;

}

**Read\_in\_file.h**

#pragma once

#include "FlightScgedule.h"

#include <fstream>

void readingfile(string& namefile, FlightSchedule& buffer);

**Read\_in\_file.cpp**

#include "Read\_in\_file.h"

void readingfile(string& namefile, FlightSchedule& buffer) // exception

{

ifstream fin(namefile);

if (!fin.is\_open())

throw new exception("Ошибка открытия файла"); //exception

Date depDate;

Time depTime;

string dstPoint;

Time arrTime;

int flightNo;

char c;

while (!fin.eof())

{

fin >> depDate.Day >> c >> depDate.Month >> c >> depDate.Year;

fin >> depTime.Second >> c >> depTime.Minute >> c >> depTime.Hour;

fin >> dstPoint;

fin >> arrTime.Second >> c >> arrTime.Minute >> c >> arrTime.Hour;

fin >> flightNo;

Flight buf(depDate, depTime, dstPoint, arrTime, flightNo);

buffer.set(buf);

}

fin.close();

};

**Console.h**

#include <iostream>

#include <string>

#include <algorithm>

#include "FlightScgedule.h"

#include "Read\_in\_file.h"

class Console {

public:

int init();

};

**Console.cpp**

#include "Console.h"

int Console::init() {

bool error = false;

FlightSchedule objFS;

cout << "0 - работа с вектором \n1 - работа со словарём" << endl;

cout << "Выберете режим работы: ";

int answMode;

cin >> answMode;

if (answMode != 1 && answMode != 0)

{

cout << "Нет такого режима работы!" << endl;

return 0;

}

objFS.choice(answMode);

string namefile = "input.txt";

try

{

readingfile(namefile, objFS);

}

catch (exception \* e)

{

cout << e << endl;

error = true;

}

if (error)

return 0;

// Ввод даты и пукнта назначения

string destonation;

Date data;

char c;

cout << "Введите дату отправления и пункт назначения рейса через пробел: ";

cin >> data.Day >> c >> data.Month >> c >> data.Year;

cin >> destonation;

cout << objFS.printinfo(data, destonation);

}

**Lab4.cpp**

#include "Console.h"

int main()

{

setlocale(LC\_ALL, "RUS");

Console c;

c.init();

return 0;

}

**Заключение**

В процессе выполнения лабораторной работы №4 мы ознакомились с классами-контейнерами библиотеки STL, увеличили свои теоретические и практические навыки работы с контейнерами STL.

Авторы Преподаватель: Тарасов

Шутенко Остап Русланович Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_ Дата: \_\_.\_\_.2020