**АННОТАЦИЯ**

Пояснительная записка по курсовой работе по дисциплине “Алгоритмизация и программирование”. Институт информационных технологий и управления в технических системах, 2021г. Количество станиц – N, рисунков – N, таблиц – N.

Разработано программное обеспечение “Обработки и хранения данных о продуктах в магазинах”. Программное обеспечение спроектировано в среде DevC++ на языке проектирования Си.

Курсовая работа состоит из титульного листа, технического задания, аннотации, содержания, введения, трёх разделов, заключения, списка литературных источников и приложения с листингом программы.

# **СОДЕРЖАНИЕ**

[СОДЕРЖАНИЕ 3](#_Toc90634043)

[ВВЕДЕНИЕ 4](#_Toc90634044)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ 6](#_Toc90634045)

[2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ 7](#_Toc90634046)

[**2.1.** **Постановка задачи на разработку программы (Вариант 15)** 7](#_Toc90634047)

[**2.2.** **Применяемые математические методы** 8](#_Toc90634048)

[**2.3.** **Описание и обоснование выбора метода организации входных, выходных и промежуточных данных** 8](#_Toc90634049)

[**2.4.** **Обоснование выбора языка и среды программирования** 9](#_Toc90634050)

[**2.5.** **Разработка модульной структуры программы** 9](#_Toc90634051)

[**2.6.** **Описание функций используемых в программе** 10](#_Toc90634052)

[**2.7.** **Описание алгоритмов функционирования программы** 13](#_Toc90634053)

# **ВВЕДЕНИЕ**

С появлением и широким распространением табличных процессоров, одним из самых известных представителей которых является Microsoft Exel, и иных программ, позволяющих обрабатывать большие объёмы табличных данных без необходимости знания пользователем языков программирования, написание узкоспециализированных программ, примером которых является разрабатываемая программа, утратило свою актуальность, что, тем не менее, никак не сказалось на возможности использования разработки программы, преимуществом которой является простота, для закрепления знаний.

Целью курсового проектирования является систематизация, закрепление и углубление знаний в области основ программирования и совершенствование практических навыков разработки программ на языке Си с использованием методологии структурного программирования.

Также целью курсового проекта является создание программы с удобным интерфейсом для пользования. Программа создана для хранения данных о продуктах в магазинах, создание и заполнение таблицы данными по варианту, также хранение данных в файле и чтение из файла.

Отчёт по курсовой работе состоит из трёх разделов: “Назначение и область применения программы”, “Технические характеристики программы” и “Выполнение программы”.

Для достижения цели на разных этапах курсового проектирования должны быть решены следующие задачи:

- выбор варианта задания и детализация поставки задачи;

- определение требований к функциям, выполняемых разрабатываемой программой;

Выбор типов и проектирование структур данных, определяющих способы представления, хранения и преобразования входных, выходных и промежуточных данных;

- разработка модульной структуры программы, определение функций модулей и способов их взаимодействия;

- написание текста программных модулей на алгоритмическом языке;

- разработка тестовых примеров;

- тестирование и отладка программы;

- разработка программных документов в соответствии с действующими стандартами.

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ**

Программа может применяться для любых организаций, у которых имеются магазины, в которых продаются продукты. Программа предназначена для организации, хранения, модификации записей о продуктах и предоставление удобного доступа к ним через консольный пользовательский интерфейс.

Программа считывает данные о продукте и записывает их в двунаправленный список. Также программа может считывать данные о продуктах из текстового или бинарного файлов или же записывать в них. Программа может выводить данные в виде таблицы, выводить на экран информацию об одном продукте по номеру чека. Способна определять товарооборот в каждом магазине присутствующему в базе данных и выводить их на экран и опционально по желанию пользователя выводить информацию о товарообороте в текстовый файл. У программы есть функции удаления информации о продукте по выбору, или же удаления всех записей. Присутствует возможность корректировки записи о продукте. По желанию пользователя данные в базе данных можно отсортировать по возрастанию или убыванию. Так же при загрузке программы можно автоматически загрузить данные, которые были в базе данных при последнем закрытии программы, так как при выходе из программы происходит автоматическое сохранение базы данных в бинарном файле.

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ**
   1. **Постановка задачи на разработку программы (Вариант 15)**

Структура записей входного файла имеет следующий вид:

* номер магазина;
* номер секции;
* номер чека;
* наименование товара;
* артикул товара;
* цена товара;
* количество товара;
* день продажи;
* месяц продажи;
* год продажи;

Разрабатываемая программа должна использовать меню-ориентированный интерфейс, обеспечивающий выполнение следующего минимального состава действий:

* начальное создание таблицы;
* просмотр таблицы с использованием скроллинга;
* удаление записи;
* корректировка записи в таблице;
* сортировка таблицы;
* поиск записи в таблице;
* сохранение таблицы в текстовый и типизированный файл;
* чтение таблицы из текстового и типизированного файла;
* обработка таблицы и просмотр результатов обработки;
* выход из программы;

При выходе из программы должно происходить автоматическое сохранение текущих данных, при запуске возможность загрузки автоматического сохранения.

Сортировка должна происходить по возрастанию и убыванию по желанию пользователя.

При обработке таблицы необходимо определить товарооборот для каждого из магазинов присутствующих в базе данных. У пользователя должна быть возможность вывести информацию о товарообороте в текстовый файл.

## **Применяемые математические методы**

В ходе разработки, при вычислении итоговой суммы по одному продукту происходило умножение количества продуктов на цену продукта (рис. 2.2.1).

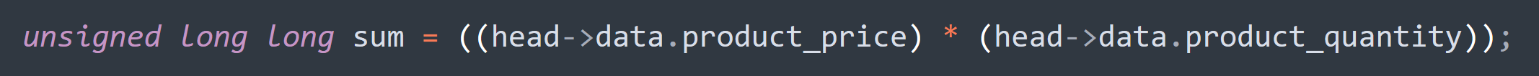


Рисунок 2.2.1 – Вычисление итоговой суммы продукта

Так же при вычислении товарооборота активно применялось сложение итоговых сумм продуктов в одном магазине (рис. 2.2.2).



Рисунок 2.2.2 – Подсчёт товарооборота

* 1. **Описание и обоснование выбора метода организации входных, выходных и промежуточных данных**

Для хранения данных в памяти было решено использовать двунаправленный список, для упрощённой, ускоренной и удобной работы с программой. Во избежание дополнительных выделений памяти программа не нуждается в промежуточных данных, а выходные данные аналогичны входным. Для того чтобы программа не нуждалась в промежуточных данных в изначальный вариант элемента списка было добавлено два указателя на следующий и предыдущий элемент списка.

Для хранения строк было решено использовать массивы символов, что позволяет экономно использовать память, так как строка в таком случае содержит ограниченное количество символов. Это в свою очередь позволяет экономить память при создании текстовых и бинарных файлов.

* 1. **Обоснование выбора языка и среды программирования**

Для разработки был выбран язык программирования Си, так как является более удобным для поставленной задачи.

Средой разработки была выбрана DevC++, так как у неё минималистичный и удобны интерфейс, а так же подсвечиваются некоторые ошибки на этапе разработки программы, в отличии от блокнота.

Вся разработка велась на двух персональных ЭВМ с операционной системой Microsoft Windows 10.

* 1. **Разработка модульной структуры программы**

В процессе разработки программы для упрощения навигации в коде программы было принято решение разделить программу на 10 частей:

* первая часть specialFunctions.c, содержащая специальные функции
* вторая часть workWithTable.c, содержащая основные функции для работы со списком
* третья часть saveAndDownloadTable.c, содержащая функции для сохранения и загрузки файлов
* четвертая часть setData.c, содержащая функции ввода информации в поля элемента
* пятая часть заголовочный файл structures.h, содержащий описание структур и подключающая библиотеки
* шестая часть заголовочный файл specialFunctions.h, содержащий прототипы функций и подключение необходимых заголовочных файлов
* седьмая часть заголовочный файл workWithTable.h, содержащий прототипы функций и подключение необходимых заголовочных файлов
* восьмая часть заголовочный файл setData.h, содержащий прототипы функций и подключение необходимых заголовочных файлов
* девятая часть заголовочный файл saveAndDownloadTable.h, содержащий прототипы функций и подключение необходимых заголовочных файлов
* десятая часть main.c, содержащий главную функцию main

Схема взаимодействия этих частей программы представлена на рисунке 2.5.3.

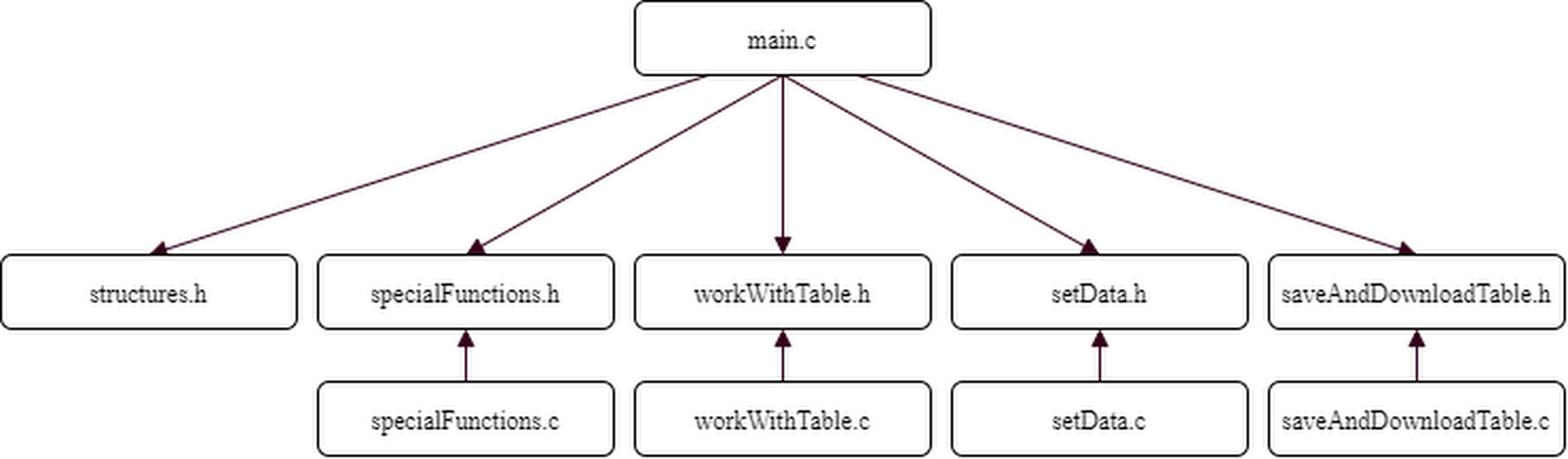


Рисунок 2.5.3 – Схема взаимодействия частей программы

* 1. **Описание функций используемых в программе**

В данном разделе описаны все функции, которые используются в программе.

**void gotoxy(int x, int y)** данная функция устанавливает координаты курсора.

**void setCursor(int i)** данная функция показывает курсор, если i = 0 и скрывает, если i = 1.

**void showMenuItems(const char \*menuItems[], const int NUM\_MENU\_ITEMS, const char \*title)** данная функция выводит элементы меню, передаваемые в качестве параметра, также параметром передаётся количество элементов меню и заглавие меню в виде строки.

**int menu(const char \*menuItems[], int numPunct, const char \*title)** данная функция возвращает выбранный пользователем номер элемента меню.

**int yesOrNo(const char \*title)** данная функция является исключительным экземпляром функции menu, так как очень часто требуется какое либо подтверждение.

**UI setNumShop()** данная функция осуществляет ввод номера магазина.

**UI setNumSection()** данная функция осуществляет ввод номера секции.

**int checkingForIdenticalReceipts(Element \*head, char buf[25])** данная функция проверяет наличие одинаковых чеков, для предотвращения наличия одинаковых чеков.

**void setNumReceipt(char \*buf, Element \*head)** данная функция осуществляет ввод номера чека.

**void setProductName(char \*buf)** данная функция осуществляет ввод наименования продукта.

**void setProductCode(char \*buf)** данная функция осуществляет ввод артикула продукта.

**UL setProductPrice()** данная функция осуществляет ввод цены товара.

**UI setProductQuantity()** данная функция осуществляет ввод количества продуктов.

**When setDate()** данная функция осуществляет ввод даты в формате день, месяц, год.

**void readTable(Element \*\*head)** данная функция осуществляет считывание таблицы из текстового файла.

**void saveTable(Element \*head)** данная функция осуществляет сохранение таблицы в текстовый файл.

**void readTableBIN(Element \*\*head)** данная функция осуществляет считывание таблицы из бинарного файла.

**void saveTableBIN(Element \*head)** данная функция осуществляет сохранение таблицы в бинарный файл.

**void autoDownloudTableBin(Element \*\*head)** данная функция осуществляет автоматическую загрузку последнего сохранения в формате .bin (типизированный файл).

**void autoSaveTableBin(Element \*head)** данная функция осуществляет автоматическое сохранение таблицы в формате .bin (типизированный файл).

**Info setInfo(Element \*head)** данная функция осуществляет ввод информации для функции добавления элемента в базу данных.

**void addNewElement(Element \*\*head, Info info)** данная функция осуществляет добавление элемента в базу данных.

**int getNumElements(Element \*head)** данная функция подсчитывает количество элементов таблицы и возвращает это значение.

**void showTable(Element \*head)** данная функция демонстрирует базу данных в виде таблицы.

**Element \*selectElement(Element \*head)** данная функция демонстрирует таблицу в которой пользователь выбирает требуемый ему элемент, функция возвращает указатель на выбранный элемент.

**void correctRecord(Element \*elem)** данная функция корректирует выбранную запись о продукте.

**void deleteElement(Element \*\*head)** данная функция удаляет выбранный элемент из базы данных.

**void deleteTable(Element \*\*head)** данная функция удаляет все элементы из базы данных.

**Element \*searchElement(Element \*head)** данная функция ищет элемент по номеру чека и возвращает указатель на этот элемент.

**void showOneElement(Element \*head)** данная функция выводит информацию об одном элементе в виде красивой таблицы.

**void sortTableAscending(Element \*\*head)** данная функция сортирует таблицу по возрастанию.

**void sortTableDescending(Element \*\*head)** данная функция сортирует таблицу по убыванию.

**void determineTurnoverForEachStore(Element \*head)** данная функция определяет товарооборот для каждого магазина.

* 1. **Описание алгоритмов функционирования программы**

Схема главной функции представлена на рисунках 2.7.1.1 – 2.7.1.3.

Поблочное описание:

1 – начало выполнения функции main;

2 – функции для поддержки русского языка;

3 – установка имени консольного приложения;

4 – установка цветов консоли и размеров консоли;

5 – создается указатель на голову и инициализируется значением NULL;

6 – инициализируются пункты первого меню;

7 ­– функция “menu” возвращает номер выбранного пункта меню и с помощью switch мы определяем дальнейшие действия;

8 – при case = 1 добавляем элемент в базу данных;

9 – с помощью функции “yesOrNo” в цикле while узнаем продолжать ли добавление элементов в базу данных;

10 – функция чтения таблицы из текстового файла, при case = 2;

11 – функция чтения таблицы из бинарного файла, при case = 3;

12 – автоматическая загрузка последнего автоматического сохранения при case = 4;

13 – инициализируются пункты главного меню;

14 – вечный цикл;

15 – функция “menu” возвращает номер выбранного пункта меню и с помощью switch мы определяем дальнейшие действия;

16 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли создаем таблицу при case = 1;

17 – проверка пуста ли таблица;

18 – если таблица не пуста инициализируются пункты дополнительного меню;

19 – функция “menu” возвращает номер выбранного пункта меню и с помощью switch мы определяем дальнейшие действия;

20 – при case = 1 удаляем таблицу с помощью функции;

21 – при case = 2 сохраняем базу данных в текстовый файл, а затем удаляем;

22 – после того как база данных точно пуста добавляем элемент в базу данных;

23 – с помощью функции “yesOrNo” в цикле while узнаем продолжать ли добавление элементов в базу данных;

24 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли показать таблицу при case = 2;

25 – функция демонстрации таблицы;

26 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли добавить элемент в таблицу при case = 3;

27 – добавляем элемент в базу данных;

28 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли удаляем элемент из базы данных при case = 4;

29 – функция удаления элемента из таблицы;

30 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли удаляем базу данных при case = 5;

31 – функция удаления базы данных;

32 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли корректируем запись в базе данных при case = 6;

33 – функция корректировки записи;

34 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли сортируем записи в базе данных при case = 7;

35 – проверка пуста ли таблица;

36 – инициализируются пункты дополнительного меню, если таблица не пуста;

37 – функция “menu” возвращает номер выбранного пункта меню и с помощью switch мы определяем дальнейшие действия;

38 – при возврате 0 функция сортировки по возрастанию;

39 – при возврате 1 функция сортировки по убыванию;

40 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли ищем запись в базе данных при case = 8;

41 – проверка пуста ли таблица;

42 – функция демонстрации одного элемента если таблица не пуста;

43 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли сохраняем таблицу в файл при case = 9;

44 – инициализируются пункты дополнительного меню;

45 – функция “menu” возвращает номер выбранного пункта меню и с помощью switch мы определяем дальнейшие действия;

46 – при возврате 0 функция сохраняется в текстовый файл;

47 – при возврате 1 функция сохраняется в бинарный файл;

48 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли считываем таблицу из файла при case = 10;

49 – проверка пуста ли таблица (равен ли указатель на голову NULL)

50 – если таблица не пуста инициализируются пункты дополнительного меню;

51 – функция “menu” возвращает номер выбранного пункта меню и с помощью switch мы определяем дальнейшие действия;

52 – при case = 1 функция удаления таблицы;

53 – при case = 2 сохранение таблицы в текстовый файл, а затем удаление;

54 – инициализируются пункты ещё одного дополнительного меню;

55 – функция “menu” возвращает номер выбранного пункта меню и с помощью switch мы определяем дальнейшие действия;

56 – при возврате 0 считывается таблица из текстового файла;

57 – при возврате единицы считывается таблица из бинарного файла;

58 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли определяем товарооборот при case = 11;

59 – функция определения товарооборота для каждого магазина в базе данных;

60 – с помощью функции “yesOrNo” спрашиваем пользователя точно ли он сделал всё что хотел при case = 12;

61 – автоматическое сохранение таблицы и очистка памяти;

62 ­­– завершение функции main.

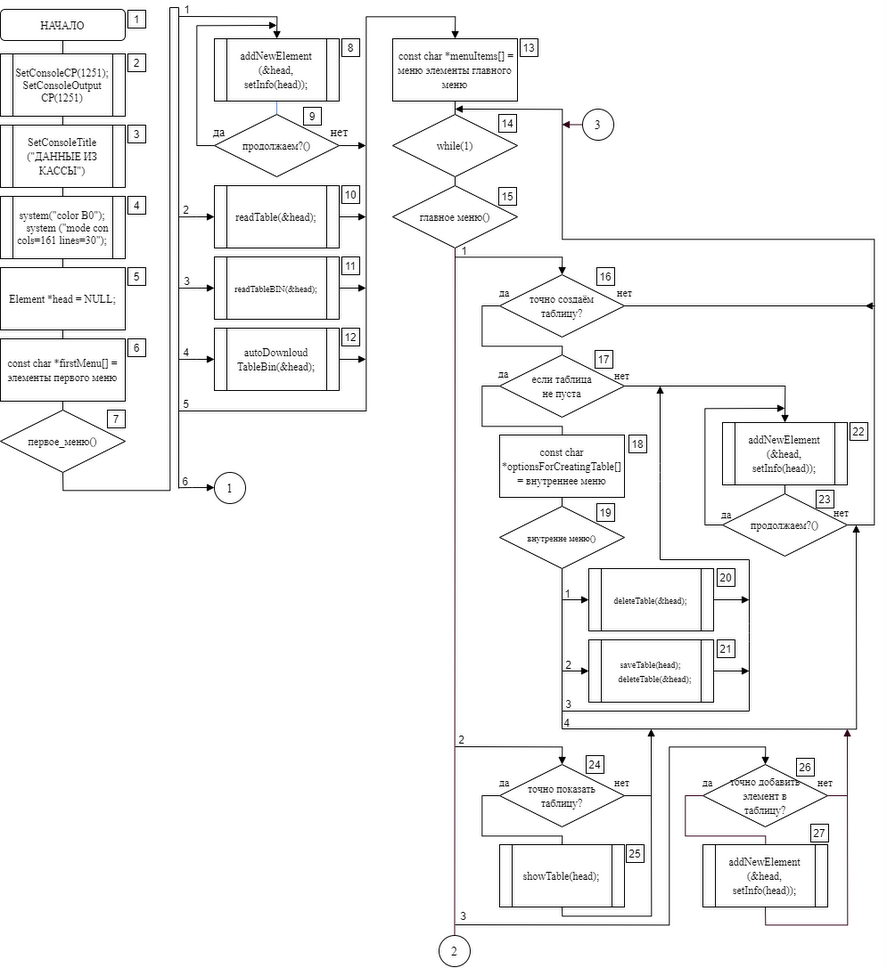


Рисунок 2.7.1.1 – Структурная схема функции main 1 часть

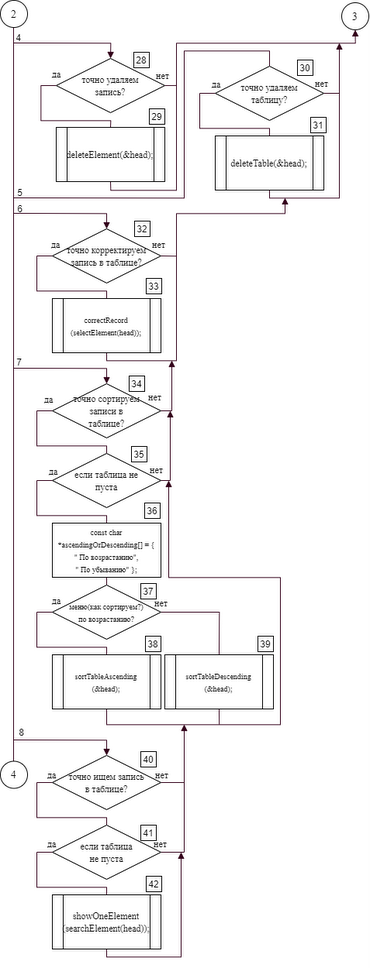


Рисунок 2.7.1.2 – Структурная схема функции main 2 часть

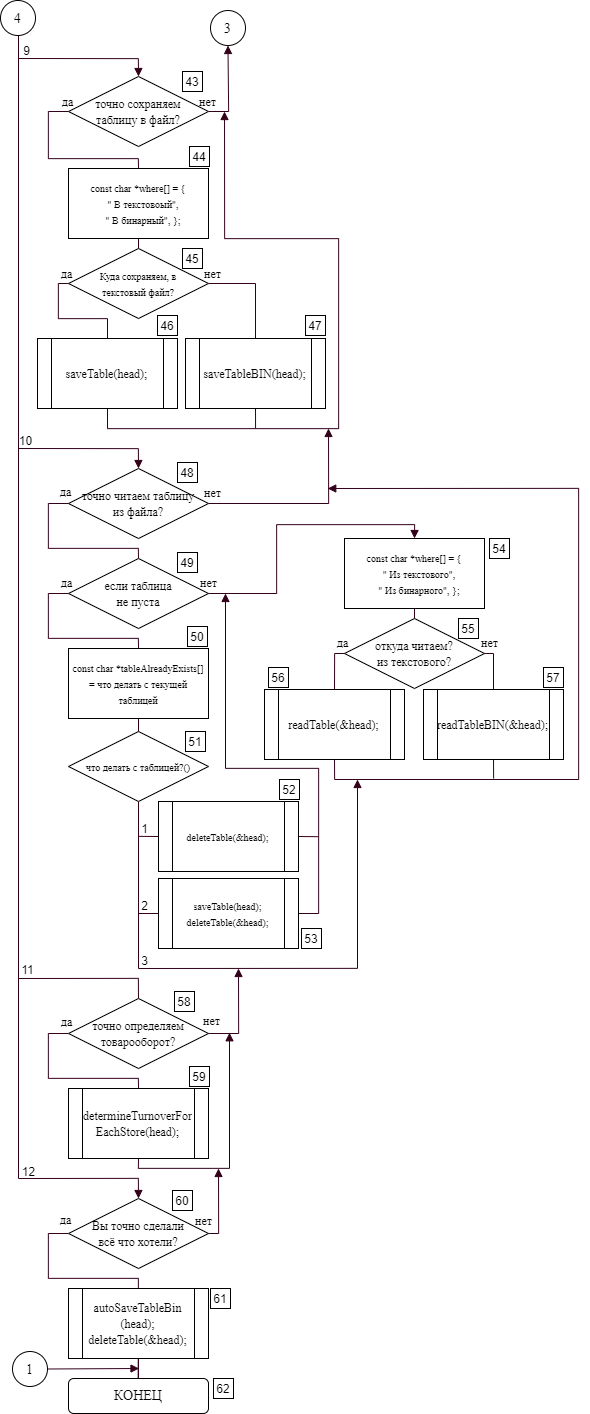


Рисунок 2.7.1.3 – Структурная схема функции main 3 часть

Схема функции добавления элемента в таблицу (рис. 2.7.2).

Поблочное описание:

1 – начало выполнения функции addNewElement;

2 – выделение памяти для нового элемента;

3 – инициализация полей элемента, указатель на следующий и предыдущий элемент = NULL;

4 – если указатель на голову списка = NULL;

5 – если да то голове присвоить temp;

6 – создание вспомогательной переменной, чтобы не изменить указатель на голову;

7 – пока следующий элемент не равен NULL;

8 – идти по списку в направлении следующего элемента;

9 – следующему элементу присвоить temp, указатель на предыдущий элемент равен текущему;

10 – конец функции добавления элемента в таблицу.

Схема функции инициализации полей элемента (рис. 2.7.3).

Поблочное описание:

1 – начало выполнения функции setInfo;

2 – создание переменной info, полю номер магазина присвоить возвращаемое значение функции setNumShop(), полю номер секции присвоить возвращаемое значение функции setNumSection();

3 – с помощью функций setNumReceipt(), setProductName() и setProductCode инициализируются соответственно поля номер чека, имя продукта и артикул продукта;

4 – полю цена продукта присвоить возвращаемое значение функции setProductPrice(), полю количество продуктов присвоить возвращаемое значение функции setProductQuantity(), полю даты продажи продукта присвоить возвращаемое значение функции setDate;

5 – конец функции инициализации полей элемента.

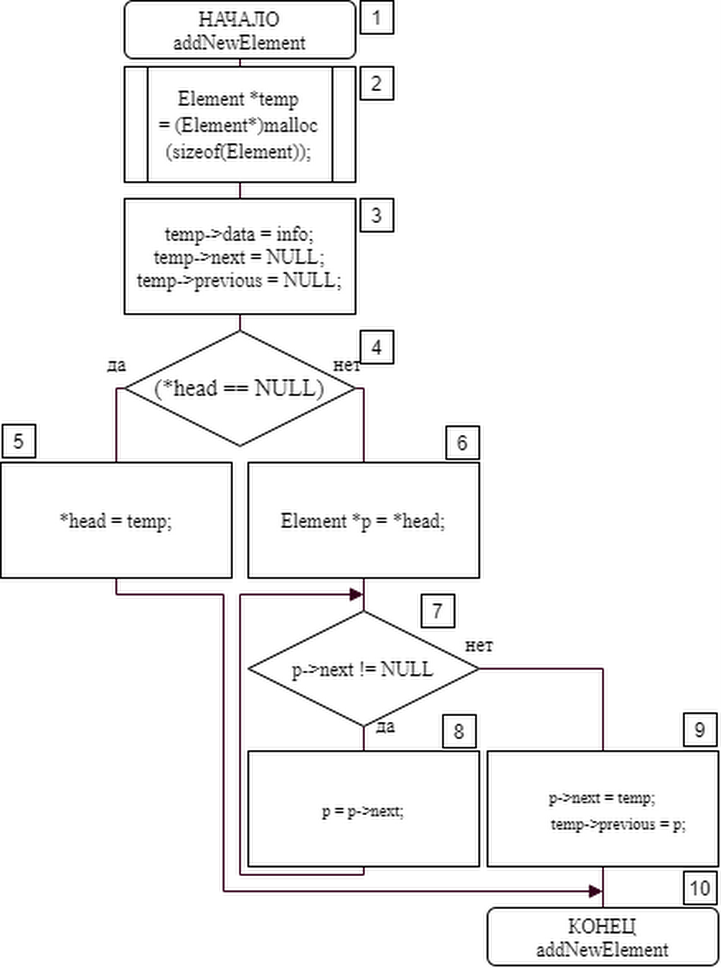


Рисунок 2.7.2 – Структурная схема функции добавления элемента в таблицу



Рисунок 2.7.3 – Структурная схема функции инициализации полей элемента

Схема функции добавления элемента в таблицу (рис. 2.7.4).

Поблочное описание:

1 – начало выполнения функции showTable;

2 – функция, которая удаляет курсор и очистка экрана;

3 – проверка пуста ли таблица;

4 – если таблица пуста, то вернуть отображение курсора;

5 – кол-во элементов для отображения = 10, взять кол-во элементов в таблице;

6 – пока символ не равен коду клавиши Esc;

7 – если нет, то вернуть курсор передав функции setCursor параметром единицу;

8 – конец функции отображения таблицы;

9 – очистить экран;

10 – вывести на экран шапку таблицы и инструкция по навигации;

11 – временной переменной temp присваивается указатель на голову;

12 – цикл for, который выполняется 10 раз (кол-во элементов для отображения);

13 – пока временная переменная temp не равна NULL;

14 – вывести элемент;

15 – переменной temp присвоить указатель на следующий элемент;

16 – считать символ;

17 – если символ esc или символ стрелки вверх/вниз;

18 – если символ оказался стрелкой;

19 – взять еще один символ (т. к. коды стрелок состоят из двух частей);

20 – если кол-во элементов в таблице > кол-ва элементов для отображения;

21 – если была введена стрелка вверх или вниз или page up или page down;

22 – вывести звук ошибки;

23 – иначе switch символ;

24 – при ch = ‘page down’ если на 19 элементов вперед не NULL;

25 – звук ошибки;

26 – переместить указатель на 10 элементов вперед;

27 – при ch = ‘стрелка вниз’ если указатель на десять шагов вперед != NULL;

28 – звук ошибки;

29 – переместить указатель на один элемент вперед;

30 – при ch = ‘page up’ если на 10 элементов назад есть элемент;

31 – звук ошибки;

32 – переместить указатель на 10 элементов назад;

33 – при ch = ‘стрелка вверх’ если предыдущий элемент не равен NULL;

34 – звук ошибки;

35 – сместить указатель на один элемент назад.

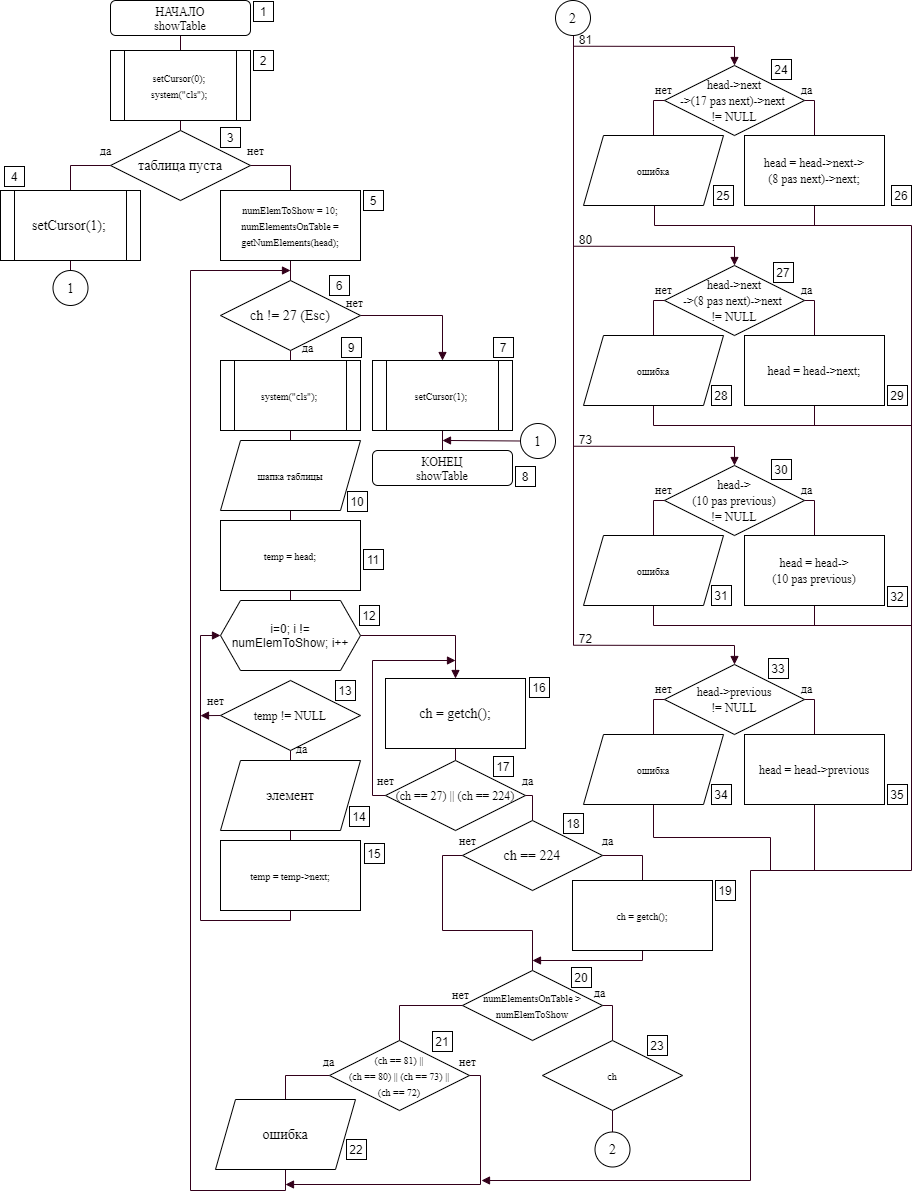


Рисунок 2.7.4 – Структурная схема функции демонстрации таблицы

Схема функции добавления элемента в таблицу (рис. 2.7.5).

Поблочное описание:

1 – начало выполнения функции sortTableAscending;

2 – пуста ли таблица, если пуста, то выход из функции;

3 – взять количество элементов в таблице;

4 – цикл for, выполняется столько раз, сколько элементов в таблице;

5 – временные переменные инициализируются для сортировки;

6 – пока numTemps меньше переменной i;

7 – если текущий элемент больше следущего;

8 – то обменять эти значения местами;

9 – увеличить счётчик и сместить указатель на один вперед;

10 – присвоить указателю на голову новый указатель на отсортированную последовательность;

11 – конец функции сортировки по возрастанию.

Схема функции добавления элемента в таблицу (рис. 2.7.6).

Поблочное описание:

1 – начало выполнения функции sortTableDescending;

2 – пуста ли таблица, если пуста, то выход из функции;

3 – взять количество элементов в таблице;

4 – цикл for, выполняется столько раз, сколько элементов в таблице;

5 – временные переменные инициализируются для сортировки;

6 – пока numTemps меньше переменной i;

7 – если текущий элемент меньше следущего;

8 – то обменять эти значения местами;

9 – увеличить счётчик и сместить указатель на один вперед;

10 – присвоить указателю на голову новый указатель на отсортированную последовательность;

11 – конец функции сортировки по убыванию.

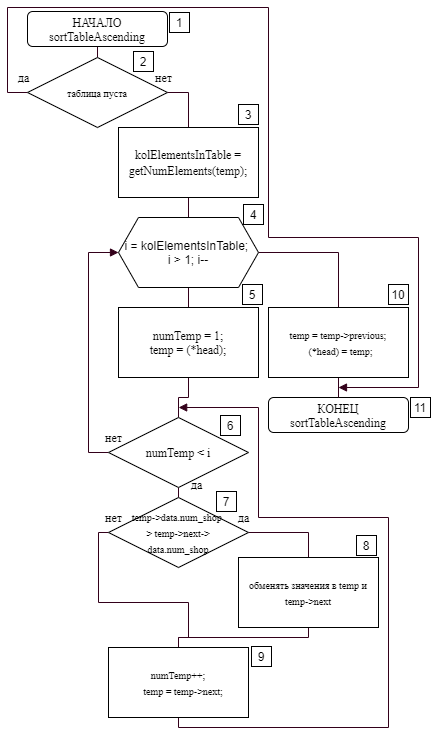


Рисунок 2.7.5 – Структурная схема функции сортировки элементов по возрастанию

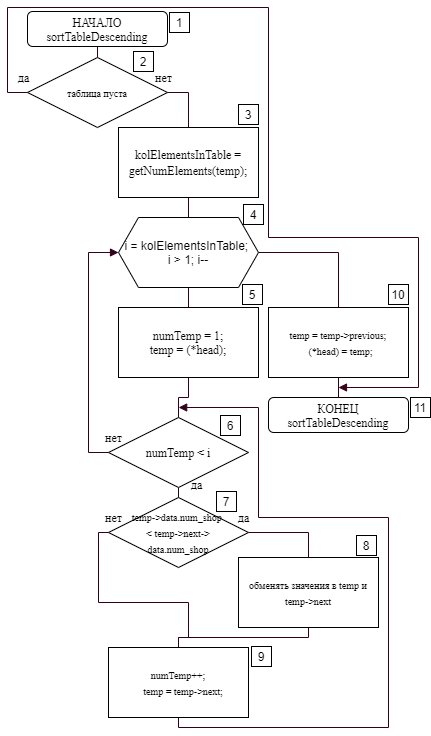


Рисунок 2.7.6 – Структурная схема функции сортировки элементов по убыванию

Схема функции добавления элемента в таблицу (рис. 2.7.7).

Поблочное описание:

1 – начало выполнения функции sortTableDescending;

2 – пуста ли таблица, если пуста, то выход из функции;

3 – взять количество элементов в таблице;

4 – цикл for, выполняется столько раз, сколько элементов в таблице;

5 – временные переменные инициализируются для сортировки;

6 – пока numTemps меньше переменной i;

7 – если текущий элемент меньше следущего;

8 – то обменять эти значения местами;

9 – увеличить счётчик и сместить указатель на один вперед;

10 – присвоить указателю на голову новый указатель на отсортированную последовательность;

11 – конец функции сортировки по убыванию.

