

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.03 | Прикладная информатика | | |
| Направленность (профиль) | |  | Прикладная информатика в химии | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | II | | | Группа | 415 |

Отчёт по контрольной работе № 1

Вариант № 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 415 |  |  |  | Шарашидзе Никита Леванович |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Федин Алексей Константинович |
|  |  | (дата, подпись) |  | Корниенко Иван Григорьевич |

Санкт-Петербург

2022

Содержание

[1 Постановка задачи 3](#_Toc10715)

[2 Исходные данные 3](#_Toc12145)

[3 Особые ситуации 3](#_Toc5812)

[4 Математические методы и алгоритмы решения задач 3](#_Toc2713)

[6 Структура программы 4](#_Toc4003)

[7 Блок-схемы алгоритмов 5](#_Toc18202)

[8 Результаты работы программы 8](#_Toc20076)

[9 Описание хода выполнения контрольной работы 12](#_Toc21184)

[10 Исходный текст программы 12](#_Toc9758)

## 1 Постановка задачи

## Необходимо разработать класс для указанной предметной области. Доступ к данным реализовать с помощью методов Set, Get, Show. При выполнении контрольной работы нельзя использовать контейнеры и алгоритмы библиотеки STL или аналогичных сторонних библиотек.

## Рейс: пункт назначения, номер рейса, тип самолета, время вылета, дни недели. Создать массив объектов. Реализовать возможность получения: – списка рейсов для заданного пункта назначения, – списка рейсов для заданного дня недели, – списка рейсов для заданного дня недели, время вылета для которых позже заданного.

## 2 Исходные данные

## В качестве исходных данных программа использует либо данные, введённые пользователем, либо данные из файла. Путь к файлу пишет пользователь.

## 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

– Пользователь выберет несуществующий пункт меню.

– Пользователь введёт некорректное значение.

– Пользователь попытается взять данные из несуществующего файла.

– Пользователь попытается создать файл для сохранения данных с уже занятым именем.

– Пользователь выйдет за пределы массива при выборе слишком большого или слишком малого количества элементов массива.

## 4 Математические методы и алгоритмы решения задач

Массив данных сортируется методом пузырька. Алгоритм сортировки пузырьковым методом состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N – 1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива.

**5 Форматы представления данных**

Таблица 1 – Переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| hour\_ | int | Количество затраченных часов |
| minute\_ | int | Количество затраченных минут |
| \_departure\_time | time\_type | Время вылета |
| \_departure\_day | string | День вылета |
| \_plane\_type | string | Тип самолёта |
| \_destination | string | Пункт назначения |

Таблица 2 – Константы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Значение** | **Описание** |
| max\_hour | int | 24 | Количество часов в сутках |
| max\_minute | int | 60 | Количество минут в часе |
| max\_plane | int | 3 | Максимально возможное количество самолётов |
| max\_day | int | 7 | Максимальное количество дней в неделе |

Таблица 3 – Классы, используемые в программе

|  |  |
| --- | --- |
| **Имя класса** | **Описание** |
| Flight | Класс, описыващий объект – самолёт |

## 6 Структура программы

Для оптимизации работы с кодом было принято решение разделить программу на модули.

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Source.cpp | Основной модуль |
| Check.cpp | Модуль для проверки данных в консоли |
| Algorithm.cpp | Модуль для алгоритмов вывода данных разными способами |
| ConsoleInput.cpp | Модуль ввода данных в консоли |
| ConsoleOutput.cpp | Модуль вывода данных из консоли |
| InputDataCheck.cpp | Модуль для проверки данных из файла |
| Menu.cpp | Модуль с меню программы |

Таблица 4 – Модули, составляющие программу

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| EnterData.cpp | Модуль с подменю для ввода пользователем данных о рейсе |
| FileInput.cpp | Модуль ввода данных из файла |
| FileOutput.cpp | Модуль вывода данных в файл |
| GetFileName.cpp | Модуль для ввода пути файла |
| Flight.cpp | Модуль, содержащий класс Flight |
| UnitTest.cpp | Модуль для юнит-тестов |

Продолжение таблицы 4.

## Таблица 5 – Функции, составляющие EnterData.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| EnterDepartureTime | Получает данные от пользователя о времени отправки рейса |
| EnterDepartureDay | Получает данные от пользователя о дне рейса |
| EnterPlaneType | Получает данные от пользователя о типе самолёта |
| EnterDestination | Получает данные от пользователя о пункте назначения |

## Таблица 6 – Функции, составляющие Check.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| CheckMenu | Проверяет введённый пункт меню |
| GetValue | Получает введённое значение для последующей проверки |
| GetPositiveValue | Проверяет введённое числовое значение на положительность |
| GetNotNegativeValue | Проверяет введённое числовое значение на неотрицательное значение |

## Таблица 7 – Функции, составляющие ConsoleInput.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| ConsoleInput | Получает данные, введённые в консоли |

## Таблица 8 – Функции, составляющие FileInput.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| FileInput | Получает данные из файла, введённого пользователем |

Таблица 9 – Функции, составляющие Algorithm.cpp

|  |  |
| --- | --- |
| DestinationList | Подменю для ввода пункта назначения |
| WeekdayAndTimeList | Подменю для ввода дня недели и времени вылета |
| WeekdayList | Подменю для ввода дня недели |
| GetDestinationList | Выводит список рейсов для введённого пункта назначения |
| GetWeekdayAndTimeList | Выводит список рейсов для заданного дня недели и времени вылета для которых позже заданного |
| GetWeekdayList | Выводит список рейсов для заданного дня недели |

## 7 Блок-схемы алгоритмов

## 

Рисунок 1 – Блок-схема алгоритма функции “выборка по дню недели”

## 

Рисунок 2 – Блок-схема алгоритма функции “выборка по пункту назначения”

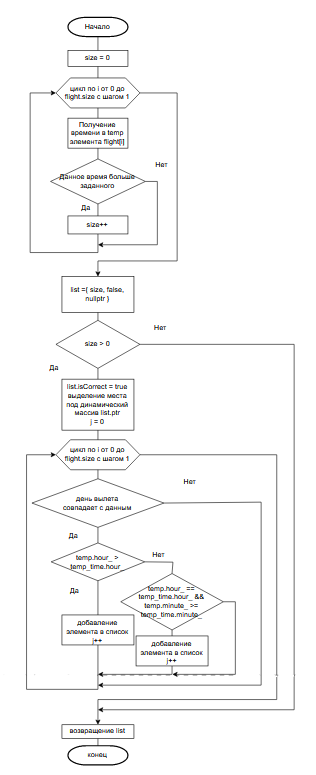


Рисунок 3 – Блок-схема алгоритма функции “выборка по дню недели и времени вылета”

## 8 Результаты работы программы

Экранные копии работы программы:

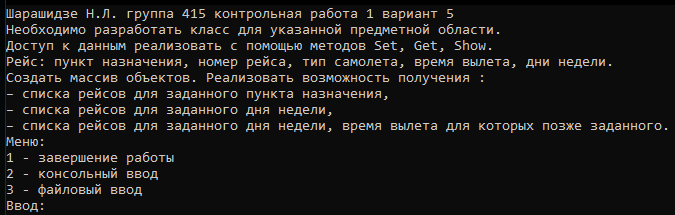


Рисунок 4 – Экранная копия меню программы

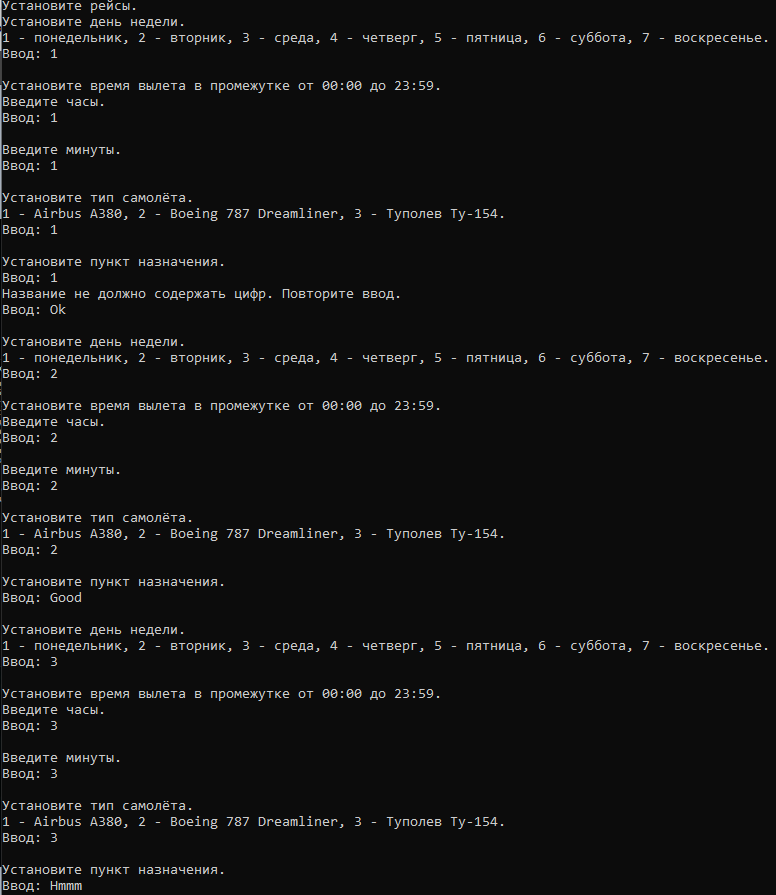


Рисунок 5 – Экранная копия ввода количества рейсов и информации о них

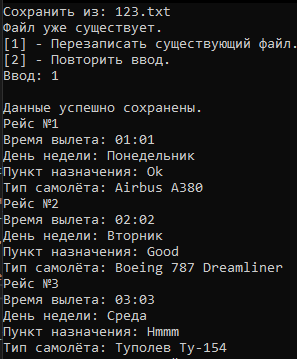


Рисунок 6 – Экранная копия сохранения полученного списка рейсов в файл и вывод общего списка всех введённых рейсов

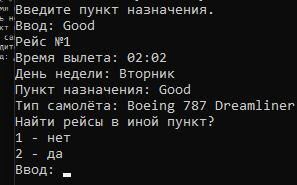


Рисунок 7 – Экранная копия вывода списка рейсов в ведённый пункт назначения

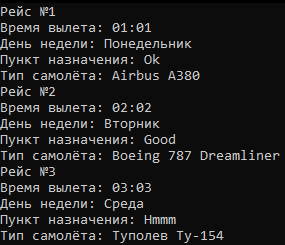


Рисунок 8 – Экранная копия вывода информации о рейсах из файла

# 9 Описание хода выполнения контрольной работы

1) В ходе контрольной работы было создано решение (Solution) в

интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был создан проект.

1. При написании блока ввода данных была выявлена проблема загрязнения потока ввода остаточными данными. Было найдено решение – очищение потока ввода с помощью функций cin.clear() и cin.ignore().
2. При создании общего списка всех рейсов возникли трудности из-за невозможности работать со статическим массивом и с библиотекой vector. Проблема была решена изучение динамических массивов и их дальнейшим использованием.
3. В ходе написания меню программы была обнаружена проблема с отсутствием русской локали. После изучения проблемы была использована функция setlocale(LC\_ALL, "Russian"), позволяющая выводить в консоли русский текст.

## 10 Исходный текст программы

**[ Начало Source.cpp--- ]**

#include "Flight.h"

#include "GetFileName.h"

#include "Items.h"

#include "ConsoleInput.h"

#include "FileInput.h"

#include "ConsoleOutput.h"

#include "FileOutput.h"

#include "Algorithm.h"

#include "Menu.h"

#include <iostream>

#include <string>

#include <windows.h>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "RU");

std::cout << "Шарашидзе Н.Л. группа 415 контрольная работа 1 вариант 5" << std::endl;

std::cout << "Необходимо разработать класс для указанной предметной области." << std::endl <<"Доступ к данным реализовать с помощью методов Set, Get, Show."

<< std::endl << "Рейс: пункт назначения, номер рейса, тип самолета, время вылета, дни недели." << std::endl <<"Создать массив объектов. Реализовать возможность получения :"

<< std::endl << "– списка рейсов для заданного пункта назначения,"

<< std::endl << "– списка рейсов для заданного дня недели,"

<< std::endl << "– списка рейсов для заданного дня недели, время вылета для которых позже заданного." << std::endl;

while (true)

{

dynamic\_array flight = { 0, true, nullptr };

inputChoice input\_choice = InputMenu();

system("cls");

switch (input\_choice)

{

case inputChoice::console\_input:

flight = ConsoleInput();

break;

case inputChoice::file\_input:

flight = FileInput();

break;

case inputChoice::quit:

return 0;

}

if (!flight.isCorrect)

{

system("cls");

continue;

}

binaryChoice file\_output\_choice = OutputMenu();

system("cls");

if (file\_output\_choice == yes)

{

if (!FileOutput(flight))

{

delete[] flight.ptr;

system("pause");

system("cls");

continue;

}

}

ConsoleOutput(flight);

while (true)

{

binaryChoice enter\_list\_choice = EnterListMenu();

if (enter\_list\_choice == binaryChoice::no)

break;

listChoice list\_choice = ListMenu();

system("cls");

switch (list\_choice)

{

case listChoice::destination\_sort:

DestinationList(flight);

break;

case listChoice::weekday\_and\_time\_sort:

WeekdayAndTimeList(flight);

break;

case listChoice::weekday\_sort:

WeekdayList(flight);

break;

}

}

delete[] flight.ptr;

binaryChoice end\_program\_choice = EndProgramMenu();

if (end\_program\_choice == binaryChoice::yes)

return 0;

system("cls");

}

}

**[---Конец Source.cpp]**

**[ Начало InputDataCheck.h--- ]**

#pragma once

#include <iostream>

#include <fstream>

#include <string>

const int error = 1;

int getLineI(std::ifstream& file);

double getLineD(std::ifstream& file);

std::string getLineS(std::ifstream& file);

**[---Конец InputDataCheck.h]**

**[ Начало InputDataCheck.cpp--- ]**

#include "InputDataCheck.h"

int getLineI(std::ifstream& file)

{

std::string temp\_s = "";

int temp\_i = 0;

try

{

std::getline(file, temp\_s);

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла." << std::endl;

throw error;

}

try

{

temp\_i = stoi(temp\_s);

if (temp\_i < 0) throw error;

else return temp\_i;

}

catch (int)

{

std::cout << "Некорректный формат данных." << std::endl;

throw error;

}

}

double getLineD(std::ifstream& file)

{

std::string temp\_s = "";

double temp\_d = 0.0;

try

{

std::getline(file, temp\_s);

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла." << std::endl;

throw error;

}

try

{

temp\_d = stod(temp\_s);

if (temp\_d < 0) throw error;

else return temp\_d;

}

catch (int)

{

std::cout << "Некорректный формат данных." << std::endl;

throw error;

}

}

std::string getLineS(std::ifstream& file)

{

std::string temp\_1 = "";

try

{

std::getline(file, temp\_1);

return temp\_1;

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла." << std::endl;

throw error;

}

}

**[---Конец InputDataCheck.cpp]**

**[ Начало Algorithm.h--- ]**

#pragma once

#include "Items.h"

#include "Flight.h"

#include "ConsoleOutput.h"

#include <string>

#include <iostream>

void DestinationList(const dynamic\_array flight);

void WeekdayAndTimeList(const dynamic\_array flight);

void WeekdayList(const dynamic\_array flight);

dynamic\_array GetDestinationList(const dynamic\_array flight, std::string value);

dynamic\_array GetWeekdayAndTimeList(const dynamic\_array flight, time\_type temp\_time, int number);

dynamic\_array GetWeekdayList(const dynamic\_array flight, int number);

**[---Конец Algorithm.h]**

**[ Начало Algorithm.cpp--- ]**

#include "Algorithm.h"

void DestinationList(const dynamic\_array flight)

{

while (true)

{

ConsoleOutput(flight);

std::string value = "";

std::cout << "Введите пункт назначения." << std::endl;

std::cout << "Ввод: ";

std::getline(std::cin, value);

dynamic\_array list = { 0, true, nullptr };

list = GetDestinationList(flight, value);

if (list.size > 0)

{

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

{

list.ptr[i].ShowFlight(i);

}

}

else

std::cout << "Рейсы в данный пункт назначения отсутствуют." << std::endl;

std::cout << "Найти рейсы в иной пункт?\n1 - нет\n2 - да" << std::endl;

binaryChoice end\_function = static\_cast<binaryChoice>(CheckMenu(two));

system("cls");

if (end\_function == binaryChoice::no)

return;

}

}

void WeekdayAndTimeList(const dynamic\_array flight)

{

while (true)

{

ConsoleOutput(flight);

std::cout << "Выберете день недели." << std::endl;

std::cout << "1 - понедельник, 2 - вторник, 3 - среда, 4 - четверг, 5 - пятница, 6 - суббота, 7 - воскресенье." << std::endl;

int number = CheckMenu(max\_day) - 1;

std::cout << "Выберете время в промежутке от 00:00 до 23:59." << std::endl;

std::cout << "Введите часы." << std::endl;

time\_type temp\_time = { 0, 0 };

while (true)

{

std::cout << "Ввод: ";

temp\_time.hour\_ = GetNotNegativeInt();

if (temp\_time.hour\_ > max\_hour - 1)

{

std::cout << "Введите число не больше 23." << std::endl;

continue;

}

break;

}

std::cout << "Введите минуты." << std::endl;

while (true)

{

std::cout << "Ввод: ";

temp\_time.minute\_ = GetNotNegativeInt();

if (temp\_time.minute\_ > max\_minute - 1)

{

std::cout << "Введите число не более 60." << std::endl;

continue;

}

break;

}

dynamic\_array list = { 0, true, nullptr };

list = GetWeekdayAndTimeList(flight, temp\_time , number);

if (list.size > 0)

{

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

{

list.ptr[i].ShowFlight(i);

}

}

else

std::cout << "Рейсы в данный день недели отсутсвуют." << std::endl;

std::cout << "Найти рейсы в иное время?\n1 - Нет\n2 - Да" << std::endl;

binaryChoice end\_function = static\_cast<binaryChoice>(CheckMenu(two));

system("cls");

if (end\_function == binaryChoice::no)

return;

}

}

void WeekdayList(const dynamic\_array flight)

{

while (true)

{

ConsoleOutput(flight);

std::cout << "Установите день недели." << std::endl;

std::cout << "1 - понедельник, 2 - вторник, 3 - среда, 4 - четверг, 5 - пятница, 6 - суббота, 7 - воскресенье." << std::endl;

int number = CheckMenu(max\_day) - 1;

dynamic\_array list = { 0, true, nullptr };

list = GetWeekdayList(flight, number);

if (list.size > 0)

{

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

{

list.ptr[i].ShowFlight(i);

}

}

else

std::cout << "Рейсы в данный день недели отсутсвуют." << std::endl;

std::cout << "Найти рейсы в иное время?\n1 - Нет\n2 - Да" << std::endl;

binaryChoice end\_function = static\_cast<binaryChoice>(CheckMenu(two));

system("cls");

if (end\_function == binaryChoice::no)

return;

}

}

dynamic\_array GetDestinationList(const dynamic\_array flight, std::string value)

{

int size = 0;

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

{

if (flight.ptr[i].GetDestination() == value)

{

size++;

}

}

dynamic\_array list = { size, false, nullptr };

if (size > 0)

{

list.isCorrect = true;

list.ptr = new Flight[size];

int j = 0;

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

{

if (flight.ptr[i].GetDestination() == value)

{

list.ptr[j] = flight.ptr[i];

j++;

}

}

}

return list;

}

dynamic\_array GetWeekdayAndTimeList(const dynamic\_array flight, time\_type temp\_time, int number)

{

int size = 0;

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

{

if (flight.ptr[i].GetDepartureDay() == week\_day[number])

{

time\_type temp = flight.ptr[i].GetDepartureTime();

if ((temp.hour\_ > temp\_time.hour\_) || (temp.hour\_ == temp\_time.hour\_ && temp.minute\_ >= temp\_time.minute\_))

{

size++;

}

}

}

dynamic\_array list = { size, false, nullptr };

if (size > 0)

{

list.isCorrect = true;

list.ptr = new Flight[size];

int j = 0;

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

{

if (flight.ptr[i].GetDepartureDay() == week\_day[number])

{

time\_type temp = flight.ptr[i].GetDepartureTime();

if (temp.hour\_ > temp\_time.hour\_)

{

list.ptr[j] = flight.ptr[i];

j++;

}

else if (temp.hour\_ == temp\_time.hour\_ && temp.minute\_ >= temp\_time.minute\_)

{

list.ptr[j] = flight.ptr[i];

j++;

}

}

}

}

return list;

}

dynamic\_array GetWeekdayList(const dynamic\_array flight, int number)

{

int size = 0;

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

{

if (flight.ptr[i].GetDepartureDay() == week\_day[number])

{

size++;

}

}

dynamic\_array list = { size, false, nullptr };

if (size > 0)

{

list.isCorrect = true;

list.ptr = new Flight[size];

int j = 0;

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

{

if (flight.ptr[i].GetDepartureDay() == week\_day[number])

{

list.ptr[j] = flight.ptr[i];

j++;

}

}

}

return list;

}

**[---Конец Algorithm.cpp]**

**[ Начало Check.h--- ]**

#pragma once

#include <iostream>

#include <string>

const int error\_ = 1;

enum numberOfChoice

{

two = 2,

three = 3

};

int CheckMenu(int items);

int GetNotNegativeInt();

int GetPositiveInt();

int GetInt();

**[---Конец Check.h]**

**[ Начало Check.cpp--- ]**

#include "Check.h"

int CheckMenu(int items)

{

std::cout << "Ввод: ";

while (true)

{

int value = 0;

try

{

value = GetPositiveInt();

if (value <= items && value >= 1)

return value;

else

throw value;

}

catch (int ex\_value)

{

std::cout << std::endl;

std::cout << ex\_value << " – не существующий пункт меню. Повторите ввод: ";

return 0;

}

}

}

template<typename T>

T GetValue()

{

T temp = static\_cast<T>(0.0);

do

{

try

{

std::cin >> temp;

std::cout << std::endl;

if (std::cin.peek() != '\n')

throw std::exception("IncorrectValue");

return temp;

}

catch (const std::exception&)

{

std::cout << "Введены некорректные данные. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

} while (true);

}

template <typename T>

T GetPositiveValue()

{

T temp = static\_cast<T>(0.0);

do

{

try

{

temp = GetValue<T>();

if (temp <= 0) throw error\_;

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

return temp;

}

catch (const int&)

{

std::cout << "Число должно быть положительным. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

} while (true);

}

template <typename T>

T GetNotNegativeValue()

{

T temp = static\_cast<T>(0.0);

do

{

try

{

temp = GetValue<T>();

if (temp < 0) throw error\_;

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

return temp;

}

catch (const int&)

{

std::cout << "Число не должно быть отрицательным. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

} while (true);

}

int GetNotNegativeInt()

{

return GetNotNegativeValue<int>();

}

int GetPositiveInt()

{

return GetPositiveValue<int>();

}

int GetInt()

{

return GetValue<int>();

}

**[---Конец Check.cpp]**

**[ Начало ConsoleInput.h--- ]**

#pragma once

#include "Flight.h"

#include "Items.h"

#include "Check.h"

#include "EnterData.h"

dynamic\_array ConsoleInput();

**[---Конец ConsoleInput.h]**

**[ Начало ConsoleInput.cpp--- ]**

#include "ConsoleOutput.h"

void ConsoleOutput(const dynamic\_array temp)

{

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

{

temp.ptr[i].ShowFlight(i);

}

}

**[---Конец ConsoleInput.cpp]**

**[ Начало EnterData.h--- ]**

#pragma once

#include "Check.h"

#include "Flight.h"

#include "Items.h"

void EnterDepartureTime(Flight& object);

void EnterDepartureDay(Flight& object);

void EnterPlaneType(Flight& object);

void EnterDestination(Flight& object);

**[---Конец EnterData.h]**

**[ Начало EnterData.cpp--- ]**

#include "EnterData.h"

void EnterDepartureTime(Flight& object)

{

time\_type temp = { 0, 0 };

std::cout << "Установите время вылета в промежутке от 00:00 до 23:59." << std::endl;

std::cout << "Введите часы." << std::endl;

while (true)

{

std::cout << "Ввод: ";

temp.hour\_ = GetPositiveInt();

if (temp.hour\_ > max\_hour - 1)

{

std::cout << "Введите число не больше 23." << std::endl;

continue;

}

break;

}

std::cout << "Введите минуты." << std::endl;

while (true)

{

std::cout << "Ввод: ";

temp.minute\_ = GetInt();

if (temp.minute\_ > max\_minute - 1)

{

std::cout << "Введите число не более 60." << std::endl;

continue;

}

break;

}

object.SetDepartureTime(temp);

}

void EnterDepartureDay(Flight& object)

{

std::cout << "Установите день недели." << std::endl;

std::cout << "1 - понедельник, 2 - вторник, 3 - среда, 4 - четверг, 5 - пятница, 6 - суббота, 7 - воскресенье." << std::endl;

int number = CheckMenu(max\_day) - 1;

object.SetDepartureDay(week\_day[number]);

}

void EnterPlaneType(Flight& object)

{

std::cout << "Установите тип самолёта." << std::endl;

std::cout << "1 - Airbus A380, 2 - Boeing 787 Dreamliner, 3 - Туполев Ту-154." << std::endl;

int number = CheckMenu(max\_plane) - 1;

object.SetPlaneType(plane\_type[number]);

}

void EnterDestination(Flight& object)

{

std::string temp = "";

std::cout << "Установите пункт назначения." << std::endl;

while(true)

{

bool is\_correct = true;

std::cout << "Ввод: ";

getline(std::cin, temp);

for (const char c : temp)

{

if (isdigit(c))

{

std::cout << "Название не должно содержать цифр. Повторите ввод." << std::endl;

is\_correct = false;

break;

}

}

if (!is\_correct)

continue;

break;

}

object.SetDestination(temp);

}

**[---Конец EnterData.cpp]**

**[ Начало FileInput.h--- ]**

#pragma once

#include "GetFileName.h"

#include "Flight.h"

#include "Items.h"

#include "InputDataCheck.h"

dynamic\_array FileInput();

**[---Конец FileInput.h]**

**[ Начало FileInput.cpp--- ]**

#include "FileInput.h"

dynamic\_array FileInput()

{

system("cls");

std::ifstream file;

std::string file\_name;

file.exceptions(std::ifstream::badbit | std::ifstream::failbit);

dynamic\_array temp = { 0, true, nullptr };

while (true)

{

try

{

file\_name = GetFileName(input);

file.open(file\_name);

std::cout << "Файл успешно открыт." << std::endl;

break;

}

catch (const std::exception&)

{

std::cout << "Ошибка доступа.\n" << "Хотите попробовать снова? 1 – Нет, 2 – Да" << std::endl;

std::cout << "Ввод: ";

int user\_choice = CheckMenu(two);

if (user\_choice != restart)

temp.isCorrect = false;

return temp;

}

}

Flight object;

try

{

int size = getLineI(file);

if (size > max\_size || size <= 0)

throw error;

temp.size = size;

temp.ptr = new Flight[size];

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

{

std::string day = getLineS(file);

bool isValid = false;

for (int j = 0; j < max\_day; j++)

{

if (day == week\_day[j])

{

isValid = true;

break;

}

}

if (!isValid)

throw error;

int hour = getLineI(file);

if (hour > max\_hour - 1)

throw error;

int minute = getLineI(file);

if (minute > max\_minute - 1)

throw error;

time\_type time\_ = { hour, minute };

std::string destination = getLineS(file);

for (char& c : destination)

{

if (isdigit(c))

throw error;

}

std::string plane = getLineS(file);

isValid = false;

for (int j = 0; j < max\_plane; j++)

{

if (plane == plane\_type[j])

{

isValid = true;

break;

}

}

if (!isValid)

throw error;

object.SetDepartureDay(day);

object.SetDepartureTime(time\_);

object.SetPlaneType(plane);

object.SetDestination(destination);

temp.ptr[i] = object;

}

std::cout << "Данные успешно считаны." << std::endl;

file.close();

return temp;

}

catch (int)

{

std::cout << "Некорректный формат данных." << std::endl;

file.close();

delete[] temp.ptr;

temp = { 0, false, nullptr };

system("pause");

return temp;

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла." << std::endl;

file.close();

temp.isCorrect = false;

return temp;

}

}

**[---Конец FileInput.cpp]**

**[ Начало FileOutput.h--- ]**

#pragma once

#include "GetFileName.h"

#include "Flight.h"

#include "Items.h"

bool FileOutput(dynamic\_array temp);

**[---Конец FileOutput.h]**

**[ Начало FileOutput.cpp--- ]**

#include "FileOutput.h"

bool FileOutput(dynamic\_array temp)

{

std::ofstream file;

file.exceptions(std::ofstream::badbit | std::ofstream::failbit);

bool is\_stream\_opened = false;

std::string file\_name = "";

do

{

try

{

file\_name = GetFileName(output);

file.open(file\_name);

is\_stream\_opened = true;

}

catch (const std::exception&)

{

std::cout << "Ошибка при попытке открыть файл." << std::endl;

continue;

}

} while (!is\_stream\_opened);

try

{

file << temp.size << std::endl;

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

{

file << temp.ptr[i].GetDepartureDay() << std::endl;

time\_type value = temp.ptr[i].GetDepartureTime();

file << value.hour\_ << std::endl;

file << value.minute\_ << std::endl;

file << temp.ptr[i].GetDestination() << std::endl;

file << temp.ptr[i].GetPlaneType() << std::endl;

}

std::cout << "Данные успешно сохранены." << std::endl;

file.close();

return true;

}

catch(std::exception&)

{

std::cout << "Ошибка при записи информации в файл. Данные удалены." << std::endl;

file.close();

return false;

}

}

**[---Конец FileOutput.cpp]**

**[ Начало Flight.h--- ]**

#pragma once

#include "Check.h"

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

const int max\_hour = 24;

const int max\_minute = 60;

const int max\_plane = 3;

const int max\_day = 7;

const std::string week\_day[max\_day] = { "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота", "Воскресенье" };

const std::string plane\_type[max\_plane] = { "Airbus A380",

"Boeing 787 Dreamliner",

"Туполев Ту-154" };

typedef struct

{

int hour\_;

int minute\_;

} time\_type;

class Flight

{

private:

time\_type \_departure\_time;

std::string \_departure\_day;

std::string \_plane\_type;

std::string \_destination;

public:

Flight() = default;

Flight(time\_type dt, std::string dd, std::string pt, std::string dst);

void SetDepartureTime(time\_type value);

time\_type GetDepartureTime();

void SetDepartureDay(std::string value);

std::string GetDepartureDay();

void SetPlaneType(std::string value);

std::string GetPlaneType();

void SetDestination(std::string value);

std::string GetDestination();

void ShowFlight(int number);

bool operator==(const Flight& other);

bool operator!=(const Flight& other);

};

**[---Конец Flight.h]**

**[ Начало Flight.cpp--- ]**

#include "Flight.h"

Flight::Flight(time\_type dt, std::string dd, std::string pt, std::string dst) : \_departure\_time(dt), \_departure\_day(dd), \_plane\_type(pt), \_destination(dst)

{

}

void Flight::SetDepartureTime(time\_type value)

{

\_departure\_time = value;

}

time\_type Flight::GetDepartureTime()

{

return \_departure\_time;

}

void Flight::SetDepartureDay(std::string value)

{

\_departure\_day = value;

}

std::string Flight::GetDepartureDay()

{

return \_departure\_day;

}

void Flight::SetPlaneType(std::string value)

{

\_plane\_type = value;

}

std::string Flight::GetPlaneType()

{

return \_plane\_type;

}

void Flight::SetDestination(std::string value)

{

\_destination = value;

}

std::string Flight::GetDestination()

{

return \_destination;

}

void Flight::ShowFlight(int number)

{

std::cout << "Рейс №" << number + 1 << std::endl;

std::cout << "Время вылета: "; std::cout << std::setfill('0') << std::setw(2) << \_departure\_time.hour\_ << ":" << std::setfill('0') << std::setw(2) << \_departure\_time.minute\_ << std::endl;

std::cout << "День недели: " << \_departure\_day << std::endl;

std::cout << "Пункт назначения: " << \_destination << std::endl;

std::cout << "Тип самолёта: " << \_plane\_type << std::endl;

}

bool Flight::operator==(const Flight& other)

{

return \_departure\_time.hour\_ == other.\_departure\_time.hour\_ && \_departure\_time.minute\_ == other.\_departure\_time.minute\_ && \_departure\_day == other.\_departure\_day && \_plane\_type == other.\_plane\_type && \_destination == other.\_destination;

}

bool Flight::operator!=(const Flight& other)

{

return !(\*this == other);

}

**[---Конец Flight.cpp]**

**[ Начало GetFileName.h--- ]**

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <filesystem>

#include "Check.h"

using namespace std::filesystem;

enum

{

output = 0,

input = 1

};

const int restart = 2;

std::string GetFileName(int stream\_type);

**[---Конец GetFileName.h]**

**[ Начало GetFileName.cpp--- ]**

#include "GetFileName.h"

std::string FileInput()

{

while (true)

{

std::string file\_path;

std::cout << "Ââîä: ";

getline(std::cin, file\_path);

if (file\_path.find(".txt") >= std::string::npos)

{

std::cout << "Íå òåêñòîâûé ôàéë. Ïîâòîðèòå ââîä." << std::endl;

continue;

}

if (!std::ifstream(file\_path)) {

std::cout << "Ôàéë íå ñóùåñòâóåò. Ïîâòîðèòå ââîä." << std::endl;

continue;

}

std::error\_code ec;

if (!is\_regular\_file(file\_path, ec))

{

std::cout << "Àäðåñ ñîäåðæèò íåäîïóñòèìûå çíà÷åíèÿ. Ïîâòîðèòå ââîä." << std::endl;

continue;

}

std::ifstream myFile(file\_path);

if (!myFile) {

std::cout << "Çàãðóçêà çàïðåùåíà. Ïîâòîðèòå ââîä." << std::endl;

myFile.close();

continue;

}

myFile.close();

return file\_path;

}

}

std::string FileOutput()

{

const int get\_name = 2;

while (true)

{

std::string file\_path;

std::cout << "Ñîõðàíèòü èç: ";

getline(std::cin, file\_path);

if (file\_path.find(".txt") >= std::string::npos)

{

std::cout << "Íå òåêñòîâûé ôàéë. Ïîâòîðèòå ââîä." << std::endl;

continue;

}

if (std::ifstream(file\_path))

{

std::cout << "Ôàéë óæå ñóùåñòâóåò." << std::endl;

std::cout << "[1] - Ïåðåçàïèñàòü ñóùåñòâóþùèé ôàéë." << std::endl;

std::cout << "[2] - Ïîâòîðèòü ââîä." << std::endl;

int tryAnotherFile = CheckMenu(2);

if (tryAnotherFile == get\_name)

{

continue;

}

}

std::ofstream myFile(file\_path, std::ofstream::app);

std::error\_code ec;

if (!is\_regular\_file(file\_path, ec))

{

std::cout << "Àäðåñ ñîäåðæèò íåäîïóñòèìûå çíà÷åíèÿ. Ïîâòîðèòå ââîä." << std::endl;

continue;

}

if (!myFile)

{

std::cout << "Çàïèñü çàïðåùåíà. Ïîâòîðèòå ââîä." << std::endl;

myFile.close();

continue;

}

myFile.close();

return file\_path;

}

}

std::string GetFileName(int stream\_type)

{

while (true) {

std::string file\_path;

switch (stream\_type)

{

case input:

return file\_path = FileInput();

case output:

return file\_path = FileOutput();

}

}

}

**[---Конец GetFileName.cpp]**

**[ Начало Items.h--- ]**

#pragma once

#include "Flight.h"

enum inputChoice

{

quit = 1,

console\_input = 2,

file\_input = 3

};

enum binaryChoice

{

no = 1,

yes = 2

};

enum listChoice

{

destination\_sort = 1,

weekday\_sort = 2,

weekday\_and\_time\_sort = 3

};

typedef struct

{

int size;

bool isCorrect;

Flight\* ptr;

} dynamic\_array;

const int max\_size = 20;

**[---Конец Items.h]**

**[ Начало Menu.h--- ]**

#pragma once

#include "Items.h"

#include "Check.h"

inputChoice InputMenu();

binaryChoice OutputMenu();

binaryChoice EnterListMenu();

listChoice ListMenu();

binaryChoice EndProgramMenu();

**[---Конец Menu.h]**

**[ Начало Menu.cpp--- ]**

#include "Menu.h"

inputChoice InputMenu()

{

std::cout << "Меню:" << std::endl << "1 - завершение работы" << std::endl << "2 - консольный ввод" << std::endl << "3 - файловый ввод" << std::endl;

inputChoice input = static\_cast<inputChoice>(CheckMenu(three));

return input;

}

binaryChoice OutputMenu()

{

std::cout << "Сохранить исходные данные в файле:" << std::endl << "1 - нет" << std::endl << "2 - да" << std::endl;

binaryChoice output = static\_cast<binaryChoice>(CheckMenu(two));

return output;

}

binaryChoice EnterListMenu()

{

std::cout << "Рассмотреть список рейсов:" << std::endl << "1 - нет" << std::endl << "2 - да" << std::endl;

binaryChoice list = static\_cast<binaryChoice>(CheckMenu(two));

return list;

}

listChoice ListMenu()

{

std::cout << "Вывести список:" << std::endl << "1 - список по пункту назначения" << std::endl << "2 - список по дню недели" << std::endl << "3 - список по дню недели и времен" << std::endl;

listChoice list = static\_cast<listChoice>(CheckMenu(three));

return list;

}

binaryChoice EndProgramMenu()

{

std::cout << "Закончить программу" << std::endl << "1 - нет" << std::endl << "2 - да" << std::endl;

binaryChoice end\_program = static\_cast<binaryChoice>(CheckMenu(two));

return end\_program;

}

**[---Конец Menu.cpp]**

**[ Начало UnitTest.cpp--- ]**

#include "pch.h"

#include "CppUnitTest.h"

#include "../Project1/Check.cpp"

#include "../Project1/Algorithm.cpp"

#include "../Project1/Flight.h"

#include "../Project1/Flight.cpp"

#include "../Project1/Items.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest)

{

private:

Flight temp\_start[5] =

{

Flight({ 9, 5 }, week\_day[2], plane\_type[0], "Москва"),

Flight({ 10, 20 }, week\_day[2], plane\_type[2], "Москва"),

Flight({ 14, 0 }, week\_day[5], plane\_type[1], "Минск"),

Flight({ 20, 30 }, week\_day[2], plane\_type[2], "Новосибирск"),

Flight({ 6, 0 }, week\_day[5], plane\_type[0], "Волгоград")

};

dynamic\_array start\_list = { 5, true, temp\_start };

Flight temp\_true1[2]

{

Flight({ 9, 5 }, week\_day[2], plane\_type[0], "Москва"),

Flight({ 10, 20 }, week\_day[2], plane\_type[2], "Москва")

};

dynamic\_array test1\_true\_list = { 2, true, temp\_true1 };

Flight temp\_true2[2]

{

Flight({ 10, 20 }, week\_day[2], plane\_type[2], "Москва"),

Flight({ 20, 30 }, week\_day[2], plane\_type[2], "Новосибирск")

};

dynamic\_array test2\_true\_list = { 2, true, temp\_true2 };

Flight temp\_true3[2]

{

Flight({ 14, 0 }, week\_day[5], plane\_type[1], "Минск"),

Flight({ 6, 0 }, week\_day[5], plane\_type[0], "Волгоград")

};

dynamic\_array test3\_true\_list = { 2, true, temp\_true3 };

public:

bool equivalentFlightList(Flight\* list1, Flight\* list2, int size)

{

for (int i = 0; i < size; i++) {

if (list1[i] != list2[i]) {

return false;

}

}

return true;

}

TEST\_METHOD(test1)

{

dynamic\_array end\_list\_test = { 0, true, nullptr };

end\_list\_test = GetDestinationList(start\_list, "Москва");

Assert::IsTrue(equivalentFlightList(test1\_true\_list.ptr, end\_list\_test.ptr, 2));

}

TEST\_METHOD(test2)

{

dynamic\_array end\_list\_test = { 0, true, nullptr };

end\_list\_test = GetWeekdayAndTimeList(start\_list, { 10, 10 }, 2);

Assert::IsTrue(equivalentFlightList(test2\_true\_list.ptr, end\_list\_test.ptr, 2));

}

TEST\_METHOD(test3)

{

dynamic\_array end\_list\_test = { 0, true, nullptr };

end\_list\_test = GetWeekdayList(start\_list, 5);

Assert::IsTrue(equivalentFlightList(test3\_true\_list.ptr, end\_list\_test.ptr, 2));

}

};

};

**[---Конец UnitTest.cpp]**