**Введение**

Базы данных всегда были важнейшей темой при изучении информационных систем. Однако в последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Технологии баз данных увели Интернет-приложения далеко от простых брошюрных публикаций, которые характеризовали ранние приложения. В то же время Интернет-технология обеспечивает пользователям стандартизированные и доступные средства публикации содержимого баз данных. Правда, ни одна из этих новых разработок не отменяет необходимости в классических приложениях баз данных, которые появились еще до развития Интернета для нужд бизнеса. Это только расширяет важность знания баз данных.

Цель базы данных — помочь людям и организациям вести учет определенных вещей. На первый взгляд, эта цель кажется скромной, и вы, возможно, удивитесь, зачем нам нужна такая сложная технология и целый курс, посвященный этому предмету. Большинство из нас может вспомнить ситуации, в которых нам требуется отслеживать некоторые вещи. На самых ранних стадиях развития информационных технологий использовались списки — набитые на перфокарте и написанные на магнитной ленте. Со временем, однако, стало ясно, что только немногие проблемы можно решить с помощью таких списков. В следующем разделе мы обсудим такие проблемы, а затем опишем, как построить базы данных для их решения.

Процесс подготовки и решения задач на компьютерах состоит из нескольких этапов:

- постановка задачи;

- построение модели;

- разработка алгоритма;

- написание и отладка программы на языке программирования;

- тестирование программы.

При разработке программ применяют метод структурного программирования: разделение задачи на относительно независимые части и выполнение отладки этих частей. При этом достигается качество и быстрота разработки программ.

Развитие вычислительной техники с самого начала было направлено на облегчение и ускорение ввода, поиска данных. Базы данных используются во многих предприятиях, отраслях науки и т.д. Разработано множество готовых языков программирования с помощью которых можно создавать базы данных.

# 1 Разработка эскизного и технического проектов программы

# 1.1 Задание

Цель курсовой работы - написать приложение, позволяющее работать со структурой данных по закупке чая содержащей следующие поля: a) порядковый номер б) тип чая в) форма упаковки г) страна производитель д) цена за шт. е) количество штук.

Результатом решения являются:

- исполняемый файл программы, полученный в режиме «Release», прилагаемый к пояснительной записке в электронном виде.

- возможность сортировать, добавлять, удалять, сохранять и загружаться данные внутри рабочей программы.

Для реализации файлового ввода-вывода в программу следует включить заголовок <fstream>. В нем определены некоторые классы, в частности, ifstream , ofstream и fstream . Эти классы являются производными от классов istream , ostream и iostream соответственно. Следует помнить, что классы istream , ostream и iostream , в свою очередь, являются производными от класса ios.

Чтение и запись текстовых файлов осуществляются очень легко. Для этого достаточно применить операторы “<<” и как это обычно делается для консольного ввода-вывода, только вместо потоков cin и cout необходимо подставить поток, связанный с файлом. Например, программа «Закупка чая» создает короткий файл, содержащий порядковый номер, тип чая, форма упаковки, страна производитель, цена за 1 шт. и количество чая.

# 1.2 Назначение и область применения

Назначение программы – «Ведомость по закупке чая» позволяет производить ввод записей о чае в базу данных, их сортировку и вывод, а также удаление записей из базы данных, сохранение в файл для последующего продолжения работы с ними или передачи другому оператору.

На первом этапе проектирования базы данных необходимо определить цель создания базы данных, основные ее функции и информацию, которую она должна содержать.

База данных разработана для чайного интернет магазина «Боффо», занимающегося покупкой, хранением и продажей продукции. Продукция поступает непосредственно с заводов-изготовителей и реализуется клиентам, нуждающимся в данном виде чая. Данная информационная система будет реализована в различных розничных и интернет магазинах чая. Пользователем этой системы будет человек, занимающийся закупкой чая, а также текущей выпиской расходных документов и оприходованием вновь поступивших товаров.

# 1.3 Технические характеристики

# 1.3.1 Постановка задачи

Необходимо написать программу, позволяющее работать со структурой данных по закупке чая содержащей следующие поля. Цель: приобретения навыков работы со структурами и базами данных, а также возможной доработкой программы и использования ее в работе с другими базами данных.

Чай имеет следующие параметры: a) порядковый номер б) тип чая в) форма упаковки г) страна производитель д) цена за 1 шт. е) количество шт.

Программа выполняет следующие операции:

1. Вывод баз данных в консоли;
2. Добавление параметров чая в базе данных;
3. Удаление параметров чая в базе данных;
4. Поиск по параметрам;
5. Сортировка по параметрам.
6. Вывод информации о разработчике программы.

# 

# 1.3.2 Описание алгоритмов

Данная программа состоит из нескольких алгоритмов, ознакомиться с ними и их описанием вы сможете ниже.

1. Программа написанная по алгоритму «Меню» управляется с помощью номера команды (рис.1). Номера команды вводится с клавиатуры и соответствуют следующим действиям:

1 – Вывод баз данных на экран;

2 – Добавление параметров чая;

3 – поиск по заданному параметру (и при желании удалить параметры найденного чая);

4 – сортировка по заданному параметру.

5 – вывод информации о разработчике программы.

Добавление чая

1. «Алгоритм для выхода в меню» изображен на рис. 2. и работает следующим образом: пока команда не равна 0, функция просит заново ввести номер команды и не выходит в меню.
2. «Алгоритм добавления параметров продуктов» изображен на рис. 3. и работает следующим образом:

3.1. Создание нового динамического массива с размером n+nAdd, где n – количество чая до добавления, nAdd - количество добавляемого чая;

3.2. Присваивание всех значении старого массива на новый;

3.3. Инициализация параметров добавлемого чая.

1. «Алгоритм удаления параметров чая» изображен на рис. 4. и работает следующим образом:
   1. Замена удаляемого элемента массива на последний;
   2. Создание нового динамического массива с размером n-1, где n – количество чая до удаления;
   3. Инициализация всех старых элементов на новый, кроме последнего.
2. «Алгоритм сортировки» изображен на рис. 5. и сортирует список по заданному параметру (по типу чая, по производителю, по цене за шт. , по форме упаковки);
3. «Алгоритм поиска» изображен на рис. 6. и ищет элемент массива по заданному параметру (по порядковому номеру, по типу чая, по форме упаковки, по производителю).

Алгоритмы

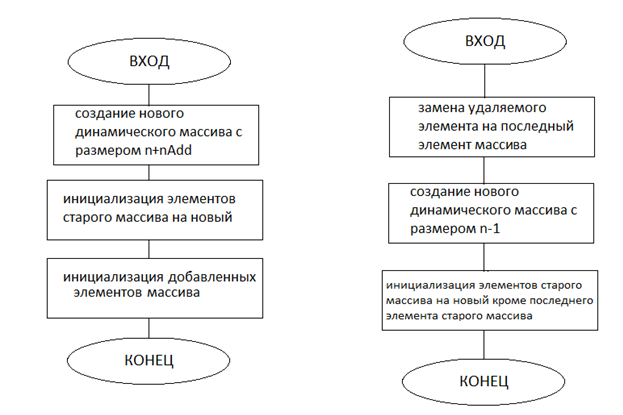
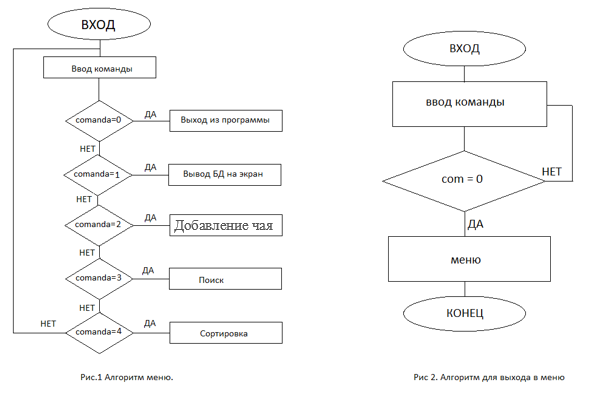


Рис.3. Алгоритм добавления параметров чая Рис.4. Алгоритм удаления параметров чая

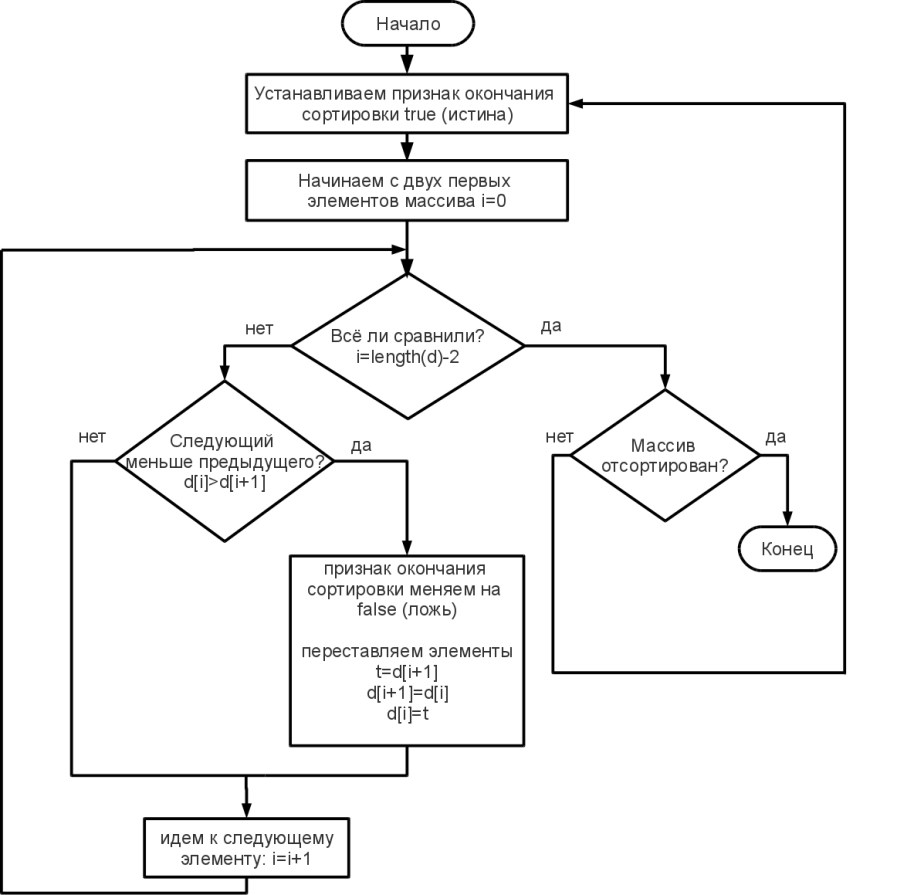


Рис. 5. Алгоритм сортировки

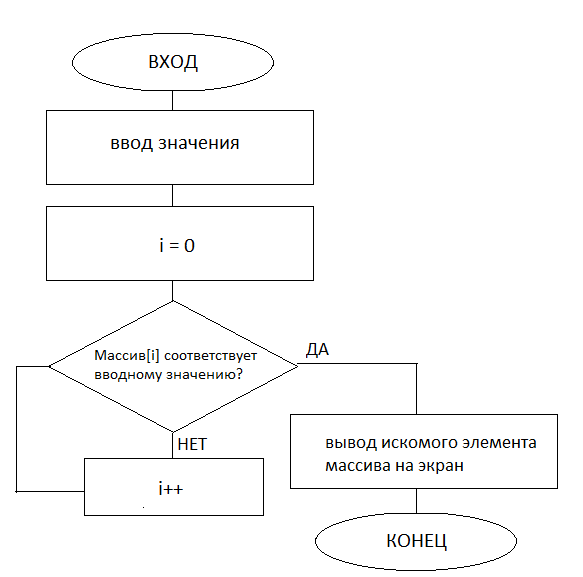


Рис. 6. Алгоритм поиска

# 1.3.3 Организация входных и выходных данных

Входные данные – данные, с которыми работает программа - Product,

которые состоят из:

1. Порядковый номер
2. Тип чая
3. Форма упаковки
4. Страна производитель
5. Цена за 1 шт.
6. Количество шт.

Для a, f, e – используется целочисленные переменные в структуре,

для b,c,d – строковые переменные.

struct Tea{

int N;

string TypeTea;

string PackingForm;

string Manufacturer;

int PriceForOne;

int Quantity;

};

Ввод переменных а, e, f осуществляется с помощью оператора cin >>, ввод переменных b, c, d осуществляется с помощью функции cin.getline().

Выходные данные – результат работы с базой данных представлен целочисленными, вещественными и строковыми переменными, выводимыми при помощи функций вывода в консоли. Вывод разбавляется текстом, выводимым при помощи cout.

Для работы программы необходимо создать файл с именем «tee.db» и в него был записан текст следующим образом: элементы следующего массива должны быть переведены на новую строку и поля структур должны быть разделены символом " ; ", так как этот символ переводит функцию на считываний следующего поля структуры, а перевод на новую строку переводит на считывании нового элемента массива и считает количество. В программе переменная n (количество строк) равна 0, при переводе на новую строку возрастает на единицу.

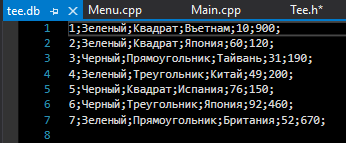


Рис. 7. Результат работы программы «Ведомость по закупкам чая».

Результат работы оператора(Рис. 7): удаление/добавление и сортировка записей в базе данных можно записать в файл tee.db.

# 1.3.4 Выбор состава технических и программных средств

Средства, используемые в данной работе, содержатся в интегрированной среде разработки (IDE) Microsoft Visual C++ 2017 (далее VS). Эта среда предоставляет набор инструментов, которые помогают в создании и изменении кода, а также в обнаружении и исправлении ошибок. В VS для организации работы служат проекты и решения. При написании программы на языке C++ с помощью VS первым этапом является выбор типа проекта. Для каждого типа проекта VS устанавливает параметры компилятора и генерирует стартовый код.

# 

# 2 Разработка рабочего проекта

# 2.1 Разработка программы

Программа «Ведомомсть по закупкам чая» предназначена для функционирования в среде Windows. Это приложение, хотя и разрабатывалось в среде визуального программирования, но использует консольное окно. Таким образом, программирование строится на процессе написания кода, придающего консольному окну и программе в целом необходимую функциональность.

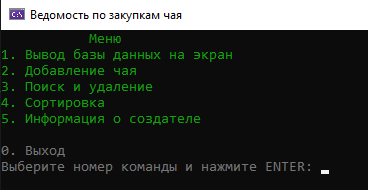


Рис. 8. Меню программы «Ведомомсть по закупкам чая».

Меню данной программы изображено на рис. 8. и объединяет нескольких функции и их вызывает введенная команда. Если номер команды не существует или вводим буквы в программе, то функция void menu(int n, Tea \*tee) вызывает самого себя.

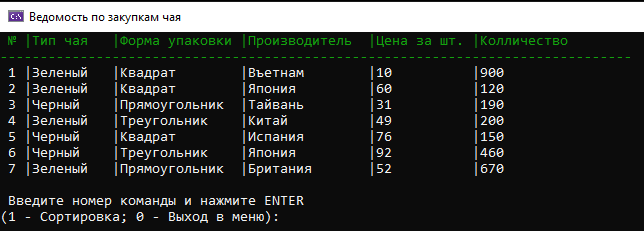


Рис. 9. Вывод баз данных в консоли.

Вывод баз данных в консоли изображен рис. 9 и осуществляется при помощи функции void Draw(int n, Tea \*tee) при выборе команд “1” в меню.

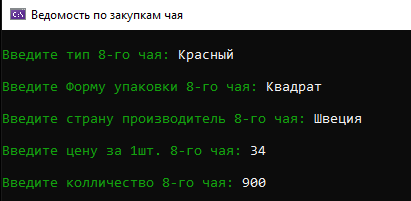


Рис. 10. Заполнение данных добавляемого чая.

Заполнение данных добавляемого чая изображено на рис. 10 и осуществляется при помощи функции Tea\* AddTea(int n, Tea \*tee) при выборе команд “2” в меню.

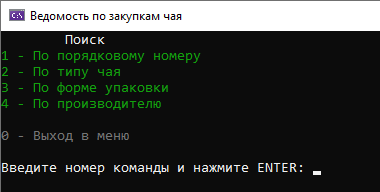


Рис. 11. Поиск.

Подменю поиска изображенo на рис. 11. Поиск чая осуществляется при помощи функции void SearchTea(int n, Tea \*tee) при выборе команд “3” в меню. Поиск осуществляется по следующим параметрам: 1)по порядковому номеру; 2)по типу чая; 3)по форме упаковки; 4)по произвдителю .

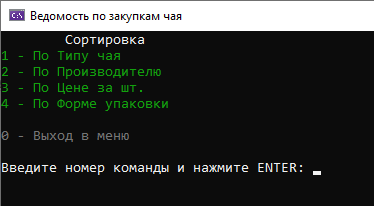


Рис. 12. Сотрировка.

Подменю сортировки изображенo на рис. 12. Сортировка чая осуществляется при помощи функции Tea \*SortTea(int n, Tea \*tee) при выборе команд “4” в меню. В данной функции используется Сортировка вставками ([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) *Insertion sort*). Сортировка осуществляется по следующим параметрам: 1)по типу чая; 2)по производителю; 3)по цене ща шт. ; 4)по форме упаковки.

При выборе сортировки по цене за шт. программа предлагает выбрать команду сортировки: 1- по возрастанию или 2 – по убыванию.

В данной функции используется Сортировка вставками.

Сортировка вставками ([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) *Insertion sort*) — [алгоритм сортировки](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. [Вычислительная сложность](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) — .

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

Данный алгоритм можно ускорить при помощи использования бинарного поиска для нахождения места текущему элементу в отсортированной части. Проблема с долгим сдвигом массива вправо решается при помощи смены указателей.

# 2.2 Спецификация программы

|  |  |  |
| --- | --- | --- |
| Обозначения | Наименование | Примечание |
| tee.exe | Исполняемая программа | Входная точка в программу |
| main.cpp | Исходный код программы | Занимается вызовом основных вычислительных функций |
| menu.cpp | Исходный код программы | Занимается вызовом основных вычислительных функций |
| Tee.h | Файл включений | Содержит все используемые в программе библиотеки, структуру и некоторые глобальные переменные. |
| tee.db | Файл базы данных | Содержит базу данных записанную туда изначально или в ходе работы программы |
|  |  |
| tee.sln | Файл проекта | Файл решений проекта (рабочая область проекта), находится в главном каталоге проекта |
| tee.iobj | Файл проекта | Файл инкрементальной генерации кода времени соединения LTCG |
| tee.ipdb | Файл проекта | Файл инкрементальной генерации кода времени соединения LTCG |
| tee.pdb | Файл проекта | Файл инкрементальной генерации кода времени соединения LTCG |
| tee.vcxproj | Файл проекта | Файл проекта. Хранит информацию, относящуюся к каждому проекту. (В более ранних версиях этот файл назывался Имя\_проекта.vcproj или Имя\_проекта.dsp |
| tee.vcxproj.filters | Файл проекта | Файл фильтров. Указывает, куда поместить файл, который добавляется в решение. Например, H-файл помещается в узле **Файлы заголовков**. |
| tee.vcxproj.user | Файл проекта | Файл пользователя миграции. После миграции проекта из Visual Studio 2008 в этом файле появляются данные, преобразованные из любых VSPROPS-файлов. |
| tee.suo | Файл проекта | Файл параметров решения. Сохраняет настройки решения, чтобы при любом открытии проекта или файла в решении оно выглядело и вело себя необходимым образом. |

# 

# 2.3 Описание программы

При создании программы «Ведомость по закупкам чая» использовался модульный принцип построения программ. Все исходные модули логически разделены на файлы. Программа состоит из трех файлов: «main.cpp», «menu.cpp», «tee.h». Такое разделение позволяет поместить каждую функцию в отдельный .cpp файл что делает код удобным для восприятия и работы.

Заголовочный файл tee.h» содержит прототипы всех функций, все подключаемые библиотеки, описание структуры и глобальной переменной n. Далее этот файл подключается ко всем функциям.

Файл «main.срр» - главный, является по своей сути управляющим. Здесь осуществляется вход в программу. Этот модуль содержит настройки консольного окна, локализацию, объявления массивов структур, а так же вызывает функцию menu, находящуюся в файле «menu.cpp»

Программа состоит из следующих функции:

1. main () – точка старта программы.
2. void menu(int n, Tea \*tee) – управляется номером команды для использования нужных команд. Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
3. void WriteBeginText() – создает файл и записывает строк. Входной параметр – Отсутствует; Выходной параметр – Отсутствует.
4. void to\_menu(int n, Tea \*tee) – предназначена для выхода в меню. Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
5. void gotoxy(int x, int y) – выравнивает баз данных в консоли. Входной параметр – Задающие координаты для выравнивания БД в консоли; Выходной параметр – Отсутствует.
6. int number() – считает количество строк из файла. Входной параметр – Отсутствует; Выходной параметр – Количество продуктов.
7. Product \*Initialization(int n, Tea \*tee) – инициализирует массив структуры данных. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
8. void RecordNewTeaFile(int n, Tea \*tee) – записывает в файл изменения. Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
9. Product \*Detete(int n, Tea \*tee, int comanda\_del) – удаляет comanda\_del-го элемента массива Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
10. Product \*AddTea(int n, Tea \*tee) – предназначена для добавления нужного количество элементов массива структуры данных. Входной параметр – Массив структуры данных. Выходной параметр – Отсутствует.
11. void Draw(int n, Tea \*tee) – выводит баз данных в консоли. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
12. void SearchTea(int n, Tea \*tee) – предназначена для поиска элемента массива(и при желаний для дальнейшего удаления элемента массива). Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
13. void SortStr(string \*S, int N, Tea \*tee) – сортирует структуру по алфавиту по заданным параметрам. Входные параметры – 1.Массив структуры данных; 2. Массив строк; Выходной параметр - Отсутствует.
14. Tea \*SortTea(int n, Tea \*tee) – сортирует структуру по алфавиту/значению по заданным параметрам. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
15. void information(int n, Tea \*tee) - выводит информацию о разработчике программы. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.

**2.4 Тестирование программы**

Тестирование (testing) программного обеспечения (ПО) - это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. В более широком смысле, тестирование ПО - это техника контроля качества программного продукта, включающая в себя проектирование тестов, выполнение тестирования и анализ полученных результатов.

# 

# 3 Внедрение

Программа «Ведомость по закупке чая» - консольное приложение, занимает объем не более 38 КБ. Для ее выполнения необходим минимальный состав аппаратурных средств: операционная система Windows 7, память 4 Мбайт, память видеокарты 512 Кбайт, жесткий диск 544 Мбайт, экран, клавиатура, мышь.

Для работы программы необходимо создать файл с именем «tee.db» и в него был записан текст следующим образом: элементы следующего массива должны быть переведены на новую строку и поля структур должны быть разделены символом “;”, так как этот символ переводит функцию на считывание следующего поля структуры, а перевод на новую строку переводит на считывании нового элемента массива и считает количество. В программе переменная сначала n=0, при переводе на новую строку возрастает на единицу больше.

При наличии файла «tee.db» программа вызывает функцию void menu(int n, Tea \*tee), которая объединяет нескольких функции и загружает соответствующего массива структур. Функция void menu(int n, Tea \*tee) позволяет удалить и добавить элементы при помощи следующей функции Tea \*AddTea(int n, Tea \*tee) и Tea \*Detete(int n, Tea \*tee, int comanda\_del). comanda\_del – индекс массива который сначала заменяется на последний элемент массива, потом удаляется.

При добавлении данный программа пишет следующее сообщение " Данные добавленного чая сохранены", а при удалении – "Данные удалены успешно".

После выполнения любой команды программа позволяет вернуть в главное меню с помощью функции void to\_menu(int n, Tea \*tee), которая выводит следующее сообщение "Для выхода в меню введите "0" и нажмите ENTER: ".

# 

# Список использованной литературы

1. Архангельский, А.Я. Язык C++ в C++ Builder : справочное и методическое пособие / А.Я. Архангельский .-- М. : Бином, 2017 – 942 с.

1. Бьерн Страуструп – Язык программирования C++. Специальное издание, 2017

3. Дейтел, Х.М.; Дейтел, П.Дж. Как программировать на С++; М.: Бином; Издание 4-е, 2015. **- 390** c

1. Саттер, Герб Новые сложные задачи на C++; Вильямс, 2016. - 272 c.
2. Кубенский, А.А. Структуры и алгоритмы обработки данных: объектно- ориентированный подход и реализация на С++ / А.А. Кубенский,- СПб.: БХВ- Петербург, 2004- 464 с.
3. Спольски Дж. Джоэл: и снова о программировании: Пер. с англ. — СПб.: Символ-Плюс, 2015. — 320 с.
4. А. Мешков, Ю. Тихомиров «Visual C++ и MFC» - СПб.: БХВ-Петербург. 2016 — 1017с.

# Приложение

# Листинг программы

# «База данных Ведомости по закупке чая»

**=================Заголовочный файл tee.h====================**

#ifndef Tee

#define Tee

#include<stdio.h>

#include<conio.h>

#include<string>

#include<iostream>

#include<fstream>

#include<windows.h>

using namespace std;

struct Tea{

int N;//Порядковый номер записи

string TypeTea;//тип чая

string PackingForm;//форма упаковки

string Manufacturer;//страна производитель

int PriceForOne;//цена за штуку

int Quantity;//Колличество штук

};

void menu(int n, Tea\* tee);

void Draw(int n, Tea\* tee);

void WriteBeginText();

void to\_menu(int n, Tea\* tee);

void gotoxy(int x, int y);

void RecordNewTeaFile(int n, Tea\* tee);

int number();

Tea\* Initialization(int n, Tea\* tee);

Tea\* Detete(int n, Tea\* tee, int comanda\_del);

Tea\* AddTea(int n, Tea\* tee);

void SortStr(string\* S, int N, Tea\* tee);

Tea\* SortTea(int n, Tea\* tee);

void SearchTea(int n, Tea\* tee);

void information(int n, Tea\* tee);

#endif

**==================Файл исходного кода Main.cpp================**

#include "Tee.h"

int n = 0; // Количество продуктов

string FailName = "tee.db"; // Путь к файлу и/или имя файла

int main()

{

system("title Ведомость по закупкам чая"); //Заголовок

setlocale(LC\_ALL, "ru");

WriteBeginText();

ifstream file; // Чтение файла

file.open(FailName); // Открытие файла

if (file.is\_open() == true) // Если файл открыт можно работать программой

{

file.close(); // Закрытие файла

n = number();

Tea\* tee = new Tea[n];

Initialization(n, tee);

menu(n, tee);

//Draw(n, tee);

delete[] tee; //удаление динамической памяти

\_getch(); // Задержка программы

}

else{

cout << "\t ------------------" << "\n\t | Файл не открыт | \n" << "\t ------------------";

Sleep(2000); // Если файл не открывается программа выключится через 2 секунды.

}

return 0;

}

==================**Файл исходного кода Menu.cpp===============**

#include "Tee.h"

string FileName = "tee.db"; // Путь к файлу и/или имя файла

HDC hdc = GetDC(GetConsoleWindow());

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE); //получаем дескриптор

//Вывод меню, выбор действия и перенаправление на функции в зависимости от выбора

void menu(int n, struct Tea\* tee)

{ /\*

Данная функция управляется номером команды для использования нужных команд

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 2); //Зеленый

cout << "\t Меню\n";

cout << "1. Вывод базы данных на экран\n";

cout << "2. Добавление чая\n";

cout << "3. Поиск и удаление\n";

cout << "4. Сортировка\n";

cout << "5. Информация о создателе\n\n";

SetConsoleTextAttribute(hStdOut, 8); // Серый

cout << "0. Выход\n";

int comanda;

cout << "Выберите номер команды и нажмите ENTER: ";

while (!(cin >> comanda))

{

cin.clear();

while (cin.get() != '\n');

menu(n, tee);

}

switch (comanda)

{

case 0: exit(0); //выход

case 1: Draw(n, tee); break; //1 - вывод базы данных на экран(menu)

case 2: AddTea(n, tee); break; //2 - добавление чая(menu)

case 3: SearchTea(n, tee); break; //3 - поиск и удаление чая(menu)

case 4: SortTea(n, tee); break; //4 - сортировка(menu)

case 5: information(n, tee); break; //5 - информация о создателе

default: menu(n, tee);

}

}

//1 - вывод базы данных на экран(menu)

void Draw(int n, struct Tea\* tee)

{

/\*

Данная функция выводит БД в консоли

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 2);

gotoxy(1, 0);

cout << "№ |";

gotoxy(4, 0);

cout << "Тип чая |";

gotoxy(15, 0);

cout << "Форма упаковки |";

gotoxy(31, 0);

cout << "Производитель |";

gotoxy(47, 0);

cout << "Цена за шт. |";

gotoxy(60, 0);

cout << "Колличество ";

cout << "\n-------------------------------------------------------------------------------";

SetConsoleTextAttribute(hStdOut, 15);

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

{

gotoxy(1, i + 2);

cout << i + 1;

gotoxy(4, i + 2);

cout << tee[i].TypeTea;

gotoxy(15, i + 2);

cout << tee[i].PackingForm;

gotoxy(31, i + 2);

cout << tee[i].Manufacturer;

gotoxy(47, i + 2);

cout << tee[i].PriceForOne;

gotoxy(60, i + 2);

cout << tee[i].Quantity;

gotoxy(3, i + 2);

cout << "|";

gotoxy(14, i + 2);

cout << "|";

gotoxy(30, i + 2);

cout << "|";

gotoxy(46, i + 2);

cout << "|";

gotoxy(59, i + 2);

cout << "|";

gotoxy(78, i + 2);

}

SetConsoleTextAttribute(hStdOut, 15);

int com;

cout << "\n\n Введите номер команды и нажмите ENTER \n(1 - Сортировка; 0 - Выход в меню): ";

while (!(cin >> com))

{

cin.clear();

while (cin.get() != '\n');

Draw(n, tee);

}

switch (com)

{

case 0: menu(n, tee); break;

case 1: SortTea(n, tee); break;

default: Draw(n, tee);

}

}

//Данные для БД

void WriteBeginText()

{

/\*

Данная функция создает файл и записывает строк

Входной параметр - Отсутствует;

Выходной параметр - Отсутствует.

\*/

ofstream fout;

fout.open(FileName);

fout << "1;Зеленый;Квадрат;Въетнам;10;900;" << endl;

fout << "2;Зеленый;Квадрат;Япония;60;120;" << endl;

fout << "3;Черный;Прямоугольник;Тайвань;31;190;" << endl;

fout << "4;Зеленый;Треугольник;Китай;49;200;" << endl;

fout << "5;Черный;Квадрат;Испания;76;150;" << endl;

fout << "6;Черный;Треугольник;Япония;92;460;" << endl;

fout << "7;Зеленый;Прямоугольник;Британия;52;670;" << endl;

fout.close();

}

//выход в главное меню и проверка на ввод(нет ссылок)

void to\_menu(int n, struct Tea\* tee)

{

/\*

Данная функция предназначена для выхода в меню

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

SetConsoleTextAttribute(hStdOut, 15);

int comanda;

cout << "\n\t\t\t\tДля выхода в меню введите \"0\" и нажмите ENTER: ";

while (!(cin >> comanda) || comanda != 0)

{

cin.clear(); //Очишает ошибочно введенной команды

while (cin.get() != '\n');

cout << "\n\t\t\t\tДля выхода в меню введите \"0\" и нажмите ENTER: ";

}

menu(n, tee);

}

//выравнивание базы данных(нет ссылок)

void gotoxy(int x, int y)

{

/\*

Данная функция выравнивает БД в консоли

Входной параметр - Задающие координаты для выравнивания БД в консоли;

Выходной параметр - Отсутствует.

\*/

COORD p = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), p);

}

//Считывание кол-ва строк из файла

int number()

{

/\*

Данная функция считает количество строк из файла

Входной параметр - Отсутствует;

Выходной параметр - Количество продуктов.

\*/

int count = 0;

ifstream file;

file.open(FileName);

while (!file.eof())

{

if (file.get() == '\n')

count++;

}

Tea\* Pr = new Tea[count];

return count;

}

//Инициализация массивоф данными

Tea\* Initialization(int n, struct Tea\* tee)

{

/\*

Данная функция инициализирует массив структуры данных

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

ifstream fin;

fin.open(FileName);

char N1[3], Quantity1[20], PriceForOne1[20],PackingForm1[20], Manufacturer1[20], TypeTea1[20];

//------------------------------------------------------------------------------

for (int j = 0; j < n; j++)

{

//---------Инициализация масивов символов к нулевому символу--------------------

for (int i = 0; i < 3; i++) {

N1[i] = '\0';

}

for (int i = 0; i < 20; i++) {

PackingForm1[i] = Manufacturer1[i] = TypeTea1[i] = '\0';

}

for (int i = 0; i < 20; i++) {

PriceForOne1[i] = Quantity1[i] = '\0';

}

int count = 0;

char ch;

while ((ch = fin.get()) != ';')

{

N1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

TypeTea1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

PackingForm1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Manufacturer1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

PriceForOne1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Quantity1[count] = ch; count++;

}

count = 0;

tee[j].N = atoi(N1);

tee[j].TypeTea = TypeTea1;

tee[j].PackingForm = PackingForm1;

tee[j].Manufacturer = Manufacturer1;

tee[j].PriceForOne = atoi(PriceForOne1);

tee[j].Quantity = atoi(Quantity1);

}

fin.close();

return NULL;

}

//2 - добавление чая(menu)

Tea\* AddTea(int n, struct Tea\* tee)

{

/\*

Данная функция предназначена для добавления нужного количество элементов массива структуры данных

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

int nAdd;

SetConsoleTextAttribute(hStdOut, 2);

cout << "Введите количество добавляемого чая: ";

while (!(cin >> nAdd))

{

cin.clear();

while (cin.get() != '\n');

cout << endl << "\t\tОШИБКА!\nВведите заново количество добавленных студентов: ";

}

Tea\* TeaNew = new Tea[n + nAdd];

for (int i(0); i < n; i++)

{

TeaNew[i] = tee[i];

}

system("cls");

delete[] tee;

tee = TeaNew;

for (int i = n; i < (n + nAdd); i++)

{

char Type[40];

char Form[40];

char Manufact[40];

char f[48];

SetConsoleCP(1251); //Перевод в кодировку

SetConsoleOutputCP(1251);

SetConsoleTextAttribute(hStdOut, 2);

//=============================

cout << endl << "Введите тип " << i + 1 << "-го чая: ";

SetConsoleTextAttribute(hStdOut, 15);

cin.getline(f, 48);

cin.getline(Type, 40);

tee[i].TypeTea = (string)Type;

SetConsoleTextAttribute(hStdOut, 2);

//==============================

cout << endl << "Введите Форму упаковки " << i + 1 << "-го чая: ";

SetConsoleTextAttribute(hStdOut, 15);

cin.getline(Form, 40);

tee[i].PackingForm = (string)Form;

SetConsoleTextAttribute(hStdOut, 2);

//==============================

cout << endl << "Введите страну производитель " << i + 1 << "-го чая: ";

SetConsoleTextAttribute(hStdOut, 15);

cin.getline(Manufact, 40);

tee[i].Manufacturer = (string)Manufact;

SetConsoleTextAttribute(hStdOut, 2);

//==============================

cout << endl << "Введите цену за 1шт. " << i + 1 << "-го чая: ";

SetConsoleTextAttribute(hStdOut, 15);

cin >> tee[i].PriceForOne;

SetConsoleTextAttribute(hStdOut, 2);

//==============================

cout << endl << "Введите колличество " << i + 1 << "-го чая: ";

SetConsoleTextAttribute(hStdOut, 15);

cin >> tee[i].Quantity;

SetConsoleTextAttribute(hStdOut, 2);

system("cls");

}

ofstream fout;

fout.open(FileName, ios\_base::app);

for (int i = n; i < (n + nAdd); i++)

{

fout << i + 1 << ";" << TeaNew[i].TypeTea << ";" << TeaNew[i].PackingForm << ";" << TeaNew[i].Manufacturer;

fout << ";" << TeaNew[i].PriceForOne << ";" << TeaNew[i].Quantity << ";\n";

}

fout.close();

n += nAdd;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\t\t\t\t\t | Данные сохранены | \n";

to\_menu(n, tee);

return 0;

}

//3 - поиск и удаление чая(menu)

void SearchTea(int n, struct Tea\* tee)

{

/\*

Данная функция предназначена для поиска элемента массива(и при желаний для дальнейшего удаления элемента массива)

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

SetConsoleCP(1251); //Перевод в кодировку

SetConsoleOutputCP(1251);

system("cls");

SetConsoleTextAttribute(hStdOut, 15);

cout << "\tПоиск\n";

SetConsoleTextAttribute(hStdOut, 2);

cout << "1 - По порядковому номеру\n";

cout << "2 - По типу чая \n";

cout << "3 - По форме упаковки \n";

cout << "4 - По производителю \n";

SetConsoleTextAttribute(hStdOut, 8);

cout << "\n0 - Выход в меню\n";

int comanda;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\nВведите номер команды и нажмите ENTER: ";

while (!(cin >> comanda))

{

cin.clear();

while (cin.get() != '\n');

SearchTea(n, tee);

}

switch (comanda)

{

case 1:

{

system("cls");

int No, No1;

cout << "\n\tВведите № искомого чая (1 - " << n << "): ";

while (!(cin >> No1) || No1 > n || No1 < 1)

{

cin.clear();

while (cin.get() != '\n');

system("cls");

cout << "Вы ввели неправильно порядковый номера \n";

cout << "\n\tВведите заново № искомого чая (1 - " << n << "): ";

}

No = No1 - 1;

system("cls");

cout << "\t№ ";

SetConsoleTextAttribute(hStdOut, 10);

cout << No1;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\tТип чая: " << tee[No].TypeTea << "\n\tФорма упаковки: " << tee[No].PackingForm << "\n\tПроизводитель: " << tee[No].Manufacturer;

cout << "\n\tЦена за 1 шт.: " << tee[No].PriceForOne << "\n\tКолличество: " << tee[No].Quantity << endl << endl;

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, tee); break;

case 1: Detete(n, tee, No); break;

}

};

break;

case 2:

{

system("cls");

string Tip;

int ComDel = 0;

cout << "\n\tВведите тип чая: ";

cin >> Tip;

int checking = 0;

system("cls");

for (int i(0); i < n; i++)

{

if (Tip == tee[i].TypeTea)

{

cout << "\t№ " << (i + 1);

cout << "\n\tТип чая: ";

SetConsoleTextAttribute(hStdOut, 10);

cout << tee[i].TypeTea;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\tФорма коробки: " << tee[i].PackingForm << "\n\tПроизводитель: " << tee[i].Manufacturer;

cout << "\n\tЦена за 1 шт.: " << tee[i].PriceForOne << "\n\tКолличество: " << tee[i].Quantity << endl << endl;

ComDel = i;

checking++;

}

}

if (checking == 0)

{

system("cls");

cout << "\n\t Такой чай не найден\n";

to\_menu(n, tee);

}

if (checking > 1)

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

if (com == 0) {

menu(n, tee);

}

if (com == 1) {

cout << "\n Введите номер удаляемого элемента: ";

int c; cin >> c;

ComDel = c - 1;

Detete(n, tee, ComDel);

}

}

else

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, tee); break;

case 1: Detete(n, tee, ComDel); break;

}

}

};

break;

case 3:

{

system("cls");

string Forma;

int checking = 0;

int ComDel = 0;

cout << "\n\tВведите форму упаковки: ";

while (!(cin >> Forma))

{

cin.clear();

while (cin.get() != '\n');

system("cls");

cout << "\n\tВведите форму упаковки: ";

}

system("cls");

for (int i(0); i < n; i++)

{

if (Forma == tee[i].PackingForm)

{

cout << "\t№ " << (i + 1) << "\n\tТип чая: " << tee[i].TypeTea << "\n\tФорма упаковки: " << tee[i].PackingForm;

cout << "\n\tПроизводитель: " << tee[i].Manufacturer << "\n\tКолличество" << tee[i].Quantity << "\n\t:Цена за 1 шт.: ";

SetConsoleTextAttribute(hStdOut, 10);

cout << tee[i].PriceForOne << endl << endl;

SetConsoleTextAttribute(hStdOut, 15);

ComDel = i;

checking++;

}

}

if (checking == 0)

{

system("cls");

cout << "\n\tЧай не найден\n";

to\_menu(n, tee);

}

if (checking > 1)

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

if (com == 0) {

menu(n, tee);

}

if (com == 1) {

cout << "\n Введите номер удаляемого элемента: ";

int c; cin >> c;

ComDel = c - 1;

Detete(n, tee, ComDel);

}

}

else

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, tee); break;

case 1: Detete(n, tee, ComDel); break;

}

}

};

break;

case 4:

{

system("cls");

string Proiz;

int ComDel = 0;

cout << "\n\tВведите производителя: ";

cin >> Proiz;

int checking = 0;

system("cls");

for (int i(0); i < n; i++)

{

if (Proiz == tee[i].Manufacturer)

{

cout << "\t№ " << (i + 1);

cout << "\n\tПроизводитель: ";

SetConsoleTextAttribute(hStdOut, 10);

cout << tee[i].Manufacturer;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\tТип чая: " << tee[i].TypeTea << "\n\tФорма упаковки: " << tee[i].PackingForm;

cout << "\n\tЦена за 1шт.: " << tee[i].PriceForOne << "\n\tКолличество: " << tee[i].Quantity << endl << endl;

ComDel = i;

checking++;

}

}

if (checking == 0)

{

system("cls");

cout << "\n\t Производитель не найден\n";

to\_menu(n, tee);

}

if (checking > 1)

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

if (com == 0) {

menu(n, tee);

}

if (com == 1) {

cout << "\n Введите номер удаляемого элемента: ";

int c; cin >> c;

ComDel = c - 1;

Detete(n, tee, ComDel);

}

}

else

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, tee); break;

case 1: Detete(n, tee, ComDel); break;

}

}

}

break;

case 0: menu(n, tee);

default: SearchTea(n, tee);

}

}

//Запись изменений в файл

void RecordNewTeaFile(int n, Tea\* tee)

{

/\*

Данная функция записывает в файл изменения

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

ofstream fout;

fout.open(FileName);

for (int i = 0; i < (n - 1); i++)

{

fout << i + 1 << ";" << tee[i].TypeTea << ";" << tee[i].PackingForm << ";" << tee[i].Manufacturer;

fout << ";" << tee[i].PriceForOne << ";" << ";\n";

}

fout.close();

}

//Удаление чая +

Tea\* Detete(int n, Tea\* tee, int comanda\_del)

{

/\*

Данная функция удаляет comanda\_del-го элемента массива

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

//------------------УДАЛЕНИЕ ЭЛЕМЕНТА-----------------------------------------------------------------------

//------- Шаг 1. Замена удаляемого элемента на крайний элемент

string temp\_Tea = tee[n - 1].TypeTea;

tee[n - 1].TypeTea = tee[comanda\_del].TypeTea;

tee[comanda\_del].TypeTea = temp\_Tea;

string temp\_Form = tee[n - 1].PackingForm;

tee[n - 1].PackingForm = tee[comanda\_del].PackingForm;

tee[comanda\_del].PackingForm = temp\_Form;

string temp\_Manuf = tee[n - 1].Manufacturer;

tee[n - 1].Manufacturer = tee[comanda\_del].Manufacturer;

tee[comanda\_del].Manufacturer = temp\_Manuf;

int temp\_Price = tee[n - 1].PriceForOne;

tee[n - 1].PriceForOne = tee[comanda\_del].PriceForOne;

tee[comanda\_del].PriceForOne = temp\_Price;

int temp\_Qua = tee[n - 1].Quantity;

tee[n - 1].Quantity = tee[comanda\_del].Quantity;

tee[comanda\_del].Quantity = temp\_Qua;

//-------- Шаг 2. Новый масив

Tea\* TeeNew = new Tea[n - 1];

//------- Шаг 3. Инициализация всех элементов к новому массиву кроме крайного

for (int i(0); i < (n - 1); i++)

{

TeeNew[i] = tee[i];

}

//------ Шаг 4. Удаление старого массива

delete[] tee;

tee = TeeNew;

//---------------Запись в файл

RecordNewTeaFile(n, tee);

system("cls");

cout << "\n\n\n\n\t\tДанные удалены успешно.\n";

n--;

to\_menu(n, tee);

return NULL;

}

//4 - сортировка чая(menu)

Tea\* SortTea(int n, struct Tea\* tee)

{

/\*

Данная функция сортирует структуру по алфавиту/значению по заданным параметрам

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 15);

cout << "\tСортировка\n";

SetConsoleTextAttribute(hStdOut, 2);

cout << "1 - По Типу чая \n";

cout << "2 - По Производителю\n";

cout << "3 - По Цене за шт.\n";

cout << "4 - По Форме упаковки\n";

SetConsoleTextAttribute(hStdOut, 8);

cout << "\n0 - Выход в меню\n";

int comanda;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\nВведите номер команды и нажмите ENTER: ";

while (!(cin >> comanda))

{

cin.clear();

while (cin.get() != '\n');

SortTea(n, tee);

}

switch (comanda)

{

case 0: menu(n, tee); break;

case 1:

{

system("cls");

string\* str1 = new string[n];

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

{

str1[i] = tee[i].TypeTea;

}

SortStr(str1, n, tee);

Draw(n, tee);

};

break;

case 2:

{

system("cls");

string\* str1 = new string[n];

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

{

str1[i] = tee[i].Manufacturer;

}

SortStr(str1, n, tee);

Draw(n, tee);

};

break;

case 3:

{

system("cls");

SetConsoleTextAttribute(hStdOut, 3);

cout << "\t\t\t\t\t1 - Возрастанию\n";

cout << "\t\t\t\t\t2 - Убыванию\n";

SetConsoleTextAttribute(hStdOut, 15);

int com;

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

while (!(cin >> com))

{

cin.clear();

while (cin.get() != '\n');

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 1:

{

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

{

for (int j = i + 1; j < n; j++)

if (tee[i].PriceForOne > tee[j].PriceForOne)

{

swap(tee[i], tee[j]);

}

}

};

break;

case 2:

{

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

{

for (int j = i + 1; j < n; j++)

if (tee[i].PriceForOne < tee[j].PriceForOne)

{

swap(tee[i], tee[j]);

}

}

};

break;

default: SortTea(n, tee);

}

};

break;

case 4:

{

system("cls");

string\* str1 = new string[n];

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

{

str1[i] = tee[i].PackingForm;

}

SortStr(str1, n, tee);

Draw(n, tee);

}

break;

default: SortTea(n, tee);

}

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

{

tee[i].N = i + 1;

}

RecordNewTeaFile(n, tee);

Draw(n, tee);

return NULL;

}

//Сортировка

void SortStr(string\* S, int N, Tea\* tee)

{

/\*

Данная функция сортирует структуру по алфавиту по заданным параметрам

Входные параметры - 1.Массив структуры данных; 2. Массив строк;

Выходной параметр - Отсутствует.

\*/

for (int i = 0; i < N - 1; i++) {

for (int j = i + 1; j < N; j++)

{

if (strcmp(S[i].c\_str(), S[j].c\_str()) > 0) {

swap(tee[i], tee[j]);

}

}

}

/\*for (int i = 0; i < N - 1; i++) {

for (int j = i + 1; j < N; j++)

{

if (strcmp(S[i].c\_str(), S[j].c\_str()) > 0) {

swap(tee[i], tee[j]);

}

}

}\*/

}

//5 - Вывод данных о создателе

void information(int n, struct Tea\* tee)

{

/\*

Данная функция показывает информацию о разработчике программы;

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 2);

gotoxy(0, 1);

cout << "Информация о создателе";

gotoxy(0, 3);

cout << "ФИО разработчика - Семиков Алексей ";

gotoxy(0, 4);

cout << "Курс - 2\n";

gotoxy(0, 5);

cout << "Возраст - 19 лет";

SetConsoleTextAttribute(hStdOut, 8);

int comanda;

cout << "\n\nДля выхода в меню введите \"0\" и нажмите ENTER: ";

while (!(cin >> comanda) || comanda != 0)

{

cin.clear(); //Очишает ошибочно введенной команды

while (cin.get() != '\n');

gotoxy(30, 10);

cout << "Для выхода в меню введите \"0\" и нажмите ENTER: ";

}

menu(n, tee);

}