Почернин В. С. 3530904/00004

Задание 9. ООП. Вариант 11

1. **Общая постановка задачи**
    1. В соответствии с выбранным вариантом задания требуется разработать класс (классы).
    2. Для разрабатываемого класса необходимо выполнить:
        2.1 . Описание модулей класса.
        2.2 . Конструктор без параметров, конструктор с параметрами.
        2.3 . Конструктор копирования, оператор присваивания, деструктор.
        2.4 . Перегрузку бинарных операторов (всех, которые имеют смысл).
        2.5 . Перегрузку операторов отношения.
        2.6 . Перегрузку операторов инкремента и декремента.
        2.7 . Перегрузку операторов << и >> через дружественные функции.
    3. Создать массив из объектов разработанного класса. Ввести в массив данные из текстового файла.
    4. Написать функцию, находящую «максимальный/минимальный» объект из неупорядоченного массива. Для сравнения использовать перегруженный оператор отношения.
    5. Написать шаблон функции, выполняющей сортировку массива по возрастанию (метод сортировки указан варианте задания). Применить функцию к массиву объектов. Результат вывести в файл в виде таблицы (с выводом заголовков столбцов).
    6. **Создать шаблон класса связного списка. Метод добавления должен позволить сформировать упорядоченный по заданному полю(ключу) связный список. Правило упорядочивания и поле ключа должны быть согласованы с преподавателем.** Создать связный список объектов. Вывести список в файл в виде таблицы (с выводом заголовков столбцов). **Рекомендуется использование стандартных контейнеров STL.**
    7. Предусмотреть генерацию и обработку исключительных ситуаций, связанных, например, с проверкой значения полей перед инициализацией и присваиванием. Для исключений должны быть написаны отдельные классы. **Требования к правилам формирования отдельных полей и к использованию библиотечных классов, а также спецификация программы должны быть должны быть согласованы с преподавателем.**
    8. Программа должна быть написана в соответствии со стилем программирования: **C++ Programming Style Guidelines (http://geosoft.no/development/cppstyle.html).**
    9. **Отчет по лабораторной работе должен содержать:**
        9.1 Общая постановка задачи.
        9.2 Требования.
        9.3 Спецификация.
        9.4 Тест план.
        9.5 Программа.
        9.6 Итоги.
    10. Вариант 11: разработать класс «Маршрут», содержащий информацию:
        - Название начального пункта маршрута.
        - Название конечного пункта маршрута.
        - Номер маршрута.
            - В пункте 4 сравнивать объекты по номеру маршрута.
            - В пункте 5 сортировку массива выполнять методом простых вставок.

- В пункте 6 составить список названий конечных пунктов маршрутов и числа маршрутов, ведущих в них. Вывести в порядке убывания числа маршрутов.

11. Для дополнительных баллов:
    11.1 Создать собственный класс строк.
    11.2 Создать собственный шаблон класса динамического массива.

2. Требования

1. Выходной файл создается в корне программы с названием output.txt.
2. Входной файл должен находиться в корне программы и быть назван input.txt.
3. Структура входного файла должна быть следующей:
   - На первой строчке файла должно быть записано одно целое положительное число N – количество считываемых маршрутов. (В итоге, в файле должно быть ровно N + 1 строк)
   - На следующих N строках должны быть описаны маршруты в следующем формате: «Номер маршрута»(целое положительное число) «Название начального пункта маршрута»(строка) «Название конечного пункта маршрута»(строка).
   - Названия начального и конечного пунктов маршрута должны быть записаны кириллицей с большой буквы. В середине названия допускается наличия символов «-».
4. При правильных входных данных должен осуществляться вывод данных в файл output.txt. Вывод состоит из четырех сущностей.

3. Спецификация

1. Выходной файл создается в корне программы с названием output.txt.
   1.1 В случае, если выходной файл не может быть создан или открыт, сообщение «Ошибка файла: output.txt не открыт», завершение программы с кодом -1.
2. Входной файл должен находиться в корне программы и быть назван input.txt.
   2.1 В случае, если входной файл отсутствует или назван по-другому, сообщение «Ошибка файла: input.txt не открыт», завершение программы с кодом -1.
3. Структура входного файла должна быть следующей:

   - На первой строчке файла должно быть записано одно целое положительное число N – количество считываемых маршрутов. (В итоге, в файле должно быть ровно N + 1 строк)
   - На следующих N строках должны быть описаны маршруты в следующем формате: «Номер маршрута»(целое положительное число) «Название начального пункта маршрута»(строка) «Название конечного пункта маршрута»(строка).
   - Названия начального и конечного пунктов маршрута должны быть записаны кириллицей с большой буквы. В середине названия допускается наличия символов «-».

   3.1 В случае, если на первой строке нет целого положительного числа или оно задано некорректно, сообщение «Ошибка ввода: Некорректная запись количества строк», завершение программы с кодом 1.
   3.2 В случае, если в файле оказывается не N+1 строк, сообщение «Ошибка ввода: Некорректная структура файла», завершение программы с кодом 1.
   3.3 В случае, если какая-либо из строчек, задающих маршруты, задана некорректно (например, имеет больше трёх элементов), сообщение «Ошибка ввода: Некорректная структура файла», завершение программы с кодом 1.
   3.4 В случае, если один из номеров маршрута задан некорректно, сообщение «Ошибка ввода: Некорректный номер маршрута», завершение программы с кодом 1.

3.5 В случае, если начало маршрута в какой-либо строке задано некорректно, сообщение «Ошибка ввода: Некорректное название начала маршрута», завершение программы с кодом 1.

3.6 В случае, если конец маршрута в какой-либо строке задан некорректно, сообщение «Ошибка ввода: Некорректное название конца маршрута», завершение программы с кодом 1.

4. При правильных входных данных должен осуществляться вывод данных в файл output.txt. Вывод состоит из четырех сущностей:

4.1 В случае, если входной файл задан правильно, выводится первая сущность – изначальный массив маршрутов (таблицей).

4.2 В случае, если входной файл задан правильно, выводится вторая сущность – максимальный и минимальный номера маршрутов.

4.3 В случае, если входной файл задан правильно, выводится третья сущность – отсортированный по возрастанию номеров маршрута массив маршрутов (таблицей).

4.4 В случае, если входной файл задан правильно, выводится четвертая сущность – список названий конечных пунктов маршрутов и числа маршрутов, ведущих в них в порядке убывания числа маршрутов (таблицей).

4. Тест план

Номер теста	Спецификация	Данные	Ожидаемый результат
1.	1.1	Выходной файл output.txt уже создан, но поставлена галочка «только на чтение»	Сообщение «Ошибка файла: output.txt не открыт», завершение программы с кодом -1.
2.	2.1	Входной файл назван inpweup.txt (с ошибкой)	Сообщение «Ошибка файла: input.txt не открыт», завершение программы с кодом -1.

3.	3.1	`A числа-то нет`
		`8 Хасанская Ладожская`
		`23 Ленская Коммуны`
		`128 Купчино Вязники`
			Сообщение «Ошибка ввода: Некорректная запись количества строк», завершение программы с кодом 1.

4.	3.2	`3`
		`8 Хасанская Ладожская`
		`23 Ленская Коммуны`
		`128 Купчино Вязники`
		`100 Лишняя Строка`
			Сообщение «Ошибка ввода: Некорректная структура файла», завершение программы с кодом 1.

5.	3.2	`3`
		`8 Хасанская Ладожская`
		`23 Строк Недостает`
			Сообщение «Ошибка ввода: Некорректная структура файла», завершение программы с кодом 1.

6.	3.3	`20`
		`8 Хасанская Ладожская`
		`23 Ленская Коммуны`
		`128 Купчино Вязники a б`
			Сообщение «Ошибка ввода: Некорректная структура файла», завершение программы с кодом 1.

7.	3.4	`3`
		`8 Хасанская Ладожская`
		`23 Ленская Коммуны`
		`1ошибка1 Купчино Вязники`
			Сообщение «Ошибка ввода: Некорректный номер маршрута», завершение программы с кодом 1.

8.	3.5	`3`
		`8 Хасанская Ладожская`
		`23 Ленская Коммуны`
		`128 Ку%пчино Вязники`
			Сообщение «Ошибка ввода: Некорректное название начала маршрута», завершение программы с кодом 1.

9.	3.6	`3`
		`8 Хасанская Ладожская`
		`23 Ленская Коммуны`
		`128 Купчино Вязники???`
			Сообщение «Ошибка ввода: Некорректное название конца маршрута», завершение программы с кодом 1.

<table>
<tr>
<td>10.</td>
<td>4.1</td>
<td>20
8 Хасанская Ладожская
23 Ленская Коммуны
128 Купчино Вязники
32 Пушкино Ладожская
1223 Колпино Вязники
921 Владимирский Северный
112 Большевиков Дыбенко
1 Питер Москва
96 Новосибирск Ладожская
55 Москва Северный
655 Коммуны Ладожская
484 Косыгина Вязники
857 Владивосток Москва
686 Крестовский Ладожская
522 Купчино Гражданский
614 Садовая Ладожская
368 Ладожская Комендантский
379 Гражданский Косыгина
807 Кудрово Мурино
443 Ириновский Ладожская</td>
<td>Вывод первой сущности – изначального массива маршрута (таблицей):
Изначальный массив:

|Номер маршрута|Начало маршрута|Конец маршрута|
8	Хасанская	Ладожская
23	Ленская	Коммуны
128	Купчино	Вязники
32	Пушкино	Ладожская
1223	Колпино	Вязники
921	Владимирский	Северный
112	Большевиков	Дыбенко
1	Питер	Москва
96	Новосибирск	Ладожская
55	Москва	Северный
655	Коммуны	Ладожская
484	Косыгина	Вязники
857	Владивосток	Москва
686	Крестовский	Ладожская
522	Купчино	Гражданский
614	Садовая	Ладожская
368	Ладожская	Комендантский
379	Гражданский	Косыгина
807	Кудрово	Мурино
443	Ириновский	Ладожская	 ---</td>
</tr>
<tr>
<td>11.</td>
<td>4.2</td>
<td>20
8 Хасанская Ладожская
23 Ленская Коммуны
128 Купчино Вязники
32 Пушкино Ладожская
1223 Колпино Вязники
921 Владимирский Северный
112 Большевиков Дыбенко
1 Питер Москва
96 Новосибирск Ладожская
55 Москва Северный
655 Коммуны Ладожская
484 Косыгина Вязники
857 Владивосток Москва
686 Крестовский Ладожская
522 Купчино Гражданский
614 Садовая Ладожская
368 Ладожская Комендантский
379 Гражданский Косыгина
807 Кудрово Мурино
443 Ириновский Ладожская</td>
<td>Вывод второй сущности: максимального и минимального номеров маршрута:
Максимальный номер маршрута: 1223
Минимальный номер маршрута: 1</td>
</tr>
</table>

12.	4.3	20 8 Хасанская Ладожская 23 Ленская Коммуны 128 Купчино Вязники 32 Пушкино Ладожская 1223 Колпино Вязники 921 Владимирский Северный 112 Большевиков Дыбенко 1 Питер Москва 96 Новосибирск Ладожская 55 Москва Северный 655 Коммуны Ладожская 484 Косыгина Вязники 857 Владивосток Москва 686 Крестовский Ладожская 522 Купчино Гражданский 614 Садовая Ладожская 368 Ладожская Комендантский 379 Гражданский Косыгина 807 Кудрово Мурино 443 Ириновский Ладожская	Вывод третьей сущности – отсортированного по возрастанию номеров маршрута массива (таблицей): Массив после сортировки: --- |Номер маршрута|Начало маршрута|Конец маршрута| | 1| Питер| Москва| | 8| Хасанская| Ладожская| | 23| Ленская| Коммуны| | 32| Пушкино| Ладожская| | 55| Москва| Северный| | 96| Новосибирск| Ладожская| | 112| Большевиков| Дыбенко| | 128| Купчино| Вязники| | 368| Ладожская| Комендантский| | 379| Гражданский| Косыгина| | 443| Ириновский| Ладожская| | 484| Косыгина| Вязники| | 522| Купчино| Гражданский| | 614| Садовая| Ладожская| | 655| Коммуны| Ладожская| | 686| Крестовский| Ладожская| | 807| Кудрово| Мурино| | 857| Владивосток| Москва| | 921| Владимирский| Северный| | 1223| Колпино| Вязники| ---
13.	4.4	20 8 Хасанская Ладожская 23 Ленская Коммуны 128 Купчино Вязники 32 Пушкино Ладожская 1223 Колпино Вязники 921 Владимирский Северный 112 Большевиков Дыбенко 1 Питер Москва 96 Новосибирск Ладожская 55 Москва Северный 655 Коммуны Ладожская 484 Косыгина Вязники 857 Владивосток Москва 686 Крестовский Ладожская 522 Купчино Гражданский 614 Садовая Ладожская 368 Ладожская Комендантский 379 Гражданский Косыгина 807 Кудрово Мурино 443 Ириновский Ладожская	Вывод четвертой сущности – списка названий конечных пунктов маршрутов и числа маршрутов, ведущих в них в порядке убывания числа маршрутов (таблицей): Список названий конечных пунктов маршрутов и числа маршрутов, ведущих в них в порядке убывания числа маршрутов: --- |Конец маршрута|Число машрутов| | Ладожская| 7| | Вязники| 3| | Москва| 2| | Северный| 2| | Коммуны| 1| | Дыбенко| 1| | Комендантский| 1| | Косыгина| 1| | Гражданский| 1| | Мурино| 1| ---

5. Программа

OOP.cpp

```
// Почернин Владислав Сергеевич.
// Вариант 11.

#include <iostream>
#include <streams>
#include <vector>

#include "Route.h"
#include "Functions.h"
#include "MyExceptions.h"
#include "MyString.h"
#include "MyArray.h"
#include "ExceptionNames.h"

int main()
```

```
{
  setlocale(LC_ALL, "Russian");
  MyString inputFileName("input.txt");
  MyString outputFileName("output.txt");
  std::ofstream fout;
  try
  {
    fout.open(outputFileName.get());
    if (!fout)
    {
      throw (outputFileName + MyString(ERROR_FILE_IS_NOT_OPEN));
    }
  }
  catch (const MyString& error)
  {
    std::cout << ERROR_WITH_FILE << error.get();
    return -1;
  }
  MyArray<Route> routes;
  try
  {
    fillMyArrayByFile(routes, inputFileName);
  }
  catch (const MyString& error)
  {
    std::cout << ERROR_WITH_FILE << error.get();
    return -1;
  }
  catch (InvalidInput& ex)
  {
    std::cout << ERROR_WITH_INPUT << ex.what();
    return 1;
  }

  fout << "Изначальный массив:" << std::endl;
  showRouteArray(routes, fout);

  fout << std::endl << "Максимальный номер маршрута: " << getMaxRoute(routes).getNumber();
  fout << std::endl << "Минимальный номер маршрута: " << getMinRoute(routes).getNumber() <<
std::endl << std::endl;

  sortMyArray(routes);
  fout << "Массив после сортировки:" << std::endl;
  showRouteArray(routes, fout);

  std::vector<std::pair<MyString, int>> pairs;
  setPairs(pairs, routes);
  sortPairs(pairs);
  fout << std::endl << "Список названий конечных пунктов маршрутов и числа маршрутов, ведущих в них
в порядке убывания числа маршрутов:" << std::endl;
  showPairs(pairs, fout);

  fout.close();

  return 0;
}
```

Route.h

```
#ifndef ROUTE
#define ROUTE

#include <iostream>
#include "MyString.h"

class Route
{
private:
```

```
  MyString start_; // Название начального пункта маршрута.
  MyString finish_; // Название конечного пункта маршрута.
  int number_; // Номер маршрута.
public:
#ifdef _MSC_VER
#pragma region Конструкторы и деструктор.
#endif
  // Конструктор без параметров.
  Route();

  // Конструктор с параметрами.
  Route(const MyString& start, const MyString& finish, int number);

  // Конструктор копирования.
  Route(const Route& route);

  // Деструктор.
  ~Route();
#ifdef _MSC_VER
#pragma endregion
#endif

#ifdef _MSC_VER
#pragma region Геттеры и сеттеры.
#endif
  // Получить название начального пункта маршрута.
  MyString getStart() const;

  // Получить название конечного пункта маршрута.
  MyString getFinish() const;

  // Получить номер маршрута.
  int getNumber() const;

  // Установить название начального пункта маршрута.
  void setStart(const MyString& start);

  // Установить название конечного пункта маршрута.
  void setFinish(const MyString& finish);

  // Установить номер маршрута.
  void setNumber(int number);

  // Превратить маршрут в маршрут в парк.
  void toThePark();
#ifdef _MSC_VER
#pragma endregion
#endif

#ifdef _MSC_VER
#pragma region Перегрузка бинарных операторов (всех, которые имеют смысл).
#endif
  // Перегрузка оператора присваивания.
  Route operator= (const Route& route2);

  // Перегрузка опреатора равенства.
  bool operator==(const Route& route2) const;

  // Перегрузка опреатора неравенства.
  bool operator!=(const Route& route2) const;

  // Перегрузка опретора больше.
  bool operator>(const Route& route2) const;

  // Перегрузка опретора меньше.
  bool operator<(const Route& route2) const;

  // Перегрузка опретора больше или равно.
  bool operator>=(const Route& route2) const;
```

```
  // Перегрузка опретора меньше или равно.
  bool operator<=(const Route& route2) const;
#ifdef _MSC_VER
#pragma endregion
#endif

#ifdef _MSC_VER
#pragma region Перегрузка ++ -- и потока.
#endif
  // Перегрузка оператора префиксного инкремента.
  Route operator++();
  // Перегрузка оператора постфиксного инкремента.
  Route operator++(int notused);

  // Перегрузка оператора префиксного декремента.
  Route operator--();
  // Перегрузка оператора постфиксного декремента.
  Route operator--(int notused);

  // Перегрузка оператора <<.
  friend std::ostream& operator<< (std::ostream& out, const Route& route);

  // Перегрузка оператора >>.
  friend std::istream& operator>> (std::istream& in, Route& route);
#ifdef _MSC_VER
#pragma endregion
#endif
};

#endif
```

Route.cpp

```
#include <iomanip>

#include "Route.h"
#include "MyExceptions.h"
#include "Functions.h"
#include "ExceptionNames.h"

#ifdef _MSC_VER
#pragma region Конструкторы и деструктор.
#endif
// Конструктор без параметров.
Route::Route() : start_(""), finish_("В парк"), number_(0) {}

// Конструктор с параметрами.
Route::Route(const MyString& start, const MyString& finish, int number) : start_(start),
finish_(finish), number_(number)
{
  if (number_ <= 0)
  {
    throw InvalidRoute(ERROR_INCORRECT_ROUTE_NUMBER);
  }
  if ((start_ == "") || !isCorrectRouteName(start_) || (finish_ == "") ||
!isCorrectRouteName(finish_))
  {
    throw InvalidRoute(ERROR_INCORRECT_ROUTE_POINTS_NAME);
  }
}

// Конструктор копирования.
Route::Route(const Route& route) : start_(route.start_), finish_(route.finish_),
number_(route.number_)
{
  if (number_ <= 0)
  {
```

```
    throw InvalidRoute(ERROR_INCORRECT_ROUTE_NUMBER_WHEN_COPYING);
  }
  if ((start_ == "") || !isCorrectRouteName(start_) || (finish_ == "") ||
!isCorrectRouteName(finish_))
  {
    throw InvalidRoute(ERROR_INCORRECT_ROUTE_POINTS_NAME_WHEN_COPYING);
  }
}

Route::~Route() {}
#ifdef _MSC_VER
#pragma endregion
#endif

#ifdef _MSC_VER
#pragma region Геттеры и сеттеры.
#endif
// Получить название начального пункта маршрута.
MyString Route::getStart() const
{
  return start_;
}

// Получить название конечного пункта маршрута.
MyString Route::getFinish() const
{
  return finish_;
}

// Получить номер маршрута.
int Route::getNumber() const
{
  return number_;
}

// Установить название начального пункта маршрута.
void Route::setStart(const MyString& start)
{
  start_ = start;
  if ((start_ == "") || !isCorrectRouteName(start_))
  {
    throw(InvalidRoute(ERROR_INCORRECT_ROUTE_START_NAME));
  }
}

// Установить название конечного пункта маршрута.
void Route::setFinish(const MyString& finish)
{
  finish_ = finish;
  if ((finish_ == "") || !isCorrectRouteName(finish_))
  {
    throw(InvalidRoute(ERROR_INCORRECT_ROUTE_FINISH_NAME));
  }
}

// Установить номер маршрута.
void Route::setNumber(int number)
{
  number_ = number;
  if (number_ <= 0)
  {
    throw(InvalidRoute(ERROR_INCORRECT_ROUTE_NUMBER));
  }
}

// Превратить маршрут в маршрут в парк.
void Route::toThePark()
{
  start_ = "";
```

```
  finish_ = "В парк";
  number_ = 0;
}
#ifdef _MSC_VER
#pragma endregion
#endif


#ifdef _MSC_VER
#pragma region Перегрузка бинарных операторов (всех, которые имеют смысл).
#endif
// Перегрузка оператора присваивания.
Route Route::operator= (const Route& route2)
{
  if (this == &route2)
  {
    return *this;
  }

  start_ = route2.start_;
  finish_ = route2.finish_;
  number_ = route2.number_;

  if (number_ <= 0)
  {
    throw InvalidRoute(ERROR_INCORRECT_ROUTE_NUMBER_WHEN_ASSIGN);
  }
  if ((start_ == "") || !isCorrectRouteName(start_) || (finish_ == "") ||
!isCorrectRouteName(finish_))""))
  {
    throw InvalidRoute(ERROR_INCORRECT_ROUTE_POINTS_NAME_WHEN_ASSIGN);
  }

  return *this;
}

// Перегрузка опреатора равенства.
bool Route::operator==(const Route& route2) const
{
  return ((start_ == route2.start_) && (finish_ == route2.finish_) && (number_ == route2.number_));
}

// Перегрузка опреатора неравенства.
bool Route::operator!=(const Route& route2) const
{
  return ((start_ != route2.start_) || (finish_ != route2.finish_) || (number_ != route2.number_));
}

// Перегрузка опретора больше.
bool Route::operator>(const Route& route2) const
{
  return (number_ > route2.number_);
}

// Перегрузка опретора меньше.
bool Route::operator<(const Route& route2) const
{
  return (number_ < route2.number_);
}

// Перегрузка опретора больше или равно.
bool Route::operator>=(const Route& route2) const
{
  return (number_ >= route2.number_);
}

// Перегрузка опретора меньше или равно.
bool Route::operator<=(const Route& route2) const
{
```

```
  return (number_ <= route2.number_);
}
#ifdef _MSC_VER
#pragma endregion
#endif

#ifdef _MSC_VER
#pragma region Перегрузка ++ -- и потока.
#endif
// Перегрузка оператора префиксного инкремента.
Route Route::operator++()
{
  number_ += 1;
  return *this;
}

// Перегрузка оператора постфиксного инкремента.
Route Route::operator++(int notused)
{
  Route temp = Route(*this);
  number_ += 1;
  return temp;
}

// Перегрузка оператора префиксного декремента.
Route Route::operator--()
{
  number_ -= 1;
  return *this;
}

// Перегрузка оператора постфиксного декремента.
Route Route::operator--(int notused)
{
  Route temp = Route(*this);
  number_ -= 1;
  return temp;
}

// Перегрузка оператора <<.
std::ostream& operator<< (std::ostream& out, const Route& route)
{
  return (out << '|' << std::setw(14) << route.number_ << '|' << std::setw(15) << route.start_ <<
'|' << std::setw(14) << route.finish_ << '|');
}

std::istream& operator>> (std::istream& in, Route& route)
{
  size_t number;
  MyString start;
  MyString finish;
  in >> number;
  if (!in || ((in.peek() != ' ') && in.peek() != '\n'))
  {
    throw (InvalidInput(ERROR_INCORRECT_ROUTE_NUMBER));
  }
  in >> start;
  if (!in || !isCorrectRouteName(start))
  {
    throw (InvalidInput(ERROR_INCORRECT_ROUTE_START_NAME));
  }
  in >> finish;
  if ((!in.eof() && !in) || !isCorrectRouteName(finish))
  {
    throw (InvalidInput(ERROR_INCORRECT_ROUTE_FINISH_NAME));
  }
  route.number_ = number;
  route.start_ = start;
  route.finish_ = finish;
```

```
  return in;
}
#ifdef _MSC_VER
#pragma endregion
#endif
```

MyString.h

```
#ifndef MY_STRING
#define MY_STRING

#include <iostream>

class MyString
{
public:
  // Конструктор без параметров.
  MyString();

  // Конструктор с параметром (из обычной строки).
  MyString(const char* string);

  // Консутрктор копирования.
  MyString(const MyString& myString);

  // Деструктор.
  ~MyString();

  // Получить строку.
  char* get() const;

  // Получить размер строки.
  size_t getSize() const;

  // Перегрузка оператора =.
  MyString& operator=(const MyString& myString2);

  // Перегрузка оператора +.
  MyString operator+(const MyString& myString2) const;

  // Перегрузка опретора ==.
  bool operator==(const MyString& myString2) const;

  // Перегрузка опретора !=.
  bool operator!=(const MyString& myString2) const;

  // Перегрузка оператора [].
  char& operator[](size_t index);

  // Перегрузка оператора <<.
  friend std::ostream& operator<< (std::ostream& out, const MyString& myString);

  // Перегрузка оператора >>.
  friend std::istream& operator>> (std::istream& in, MyString& myString);
private:
  char* string_;
  size_t size_;
};


#endif
```

MyString.cpp

```
#include <windows.h>
#include <exception>

#include "Functions.h"
```

```cpp
#include "ExceptionNames.h"
#include "MyString.h"

// Конструктор без параметров.
MyString::MyString() : string_(nullptr), size_(0) {}

// Конструктор с параметром (из обычной строки).
MyString::MyString(const char* string)
{
  size_t size = myStrLen(string);
  string_ = new char[size + 1];
  size_ = size;

  for (size_t i = 0; i < size; i++)
  {
    string_[i] = string[i];
  }

  string_[size] = '\0';
}

// Консутрктор копирования.
MyString::MyString(const MyString& myString)
{
  size_t size = myString.size_;
  string_ = new char[size + 1];
  size_ = size;

  for (size_t i = 0; i < size; i++)
  {
    string_[i] = myString.string_[i];
  }

  string_[size] = '\0';
}

// Деструктор.
MyString::~MyString()
{
  delete[] string_;
}

// Получить строку.
char* MyString::get() const
{
  return string_;
}

// Получить размер строки.
size_t MyString::getSize() const
{
  return size_;
}

// Перегрузка оператора =.
MyString& MyString::operator=(const MyString& myString2)
{
  if (this == &myString2)
  {
    return *this;
  }

  // Очистка памяти.
  if (string_ != nullptr)
  {
    delete[] string_;
  }

  size_ = myString2.size_;
```

```
  string_ = new char[size_ + 1];

  for (size_t i = 0; i < size_; i++)
  {
    string_[i] = myString2.string_[i];
  }

  string_[size_] = '\0';

  return *this;
}

// Перегрузка оператора +.
MyString MyString::operator+(const MyString& myString2) const
{
  size_t size1 = size_;
  size_t size2 = myString2.size_;
  size_t newSize = size1 + size2;

  MyString newString;
  newString.size_ = newSize;
  newString.string_ = new char[newSize + 1];
  size_t i = 0;
  for (size_t j = 0; j < size1; j++)
  {
    newString.string_[i] = string_[i];
    i++;
  }
  for (size_t j = 0; (j < size2) && (i < newSize) ; j++)
  {
    newString.string_[i] = myString2.string_[j];
    i++;
  }
  newString.string_[newSize] = '\0';

  return newString;
}

// Перегрузка опретора ==.
bool MyString::operator==(const MyString& myString2) const
{
  if (size_ != myString2.size_)
  {
    return false;
  }

  for (size_t i = 0; i < size_; i++)
  {
    if (string_[i] != myString2.string_[i])
    {
      return false;
    }
  }

  return true;
}

// Перегрузка опретора !=.
bool MyString::operator!=(const MyString& myString2) const
{
  if (size_ != myString2.size_)
  {
    return true;
  }

  for (size_t i = 0; i < size_; i++)
  {
    if (string_[i] != myString2.string_[i])
    {
```

```
      return true;
    }
  }

  return false;
}

// Перегрузка оператора [].
char& MyString::operator[](size_t index)
{
  if ((index < 0) || (index >= size_))
  {
    throw (std::out_of_range(ERROR_INCORRECT_INDEX_OUT_OF_RANGE));
  }

  return string_[index];
}

// Перегрузка оператора <<.
std::ostream& operator<< (std::ostream& out, const MyString& myString)
{
  return (out << myString.string_);
}

// Перегрузка оператора >>.
std::istream& operator>> (std::istream& in, MyString& myString)
{
  SetConsoleCP(1251);
  SetConsoleOutputCP(1251);

  char* input = new char[256]; // Выделяем считывающей строке оверсайз памяти.
  char symbol = ' ';
  while ((!in.eof()) && ((in.peek() == ' ') || (in.peek() == '\n'))) // Пока впереди есть
разделители - убираем их из потока.
  {
    symbol = in.get();
  }
  symbol = in.get(); // Будем посимвольно считывать сюда.
  size_t i = 0;
  while ((symbol != ' ') && (symbol != '\n')) // Пока мы не считали разделитель.
  {
    input[i] = symbol; // Записываем очередной символ в считывающую строку.
    if (in.eof()) // Чтобы последний символ последней строки не считывался бесконечно.
    {
      break;
    }
    symbol = in.get(); // И считываем очередной символ.
    i++;
  }
  input[i] = '\0'; // После того, как дошли до разделителя, записываем конечный ноль.

  while ((!in.eof()) && ((in.peek() == ' ') || (in.peek() == '\n'))) // Пока впереди есть
разделители - убираем их из потока.  (Конец потока ввода - Ctrl+Z!!!)
  {
    symbol = in.get();
  }


  delete[] myString.string_;
  myString.size_ = myStrLen(input);
  myString.string_ = new char[myString.size_ + 1];
  for (size_t i = 0; i < myString.size_; i++)
  {
    myString.string_[i] = input[i];
  }
  myString.string_[myString.size_] = '\0';

  delete[] input;
  return in;
```

```
}
```

MyArray.h

```
#ifndef MY_ARRAY
#define MY_ARRAY

#include <exception>

#include "ExceptionNames.h"

template <typename Data>
class MyArray
{
public:
  // Конструктор без параметров.
  MyArray() : size_(0), data_(nullptr) {}

  // Конструктор с параметрами.
  MyArray(size_t size) : size_(size), data_(new Data[size]) {}

  // Конструктор копирования.
  MyArray(const MyArray<Data>& array2);

  // Конструктор копирования с переносом.
  MyArray(MyArray<Data>&& array2) noexcept;

  // Деструктор.
  ~MyArray()
  {
    delete[] data_;
  }

  // Получить размер массива.
  size_t getSize() const;

  // Заново выделить память у массива (при этом данные пропадают).
  void reAllocate(size_t size);

  // Метод добавления элемента в массив.
  void pushBack(const Data& object);

  // Перегрузка оператора [].
  Data& operator[](size_t index) const;

  // Перегрузка оператора =.
  MyArray<Data>& operator=(const MyArray<Data>& array2);

  // Перегрузка оператора +.
  MyArray<Data> operator+(const MyArray<Data>& array2);

  // Перегрузка оператора = с переносом.
  MyArray<Data>& operator=(MyArray<Data>&& array2) noexcept;
private:
  size_t size_;
  Data* data_;
};

// Конструктор копирования.
template <typename Data>
MyArray<Data>::MyArray(const MyArray<Data>& array2)
{
  if (array2.size_ == 0)
  {
    size_ = 0;
    data_ = nullptr;
  }
  else
```

```
  {
    size_ = array2.size_;
    data_ = new Data[array2.size_];
    for (size_t i = 0; i < size_; i++)
    {
      data_[i] = array2.data_[i];
    }
  }
}

// Конструктор копирования с переносом.
template <typename Data>
MyArray<Data>::MyArray(MyArray<Data>&& array2) noexcept
{
  size_ = array2.size_;
  data_ = array2.data_;
  array2.size_ = 0;
  array2.data_ = nullptr;
}

// Перегрузка оператора [].
template <typename Data>
Data& MyArray<Data>::operator[](size_t index) const
{
  if (index >= size_)
  {
    throw (std::out_of_range(ERROR_INCORRECT_INDEX_OUT_OF_RANGE));
  }

  return data_[index];
}

// Получить размер массива.
template <typename Data>
size_t MyArray<Data>::getSize() const
{
  return size_;
}

// Заново выделить память у массива (при этом данные пропадают).
template <typename Data>
void MyArray<Data>::reAllocate(size_t size)
{
  if (data_ != nullptr)
  {
    delete[] data_;
  }

  size_ = size;
  data_ = new Data[size];
}

// Метод добавления элемента в массив.
template <typename Data>
void MyArray<Data>::pushBack(const Data& object)
{
  Data* newData = new Data[size_ + 1];
  for (size_t i = 0; i < size_; i++)
  {
    newData[i] = data_[i];
  }
  newData[size_] = object;

  delete[] data_;
  data_ = newData;
  size_ = size_ + 1;
}

// Перегрузка оператора =.
```

```cpp
template <typename Data>
MyArray<Data>& MyArray<Data>::operator=(const MyArray<Data>& array2)
{
  delete[] data_;
  if (array2.size_ == 0)
  {
    size_ = 0;
    data_ = nullptr;
  }
  else
  {
    size_ = array2.size_;
    data_ = new Data[array2.size_];
    for (size_t i = 0; i < array2.size_; i++)
    {
      data_[i] = array2.data_[i];
    }
  }

  return *this;
}

// Перегрузка оператора +.
template <typename Data>
MyArray<Data> MyArray<Data>::operator+(const MyArray<Data>& array2)
{
  MyArray<Data> temp = MyArray<Data>(size_ + array2.size_);
  for (size_t i = 0; i < size_; i++)
  {
    temp.data_[i] = data_[i];
  }
  for (size_t i = 0; i < array2.size_; i++)
  {
    temp.data_[size_ + i] = array2.data_[i];
  }

  return temp;
}


// Перегрузка оператора = с переносом.
template <typename Data>
MyArray<Data>& MyArray<Data>::operator=(MyArray<Data>&& array2) noexcept
{
  if (this == &array2)
  {
    return *this;
  }
  delete[] data_;
  size_ = array2.size_;
  data_ = array2.data_;
  array2.size_ = 0;
  array2.data_ = nullptr;

  return *this;
}

#endif
```

Functions.h

```cpp
#ifndef FUNCTIONS
#define FUNCTIONS

#include <vector>
#include "Route.h"
#include "MyArray.h"
#include "MyString.h"
```

```
// Заполнить массив маршрутов из файла (MyArray массив)
void fillMyArrayByFile(MyArray<Route>& routes, const MyString& fileName);

// Вывести в поток вывода таблицу маршрутов.
void showRouteArray(const MyArray<Route>& routes, std::ostream& out);

// Получить максимальный по номеру маршрут из массива.
Route getMaxRoute(const MyArray<Route>& routes);

// Получить минимальный по номеру маршрут из массива.
Route getMinRoute(const MyArray<Route>& routes);

// Шаблон сортировки массива (мой шаблон массива).
template <typename T>
void sortMyArray(MyArray<T>& a)
{
  // Массив из 1 или 0 элементов уже отсортирован.
  if (a.getSize() <= 1)
  {
    return;
  }
  for (int i = 1; i < static_cast<int>(a.getSize()); i++) // Для каждого следующего
неотсортированного элемента найдем его место.
  {
    T temp = a[i];
    int j = 0;
    for (j = i - 1; (j >= 0) && a[j] > temp; j--) // Для элементов левее первого неотсортированного,
пока они больше его...
    {
      // Сдвигаем их вправо.
      a[j + 1] = a[j];
    }
    a[j + 1] = temp; // Ставим первый неотсортированный элемент на своё место.
  }
}

#ifdef _MSC_VER
#pragma region Вектор пар.
#endif
// Определить, есть ли в векторе пар элемент с ключем key. Если есть - положить его в переменную
index.
bool indexOfKey(const std::vector<std::pair<MyString, int>>& pairs, const MyString& key, size_t&
index);

// Сортировка вектора пар по значению.
void sortPairs(std::vector<std::pair<MyString, int>>& pairs);

// Заполнить вектор пар из массива MyArray.
void setPairs(std::vector<std::pair<MyString, int>>& pairs, const MyArray<Route>& routes);

// Вывести вектор пар в поток ввода.
void showPairs(const std::vector<std::pair<MyString, int>>& pairs, std::ostream& out);
#ifdef _MSC_VER
#pragma endregion
#endif

#ifdef _MSC_VER
#pragma region Функции для строк.
#endif
// Вычисление длины строки (массив char).
size_t myStrLen(const char* string);

// Определить, корректно ли задано название начала или конца маршрута?.
bool isCorrectRouteName(const MyString& string);

// Определение количества слов в строке.
int countOfWords(const char* string);
#ifdef _MSC_VER
```

```
#pragma endregion
#endif
#endif

// Проверить, корректная ли структура файла.
bool isCorrectFileStructure(const MyString& fileName, size_t count);
```

Functions.cpp

```
#include <fstream>
#include <iomanip>

#include "Functions.h"
#include "MyExceptions.h"
#include "ExceptionNames.h"

// Заполнить массив маршрутов из файла (MyArray массив)
void fillMyArrayByFile(MyArray<Route>& routes, const MyString& fileName)
{
  std::ifstream fin;
  fin.open(fileName.get());
  if (!fin)
  {
    throw (fileName + MyString(ERROR_FILE_IS_NOT_OPEN));
  }
  size_t nElements = 0;
  fin >> nElements;
  if (!fin || fin.peek() != '\n')
  {
    throw (InvalidInput(ERROR_INCORRECT_LINES_NUMBER));
  }
  if (!isCorrectFileStructure(fileName, nElements + 1))
  {
    throw (InvalidInput(ERROR_INCORRECT_FILE_STRUCTURE));
  }
  routes.reAllocate(nElements);
  size_t i = 0;
  while (!fin.eof() && i < nElements)
  {
    fin >> routes[i];
    i++;
  }
  fin.close();
}

// Вывести в поток вывода таблицу маршрутов.
void showRouteArray(const MyArray<Route>& routes, std::ostream& out)
{
  out << "-----------------------------------------------" << std::endl;
  out << "|Номер маршрута|Начало маршрута|Конец маршрута|" << std::endl;
  for (size_t i = 0; i < routes.getSize(); i++)
  {
    out << routes[i] << std::endl;
  }
  out << "-----------------------------------------------" << std::endl;
}

// Получить максимальный по номеру маршрут из массива.
Route getMaxRoute(const MyArray<Route>& routes)
{
  Route max = routes[0];
  for (size_t i = 0; i < routes.getSize(); i++)
  {
    if (routes[i] > max)
    {
      max = routes[i];
    }
  }
```

```
  return max;
}

// Получить минимальный по номеру маршрут из массива.
Route getMinRoute(const MyArray<Route>& routes)
{
  Route min = routes[0];
  for (size_t i = 0; i < routes.getSize(); i++)
  {
    if (routes[i] < min)
    {
      min = routes[i];
    }
  }
  return min;
}

#ifdef _MSC_VER
#pragma region Вектор пар.
#endif
// Определить, есть ли в векторе пар элемент с ключем key. Если есть - положить его в переменную
index.
bool indexOfKey(const std::vector<std::pair<MyString, int>>& pairs, const MyString& key, size_t&
index)
{
  for (size_t i = 0; i < pairs.size(); i++)
  {
    if (pairs[i].first == key)
    {
      index = i;
      return true;
    }
  }

  return false;
}

// Сортировка вектора пар по значению.
void sortPairs(std::vector<std::pair<MyString, int>>& pairs)
{
  // Массив из 1 или 0 элементов уже отсортирован.
  if (pairs.size() <= 1)
  {
    return;
  }
  for (int i = 1; i < static_cast<int>(pairs.size()); i++) // Для каждого следующего
неотсортированного элемента найдем его место.
  {
    std::pair<MyString, int> temp = pairs[i];
    int j = 0;
    for (j = i - 1; (j >= 0) && pairs[j].second < temp.second; j--) // Для элементов левее первого
неотсортированного, пока они больше его...
    {
      // Сдвигаем их вправо.
      pairs[j + 1] = pairs[j];
    }
    pairs[j + 1] = temp; // Ставим первый неотсортированный элемент на своё место.
  }
}

// Заполнить вектор пар из массива MyArray.
void setPairs(std::vector<std::pair<MyString, int>>& pairs, const MyArray<Route>& routes)
{
  size_t index = 0;
  for (size_t i = 0; i < routes.getSize(); i++)
  {
    if (indexOfKey(pairs, routes[i].getFinish(), index))
    {
      pairs[index].second++;
```

```
    }
    else
    {
      pairs.push_back(std::make_pair<MyString, int>(routes[i].getFinish(), 1));
    }
  }
}

// Вывести вектор пар в поток ввода.
void showPairs(const std::vector<std::pair<MyString, int>>& pairs, std::ostream& out)
{
  out << "------------------------------" << std::endl;
  out << "|Конец маршрута|Число машрутов|" << std::endl;
  for (size_t i = 0; i < pairs.size(); i++)
  {
    out << "|" << std::setw(14) << pairs[i].first << "|" << std::setw(14) << pairs[i].second << "|"
<< std::endl;
  }
  out << "------------------------------" << std::endl;
}
#ifdef _MSC_VER
#pragma endregion
#endif

#ifdef _MSC_VER
#pragma region Функции для строк.
#endif
// Вычисление длины строки (массив char).
size_t myStrLen(const char* string)
{
  size_t i = 0;
  while (string[i] != '\0')
  {
    i++;
  }
  return i;
}

// Определить, корректно ли задано название начала или конца маршрута?.
bool isCorrectRouteName(const MyString& string)
{
  if (string.getSize() < 2)
  {
    return false;
  }

  int ch = static_cast<int>(string.get()[0]);
  int leftBorder = static_cast<int>('A');
  int rightBorder = static_cast<int>('Я');
  if (((ch < leftBorder) || (ch > rightBorder)))
  {
    return false;
  }

  leftBorder = static_cast<int>('a');
  rightBorder = static_cast<int>('я');
  for (size_t i = 1; i < string.getSize(); i++)
  {
    ch = static_cast<int>(string.get()[i]);
    if (((ch < leftBorder) || (ch > rightBorder)) && (ch != static_cast<int>('-')))
    {
      return false;
    }
  }

  return true;
}

// Определение количества слов в строке.
```

```
int countOfWords(const char* string)
{
  int result = 0;
  bool isWord = false; // Проверка, записывается ли сейчас слово.
  for (size_t i = 0; i < myStrLen(string); i++)
  {
    if (string[i] == ' ')
    {
      isWord = false;
      continue;
    }
    if (!isWord)
    {
      result++;
      isWord = true;
    }
  }

  return result;
}
#ifdef _MSC_VER
#pragma endregion
#endif

// Проверить, корректная ли структура файла.
bool isCorrectFileStructure(const MyString& fileName, size_t countOfLines)
{
  std::ifstream fin;
  fin.open(fileName.get());
  if (fin.eof())
  {
    return false;
  }
  char* temp = new char[256];

  fin.getline(temp, 256);
  if (fin.eof() || (countOfWords(temp) > 1))
  {
    delete[] temp;
    return false;
  }

  for (size_t i = 0; i < countOfLines - 2; i++)
  {
    fin.getline(temp, 256);
    if (fin.eof() || (countOfWords(temp) != 3))
    {
      delete[] temp;
      return false;
    }
  }

  fin.getline(temp, 256);
  if (!fin.eof() || (countOfWords(temp) != 3))
  {
    delete[] temp;
    return false;
  }

  delete[] temp;
  return true;
}
```

MyExceptions.h

```
#ifndef MY_EXCEPTIONS
#define MY_EXCEPTIONS
```

```
// Классы исключений.

#include <exception>
#include "MyString.h"

class InvalidRoute : public std::exception
{
private:
  MyString errorMessage_;
public:
  InvalidRoute(MyString errorMessage) : errorMessage_(errorMessage) {}

  virtual const char* what() const noexcept
  {
    return errorMessage_.get();
  }
};

class InvalidInput : public std::exception
{
private:
  MyString errorMessage_;
public:
  InvalidInput(MyString errorMessage) : errorMessage_(errorMessage) {}

  virtual const char* what() const noexcept
  {
    return errorMessage_.get();
  }
};


#endif
```

ExceptionNames.h

```
#ifndef EXCEPTION_NAMES
#define EXCEPTION_NAMES
extern const char* ERROR_INCORRECT_ROUTE_POINTS_NAME;
extern const char* ERROR_INCORRECT_ROUTE_POINTS_NAME_WHEN_COPYING;
extern const char* ERROR_INCORRECT_ROUTE_POINTS_NAME_WHEN_ASSIGN;


extern const char* ERROR_INCORRECT_ROUTE_START_NAME;
extern const char* ERROR_INCORRECT_ROUTE_START_NAME_WHEN_COPYING;
extern const char* ERROR_INCORRECT_ROUTE_START_NAME_WHEN_ASSIGN;


extern const char* ERROR_INCORRECT_ROUTE_FINISH_NAME;
extern const char* ERROR_INCORRECT_ROUTE_FINISH_NAME_WHEN_COPYING;
extern const char* ERROR_INCORRECT_ROUTE_FINISH_NAME_WHEN_ASSIGN;


extern const char* ERROR_INCORRECT_ROUTE_NUMBER;
extern const char* ERROR_INCORRECT_ROUTE_NUMBER_WHEN_COPYING;
extern const char* ERROR_INCORRECT_ROUTE_NUMBER_WHEN_ASSIGN;


extern const char* ERROR_WITH_FILE;
extern const char* ERROR_WITH_INPUT;


extern const char* ERROR_FILE_IS_NOT_OPEN;
extern const char* ERROR_INCORRECT_FILE_STRUCTURE;


extern const char* ERROR_INCORRECT_LINES_NUMBER;
```

```
extern const char* ERROR_INCORRECT_INDEX_OUT_OF_RANGE;
#endif
```

ExceptionNames.cpp

```
const char* ERROR_INCORRECT_ROUTE_POINTS_NAME = "Некорректное название пунктов маршрута";
const char* ERROR_INCORRECT_ROUTE_POINTS_NAME_WHEN_COPYING = "Некорректное название пунктов маршрута
при копировании";
const char* ERROR_INCORRECT_ROUTE_POINTS_NAME_WHEN_ASSIGN = "Некорректное название пунктов маршрута
при присваивании";


const char* ERROR_INCORRECT_ROUTE_START_NAME = "Некорректное название начала маршрута";
const char* ERROR_INCORRECT_ROUTE_START_NAME_WHEN_COPYING = "Некорректное название начала маршрута
при копировании";
const char* ERROR_INCORRECT_ROUTE_START_NAME_WHEN_ASSIGN = "Некорректное название начала маршрута
при присваивании";


const char* ERROR_INCORRECT_ROUTE_FINISH_NAME = "Некорректное название конца маршрута";
const char* ERROR_INCORRECT_ROUTE_FINISH_NAME_WHEN_COPYING = "Некорректное название конца маршрута
при копировании";
const char* ERROR_INCORRECT_ROUTE_FINISH_NAME_WHEN_ASSIGN = "Некорректное название конца маршрута
при присваивании";


const char* ERROR_INCORRECT_ROUTE_NUMBER = "Некорректный номер маршрута";
const char* ERROR_INCORRECT_ROUTE_NUMBER_WHEN_COPYING = "Некорректный номер маршрута при
копировании";
const char* ERROR_INCORRECT_ROUTE_NUMBER_WHEN_ASSIGN = "Некорректный номер маршрута при
присваивании";

const char* ERROR_WITH_FILE = "Ошибка файла: ";
const char* ERROR_WITH_INPUT = "Ошибка ввода: ";


const char* ERROR_FILE_IS_NOT_OPEN = " не открыт";
const char* ERROR_INCORRECT_FILE_STRUCTURE = "Некорректная структура файла";


const char* ERROR_INCORRECT_LINES_NUMBER = "Некорректная запись количества строк";


const char* ERROR_INCORRECT_INDEX_OUT_OF_RANGE = "Некорректный индекс, выход за границы массива";
```

6. Итоги

За время написания проекта я научился многим вещам, основными из которых являются:

1. Написание собственного проекта, содержащего множество файлов.
2. Написание собственного класса с перегрузками функций, сокрытием информации и созданием защищенного от пользователя интерфейса.
3. Работа с динамической памятью, контроль утечек памяти.
4. Создание шаблонов классов/функций.

Работа над проектом также дала мне опыт в отладке программы, использовании средств контроля утечек памяти, рефакторинге кода и составлении отчета большого проекта.

Проект успешно написан и отвечает всем требованиям, предъявленным к нему.