Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

«Информационная система аэропорта»

по дисциплине

«Конструирование программ и языки программирования»

Выполнил:                                                                 Руководитель:

студент гр. 750503                                                      Марченко В.В.

Данилович Илья Артурович

Минск, 2018

ЗАДАНИЕ

по курсовому проектированию

Студенту Даниловичу Илье Артуровичу

1. Тема проекта: «Информационная система аэропорта»

2. Дата выдачи задания: 10.09.2018

3. Предоставление студентом готового проекта: 26.12.2018

4. Решаемые задачи и функционал разрабатываемого ПО:

Программа помогает пользователям лучше ориентироваться со списком рейсов в базе данных. В качестве GUI используется консольная псевдографика.

1. Выводит на экран приветствие сообщение с описание работы программы и меню выбора.
2. В таблице рейсов будет помещаться напоминание о скором вылете воздушного судна (если такие имеются).
3. Пользователь может просматривать содержащиеся в базе рейсы и сортировать их по нужному критерию.
4. Имеются функции добавления рейсов, а также их удаление.
5. Пользователь может просматривать всю историю рейсов, изменить ее и вывести в файл.
6. Пользователь может найти рейс исходя из его информации среди хранящийся в базе данных.
7. При нахождения нужного нам типа воздушного судна, будет выводится полная информация о рейсе.

5. ОС и средства разработки:

5.1 Microsoft Visual Studio 2017

5.2 Window 10

6. Другое:

6.1. Бинарный файл

7. Перечень графического материала:

7.1. «Контроль воздушного транспорта»

Схема структурная. Формат A4.

7.2. «Контроль воздушного транспорта»

Диаграмма классов. Формат A4.

СОДЕРЖАНИЕ

Введение4

1. Обзор источников5

* 1. Основные технологии5

1.2 Обзор аналогов6

2. Структурное проектирование7

3. Функциональное проектирование8

4. Разработка программных модулей11

4.1 Перемещение курсора в меню11

4.2 Вывод рейсов хранящихся в базе данных 11

4.3 Сортировка данных12

4.4 Изменение размеров окон12

4.5 Работа с файлами 12

4.6 Использование потоков для перевода строк в числа13

5. Руководство пользователя14

6. Тестирование17

Заключение18

Список использованных источников19

Приложения А20

Приложения Б21

Приложения В22

Приложения Г23

Приложения Д24

Приложения Е25

# **ВВЕДЕНИЕ**

**Управление воздушным движением (УВД)**, **Организация воздушного движения (ОрВД)** — система организационных и технических мероприятий, обеспечивающая порядок и безопасность полетов [воздушных судов](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B4%D1%83%D1%88%D0%BD%D0%BE%D0%B5_%D1%81%D1%83%D0%B4%D0%BD%D0%BE) в воздушном пространстве и обмен информацией между [авиадиспетчерами](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D0%B8%D0%B0%D0%B4%D0%B8%D1%81%D0%BF%D0%B5%D1%82%D1%87%D0%B5%D1%80) и экипажами воздушных судов с использованием средств радиосвязи, аэронавигации и ЭВМ.

Контроль воздушного транспорта играет чрезвычайно важную роль.

Поэтому данное приложение помогает ориентироваться во времени, рейсах и поиске нужных нам параметров и также анализировать данные необходимые для работы. Так же для упрощения работы есть сортировка по параметрам.

Данное программное обеспечение сделано для пользователей, работающих в сфере Гражданской Авиации и Вооруженных Сил. Это люди, которые трепетно относятся к своей работе и времени.

Данный проект программного обеспечения направлен на:

1. Возможность проверить свои знания на практике
2. Возможность углубления и освоения новых знаний, навыков, технологий (проектирование ПО, грамотное оформление кода, управление версиями и т.д.)
3. Получение опыта в довольно универсальном направлении ПО
4. Возможность иметь в своём портфолио при трудоустройстве готовый и довольно серьёзный проект
5. Возможность будущего расширения и развития приложения

Найти аналоги в данный момент, составляет трудности. Так как это развивающаяся область. Но в найденных аналогах больший упор сделан на облегчение, автоматизацию работы с ПО с помощью аппаратной части. Поэтому мне, как разработчику в программной части предоставлен большой простор для фантазии. И кроме самых обычных и ожидаемых функций я предлагаю возможность посмотреть список рейсов, которые необходимо добавить.

В будущем расширении приложении есть желание добавить связь с другими устройствами, более быструю обработку запросов, смену языков и другие функции.

1. **ОБЗОР ИСТОЧНИКОВ**

**1.1 Основные технологии**

***C++*** — компилируемый,  статически типизированный  язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

**Система управления версиями** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Version Control System, VCS* или *Revision Control System*) — [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения [исходных кодов](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

***Псевдографика***,  *псевдографические символы* — совокупность символов, включенных в набор символов компьютерного шрифта, отображающих графические примитивы (линии, прямоугольники, треугольники, кресты, различная заливка и т. п.). Псевдографические символы реализуются в рамках формата шрифта (к примеру, матрица 8×12 точек) и дополняют в таком наборе цифро-буквенные и служебные символы. Основное назначение псевдографики — графическое оформление программ с текстовым интерфейсом пользователя (в том числе и т. н. консольных) — отображение в них окон, меню, кнопок и прочих элементов интерфейса, создания рамок и таблиц.

* 1. **Обзор аналогов**

При рассмотрении аналогов было найдено множество приложений, занимающих данную область. Условно их можно разделить на различные организации, предназначенные для людей, работающих в Гражданской Авиации.

Поскольку данный проект разрабатывался с оглядкой на потребности отдельных людей, то будем рассматривать те приложения, которые рассчитаны на индивидуальное использование. Выделим наиболее общий функционал этих приложений:

1. Запись и хранения данных о рейсах.
2. Изменение их параметров.
3. Общая история изменения о рейсах.

Каждое приложение обладает собственным функционалом, что обязательно будет учтено при разработке.

1. **СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ**
2. ***Блок графического пользовательского интерфейса (меню)***

Оно открывается при запуске программы и с помощью него будет производиться управление программой. В нём можно будет выбрать интересующий пользователя пункт меню. Так же это окно будет меняться в зависимости от выбранного пункта. Управление в данном блоке будет происходить с помощью работы с псевдографикой.

1. ***Блок логического ядра (Алгоритмы)***

Написанные функции обработки информации. Здесь будут представлены различные функции необходимые в работе программы. Это будут как и банальные, фундаментальные алгоритмы при работе с программой, функционал которой похож на базу данных (сортировки, поиск, удаления, добавления), так и совершенно оригинальные.

1. ***Блок данных для работы (в оперативной памяти)***

Данные, с которыми непосредственно работают алгоритмы. С помощью этого блока будет происходить обмен информацией между различными функциональными блоками, без которого сложно представить практически любую программу.

1. ***Хранилище данных (файл)***

Файлы для сохранения информации, чтобы избежать потери данных из-за случайных аппаратных ошибок. Так же используются для возобновления работы программы после её закрытия. Каждый файл будет использоваться по своему назначению, и их работа не будет пересекаться. В каждом файле будет храниться вся нужная информация для данной функции, без надобности переходить в другие файлы. Это сделано с целью ускорения и упрощения процесса. Но из-за этого приходиться жертвоваться памятью на жестком диске, из-за некоторых повторных записей в разных файлах. Но при небольших объемах информации – это не так уж и заметно.

Схема представлена в приложении А.

1. **ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе будет приводиться описание реализуемых классов, обмена информацией между ними и работа с ними.

Первым при запуске нас встречают методы класса Menu. Так же в этом классе реализованы приветственное окно и окно показываемое при выходе из программы. На обоих экранах присутствует информация о проекте:

1. Название (см. Рисунок 3.1).
2. имя учащегося, который его разрабатывал (см. Рисунок 3.2).

Хоть и в проекте графика реализована средствами консольной графикой, так называемой псевдографикой, интерфейс довольно дружелюбный. В окнах, описываемых ранее, расположены изображения, составленные из знаков.



Рисунок 3.1



Рисунок 3.2

Все остальные методы данного класса работают с меню, в которых пользователю предоставлена свобода выбора. Он может выбрать интересующий его пункт меню, которое он видит, после чего будет перенаправлен в соответствующий метод класса, функция перемещения представлена во вложении Б. В котором будут проходить его следующие действия. После чего он будет возвращен обратно в главное меню. Это действие будет повторяться до тех пор, пока пользователь не захочет выйти из программы.

Помощь в управлении главного меню реализовано с помощью курсора и подсветки выбранного пункта меню (изображено на рисунке 3.3).

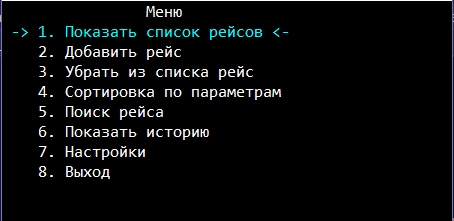


Рисунок 3.3

Таким же образом реализованы и другие меню, вызываемые из главного меню. Например, когда нам необходимо выбрать по какому-либо параметру производить поиск (сортировку). Пример реализован на рисунке 3.4.

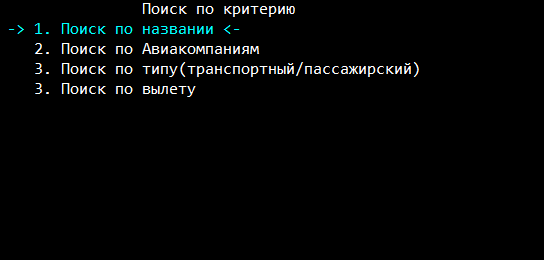


Рисунок 3.4

Следующий, самый важный класс в данной программе – Flight.С помощью объектов данного класса мы реализуем хранение того или иного рейса с помощью полей класса. Все поля данного класса – приватные. Сделано это из соображений безопасности. Поэтому чтобы обеспечить безопасный доступ к данным полям – используются «сеттеры» и «геттеры», соответственно для установки значения и получение хранимого значения. В конструкторе класса так же прописаны действия, которые нам нужны только при добавлении рейса в базу (например, для даты, когда мы добавили рейс в базу данных).

Класс, методы которого реализуют большую часть функционала программы – Database. В этот класс включен класс Flight. Отношения между этими классами строгие, так как при удалении самого объекта класса Database нам не нужно хранить объекты класса Flight. Поэтому данное отношение можно назвать композицией. В этом блоке происходит добавление информации, удаление информации, обработка имеющихся данных функциями (такими как сортировка, поиск по некоторой информации, удаление, добавление). Так же в этом классе расположены методы работы с файлами.

Вышеописанные функции в классе реализованы следующими методами:

1. show\_color – вывод на экран всех хранящихся рейсов на данный момент(помощь в определении вылета служит подсветка рейсов разными цветами);
2. addObject – добавление рейса в базу данных(при том одинаковые рейсы, но с разными вылетами, считаются разными, чтобы можно было отслеживать рейс);
3. delObject – удаление рейса по названию воздушного судна и его серийного номера, функция представлена во вложении В;
4. sort – сортировка всех рейсов по параметрам, функция представлена во вложении Г;
5. find – поиск по какому-либо параметру;
6. read\_database - загрузка из файла базы с последнего запуска программы (после загрузки файл не очищается, на случай непредвиденного сбоя, в таком случае можно будет загрузить последнее сохранённое состояние).
7. save\_in\_database – сохранение текущих данных в файл;
8. history – загрузка в объект истории добавления продуктов;
9. setting – метод настроек, в котором можно очистить файлы с которыми работает программа;

Так же, с целью сделать код более логичным, нужно было добавить класс работы с датами. Здесь реализованы методы для сравнения дат и узнавание разницы между датами.

Диаграмма классов представлена в приложении Д.

1. **РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

**4.1 Перемещение курсора в меню**

Для выделения некоторого пункта меню объявлена специальная переменная.

Зная коды основных клавиш, нам не составит труда отловить их: «↑»(стрелка вверх) – код 80; «↓»(стрелка вниз) – код 72; enter – код 13; esc – код 27. Поэтому при каждом нажатии клавиши, программа отлавливает нажатие с помощью функции \_getch(). Нажатие стрелок вверх или вниз изменяет переменную, которая отвечает за то, какой пункт меню подсвечивать. Нажатие enter вызывает метод соответствующий подсвеченному меню. Нажатие esc возвращает в главное меню.

Для изменения цвета в консоли необходимо было подключить заголовочный файл <windows.h>, а также получить дескриптор консоли.

HANDLE h;

h = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(h, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY)

* 1. **Вывод рейсов хранящихся в базе данных**

Пример изображен на рисунке 4.1

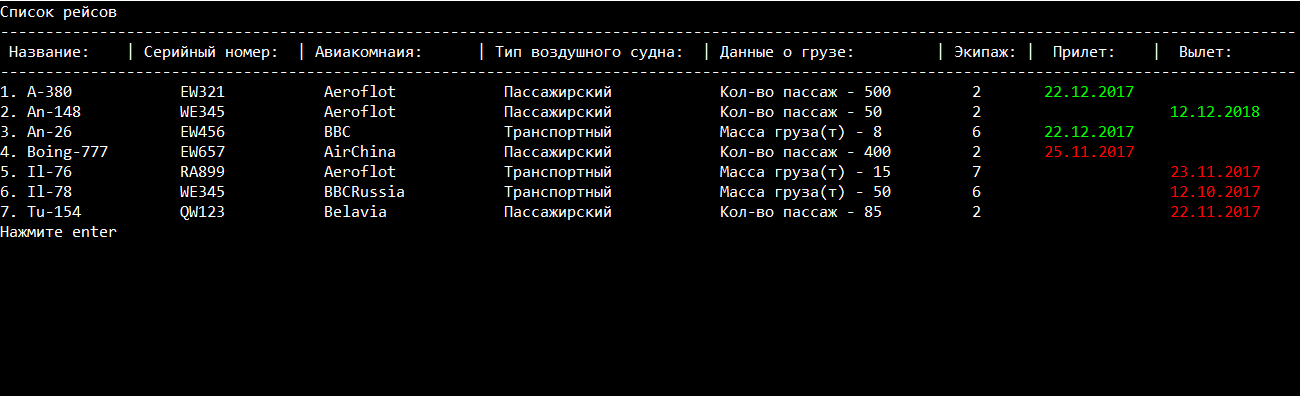


Рисунок 4.1

Просматривая все рейсы в базе можно заменить, что прилет\вылет обозначены разными цветами. Это говорится о том, что рейс, который будет осуществляться в будущем, обозначается зеленым цветом, рейс который уже был – красным, а белым – то, что сегодня.

Цвета подбирал на основе признаков, к примеру, красный цвет говорит о сигнале, предостережение (к примеру, «Внимание!»), а зеленый говорит о спокойствие, рейс еще не был осуществлен, белый самый простой, поэтому я выбрал настоящее время.

Для реализации этой функции, была подключена библиотека <ctime>. Она была подключена с целью узнавать системное время и вычисления разницы со сроком годности продукта. Программа узнаёт системное время в нужном формате (дд.мм.гг) с помощью следующих строчек:

char buffer[80];

time\_t seconds = time(NULL);

tm\* timeinfo = localtime(&seconds);

strftime(buffer, 80, "%d.%m.%Y", timeinfo);

В итоге я рассматриваю три возможных варианта разности:

* То, что было – подсвечивать красным
* То, что происходит сегодня – подсвечивать белым
* То, что будет – подсвечивать зелёным
  1. **Сортировка данных.**

При больших количествах продуктов нам интересно бы их сортировать по параметрам. За основу был взят алгоритм Пузырька. Код данной сортировки представлен во вложении.

**4.4 Изменение размеров окон.**

Вывод небольшой информации на большое окно консоли – выглядит не слишком привлекательно, так же как и полная занятость окна консоли. Поэтому было принято решение для каждого метода подобрать свой размер окна. Реализована данная функция была с помощью библиотеки<windows.h>, и функции:

system("mode 70,15");

При необходимости расширить буфер (рабочую область) консоли – для отображения большего числа записей, чем размеры консоли, были использованы следующие функции:

COORD crd = { 147, 100 };

SMALL\_RECT src = { 0, 0, crd.X, crd.Y };

Данные строчки кода добавляют скроллер, с помощью которого и происходит перемещение по рабочей области.

**4.5** **Работа с файлами.**

При сохранении данных в файл нас интересует надёжность записей (которая не даёт возможность напрямую изменить данные файла вручную, что увеличивает безопасность) и рационально занимание места на диске. Конечно, для большей безопасности необходимо использовать шифр, который защитит информацию достаточно серьезно. Но в своей программе было решено использовать работу с файлами в бинарном режиме, как самый экономичный метод по отношению к памяти на диске, так и с точки зрения защиты. Попытка сделать изменения в файле приведёт к невозможности работать с этим файлом в большинстве случаев.

В данном режиме файл открывается таким образом:

ofstream fout("Database.bin", ios::binary | ios::trunc);

И чтение, и запись происходили следующим образом (например, для переменной отражающей количество экипажа):

fout.write(reinterpret\_cast<char\*>(&count), sizeof(int));

fin.read(reinterpret\_cast<char\*>(&count), sizeof(int));

* 1. **Использование потоков для перевода строк в числа.**

Каждая программа должна быть защищена от случайно неправильно введённых данных, потому что людям свойственно ошибаться. Поэтому программа должна адекватно реагировать на такие ситуации.

Данная проблема была решена с использованием строковых величин при вводе и при соответствии строки нужному формату – перевод строки в число. Функция перевода из строки была сделана шаблонной, чтобы в последующих обновлениях программы проверка на ввод уже имела часть нужного ей кода.

template<typename T>

T fromString(const std::string& s)

{

std::istringstream iss(s);

T res;

iss >> res;

return res;

}

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Данный проект был написал с целью улучшить качество работы и структурировать некоторые её области.

Но так как проект рассчитан на активную и большую аудиторию, он работает пока что только на последних версиях Windows. За счёт запуска его только на последних версиях данной операционной система – не стоит беспокоиться об оперативной памяти необходимой программе, Вы даже не заметите эту часть памяти, потребляемой программой.

Программа не требует установки. Запускается программа с помощью Air\_Traffic\_Control.exe. Рекомендуется расположить программу в отдельной папке. Ведь уже после первой сессии в директории, в которой находиться программа, будет создано два файла. Открывать данные файлы и редактировать – крайне не рекомендуется. Это в большинстве случаев приведёт к потере всей информации хранящейся в файле.

При запуске вы увидите стартовый экран. Для продолжения и выхода в главное меню необходимо нажать Enter (см. Рисунок 5.1).



Рисунок 5.1

Перемещение по меню происходит с помощью кнопок «↑» (стрелка вверх) и «↓» (стрелка вниз). Изображено на рисунке 5.2 и 5.3

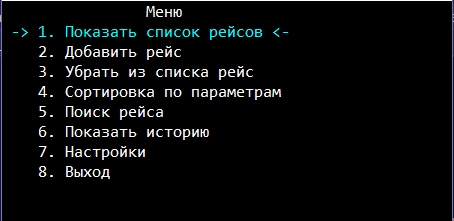


Рисунок 5.2

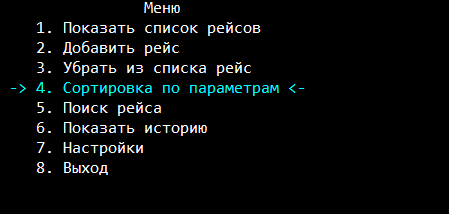


Рисунок 5.3

При желании выбрать меню необходимо нажать Enter. При желании вернутся из контекстного меню в главное – нажмите Esc.

При поиске нам нужного параметра будет высвечена полная информация о рейсе (см. Рисунок 5.4):

1. Название
2. Серийный номер
3. Компания
4. Количество пассажиров или масса груза
5. Экипаж
6. Дата
7. Расстояние
8. Скорость
9. Потребления топлива

10) Время полета

11) Остаток топлива

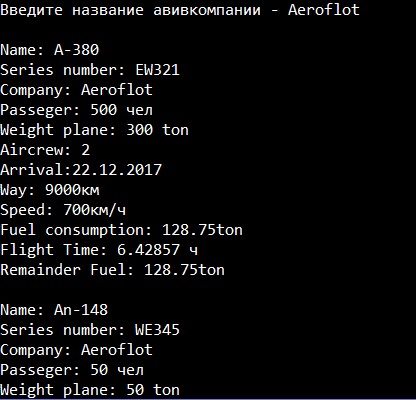


Рисунок 5.4

При желании стереть какую-то информацию – поможет окно настроек. В нём можно отчистить информацию как по какой-то одной выбранной функции, так и можно вернуть программу к самому началу, стерев все данные (см. Рисунок 5.5).

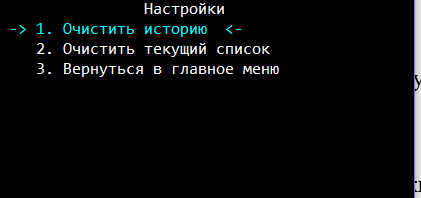


Рисунок 5.5

1. **ТЕСТИРОВАНИЕ**

Тестирование проекта было разделено на две части: совместимость с операционными системами, проверка работоспособности всех функций программы.

Совместимость с операционными системами была проверена запуском программы на разных версиях Windows. Точно можно заметить, что на версиях от Windows 7 и выше – программа работает как надо и непредвиденных ошибок не показывает. Кроссплатформенности программа не предусматривает, поэтому на GNU/Linux или на Mac OS X программа не работает.

Проверка работоспособности была проведена вручную, выбором различных функций. И проработкой различных исключительных ситуаций. Таких как: работа с пустой базой, неправильный ввод от пользователя и т. д.

Всё найденные исключительные ситуации были проанализированы и обработаны. Чтобы не допустить неожиданного выключения или зависания программы. Обработка исключительных ситуаций была реализована с помощью конструкции try\catch и с помощью if…\else. Используя try\catch, я проверяю работу открытия файла. А вот с помощью конструкции if…\else ошибки на ввод данных. При возникновении какой-то исключительной ситуации – это будет сообщено пользователю, после чего он сможет продолжить пользоваться программой. Данное решение было принято с целью упростить работу с программой. Например, при неправильном формате ввода будет просьба ввести информацию повторно, обращая внимание на нужный формат. При чтении данного сообщения, ввести информацию в неправильном формате – довольно нелогично. Но программа будет просить правильного формата ввода, пока она его не получит, что в самом крайнем случае приведёт к увеличению времени работы программы.

**ЗАКЛЮЧЕНИЕ**

В проекте реализованы все функции, которые были предусмотрены и записаны в листе заданий. Лишних функций добавлено не было. Зато написанные функции были тщательно протестированы, во избежание неожиданных исключительных ситуаций при работе программы с пользователем. Интерфейс вышел простым и интуитивно понятным, с помощью работы с клавишами используемых в большинстве других программ. При использовании псевдографики были использованы и реализованы все найденные возможности консоли. Что довольно упрощает процесс работы с программой. Например, подсветка дат несёт не только эргономичный, но и информационный характер.

Других похожих проектов в данной области найдено было не так уж и много, но по и этим имеющимся знаниям можно сказать, что проект написан на должном уровне.

Для должного конкурирования с похожими программами этому проекту не хватает привлекающего графического интерфейса и баз данных. Функционал программы от этого не поменяется, но работать станет ещё приятнее. Данные направления не были реализованы за счёт недостаточных знаний в этих областях. Но и без этого при написании программы использовались лучшие алгоритмы, чтобы обеспечить быстродействие и удобство.

В будущем проект будет усовершенствован и использоваться в сфере Гражданской Авиации и Военно - Воздушных Сил в области контроля и надзора воздушного транспорта.

# 

# **СПИСОК ИСТОЧНИКОВ**

# Герберт Шилдт - Самоучитель C++, стр. 239-270

1. Роберт Лафоре - Объектно-ориентированное программирование C++, 4-е изд (Классика Computer Science) – 2004, стр. 536-600
2. Консольное меню [электронный ресурс]: <http://www.cyberforum.ru/cpp>
3. Философия С++. Введение в стандартный С++,2004, *Брюс Эккель*
4. Pro Git, 2012, *Скотт Шакон*

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Структурная схема

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Схема программы

**ПРИЛОЖЕНИЕ В**

*(обязательное)*

Схема программы

**ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Схема программы

**ПРИЛОЖЕНИЕ Д**

*(обязательное)*

Диаграмма классов

**ПРИЛОЖЕНИЕ Е**

*(обязательное)*

Пояснительная записка