АННОТАЦИЯ

В документе описывается программа ОС семейства Windows, которая организует обработку информации о авиаперелётах.

Документ содержит в себе описание реализации данной программы, с аргументированием тех или иных её критериев. Предоставляется программа с детальным пояснением, её структуры, реализации и особенностей работы на ЭВМ. Также пояснительная записка содержит графическую интерпретацию основных алгоритмов программы и руководство к эксплуатации программы.

СОДЕРЖАНИЕ…………………………………………………………. 2

ВВЕДЕНИЕ………………………………....…………………………… 3

1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ…………………………..4
2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ………………………………….5
   1. Постановка задачи……………………………………………………..5
   2. Описание и обоснование выбора входных и выходных данных…...5
   3. Применяемые математические методы………………………………6
   4. Структура программы………………………………………………....7
   5. Входные данные……………………………………………………….8
   6. Выходные данные……………………………………………………..8
   7. Используемые функции……………………………………………….9
      1. Меню…………………………………………………………….10
      2. Ввод……………………………………………………………...11
      3. Вывод……………………………………………………………11
      4. Корректировка…………………………………………………..12
      5. Сортировка……………………………………………………...12
      6. Поиск…………………………………………………………….13
      7. Считывание данных из файла………………………………….13
      8. Запись данных в файл…………………………………………..14
      9. Вывод статистики………………………………………………15
      10. Удаление элемента……………………………………………..15

2.8 Алгоритм функций………………………………………………….. 16

ВЫПОЛНЕНИЕ ПРОГРАММЫ……………………………………...…30

* 1. Условие выполнение программы……………………………………30
  2. Загрузка и запуск программы………………………………………..30

ВЫВОДЫ………………………………………………………………….35

СПИСОК ЛИТЕРАТУРЫ………………………………………………..36

ПРИЛОЖЕНИЕ А………………………………………………………...37

ВВЕДЕНИЕ

В ходе выполнения курсовой работы была написана программа для работы с данными о авиарейсах. Программа предполагает собой различные способы ввода и вывода данных, а также разноплановая их обработка. Был реализован интерфейс, который делает работу в данной программе понятной и доступной любому пользователю.

Написание программы было реализовано при помощи языков С/С++ согласно принципам структурного программирования, которая базируется на методах нисходящего проектирования и модульного программирования. Написание программы производилось в интегрированной среде программирования Visual Studio 2013.

Программа написана согласно заданию кафедры информационных систем Севастопольского государственного университета.

1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Назначение программы «Авиарейсы» заключается в введение статистики о авиарейсах и их компонентах, в частности о номере авиарейса, марки самолёта, маршрута пути, а также цены за билет. Данная программа может использоваться в аэропортах, турбюро. В случае необходимости, пользователь имеет возможность корректировать данные, получать необходимые статистические данные, упорядочивать их. Результаты данной работы могут хранится на жёстком диске, и в случае необходимости загружаться в программу.

Программа может работать на любой операционной системе Windows, начиная с Windows XP. Для работы программы не требуется большого количества оперативной памяти, а также места на жёстком диске.

1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ
   1. Постановка задачи

Программа должна уметь получать, обрабатывать и выводить данные, представленные заданием.

По заданию даны ведения о перевозках авиапассажиров на рейсах одного аэропорта. Входные данные: структура записи (номер рейса, маршрут (например, «Севастополь − Лондон», 20 символов), марка самолета (6 символов), общие стоимостные затраты на рейс (грн.), количество пассажиров). Помимо стандартных функций, необходимо было предоставить статистику о средней стоимости перевозки одного пассажира на рейсе, а также итоговые данные по затратам и количеству пассажиров и среднюю стоимость провозки одного пассажира по аэропорту.

Основные функции программы: ввод данных (ввод с клавиатуры), вывод данных в виде таблицы на экран, работа с файлами, в частности, запись данных в текстовый и бинарный файл, и, соответственно, считывание данных из текстового или бинарного файла, сортировка данных, корректирование конкретной записи, удаление.

Также было реализованы вспомогательные функции, такие как интерфейсное оформление запросов для всех вышеупомянутых, и служебные, в частности меню, очистка памяти.

* 1. Описание и обоснование выбора входных и выходных данных

В качестве структуры данных, в который будет хранится информация о авиарейсе был выбран односвязный линейный список, который хранит в себе информационную структуру «L» и указатель на следующий элемент.

struct L{

fly data;

L\* next;

};

struct fly{

int ID;

int n;

char route[21];

char mark[7];

double cost;

};

Этот выбор аргументирован тем, что в данной программе приоритетнее минимизировать затратность, а выигрыш от быстроты поиска как, к примеру, при использовании бинарных деревьев, не имеет особой важности. В частности, ни одна из функций по данной причине не является рекурсивной и соответственно не расходует большое количество оперативной памяти.

Для написания программы использовалось среда MS Visual Studio 2013. Выбор среды аргументируется большим числом подключаемых по умолчанию библиотек, а также удобным интерфейсом и реализаций работы с многофайловым проектом.

Также, помимо основных полей, заданных вариантом, было добавлено поле «ID» представляющий собой идентификатор каждой записи, который, в свою очередь является персональным для каждой записи и не может повторяться. Он представлен целочисленным типом int.

Каждому рейсу присваивается целочисленное значение его номера, которое в последствии является характерным и общи признаком для подсчёта статистических данных о рейсе.

Маршрут рейса записывается в строку при помощи символьного массива из 21 элемента, где последняя 20-ая ячейка отведена под символ конца строки. Аналогичным образом реализовано и поле марке самолёта, которое представлено массивом размерности из 7 элементов, исходя из тех же причин.

Стоимость проезда на рейсе представляет с собой действительное число с плавающей точкой (тип double). В ходе работы программы данное значение будет выводится лишь с двумя числами после запятой.

* 1. Применяемые математические методы

Из математических методов в программе используется сортировка указателей списка, основой которой является пузырьковая сортировка.

Для подсчёта статистических данных используется математическая формула среднего арифметического чисел.

* 1. Разработки модульной структуры программы

Данная программа была написана как многофайловый проект в среде Visual Studio 2013 Community. Работа программы начиналась с функции из файла Main.cpp :

int main();

В данной функции создаётся указатель Top на первый элемент списка, который в свою очередь передаётся в функцию-меню:

int menu(L\* \*Top);

Стоит также отметить, что все файлы данной программы, кроме файла Main.cpp и заголовочного файла Struct.h имею как заголовочный файл (\*.h), так и соответствующий ему файл реализации (\*.cpp). Заголовочный файл Struct.h содежит в себе, как все основные структуры данных, так и библиотеки <iostream> и <iomanip>, предназначенные для управления вводом-выводом.

Подключение всех файлов проекта происходит в Menu.h, которая непосредственно и отвечает за выполнение программы и выбор необходимого пункта меню при помощи интегрированного меню.

2.5 Входные данные

При запуске программы пользователь может ввести данные при помощи клавиатуры, при условии, что ID у записей не будет повторяться, в противном случае программа попросит ввести данные ещё раз, указав на данную ошибку. При заполнении поля о марки самолёта или маршрута нельзя превышать описанный в варианте задания предел, иначе запись будет обрезана по количеству считываемых символов. При введении вместо цифровых данных символьных программа также выведет сообщение об ошибке.

При использовании функции необходимо вводить данные, согласно требуемым, иначе будет выводится сообщение о необходимости правильного ввода, до того, как пользователь введёт корректный запрос.

Программ может также считать данные из файла. При запуске соответствующей функции, пользователю необходимо ввести имя файла, если он находится в папке с исполняемым файлом, или же путь к файлу, если он находится в другой папке. В зависимости от расширения файла программа будет считывать его либо как бинарный, либо как текстовый. Данный выбор зависит от того, есть ли у файла расширение «txt» или нет.

2.6 Выходные данные

В последствии ввода при помощи файла или клавиатуры, можно просмотреть данные в виде таблицы при помощи функции:

void Print\_L(L\* Top);

Перед эти данные можно отсортировать или изменить поля отдельной записи. Также можно произвести вывод в текстовый или бинарный файл, в зависимости от расширения.

Можно просмотреть в виде таблицы общую статистику данных о рейсах, которая включает в себя поля: номер рейса, количество человек на этом рейсе, общая стоимость (суммирование всех стоимостей пассажиров), и средняя стоимость проезда на этом рейсе.

На экран в ходе работы изначально и после завершения работы выбранной функции выводится меню, в котором можно выбрать необходимую операцию, или завершить работу программы.

2.7 Используемые функции

2.7.1 Меню

Заголовочный файл меню содержит в себе подключение всех других функций, которые используются в программе, а также прототип собственной функции:

int menu(L\*\* Top);

В ходе работы функции в бесконечном цикле работает оператор switch, который посредством считывания определяет, какую из функций надо запустить. При вводе некорректного значения программа его проигнорирует. После завершения каждой функции, программа функция menu очищает экран и вновь выводит меню выбора следующей операции. В случае корректного завершения, функция возвращает значение 0.

2.7.2 Ввод

Для ввода данных с клавиатуры, а именно добавления нового элемента в список используются следующие функции:

void Add\_L(L\* &Top, L\* &End);

fly element(void);

Первая из функций отвечает за добавления элемента в список, при это меняются указатели на первый и на последний элемент списка. Вторая функция предназначена для оформления интерфейса считывания данных и возвращает значение информационного поля. Каждый некорректный ввод обрабатывается, при этом очищается буфер и состояние объекта ввода. Невозможно никоим образом ввести данные с одинаковым ID.

2.7.3 Вывод списка на экран в виде таблицы с заголовком осуществляется при помощи функций:

inline void print\_element(fly \_new);

inline void Table\_title();

void Print\_L(L\* Top);

Первая функция отвечает за вывод строки с данными, которые передаются ей как параметр, которыми обладает текущий элемент списка. Эти данные имеют между собой разделители и занимают определённое количество ячеек. Функция является встроенной, что улучшает производительность. Никакие данные не возвращаются.

Вторая функция отвечает за контурное обрамление таблицы и вынесена как отдельная функция по причине неоднократного использования. Печатается шапка таблицы. Является встроенной, что улучшает производительность. Функция также ничего не возвращает.

Последняя функция отвечает за прохождение по всему списку и печать его значений. При наличии хоть одного элемента выведется шапка таблицы, в противном случае она выводится не будет. Значение параметра Top передаётся как параметр значения.

2.7.4 Корректировка

При необходимости, есть возможность изменить данные записи. В этом случае, необходимо указать ID элемента, который будет изменён. Так как данный параметр сугубо персональный каждому элементу списка, то изменится только одно информационное поле, при совпадении ID. Если вышеуказанное поле было введено корректно, то в таком случае необходимо выбрать сначала изменяемый тип поля, а после ввести новое значение.

Значение параметра ID изменять нельзя.

За реализацию данных процессов отвечает функция:

void Correct(L\* &Top);

В ходе её работы ничего не возвращается, однако значение указателя Top может быть изменено. В ходе работы функции на экран выводятся указания, поясняющие, что необходимо сделать на каждой из стадий.

2.7.5 Сортировка

Сортировка списка реализуется при помощи не рекурсивной сортировки, основанной на сортировке пузырька.

void Sort(L\* &Top);

При запуске данная функция не просит вводить данные. По завершению своей работы, на экран выводится соответствующее сообщение. Суть сортировки заключается в изменении указателей next в каждой записи. Это затрачивает минимальную память, однако недостаток сортировки в том, что O(n2). Вспомогательных функций не используется.

2.7.6 Поиск

Поиск элемента по ключу выполняется при помощи функций:

void Search\_L(L\* Top);

void Search(L\* Top, char\* \_mark);

L\* Search\_n(L\* Top, int \_n);

Первая функция является основной и отвечает за ввод ключевого поля, по которому осуществляется поиск. Данным полем является марка самолёта.

После ввода, значение передаётся в функцию Search. В данной функции осуществляется сам обход списка и поиск элементов, подходящий по критерию, для вывода таблицы совпадений используется функции, описанные в пункте 2.7.3.

inline void Table\_title();

inline void print\_element(fly \_new);

В заголовочный файл по данной причине был подключен Print.h.

Также, есть функция Search\_n, которая отвечает за поиск по ключевому полю, а в частности номеру рейса. Эту функцию нельзя вызвать напрямую, однако она используется при подсчёте статистических данных, которые будут описаны в пункте 2.7.9.

2.7.7 Считывание данных из файла

Для считывание данных из файла используется функция:

void Start(L\* &Top, L\*& End)

Данная функция считывает данные из файла в список. При этом алгоритмически её можно разделить на 2 составляющие: ввод имени файла и определение его типа; открытие и считывание из файла в зависимости от его типа.

В ходе работы программы используется стороння функция очистки памяти, которая будет описана в пункте 2.7.11:

Clean\_memory(Top);

Из этого следует, что при считывании данных из файла, предыдущий список будет удалён.

Особенностью считывания из текстового файла в том, что данные по одному элементу должны находится в одной строке. Между ними может быть произвольное число пробелов, что позволяет совершать табуляцию. Функция оптимизирована тем, что происходит малозатратное посимвольное считывание и посимвольная обработка.

2.7.8 Запись данных в файл

Для записи данных в файл используется функция:

void File\_save(L\* Top)

В ходе её выполнения в файл выводятся строки с данными элементов. Корректно выводится как англоязычный, так и русскоязычный текст. Сам список, в ходе выполнения функций, остаётся без изменений. Бинарный файл может иметь любое расширение, кроме txt.

2.7.9 Вывод статистики

Для вывода статистический данных, исходя из основного односвязного списка, используем ещё одну аналогичную динамическую структуру:

struct Average{

int n;

int amount;

double sum;

double sred;

Average\* next;

};

В данной структуре поля имеют следующие назначение:

1. n - номер рейса, по которому будет подсчитываться статистика.
2. amount – количество человек на данном рейсе.
3. sum – суммарная стоимость всех людей на данном рейсе.
4. sred - среднее арифметическое всех стоимостей на данный рейс.
5. next – указатель на следующий элемент.

Для реализации корректной работы данной структуры используются следующие функции:

void Statistic(L\* Top);

Average\* Average\_New(L\* Top);

void Average\_Print(Average\* First);

void Average\_Clean(Average\*& First);

Первая функция является основной, в ходе её работы создаётся указатель на первый и на последний элемент данного списка. Данные из главного списка перегоняются в вспомогательный, при этом обрабатываясь. Число элементов нового списка будет равно числу разных номеров рейсов в исходном.

Вторая функция Average\_New возвращает указатель на элемент вспомогательного списка, при это преобразуя в себе элемент исходного. В ходе данной функции выделяется динамическая память.

Функция Average\_Print выводит значения вспомогательного списка в виде таблицы.

Функция Average\_Clean удаляет вспомогательный список по завершению работы с ним. Таким образом происходит избежание утечки памяти.

2.7.10 Удаление элемента

Удаление элемента списка осуществляется при помощи функций:

void Delete\_L(L\*& Top, L\* &End)

void Delete(L\* &Top, L\* &End, int \_ID)

Первая функция обеспечивает считывание ID, по которому будет происходить удаление и передаёт это значение в функцию №2, вместе с указателем на первый и последний элемент списка, которые передаются как параметры переменные.

В ходе работы программы на каждом этапе присутствуют сообщение о текущем состоянии и о правилах совершения следующей.

При единственном вызове функции можно удалить максимум один элемент.

2.7.11 Очистка памяти

Очистка динамической памяти при завершении работы программы осуществляет функция:

void Clean\_memory(L\* &Top)

Так как функция работает корректно, то утечки данных не происходит.

2.8 Алгоритм функций

Далее будут предоставлены алгоритмы всех основных функций, используемых в программе.

Рисунок 2.1 – Алгоритм добавления элемента

Структурная схема на рисунке 2.1 работает следующим образом:

Блок 1: Обнуление флага.

Блок 2: Создание нового элемента.

Блок 3: Проверка указателя главы списка.

Блок 4: Пока элемент не найден-делать.

Блок 5: Условие верно, если ID совпало.

Блок 6: Да – условие блока 5 выполнено, влаг равен 1.

Блок 7: Нет- условие блока 5 не выполнено, переход к следующему элементу.

Блок 8: Условие верно, если совпадение ID не было найдено.

Блок 9: Постусловие бесконечного цикла.

Блок 10: Проверка, является ли указатель на голову списку нулевым.

Блок 11: Если первого элемента нет, то он создаётся.

Блок 12: Добавляется элемент, при этом первый уже есть.



Рисунок 2.2 – Алгоритм печати элемента

Структурная схема на рисунке 2.2 работает следующим образом:

Блок 1: Печать заголовка таблицы.

Блок 2: Проверка, является ли Top нулевым (цикл while).

Блок 3: Печать текущего элемента списка.

Блок 4: Переход к следующему элементу.



Рисунок 2.3 – Алгоритм считывания из файла

Структурная схема на рисунке 2.3 работает следующим образом:

Блок 1: Очистка памяти от списка.

Блок 2: Считывание имени файла.

Блок 3: Проверка, является ли файл текстовым.

Блок 4: Открытие бинарного файла.

Блок 5: Посимвольное считывание из бинарного файла.

Блок 6: Создание нового элемента.

Блок 7: Является ли Top нулевым.

Блок 8: Занесение нового элемента в начало списка.

Блок 9: Добавление нового элемента в конец списка.

Блок 10: Открытие текстового файла.

Блок 11: Посимвольное считывание из текстового файла.

Блок 12: Преобразования считанных символов в соответствующие типы данных.

Блок 13: Проверка на наличие дубликата.

Блок 14: Добавление элемента в список.

Структурная схема на рисунке 2.4 работает следующим образом:

Блок 1: Считывание имени файла.

Блок 2: Проверка, является ли файл текстовый.

Блок 3: Открытие бинарного файла.

Блок 4: Проверка, является ли Top нулевым.

Блок 5: Запись структуры в файл.

Блок 6: Переход к следующему элементу.

Блок 7: Открытие текстового файла.

Блок 8: Проверка, является ли Top нулевым.

Блок 9: Вывод данных в строку через пробел.

Блок 10: Переход к следующему элементу.



Рисунок 2.4 – Алгоритм записи в файл

Структурная схема на рисунке 2.5 работает следующим образом:

Блок 1: Условие, выполняющие при совпадении ID.

Блок 2: Условие, выполняющиеся если данный элемент не единственный.

Блок 3: Проверка на совпадение ID со следующим элементом.

Блок 4: Условие, которое выполняется, если данный элемент-последний.

Блок 5: Обнуление флага, удаление элемента.

Блок 6: Удаление последнего элемента.

Блок 7: Удаление первого элемента.



Рисунок 2.5 – Алгоритм удаления

Структурная схема на рисунке 2.6 работает следующим образом:

Блок 1: Является ли Top нулевым.

Блок 2: Создание первого элемента вспомогательного списка.

Блок 3: Есть ещё элемента после Top.

Блок 4: Подсчёт среднего значения цены.

Блок 5: Переход к следующему элементу после Top.

Блок 6: Top не равно нулю.

Блок 7: Обнуление переменных.

Блок 8: Условие, выполняющиеся при ненулевом Temp.

Блок 9: Номера буфера и Top совпадают.

Блок 10: Переход к следующему после Temp элементу.

Блок 11: Заполнение существующих элементов вспомогательного списка.

Блок 12: Существуют ли данные о этом рейсе.

Блок 13: Создание нового элемента вспомогательного списка.

Блок 14: Переход к следующему после Top элементу.

Рисунок 2.6 – Алгоритм подсчёта статистики

Структурная схема на рисунке 2.7 работает следующим образом:

Блок 1: Проверка итератора i.

Блок 2: Проверка итератора i.

Блок 3: j принимает значение элемента после i.

Блок 4: Проверка итератора j.

Блок 5: Сравнение номеров информационных полей.

Блок 6: Перестановка элементов.

Блок 7: Проверка итератора i.

Блок 8: Проверка итератора j.

Блок 9: j принимает следующее значение.

Блок 10: i принимает следующее значение.



Рисунок 2.7 – Алгоритм сортировки

Структурная схема на рисунке 2.8 работает следующим образом:

Шаг 1: Считывание номера изменяемого элемента

Шаг 2: Нахождение указателя элемента по введённому номеру

Шаг 3: Верно, если Temp ни на что не указывает.

Шаг 4: Выбор поля редактирования

Шаг 5: Ввод нового значения

Шаг 6: Сохранение изменений



Рисунок 2.8 – Алгоритм корректировки.

Структурная схема на рисунке 2.9 работает следующим образом:

Шаг 1: Есть ли первый элемент.

Шаг 2: Одинаковые ли строки

Шаг 3: Проверка, равен ли i нулю.

Шаг 4: Печать заголовка таблицы.

Шаг 5: Печать элемента

Шаг 6: Перейти к следующему элементу



Рисунок 2.9 – Алгоритм поиска элемента

Структурная схема на рисунке 2.10 работает следующим образом:

Шаг 1: Является ли Top нулевым.

Шаг 2: Есть ли элемент после Top.

Шаг 3: Запоминание Top в Temp.

Шаг 4: Удаление элемента Top и обнуление указателя.

Шаг 5: Если текущий элемент 3 с конца.

Шаг 6: Переход к следующему элементу.

Шаг 7: Удаление следующего после Temp элемента.



Рисунок 2.10 – Алгоритм очистки памяти

1. ВЫПОЛНЕНИЕ ПРОГРАММЫ
   1. Условие выполнение программы

Программа представлена файлом OS.exe. Её можно запустить также, как и любую другую программу данного расширения: двойной клик правой кнопкой мыши или, выделив иконку, нажать клавишу Enter.

При вводе имени файла можно не указывать расширение, если файл находится в директории с исполняемым файлом. Аналогично, если не указывать путь, то все файлы для сохранения данных будут попадать в ту же папку.

Также для выполнения программы необходим минимальный состав аппаратных и программных средств:

- центральный процессор, производительностью не менее 500 МГц;

- видеокарта, с объёмом памяти не менее 128 Мб;

- монитор;

- клавиатура.

* 1. Загрузка и запуск программы

Запустив программу, на экран выводится следующее (рис.3.1):

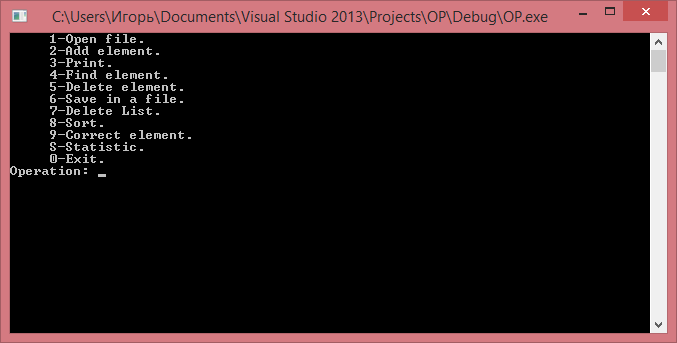


Рисунок 3.1 – Меню программы

Необходимо вывести символ из представленных на экране. В случае ввода некорректной комбинации, экран будет очищен и меню будет заново выведено на экран. Ввод является регистрозависимым. Выберем пункт добавления нового элемента (рис.3.2)

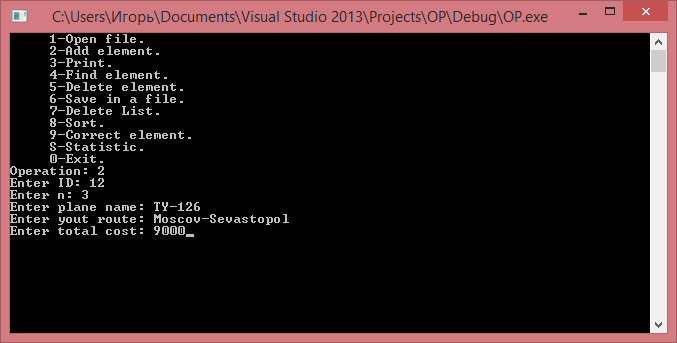


Рисунок 3.2 – Ввод данных с клавиатуры

Было введено 5 параметров, 2 из которых были символьными, а остальные - числовыми. При вводе символов или значений больше лимита, программа отменяет предыдущий ввод и повторяет его снова.

При вводе элемента с уже существующим номеров, на экран будет выведено соответствующее сообщение, после чего надо будет ввести новые данные до тех пор, пока они не будут повторяться с уже существующими. (рис.3.3)

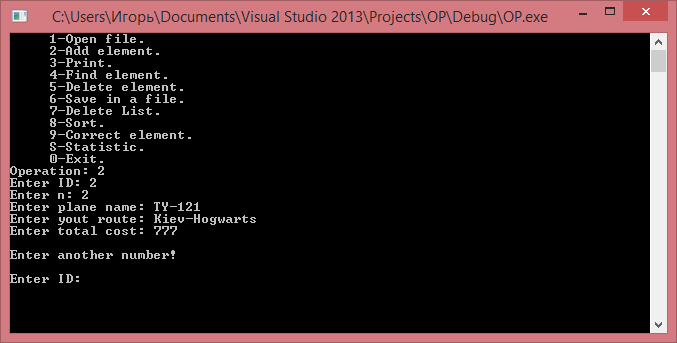


Рисунок 3.3 - Ввод дубликата

Выведем на экран таблицу данных, после заполнения методом, описанным ранее. (рис.3.4)

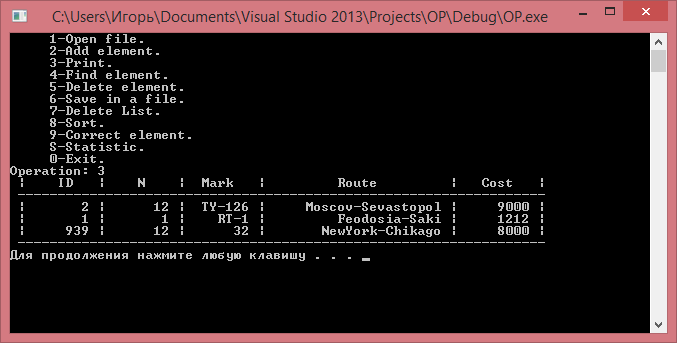


Рисунок 3.4 – Печать таблицы значений

Отсортируем данные, введя в меню «8». После этого запишем данные как в текстовый, так и в бинарный файлы. Для вывода в текстовый файл в поле вводе имени файла необходимо ввести данные в формате «\*.txt». Во всех других случаях данные запишутся в бинарный файл. (рис.3.5)

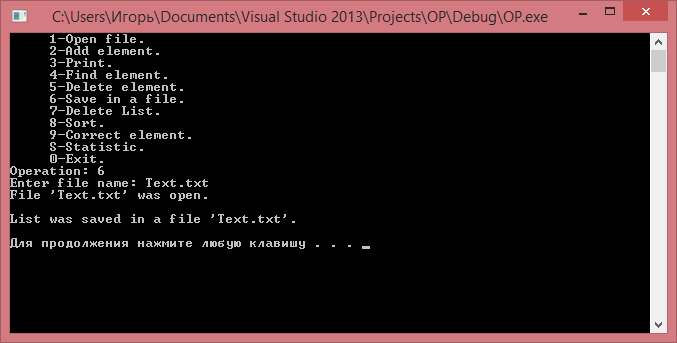


Рисунок 3.5 – Сохранение в текстовый файл

В данном случае, содержимое файла Text.txt будет выглядеть следующим образом:

2 12 TY-126 Moscov-Sevastopol 9000.000000

1 1 RT-1 Feodosia-Saki 1212.000000

939 12 32 NewYork-Chikago 8000.000000

Эти же данные можно считать. Внесём изменения в файле. При открытии файла все данные списка будут удалены. Сохраним их перед этим в бинарный файл. Далее выполняется загрузка 2 файлов. Получаем следующий список (рис.3.6)

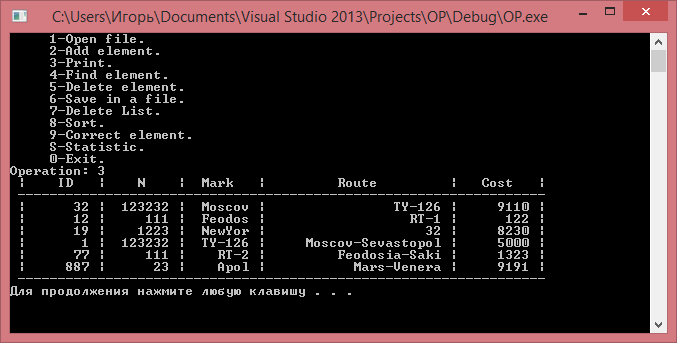


Рисунок 3.6 – Данные списка

Вызовем функцию поиска элемента. Для этого необходимо ввести марку самолёта, и все записи, в которых будет иметься совпадение, будут выведены на экран. (рис.3.7)

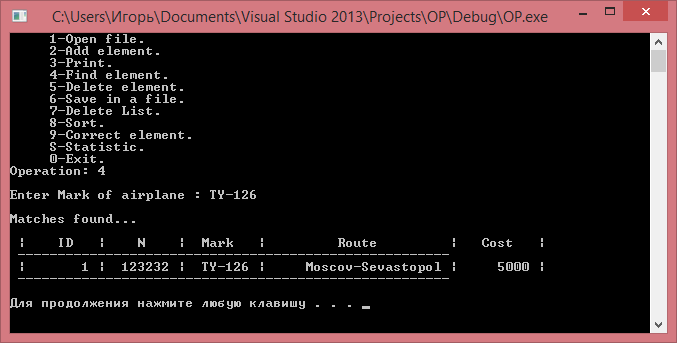


Рисунок 3.7 – Поиск элемента

При помощи ввода заглавной «S» вызовем функцию, отвечающую за статистические данные. Количество полей соответствует количеству разных номеров рейсов. Первая столбец обозначает номер рейса, второй- количество билетов на рейс, третий- суммарная стоимость билетов, четвёртый- среднее арифметическое цен билетов. (рис.3.8)

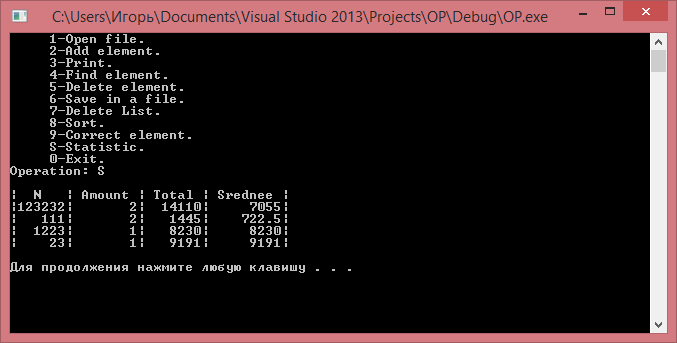


Рисунок 3.8 –Статистические данные

Используя удаление, применим его на записях, которые состоят только из цифр. После этого, при помощи функции редактирования «9» изменим данные, чтобы у всех рейсов был один номер. (рис. 3.9)

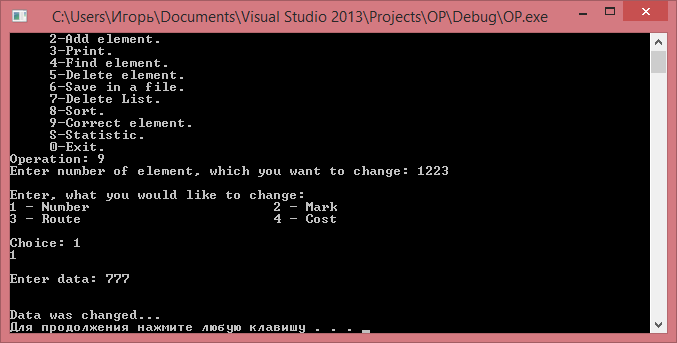


Рисунок 3.9 – Изменение элемента

В любой момент работы программы есть возможность удалить список. Для этого необходимо ввести «7» с клавиатуры. Для корректного выхода из программы, с очисткой списка, необходимо нажать ноль.

ВЫВОДЫ

Написанная программа была составлена в соответствии с постановкой задачи на курсовое проектирование. При написании программы использовались методические указания по дисциплине «Основы программирования и алгоритмические языки».

Интерфейс данной программы реализован на английском языке, что делает её универсальной. Минимизированы все сложности использования программы для пользователя. В частности, при вводе имени файла, программа сама определяет, работать ли с ним как с текстовым, или как с бинарным. Каждый этап работы программы сопровождается подсказками и описаниями состояния работы программы. После завершения, программа удаляет следы своей работы, очищая за собой динамическую память.

Тестирование и отладка программы показали, что программа решает все задачи, поставленные в техническом задании. Код полностью документирован и может быть модифицирован любым сторонним разработчиком. При необходимости можно крайне удобно модифицировать отдельные файлы, без вреда остальному коду.

СПИСОК ЛИТЕРАТУРЫ

1. Аверкин В. Программирование на C++Второе издание/ В. Аверкин, А. Бобровский, В. Веснич — Корона-Принт, Альтекс ,2003,512с.

2. Пахомов Б.Самоучитель C/C++ и C++ Builder 2007/ Б. Пахомов : БХВ-Петербург, 2008- 670 с.

3. Страуструп Б. Язык программирования C++Специальное издание/ Б.Страуструп —Бином, 2011, 1136с.

ПРИЛОЖЕНИЕ А

(обязательное)

Код программы

Файл Main.cpp:

#include "Menu.h"

int main(){

L \*Top = NULL;

menu(&Top);

return 0;

}

Файл Menu.h:

#pragma once

#include "Add.h"

#include "File\_open.h"

#include "Print.h"

#include "Delete.h"

#include "Search.h"

#include "File\_save.h"

#include "Clean\_memory.h"

#include "Sort.h"

#include "Correct.h"

#include "Statistic.h"

int menu(L\*\* Top);

Файл Menu.cpp:

#include "Menu.h"

int menu(L\* \*Top){

char m;

L\* End = \*Top;

while (1){

system("CLS");

std::cout << " 1-Open file." << std::endl;

std::cout << " 2-Add element." << std::endl;

std::cout << " 3-Print." << std::endl;

std::cout << " 4-Find element." << std::endl;

std::cout << " 5-Delete element." << std::endl;

std::cout << " 6-Save in a file." << std::endl;

std::cout << " 7-Delete List." << std::endl;

std::cout << " 8-Sort." << std::endl;

std::cout << " 9-Correct element." << std::endl;

std::cout << " S-Statistic." << std::endl;

std::cout << " 0-Exit." << std::endl;

std::cout << "Operation: ";

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cin >> m;

if (std::cin.rdbuf()->in\_avail() > 1) continue;

switch (m){

case '1':{

Start(\*Top,End);

break;

}

case '2':{

Add\_L(\*Top,End);

break;

}

case '3':{

Print\_L(\*Top);

break;

}

case '4':{

Search\_L(\*Top);

break;

}

case '5':{

Delete\_L(\*Top,\*Top);

break;

}

case '6':{

File\_save(\*Top);

break;

}

case '7':{

Clean\_memory(\*Top);

break;

}

case '8':{

Sort(\*Top);

break;

}

case '9':{

Correct(\*Top);

break;

}

case 'S':{

Statistic(\*Top);

break;

}

case '0': {

Clean\_memory(\*Top);

return 0;

break;

}

system("PAUSE");

}

}

return -1;

}

Файл Struct.h:

#ifndef \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#endif

#pragma once

#include <iostream>

#include <iomanip>

//информационное поле списка

struct fly{

int ID;

int n;

char route[21];///////////////////

char mark[7];

double cost;

};

//односвязный список

struct L{

fly data;

L\* next;

};

Файл Statistic.h:

#pragma once

#include "Struct.h"

//в структуре будут содержаться статистические данные о каждом рейсе

struct Average{

int n;

int amount;

double sum;

double sred;

Average\* next;

};

//подсчёт статистики по указанному заданию

void Statistic(L\* Top);

//очистка памяти от вспомогательного списка

void Average\_Clean(Average\*& First);

//печать значений вспомогательного списка

void Average\_Print(Average\* First);

//создание нового элемента

Average\* Average\_New(L\* Top);

Файл Statistic.cpp:

#include "Statistic.h"

void Statistic(L\* Top){

//подсчёт средней стоимости пасажиров на рейсе

Average \*First = NULL;

Average \*Temp = NULL;

Average \*Last = NULL;

int flag = 0;

if (Top != NULL){

First = new Average;

First = Average\_New(Top);

Last = First;

}

else return;

if (Top->next != NULL){

Top = Top->next;

while (Top != NULL){

Temp = First;

flag = 0;

while (Temp != NULL){

if (Top->data.n == Temp->n){

flag = 1;

Temp->amount++;

Temp->sum += Top->data.cost;

break;

}

Temp = Temp->next;

}

if (flag == 0){

Last->next = new Average;

Last->next = Average\_New(Top);

Last = Last->next;

}

Top = Top->next;

}

}

//подсчёт среднего значения цены

Temp = First;

while (Temp != NULL){

Temp->sred = Temp->sum / (double)Temp->amount;

Temp = Temp->next;

}

Average\_Print(First);

Average\_Clean(First);

delete First;

delete Temp;

system("PAUSE");

}

void Average\_Clean(Average\*& First){

Average\* Temp = NULL;

while (First != NULL){

//в наличии только первый элемент списка

if (First->next == NULL){

delete First;

break;

}

//для случая от 2 элементов и больше

Temp = First;

while (Temp->next->next != NULL){

Temp = Temp->next;

}

delete Temp->next;

Temp->next = NULL;

}

First = NULL;

}

void Average\_Print(Average\* First){

if (First != NULL){

std::cout << std::endl << "| N | Amount | Total | Srednee |" << std::endl;

while (First!=NULL){

std::cout << "|" << std::setw(6) << First->n << "|" << std::setw(8) << First->amount <<

"|" << std::setw(7) << First->sum << "|" << std::setw(9) << First->sred << "|" << std::endl;

First = First->next;

}

std::cout << std::endl;

}

}

Average\* Average\_New(L\* Top){

Average\* Temp = NULL;

Temp = new Average;

Temp->sred = 0;

Temp->amount = 1;

Temp->n = Top->data.n;

Temp->sum = Top->data.cost;

Temp->next = NULL;

return Temp;

}

Файл Sort.h:

#pragma once

#include "Struct.h"

//сортировка списка по возростанию номера

void Sort(L\* &Top);

/\*

Суть алгоритма в том, что мы переприсваиваем в непустом списке информационные поля

по возрастанию номера. Данная сортировка основана на сортировке пузырька, однако счётчикам

циклов в данном случае есть правый и левый указатели, которые ограничиваются тем, что не

должны быть равны NULL.

\*/

Файл Sort.cpp:

#include "Sort.h"

//сортировка списка по возрастанию номера

void Sort(L\* &Top){

if (Top == NULL) return;

L\* i = Top;

L\* j = NULL;

fly temp;

while (i != NULL){

if (i != NULL) j = i->next;

while (j != NULL){

if (i->data.n > j->data.n){

temp = i->data;

i->data = j->data;

j->data = temp;

}

if (j != NULL) j = j->next;

}

if (i != NULL) i = i->next;

}

std::cout << std::endl << "List was sorted." << std::endl << std::endl;

system("PAUSE");

}

Файл Search.h:

#pragma once

#include "Struct.h"

#include "Print.h"

//функция поиска элемента с вводом данных

void Search\_L(L\* Top);

//функция, которая реализует поиск, нет ввода параметров

void Search(L\* Top, char\* \_mark);

//поиск по ключевому полю

L\* Search\_n(L\* Top, int \_n);

Файл Search.cpp:

#pragma once

#include "Search.h"

//основной вызов функции поиска с передачей параметров

void Search\_L(L\* Top){

char \_mark[7] = "";

std::cout << std::endl << "Enter Mark of airplane : ";

std::cin.clear();

std::cin.ignore();

std::cin.getline(\_mark,7);

Search(Top,\_mark);

system("PAUSE");

return;

}

//функция поиска

void Search(L\* Top, char\* \_mark){

int i = 0;//bool

while (Top != NULL){

if (strcmp(\_mark, Top->data.mark) == 0){

if (!i++) {

std::cout << std::endl << "Matches found..." << std::endl << std::endl;

Table\_title();

std::cout << " ------------------------------------------------------" << std::endl;

}

print\_element(Top->data);

}

Top = Top->next;

}

if (i == 0){

std::cout << std::endl << "Matches not found..." << std::endl << std::endl;

}

else{

std::cout << " ------------------------------------------------------" << std::endl << std::endl;

}

return;

}

//поиск не по ключевому полю

L\* Search\_n(L\* Top, int \_n){

while (Top!=NULL){

if (Top->data.n == \_n) return Top;

Top = Top->next;

}

return NULL;

}

Файл Print.h:

#pragma once

#include "Struct.h"

//печать данных одного элемента

inline void print\_element(fly \_new);

//заголовок таблицы

inline void Table\_title();

//печать всех элементов

void Print\_L(L\* Top);

Файл Print.cpp:

#include "Print.h"

//единичная печать

inline void print\_element(fly \_new){

\_new.mark[6] = '\0';

\_new.route[20] = '\0';

std::cout << " | ";

std::cout << std::setw(7) << \_new.ID << " | ";

std::cout << std::setw(7) << \_new.n << " | ";

std::cout << std::setw(7) << \_new.mark << " | ";

std::cout << std::setw(21) << \_new.route << " | ";

std::cout << std::setw(8) << \_new.cost << " | " << std::endl;

return;

}

//заголовок таблицы

inline void Table\_title(){

std::cout << " | " << std::setw(7) << " ID " << " | " << std::setw(7) << " N " << " | " <<

std::setw(7) << "Mark " << " | ";

std::cout << std::setw(21) << "Route " << " | " << std::setw(8) << "Cost " << " | " << std::endl;

}

//вызов функции печати всех элементов

void Print\_L(L\* Top){

Table\_title();

std::cout << " ------------------------------------------------------------------" << std::endl;

while (Top!=NULL){

print\_element(Top->data);

Top = Top->next;

}

std::cout << " ------------------------------------------------------------------" << std::endl;

system("PAUSE");

return;

}

Файл File\_save.h:

#pragma once

#include "Struct.h"

//сохранение списка в файле

void File\_save(L\* Top);

Файл File\_save.cpp:

#include "File\_save.h"

//сохранение списка в файле

void File\_save(L\* Top){

FILE\* f;

char fname[20];

int flag = 0;

std::cin.ignore();

std::cin.clear();

std::cout << "Enter file name: ";

std::cin.getline(fname, 20);

std::cin.ignore(std::cin.rdbuf()->in\_avail());

//проверка на текстовый файл

for (int i = 19; i > 3; i--){

if (fname[i] == 't' && fname[i - 1] == 'x' && fname[i - 2] == 't' && fname[i - 3] == '.'){

flag = 1;

break;

}

}

//открытие файла

if ((flag) ? f = fopen(fname, "w+t") : f = fopen(fname, "w+b")){

std::cout << "File '" << fname << "' was open. " << std::endl;

}

else{

std::cout << "File '" << fname << "' was not open. " << std::endl;

system("PAUSE");

return;

}

if (flag == 0){

while (Top){

fwrite(&(Top->data), sizeof(fly), 1, f);

Top = Top->next;

}

}

else{

while (Top){

fprintf(f, "%d ", Top->data.ID);

fprintf(f, "%d ", Top->data.n);

fwrite(&" ", sizeof(char), 1, f);

fputs(Top->data.mark, f);

fwrite(&" ", sizeof(char), 1, f);

fputs(Top->data.route, f);

fwrite(&" ", sizeof(char), 1, f);

fprintf(f, "%f\n", Top->data.cost);

Top = Top->next;

}

}

std::cout << std::endl <<"List was saved in a file '" << fname << "'." << std::endl << std::endl;

fclose(f);

system("PAUSE");

}

Файл File\_open.h:

#pragma once

#include "Struct.h"

#include "Clean\_memory.h"

//считывание данных из файла

void Start(L\*& Top,L\* &End);

Файл File\_open.cpp:

#include "File\_open.h"

//считывание данных из файла в список

void Start(L\* &Top, L\*& End){

L\* Temp = NULL;

fly c,temp;

FILE\* f;

char fname[20];

char ch;

char str[90]="", str\_temp[30]="";

int flag = 0, i = 0;

Clean\_memory(Top);

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cout << "Enter file name: ";

std::cin.getline(fname, 20);

std::cin.ignore(std::cin.rdbuf()->in\_avail());

//проверка на текстовый файл

for (int i = 19; i > 3; i--){

if (fname[i] == 't' && fname[i - 1] == 'x' && fname[i - 2] == 't' && fname[i - 3] == '.'){

flag = 1;

break;

}

}

if ((flag) ? f = fopen(fname, "r+t") :f = fopen(fname, "r+b")){

std::cout << "File '"<< fname <<"' was open. " << std::endl;

}

else{

std::cout << "File '" << fname << "' was not open. " << std::endl;

system("PAUSE");

return;

}

i = 0;

//если файл текстовый, то выполняется следующие условие

if (flag == 1){

std::cout <<std::endl << "File has text format." << std::endl;

while (fread(&ch, sizeof(char), 1, f)){

/////////////////////////////////////////////////////////////////////////////////////////////////////

if ((ch == '\0' || ch == '\n') && i != 0){

i = 0;

//заполнение инф поля в temp

for (; str[i] == ' '; i++);//игнорирование пробелов

//int

for (int j = 0; str[i] != ' '; j++, i++){

str\_temp[j] = str[i];

str\_temp[j + 1] = '\0';

}

temp.ID = atoi(str\_temp);

for (; str[i] == ' '; i++);//игнорирование пробелов

//int

for (int j = 0; str[i] != ' '; j++, i++){

str\_temp[j] = str[i];

str\_temp[j + 1] = '\0';

}

temp.n = atoi(str\_temp);

for (; str[i] == ' '; i++);//игнорирование пробелов

//route

for (int j = 0; str[i] != ' '; j++, i++){

str\_temp[j] = str[i];

str\_temp[j + 1] = '\0';

}

for (int j = 0; str\_temp[j] != '\0' && j < 21; j++){

temp.route[j] = str\_temp[j];

temp.route[j + 1] = '\0';

}

for (; str[i] == ' '; i++);//игнорирование пробелов

//mark

for (int j = 0; str[i] != ' '; j++, i++){

str\_temp[j] = str[i];

str\_temp[j + 1] = '\0';

}

for (int j = 0; str\_temp[j] != '\0' && j < 7; j++){

temp.mark[j] = str\_temp[j];

temp.mark[j + 1] = '\0';

}

for (; str[i] == ' '; i++);//игнорирование пробелов

//double

for (int j = 0; str[i] != ' ' && str[i] != '\n' && str[i] != '\0'; j++, i++){

str\_temp[j] = str[i];

str\_temp[j + 1] = '\0';

}

temp.cost = atof(str\_temp);

//обнуление счётчика цикла

i = 0;

//создание элемента в списке

//предварительная проверка на уже такой же ID

L\* Top\_t = Top;

int flag\_t = 0;

while (Top\_t != NULL){

if (Top\_t->data.ID == temp.ID){

flag\_t = 1;

break;

}

Top\_t = Top\_t->next;

}

if (flag\_t == 1) continue;

/////////////////////////////////////////////

Temp = new L;

Temp->data = temp;

Temp->next = NULL;

if (Top == NULL){

Top = Temp;

End = Top;

continue;

}

End->next = Temp;

End = End->next;

continue;

/////////////////////////////////////////////////////////////////////////////////////////////////////

}

str[i++] = ch;

str[i] = '\0';

//проверка на пустую строку

if (str[0] == '\0' || str[0] == '\n'){

i = 0;

ch = 'a';

}

}

fclose(f);

}

else{

//файл бинарный типизированный

while (fread(&c, sizeof(fly), 1, f)){

Temp = new L;

Temp->data = c;

Temp->next = NULL;

if (Top == NULL){

Top = Temp;

End = Top;

continue;

}

End->next = Temp;

End = End->next;

}

if (Top == NULL){

std::cout << "File is empty..." << std::endl;

}

}

fclose(f);

system("PAUSE");

return;

}

Файл Delete.h:

#include "Struct.h"

//вызов функции удаления с передачей ей параметров

void Delete\_L(L\*& Top, L\* &End);

//функция удаление элемента без ввода данных

void Delete(L\* &Top, L\* &End, int \_ID);

Файл Delete.cpp:

#include "Delete.h"

//вызов функции удаления с передачей ей параметров

void Delete\_L(L\* &Top, L\* &End){

int \_ID = 0;

std::cout << std::endl << "Enter ID of airplane to delete it: ";

std::cin.clear();

std::cin.ignore();

std::cin >> \_ID;

Delete(Top, End, \_ID);

system("PAUSE");

return;

}

//функция удаление элемента без ввода данных

void Delete(L\*& Top, L\* &End, int \_ID){

L\* Temp = Top;

int flag = 1;

//первый элемент

if (Top->data.ID == \_ID){

Temp = Top;

Top = Top->next;

delete Temp;

std::cout << std::endl << "Element was deleted. " << std::endl;

return;

}

//все остальные непервые элементы

while (Temp->next != NULL){

if (Temp->next->data.ID == \_ID){

//удаляемый элемент последний

if ((Temp->next->data.ID == \_ID) && (Temp->next->data.ID == End->data.ID)){

End = Temp;

Temp = Temp->next;

std::cout << std::endl << "Element was deleted. " << std::endl;

delete Temp;

return;

}

flag = 0;

L\* Temp2 = Temp->next;

Temp->next = Temp->next->next;

delete Temp2;

std::cout << std::endl << "Element was deleted. " << std::endl;

return;

}

Temp = Temp->next;

}

//необходимый элемент не был найден в списке

if (flag){

std::cout << "Element was not found." << std::endl;

}

return;

}

Файл Correct.h:

#pragma once

#include "Struct.h"

#include "Search.h"

//изменение поля в записи

void Correct(L\* &Top);

Файл Correct.cpp:

#include "Correct.h"

void Correct(L\* &Top){

char ch;

int \_n;

L\*Temp = NULL;

fly temp;

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cout << "Enter number of element, which you want to change: ";

std::cin >> \_n;

std::cin.ignore(std::cin.rdbuf()->in\_avail());

Temp = Search\_n(Top, \_n);

if (Temp == NULL) {

std::cout << std::endl << "Element not found." << std::endl;

system("PAUSE");

return;

}

std::cout << std::endl << "Enter, what you would like to change: " << std::endl;

std::cout << "1 - Number 2 - Mark" << std::endl;

std::cout << "3 - Route 4 - Cost" << std::endl << std::endl;

std::cout << "Choice: ";

std::cin.clear();

std::cin.ignore();

std::cin >> ch;

std::cout << std::endl << "Enter data: ";

std::cin.clear();

std::cin.ignore();

temp = Temp->data;

switch (ch){

case '1': std::cin >> temp.n; break;

case '2': std::cin.getline(temp.mark, 7); break;

case '3': std::cin.getline(temp.route, 21); break;

case '4': std::cin >> temp.cost; break;

}

Temp->data = temp;

std::cout << std::endl << std::endl << "Data was changed..." << std::endl;

system("PAUSE");

}

Файл Clean\_memory.h:

#pragma once

#include "Struct.h"

//очистка динамической памяти

void Clean\_memory(L\* &Top);

Файл Clean\_memory.cpp:

#include "Clean\_memory.h"

//очистка динамической памяти

void Clean\_memory(L\* &Top){

L\* Temp = NULL;

while (Top != NULL){

//в наличии только первый элемент списка

if (Top->next == NULL){

delete Top;

break;

}

//для случая от 2 элементов и больше

Temp = Top;

while (Temp->next->next != NULL){

Temp = Temp->next;

}

delete Temp->next;

Temp->next = NULL;

}

Top = NULL;

std::cout << std::endl << "Memory was cleaned. " << std::endl << std::endl;

system("PAUSE");

}

Файл Add.h:

#pragma once

#include "Struct.h"

//добавление нового элемента в список

void Add\_L(L\* &Top, L\* &End);

//считывание данных нового файла

fly element(void);

Файл Add.cpp:

#include "Add.h"

//создание новых данных

fly element(void){

fly \_new;

std::cout << "Enter ID: ";

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cin >> \_new.ID;

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cout << "Enter n: ";

std::cin >> \_new.n;

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cout << "Enter plane name: ";

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cin.getline(\_new.mark, 7);

std::cout << "Enter your route: ";

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cin.getline(\_new.route, 21);

std::cin.ignore(std::cin.rdbuf()->in\_avail());

std::cout << "Enter total cost: ";

std::cin >> \_new.cost;

std::cin.ignore(std::cin.rdbuf()->in\_avail());

return \_new;

}

//добавление нового элемента в список

void Add\_L(L\* &Top, L\* &End){

int found = 0;

L\* Temp\_found = NULL;

L\* Temp = new L;

//цикл предназначен для избежание одинаковых ID записей

do{

found = 0;

Temp->data = element();

Temp->next = NULL;

if (Top == NULL) break;

Temp\_found = Top;

while (Temp\_found != NULL){

if (Temp\_found->data.ID == Temp->data.ID){

found = 1;

break;

}

Temp\_found = Temp\_found->next;

}

if (!found) break;

std::cout << std::endl <<"Enter another number! " << std::endl << std::endl;

} while (1);

if (Top == NULL){

Top = Temp;

End = Top;

return;

}

End->next = Temp;

End = End->next;

}