

АННОТАЦИЯ

Пояснительная записка по курсовой работе по дисциплине “Алгоритмизация и программирование”. Институт информационных технологий и управления в технических системах, 2021г. Количество страниц – 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
ВВЕДЕНИЕ.....	4
1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ.....	6
2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ.....	7
2.1. Постановка задачи на разработку программы (Вариант 15).....	7
2.2. Применяемые математические методы.....	8
2.3. Описание и обоснование выбора метода организации входных, выходных и промежуточных данных	9
2.4. Обоснование выбора языка и среды программирования	9
2.5. Разработка модульной структуры программы	10
2.6. Описание функций используемых в программе.....	11
2.7. Описание алгоритмов функционирования программы	14
3 ВЫПОЛНЕНИЕ ПРОГРАММЫ	37
3.1. Условия выполнения программы.....	37
3.2. Загрузка и запуск программы.....	37
3.3. Проверка работоспособности программы.....	42
ВЫВОДЫ	52
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	54
ПРИЛОЖЕНИЕ А	55

ВВЕДЕНИЕ

С появлением и широким распространением табличных процессоров, одним из самых известных представителей которых является Microsoft Excel, 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.

2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ

2.1. 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.

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.

2.2. Применяемые математические методы

В ходе разработки, при вычислении итоговой суммы по одному продукту происходило умножение количества продуктов на цену продукта (рис. 2.1).

```
unsigned long long sum = ((head->data.product_price) * (head->data.product_quantity));
```

Рисунок 2.1 – Вычисление итоговой суммы продукта

Так же при вычислении товарооборота активно применялось сложение итоговых сумм продуктов в одном магазине (рис. 2.2).

```
turnoverSum += (head->data.product_price * head->data.product_quantity);
```

Рисунок 2.2 – Подсчёт товарооборота

2.3. 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.

2.4. 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 удобный интерфейс, а так же подсвечиваются некоторые ошибки на этапе разработки программы, в отличие от блокнота.

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

2.5. 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 2.3.

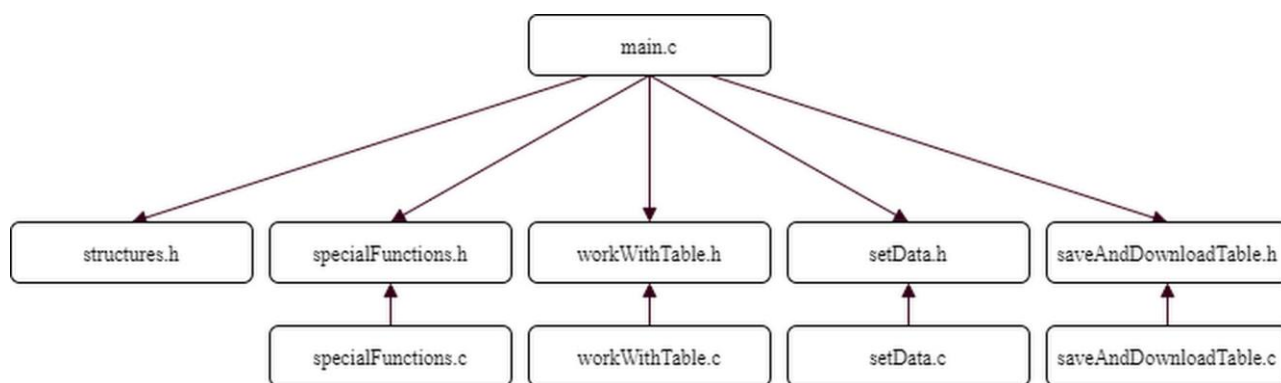


Рисунок 2.3 – Схема взаимодействия частей программы

2.6. 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.

2.7. Описание алгоритмов функционирования программы

Схема главной функции представлена на рисунках 2.4.1 – 2.4.3.

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.

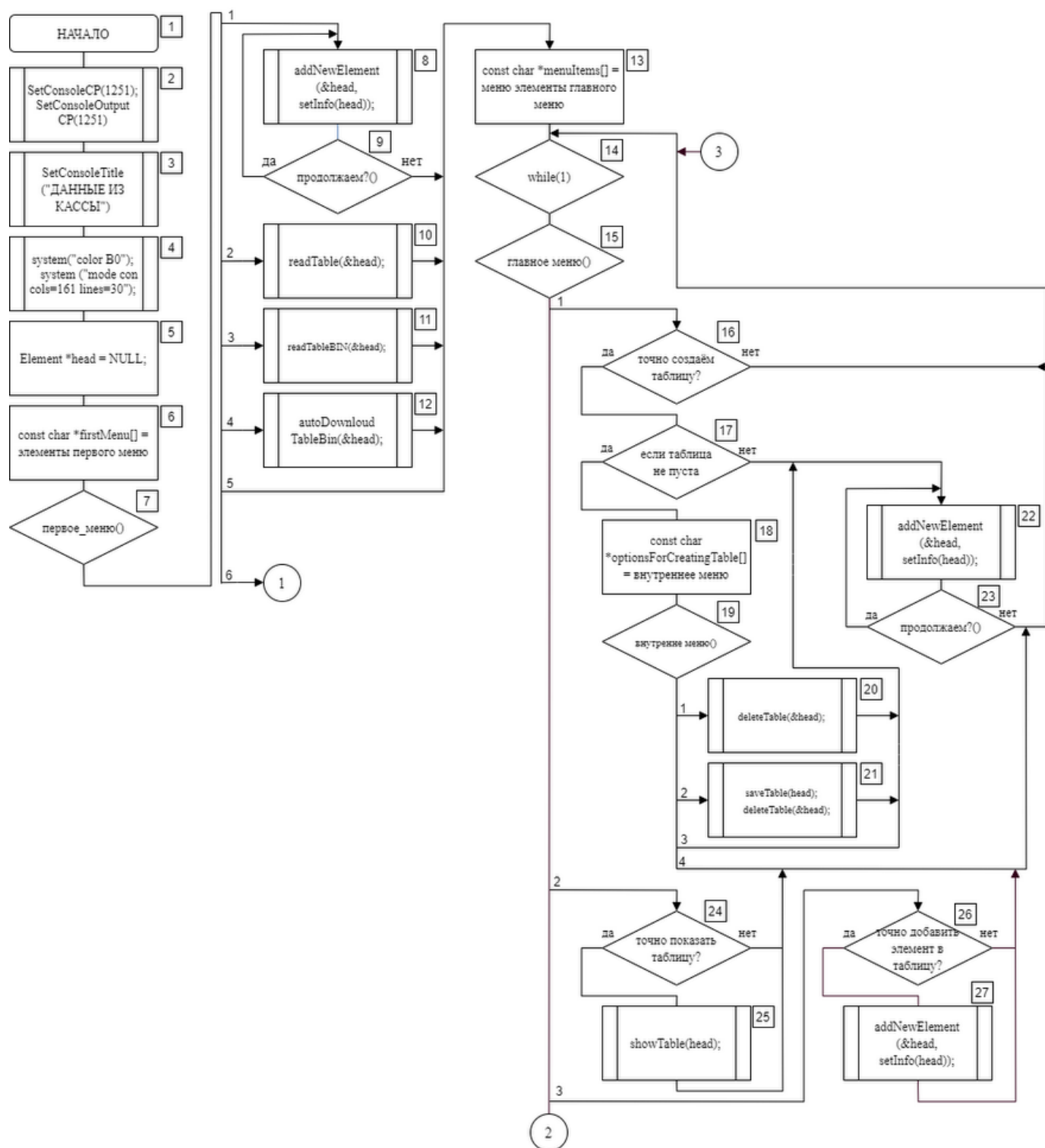
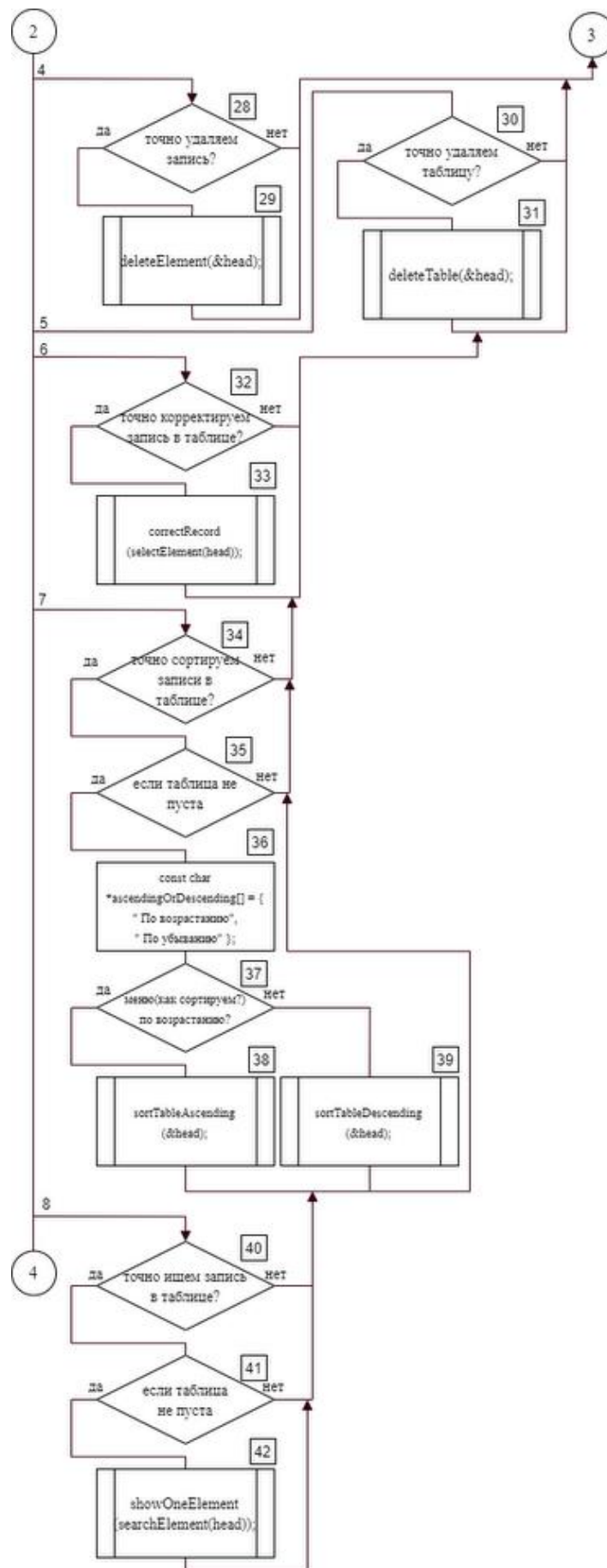


Рисунок 2.4.1 – Структурная схема функции main 1 часть

Рисунок 2.4.2 – Структурная схема функции `main` 2 часть

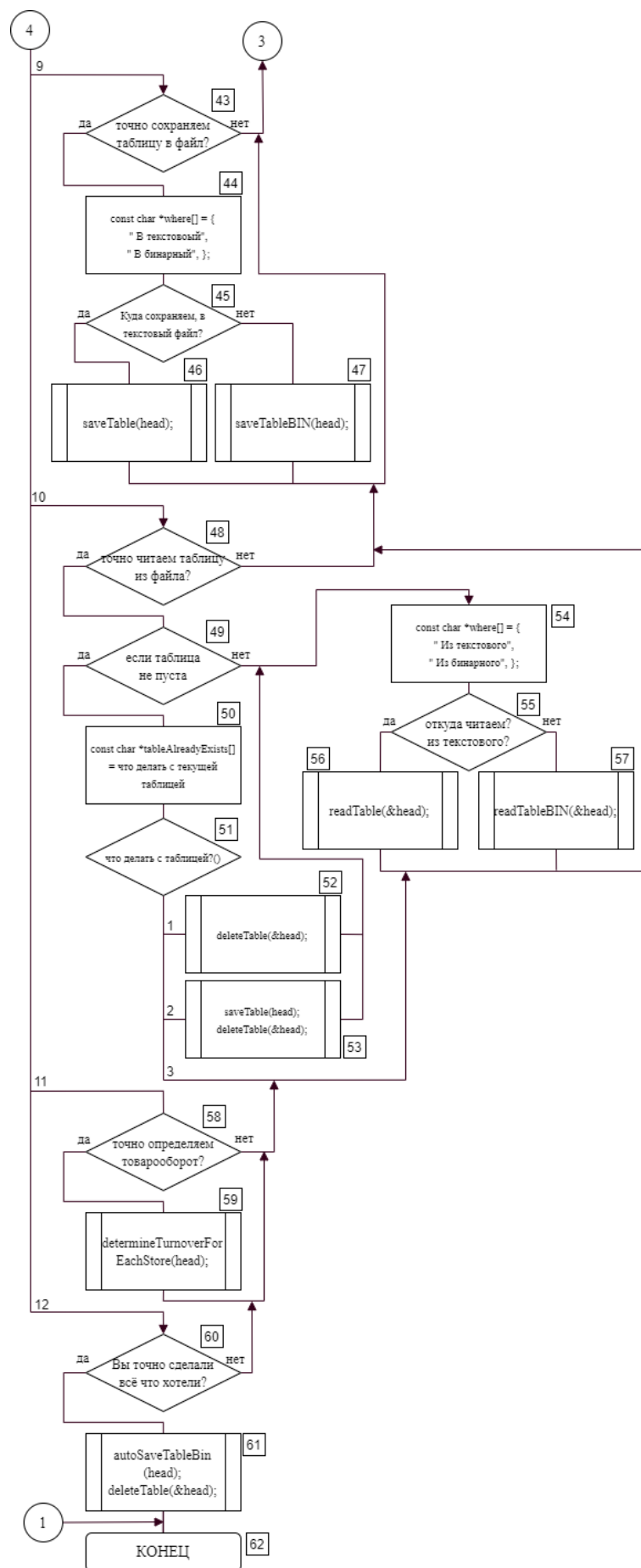


Рисунок 2.4.3 – Структурная схема функции main 3 часть

Схема функции добавления элемента в таблицу (рис. 2.5).

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. 2.6).

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.

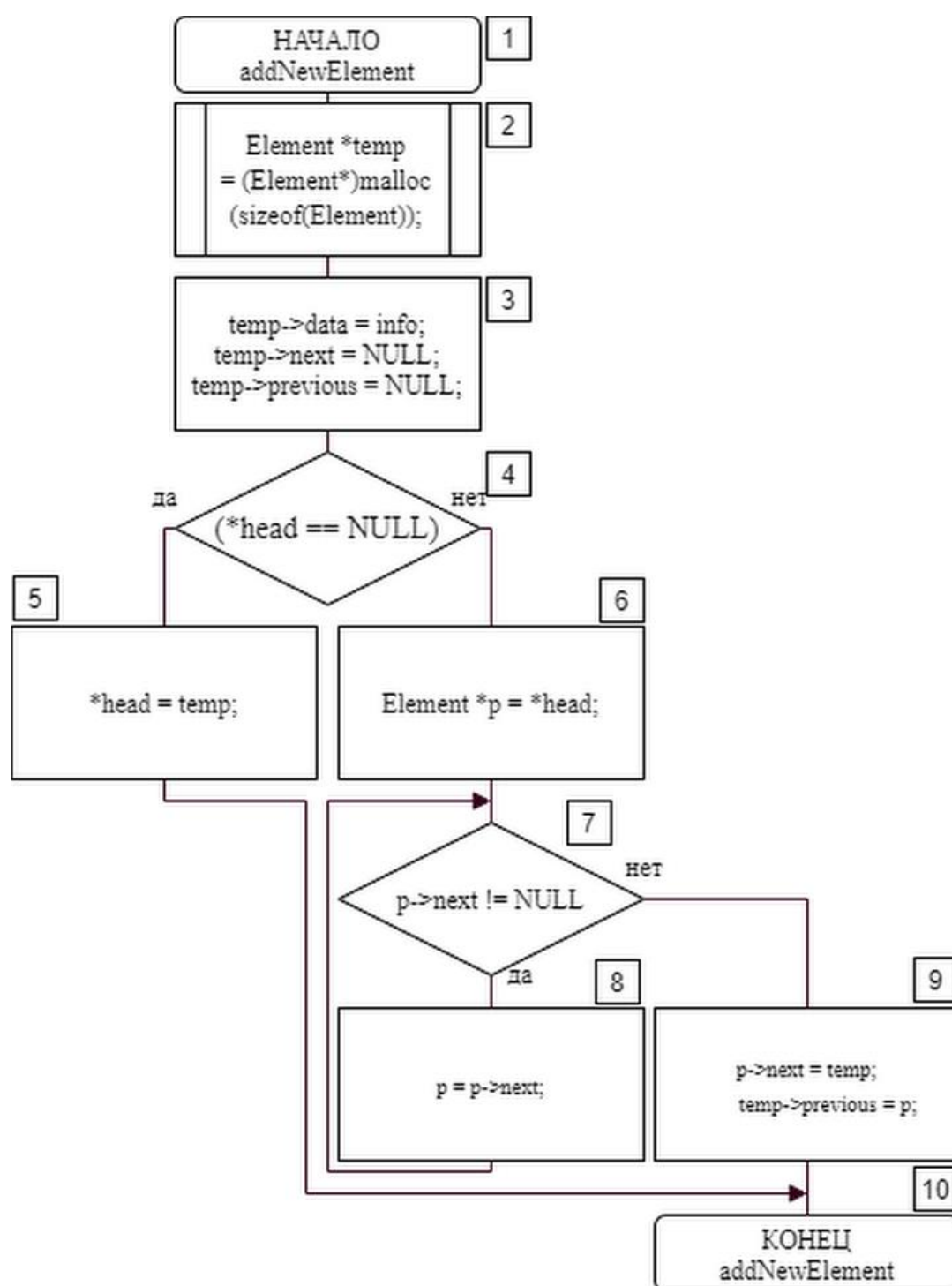


Рисунок 2.5 – Структурная схема функции добавления элемента в таблицу

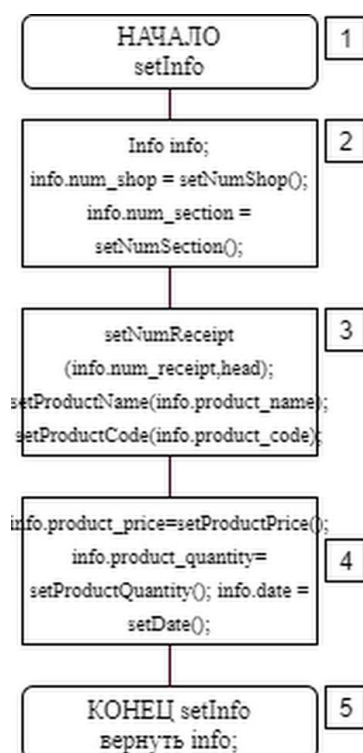


Рисунок 2.6 – Структурная схема функции инициализации полей элемента

Схема функции добавления элемента в таблицу (рис. 2.7).

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.

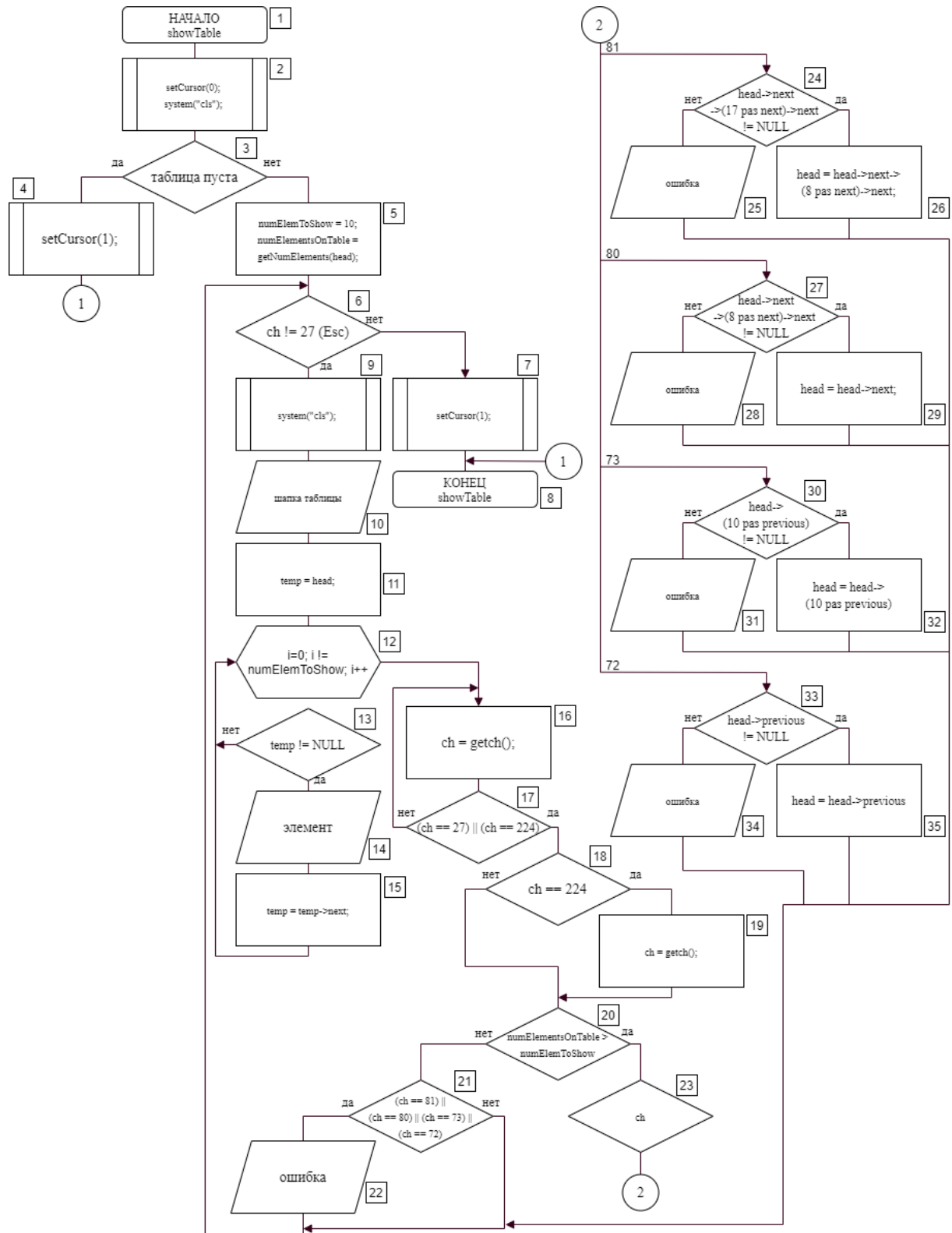


Рисунок 2.7 – Структурная схема функции демонстрации таблицы

Схема функции добавления элемента в таблицу (рис. 2.8).

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 следующего;
- 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. 2.9).

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 следующего;

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.

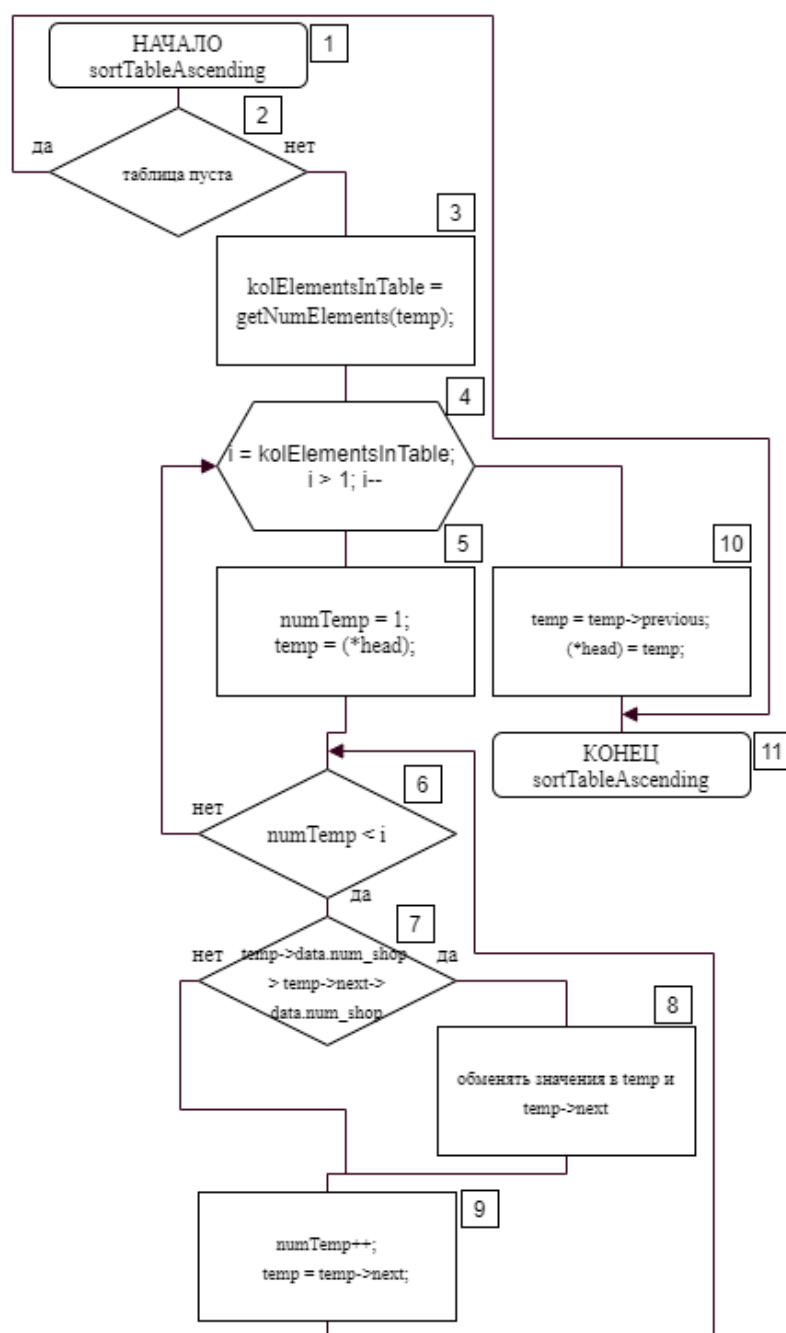


Рисунок 2.8 – Структурная схема функции сортировки элементов по возрастанию

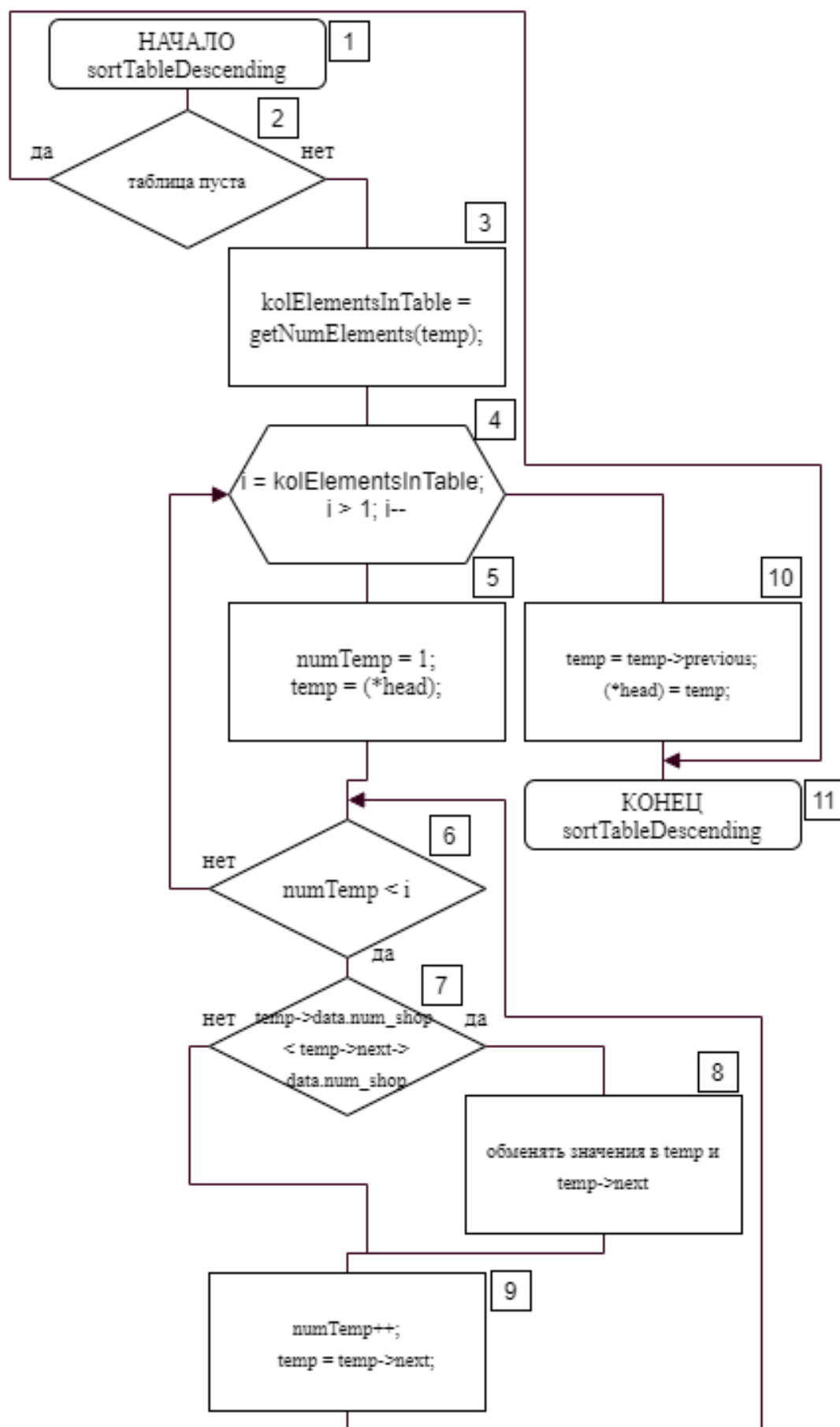


Рисунок 2.9 – Структурная схема функции сортировки элементов по убыванию

Схема функции взятия количества элементов списка (рис. 2.10).

Поблочное описание:

1 – начало выполнения функции getNumElements;

2 – пуста ли таблица, если пуста, то выход из функции, вернуть 0;

3 – иначе пока голова списка не будет указывать на NULL;

4 – добавить единицу к кол-ву элементов, сместить указатель на следующий элемент.

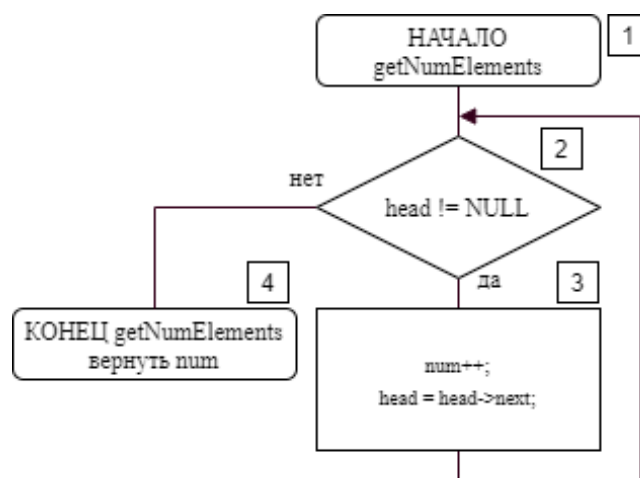


Рисунок 2.10 – Структурная схема функции взятия количества элементов списка

Схема функции удаления элемента из базы данных (рис. 2.11).

Поблочное описание:

1 – начало выполнения функции deleteElement;

2 – очистка экрана;

3 – проверка пуста ли таблица, если да то выход из функции;

4 – вызвать функцию выбора элемента в таблице и присвоить переменной удаляемого элемента возвращаемое значение этой функции;

5 – узнать является ли элемент первым или последним;

6 – если элемент первый, то проверка является ли он единственным;

7 – если он первый и единственный удалить по соответствующему сценарию;

8 – если удаляемый элемент является последним удалить по соответствующему сценарию;

9 – если элемент не первый и не последний удалить по соответствующему сценарию;

10 – убрать указатели удаленного элемента и освободить память;

11 – конец выполнения функции удаления элемента.

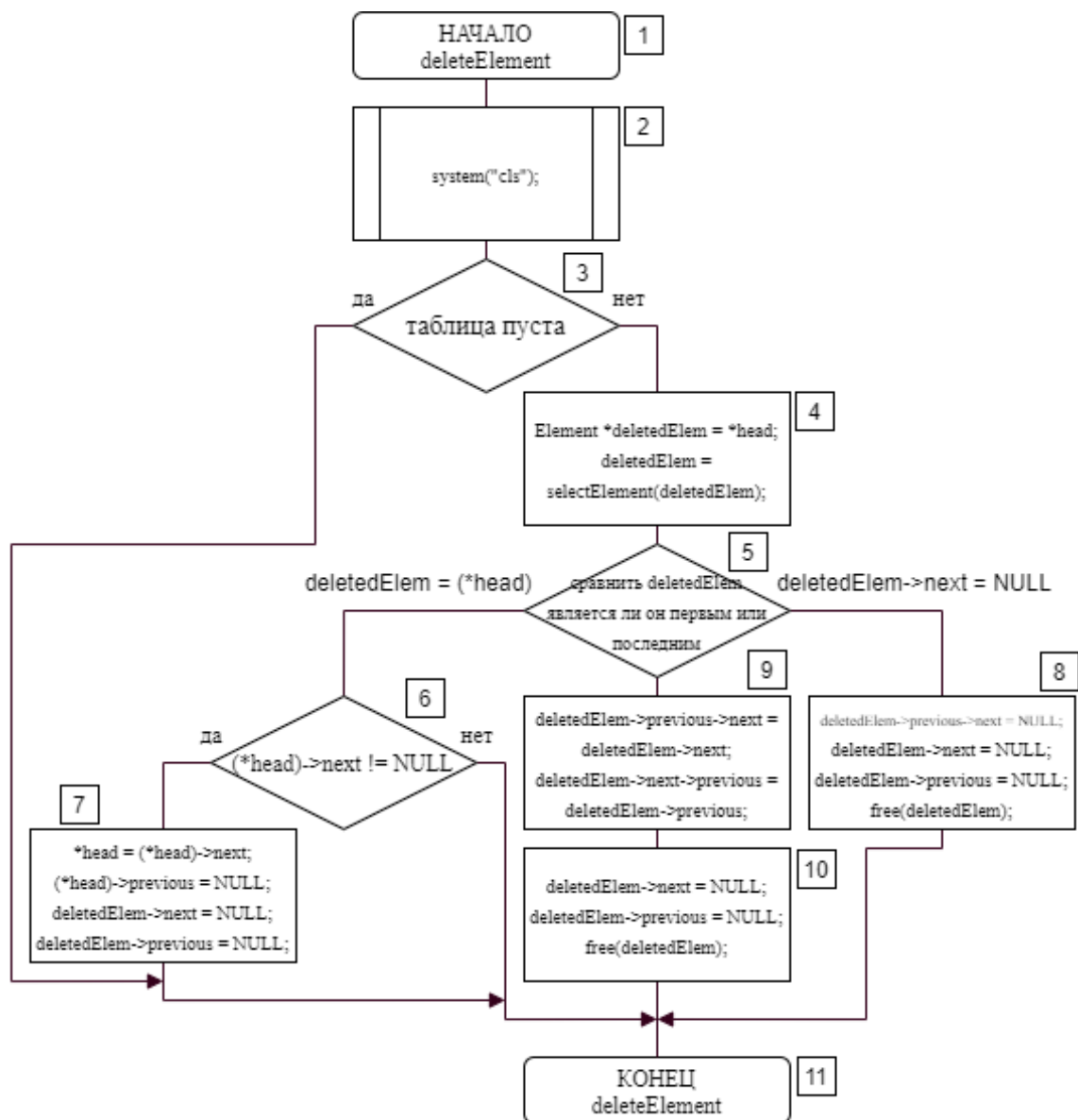


Рисунок 2.11 – Структурная схема функции удаления элемента

Схема функции удаления всех записей в базе данных (рис. 2.12).

Поблочное описание:

- 1 – начало выполнения функции deleteTable;
- 2 – если таблица пуста выйти из функции;
- 3 – создать временную переменную temp;
- 4 – пока голова не равна NULL;
- 5 – присвоить временной переменной текущий элемент и сместить голову;
- 6 – если голова не равна NULL
- 7 – временная переменная равна NULL;
- 8 – очистить память временной переменной;
- 9 – конец функции удаления таблицы.

Схема функции чтения базы данных из текстового файла (рис. 2.13).

Поблочное описание:

- 1 – начало выполнения функции readTable;
- 2 – если таблица пуста, то выйти из функции;
- 3 – очистить экран и строку для ввода информации;
- 4 – ввести путь к файлу загрузки;
- 5 – очистить поток ввода;
- 6 – открыть файл для записи;
- 7 – пока не конец файла;
- 8 – перевести строки в числа если тип поля не строка;
- 9 – добавить элемент в таблицу с помощью функции;

10 – закрыть файл;

11 – вывести пользователю информацию что работа функции окончена успешно;

12 – конец выполнения функции чтения таблицы из файла.

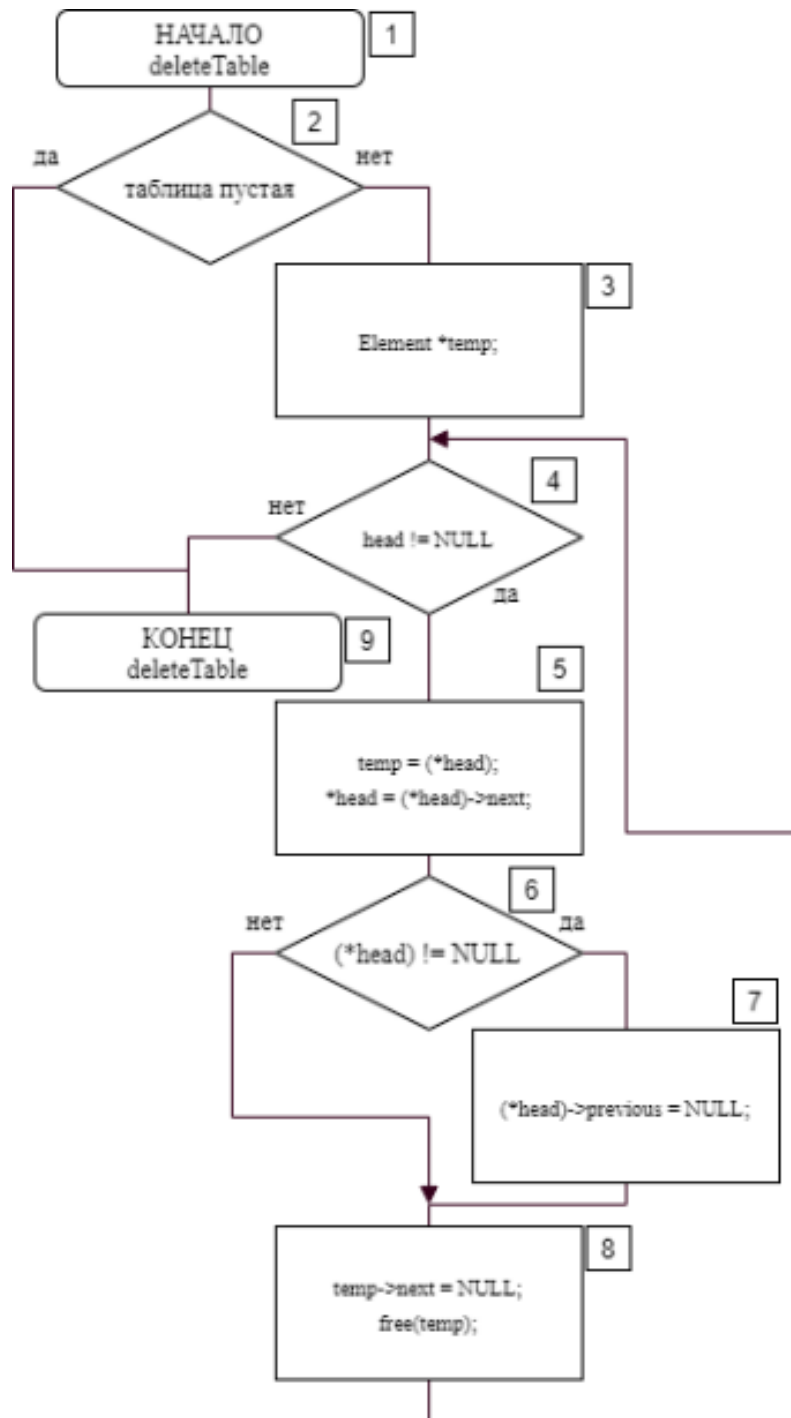


Рисунок 2.12 – Структурная схема функции удаления таблицы

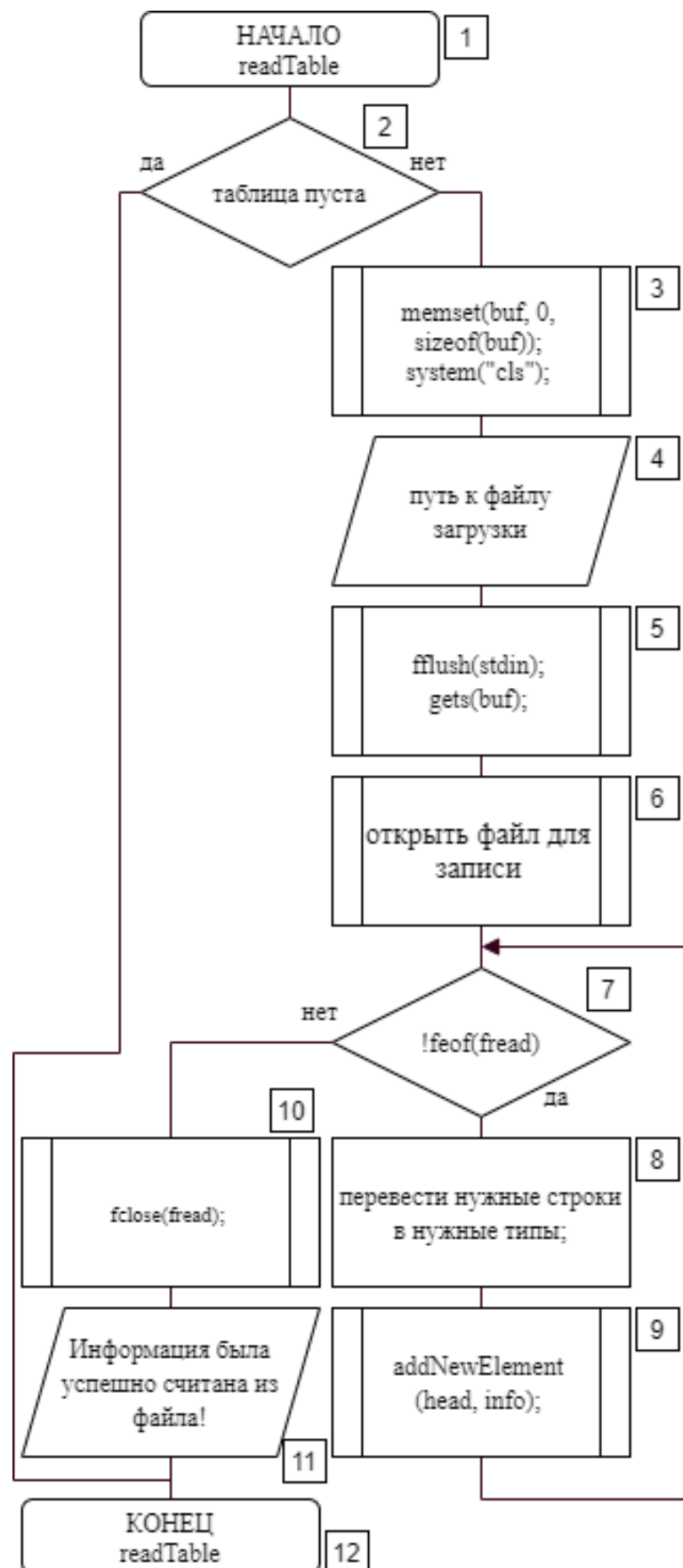


Рисунок 2.13 – Структурная схема функции чтения данных из файла

Схема функции сохранения базы данных в текстовый файл (рис. 2.14).

Поблочное описание:

- 1 – начало выполнения функции saveTable;
- 2 – если таблица пуста, то выйти из функции;
- 3 – очистить экран и строку для ввода информации;
- 4 – ввести путь к файлу сохранения;
- 5 – очистить поток ввода;
- 6 – если длина введенного пути к файлу больше 4;
- 7 – если последние 4 символа = '.txt';
- 8 – звук ошибки;
- 9 – открыть файл для чтения;
- 10 – пока указатель на голову не равен NULL;
- 11 – перевести все числа в строки, и записать все строки в файл;
- 12 – сместить указатель на голову на следующий элемент;
- 13 – закрыть файл загрузки;
- 14 – сообщить пользователю, что программа успешно окончила загрузку;
- 15 – конец выполнения функции сохранения базы данных в текстовый файл.

Схема функции демонстрации одного элемента (рис. 2.15).

Поблочное описание:

- 1 – начало выполнения функции showOneElement;
- 2 – Если указатель на передаваемый элемент = NULL, то выйти из функции;
- 3 – отобразить данные об одном элементе таблицы в виде красивой формы;

4 – Конец выполнения отображения элемента.

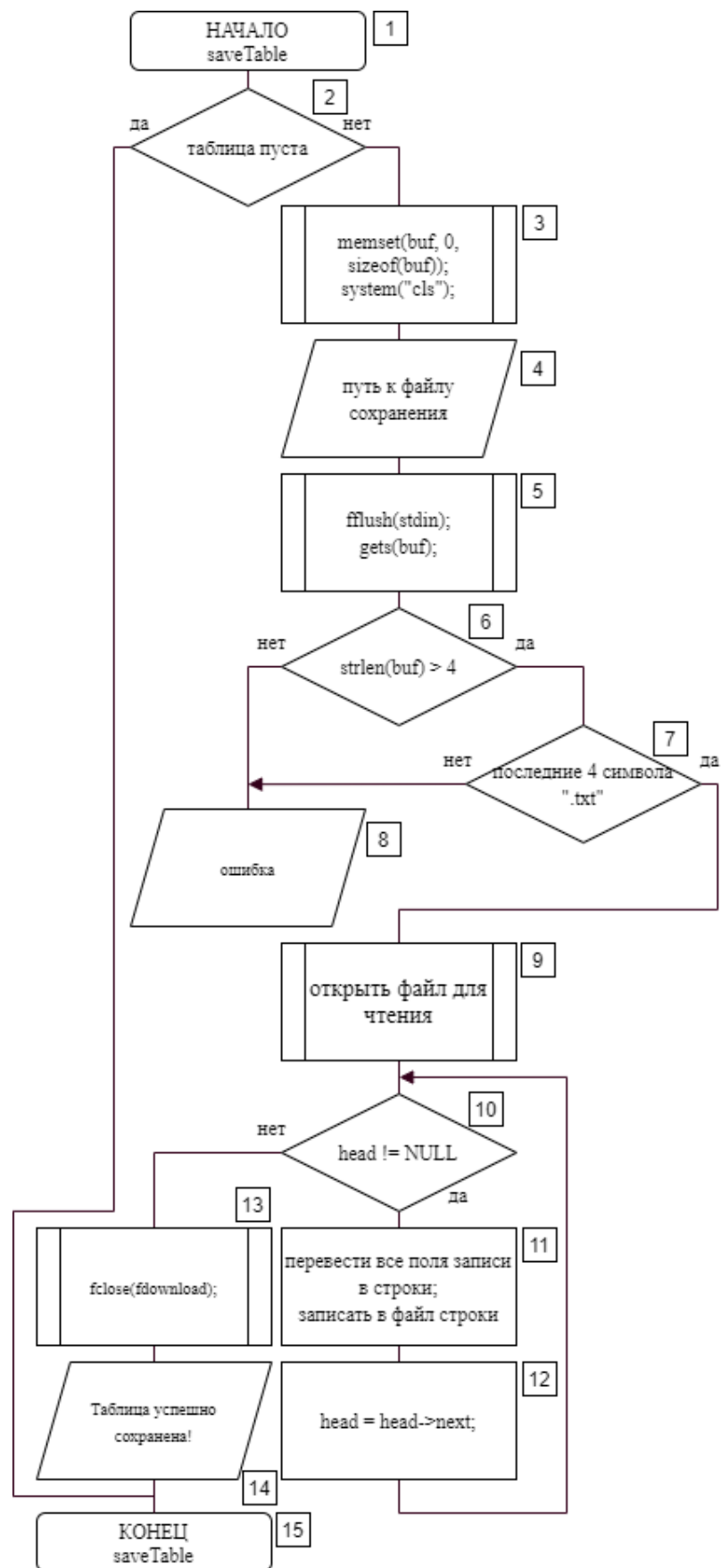


Рисунок 2.14 – Структурная схема функции сохранения таблицы в текстовый файл

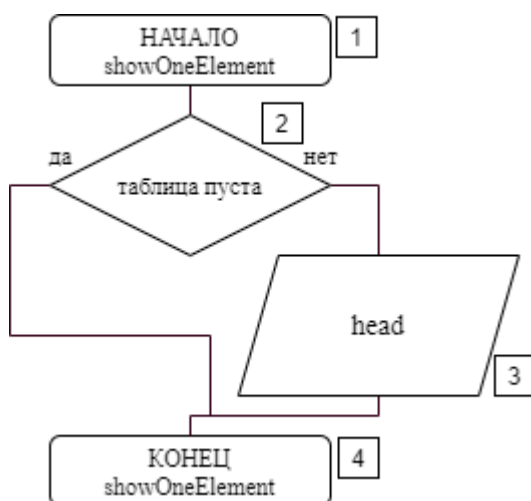


Рисунок 2.15 – Структурная схема функции демонстрации одного элемента

3 ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1. Условия выполнения программы

Так как программа изначально разрабатывалась для исполнения под ОС Windows и требует сравнительно ничтожных ресурсов, системные требования к компьютеру, на котором может осуществляться её запуск соответствуют системным требованиям ОС, приведённым ниже в таблице 3.1.

Таблица 3.1 – минимальный состав аппаратных и программных средств

Процессор	Intel core i3 пятого поколения (и выше)
ОЗУ	2 ГБ
Минимальная ёмкость жесткого диска SSD/HDD или Flash-накопителя	1 ГБ
Видеокарта с операционной памятью	Не менее 64МБ
Операционная система	Семейство Windows (8 и выше)
Устройство ввода	Клавиатура, мышь (необязательно)
Устройство вывода	Монитор с разрешением HD или больше

3.2. Загрузка и запуск программы

Программа состоит из одного файла – «Program.exe». Данный элемент является исполняемым. Ниже представлена последовательность действий для пользователя для запуска программы:

- программа может быть установлена с любого носителя или интернета;

- для работы программы не требуется специальная инсталляция;
- поместите программу на необходимую директорию на диске;
- запустите исполняемый файл «Program.exe» нажав двойным щелчком мыши по исполняемому файлу;
- альтернативным вариантом можно запустить программу через консоль Windows, для этого нажмите “win + R”, введите в строку “Открыть” слово “cmd”, с помощью команды “cd” перейдите в нужную директорию и запустите программу с помощью команды “start” (рис. 3.1), программа откроется в новом окне.



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.1348]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Danil>cd /D "D:\SevsU_IS\3_Sem\Algorithmization_and_programming\Coursework"
D:\SevsU_IS\3_Sem\Algorithmization_and_programming\Coursework>start Program.exe
D:\SevsU_IS\3_Sem\Algorithmization_and_programming\Coursework>
  
```

Рисунок 3.1 – Запуск программы

В начале запуска, исполняемого файла «Program.exe», появится консоль. В консоли будет выведено стартовое меню программы. Это точка вхождения в программу откуда начинается её работа. Навигация по меню осуществляется с помощью клавиш стрелок вверх и вниз или клавиш “Page Up” или “Page Down”, для выбора пункта меню необходимо нажать клавишу “Enter”. На рисунке 3.2 приведен пример стартового меню.

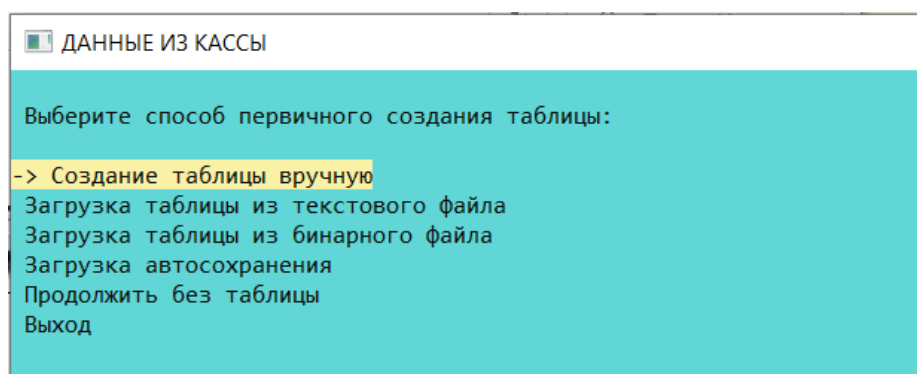
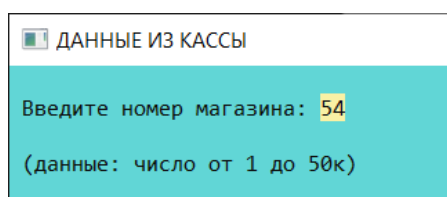


Рисунок 3.2 – Стартовое меню программы

Среди пунктов стартового меню можно наблюдать пункт создания таблицы вручную, где будет происходить ручной ввод с клавиатуры данных о продуктах. На рисунках 3.3 – 3.13 отображен ввод данных с клавиатуры, таким образом происходит ручное заполнение базы данных. На рисунке 3.3 осуществляется ввод номера магазина, если будет введено не число или число будет больше 50000 или меньше 1, то воспроизведется звук ошибки и будет осуществлен повторный ввод, до тех пор пока не будут соблюдены условия ввода. На рисунке 3.4 аналогично осуществляется ввод номера секции с отличием в том, что ограничение на ввод числа не больше 1000. На рисунке 3.4 отображен ввод номера чека, если будет введено не 20 чисел, то программа сообщит пользователю об ошибке звуковым сигналом, так же программа не позволит ввести 2 одинаковых чека в базу данных. На рисунке 3.5 осуществляется ввод наименования товара с ограничением в 50 символов. На рисунке 3.6 аналогично осуществляется ввод артикула с ограничением в 20 символов. Рисунок 3.8 содержит в себе отображение ввода цены товара с ограничением в 1 миллиард рублей. На рисунке 3.9 осуществляется ввод количества товара. На рисунках 3.10 – 3.12 осуществляется ввод даты продажи с помощью стрелок вверх или вниз, программа не позволит ввести недопустимую дату. Затем после корректного ввода всех данных элемента списка у пользователя спрашивается хочет ли он продолжить добавление элементов в список (рис. 3.13).

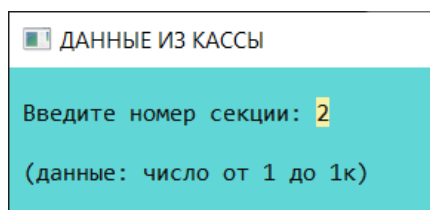


ДАННЫЕ ИЗ КАССЫ

Введите номер магазина: 54

(данные: число от 1 до 50к)

Рисунок 3.3 – Ввод номера магазина

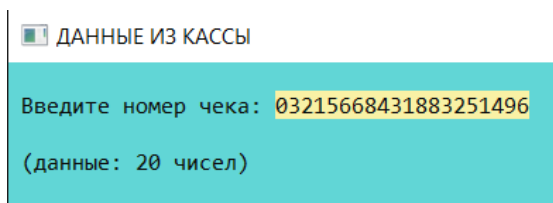


ДАННЫЕ ИЗ КАССЫ

Введите номер секции: 2

(данные: число от 1 до 1к)

Рисунок 3.4 – Ввод номера секции

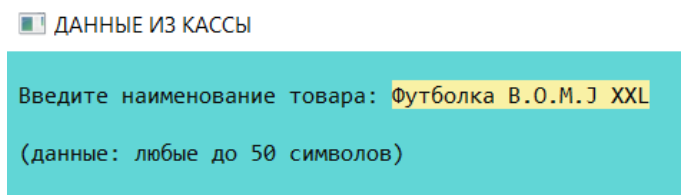


ДАННЫЕ ИЗ КАССЫ

Введите номер чека: 03215668431883251496

(данные: 20 чисел)

Рисунок 3.5 – Ввод номера чека

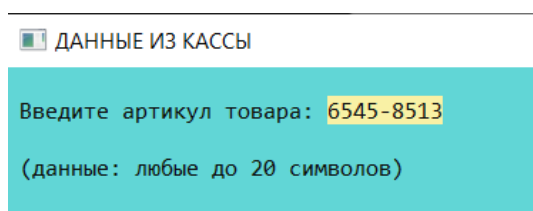


ДАННЫЕ ИЗ КАССЫ

Введите наименование товара: Футболка В.О.М.И XXL

(данные: любые до 50 символов)

Рисунок 3.6 – Ввод наименования продукта

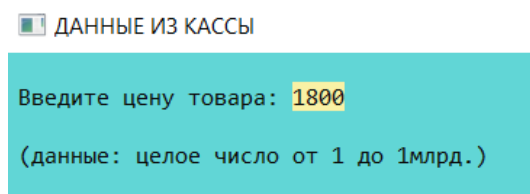


ДАННЫЕ ИЗ КАССЫ

Введите артикул товара: 6545-8513

(данные: любые до 20 символов)

Рисунок 3.7 – Ввод артикула продукта

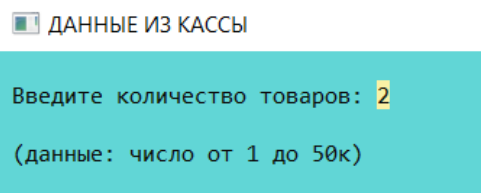


ДАННЫЕ ИЗ КАССЫ

Введите цену товара: 1800

(данные: целое число от 1 до 1млрд.)

Рисунок 3.8 – Ввод цены продукта

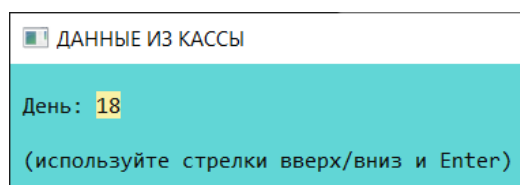


ДАННЫЕ ИЗ КАССЫ

Введите количество товаров: 2

(данные: число от 1 до 50к)

Рисунок 3.9 – Ввод количества товаров



ДАННЫЕ ИЗ КАССЫ

День: 18

(используйте стрелки вверх/вниз и Enter)

Рисунок 3.10 – Ввод дня продажи

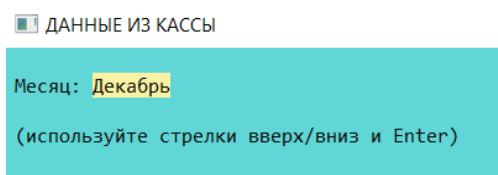


Рисунок 3.11 – Ввод месяца продажи

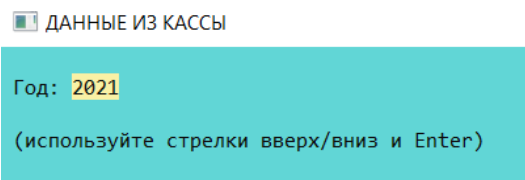


Рисунок 3.12 – Ввод года продажи

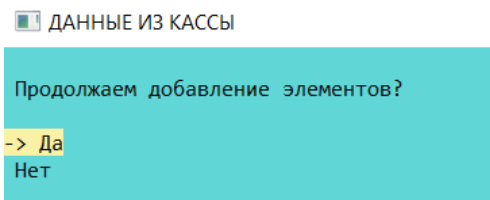


Рисунок 3.13 – Вопрос для пользователя

Альтернативно можно загрузить таблицу из текстового или бинарного файла, если у пользователя были сохранения. Так же можно загрузить последнее автосохранение, которое создавалось при последнем выходе из программы. При запуске этих пунктов меню во всех случаях аналогично запрашивается осуществить ввод пути к файлу с указанием типа (.txt или .bin), если будет указан не корректный путь, не корректный тип или доступ к файлу не сможет быть осуществлен программа выдаст звуковой сигнал и потребует осуществить повторный ввод пути к файлу (рис. 3.14). При успешном выполнении загрузки базы данных, программа сообщит об этом пользователю (рис. 3.15).

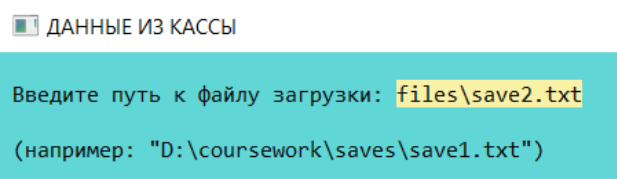


Рисунок 3.14 – Ввод пути к файлу загрузки

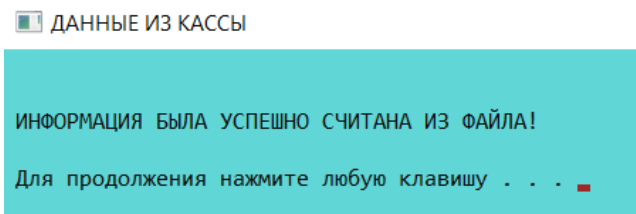


Рисунок 3.15 – Сообщение об успешном завершении выполнения функции

Можно продолжить работать с программой без таблицы или выйти из неё вовсе. Если пользователь не вышел из программы пользователю откроется доступ к главному меню (рис. 3.16).

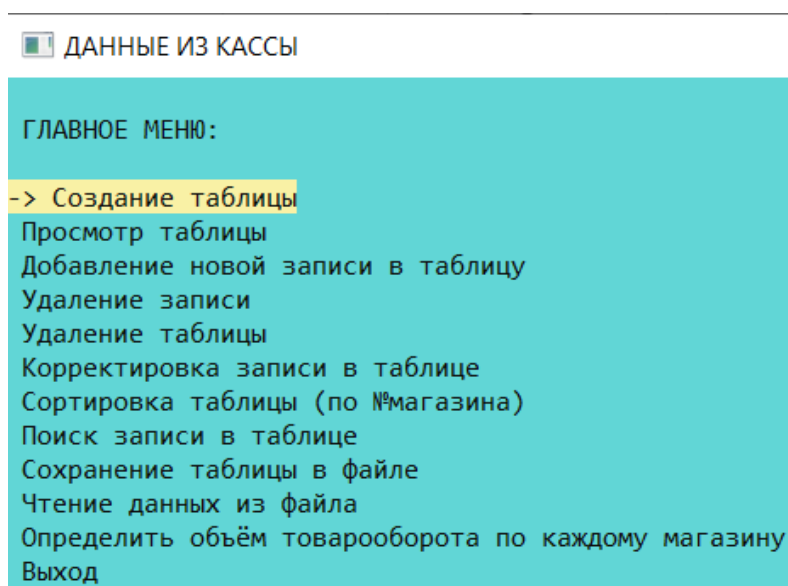


Рисунок 3.16 – Главное меню программы

3.3. Проверка работоспособности программы

После работы со вступительным меню открывается главное меню в котором присутствуют 12 пунктов навигация по которым осуществляется аналогично как и во вступительном меню. При вызове каждого из пунктов будет вызываться контекстное меню подтверждения вызываемого пункта. Так же при отсутствии таблицы будет выводиться соответствующее сообщение.

Если зайти в главное меню с пустой таблицей, то логично будет либо создать новую с помощью первого пункта “Создание таблицы” (см. рисунки 3.3 – 3.13) либо считать таблицу из файла (десятый пункт).

Было произведено чтение данных из файла. При вызове этого пункта выводится контекстное меню, которое спрашивает у пользователя точно ли он хочет считать таблицу из файла (рис. 3.17). Если выбрать “нет”, то пользователь вернется в главное меню.

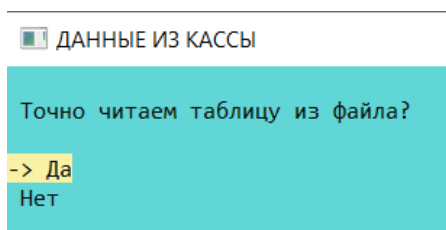


Рисунок 3.17 – Подтверждение чтения файла

Если выбрать “да”, то вызовется ещё одно контекстное меню, которое спрашивает из какого файла считывать (из текстового или бинарного). После осуществляется ввод пути к файлу загрузки и выводится сообщение (см. рис. 3.14 и 3.15).

После того как файл был загружен можно выбрать второй пункт “Просмотр таблицы” и просмотреть содержимое таблицы (рис. 3.19) перед этим подтвердив свои намерения (рис. 3.18).

Таблица отображается по 10 элементов, клавишами стрелок вверх и вниз можно скроллить содержимое таблицы по одному элементу вверх или вниз, или же клавишами “Page Up” и “Page Down” скроллить таблицу сразу по 10 элементов вверх или вниз. Если при нажатии клавиши происходит звуковой сигнал ошибки, это означает что вверху или внизу больше нет элементов и скроллинг осуществить не возможно. На рисунке 3.20 отображены следующие 10 элементов таблицы после нажатия клавиши “Page Down”. Если же попробовать посмотреть таблицу, когда её нет, то на экран выведется соответствующее сообщение (рис. 3.21). При нажатии “Esc” происходит выход в главное меню.

ДАННЫЕ ИЗ КАССЫ

Точно показать таблицу?

-> Да

Нет

Рисунок 3.18 – Подтверждение выполнения функции демонстрации таблицы

ДАННЫЕ ИЗ КАССЫ

Навигация: стрелочками вверх/вниз; Esc - выход

#МАГ	#СЕК	НОМЕР ЧЕКА	НАИМЕНОВАНИЕ ТОВАРА	АРТИКУЛ ТОВАРА	ЦЕНА ТОВАРА	КОЛ-ВО ТОВАРА	ДАТА ПРОДАЖИ
2	3	21209878451100390716	Смартфон realme GT Master Edition 8+256GB Daybreak	R3103QFEJU	32999	9	17 / 5 / 2021
54	2	83922422913761516077	Смартфон realme 8 PRO 6+128GB Infinite Blue	TRUGGGL3K0	24999	68	13 / 4 / 2020
1	3	23022113907847070278	Смартфон Xiaomi Redmi 9T 4+64GB Orange	0VNJ138K3N	12999	1	14 / 11 / 2020
35302	7	93546488932634990673	Наушники накладные Bluetooth Sony WH-CH710N	9A05UL2840	5999	2	6 / 8 / 2020
35302	10	47193263242108640906	Ноутбук Acer Swift 1 SF114-34-C857 NX.A78ER.005	F757H-68LE	30999	1	27 / 1 / 2021
69	9	83431366247819338289	Умная колонка Яндекс Станция Лайт	RZ5R9KGJLT	3999	5	23 / 6 / 2021
1	7	46404614880523268006	Телевизор Samsung UE43AU7570U	X2CG6N14IX	39999	1	24 / 12 / 2020
69	3	86160892541234015189	Робот-пылесос Tefal Smart Force X-plorer	XDS3GCK99W	9999	33	12 / 5 / 2019
35302	4	35390623124590467077	Стиральная машина узкая Candy SmartPro	ZQVJRJISWF	20490	10	10 / 11 / 2021
1	10	20991278599861401736	Цифровая смарт ТВ-приставка Sber SberBox	UYKFCIOVFO	14999	1	6 / 5 / 2021

Рисунок 3.19 – Содержимое таблицы при открытии

ДАННЫЕ ИЗ КАССЫ

Навигация: стрелочками вверх/вниз; Esc - выход

#МАГ	#СЕК	НОМЕР ЧЕКА	НАИМЕНОВАНИЕ ТОВАРА	АРТИКУЛ ТОВАРА	ЦЕНА ТОВАРА	КОЛ-ВО ТОВАРА	ДАТА ПРОДАЖИ
54	6	10118456708564495796	Наушники True Wireless Huawei Freebuds 4	J17TAMDD0Y	10999	2	20 / 12 / 2021
54	3	62415320857358091582	Электрогриль Tefal Optigrill XL	MFE13Y08E8	19999	1	5 / 12 / 2019
54	3	02775089076586255554	Электрочайник Russell Hobbs Luna Solar Red	5Q6UCQTXOJ	3999	2	17 / 3 / 2021
54	5	86284537057937829401	Электрическая зубная щетка Braun	YEK4JA8P3T	4299	1	20 / 7 / 2020
2	7	28597550613333358943	Электробритва Braun 50	1JHUM9QKAX	7499	1	21 / 7 / 2020
35302	10	71509858879165573808	Триммер Rowenta Forever Sharp	41HBX7N6AX	2499	3	17 / 1 / 2019
69	1	38831742349273340289	Эпилятор Braun 3-321 Legs & Body	2PTV84JISQ	3499	1	13 / 11 / 2019
2	5	31134049104808351202	Умная колонка Новая Яндекс .Станция Мини	1KY3VE7QX7	6999	1	1 / 11 / 2020
35302	6	77254009019130860541	Смартфон Samsung Galaxy Z Flip3 128GB Black	DZDY1171NT	89990	1	29 / 1 / 2020
35302	9	66698881620195404453	Наушники True Wireless Samsung Galaxy Buds2	Q7K6W621WY	10990	12	9 / 12 / 2020

Рисунок 3.20 – Содержимое таблицы следующие 10 элементов

ДАННЫЕ ИЗ КАССЫ

ТАБЛИЦА ПУСТА!

Для продолжения нажмите любую клавишу . . .

Рисунок 3.21 – Отображение таблицы при её отсутствии

Если выбрать первый пункт, когда таблица уже существует, то будут предложены 4 пункта меню (рис. 3.22). При выборе первого пункта текущая таблица будет удалена без сохранения и начнется процесс создания новой (см. рисунки 3.3 – 3.13). При выборе второго пункта меню вызовется процесс сохранения таблицы, затем текущая таблица удалится и начнется создание новой. При выборе третьего пункта меню начнут добавляться элементы в текущую таблицу. При выборе четвертого пункта будет осуществлен выход в главное меню.

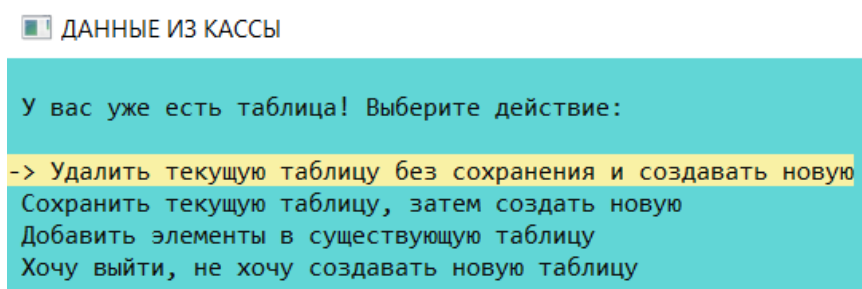


Рисунок 3.22 – Контекстное меню при наличии таблицы и попытке создания новой

При выборе третьего пункта главного меню произведется добавление одной записи в текущую таблицу (см. рисунки 3.3 – 3.12).

При выборе четвертого пункта меню “Удаление записи” вызовется контекстное меню подтверждения (рис. 3.23). Затем выведется таблица с инструкцией по управлению в шапке таблицы. Стрелками вверх и вниз выбирается удаляемый пункт меню. Клавиша Enter подтверждает удаление. При успешном удалении соответствующее уведомление выведется на экран внизу таблицы. (Рисунок 3.24)

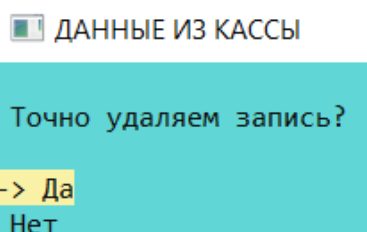


Рисунок 3.23 – Контекстное меню подтверждения

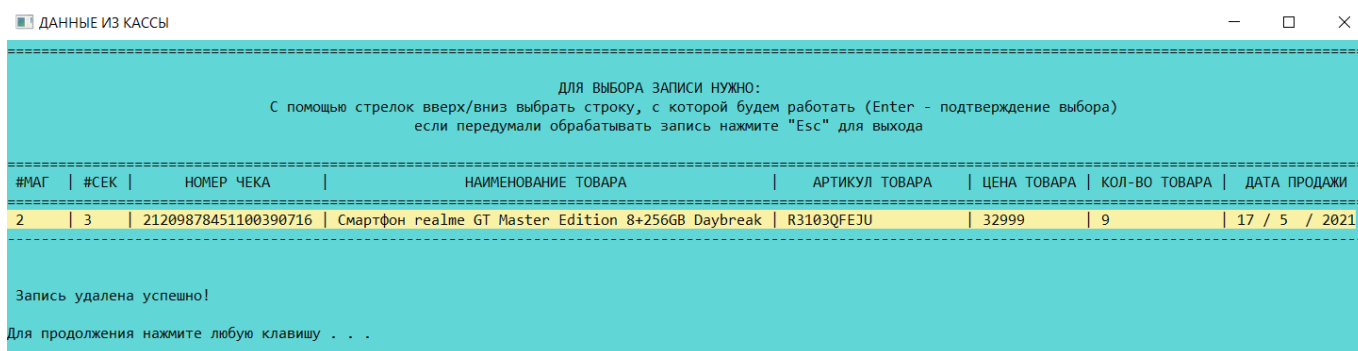


Рисунок 3.24 – Удаление записи в базе данных

При выборе пятого пункта меню текущая таблица будет полностью удалена при подтверждении этого действия (рис. 3.25).

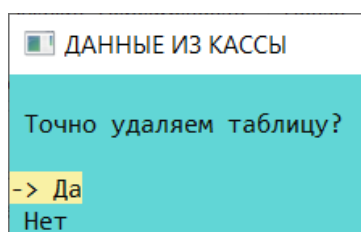


Рисунок 3.24 – Удаление всех записей в текущей базе данных

При выборе шестого пункта меню вызовется процедура корректировки записи в таблице. Открывается таблица, как в процедуре удаления, где стрелками вверх и вниз выбирается нужная запись (рис. 3.25). Далее стрелками влево и вправо выбирается поле элемента, которое будет подвержено изменению (рис. 3.26). Затем вызывается соответствующая процедура ввода данных в соответствии с выбранным полем (см. рисунки 3.3 – 3.12).

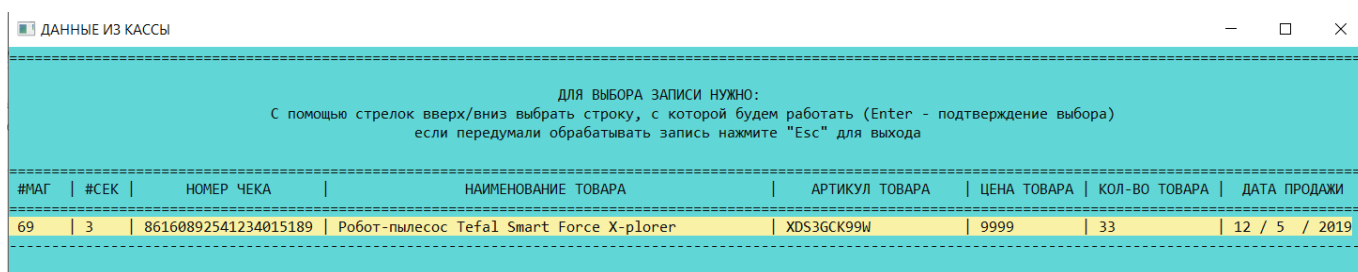


Рисунок 3.25 – Выбор обрабатываемой записи

ДАННЫЕ ИЗ КАССЫ

ДЛЯ КОРРЕКТИРОВКИ ЗАПИСИ НУЖНО:
С помощью стрелок влево/вправо выбрать корректируемую запись (Enter - подтверждение выбора)
если передумали корректировать запись нажмите "Esc" для выхода

#МАГ	#СЕК	НОМЕР ЧЕКА	НАИМЕНОВАНИЕ ТОВАРА	Артикул товара	Цена товара	кол-во товара	Дата продажи
69	3	86160892541234015189	Робот-пылесос Tefal Smart Force X-plorer	XDS3GCK99W	9999	33	12 / 5 / 2019

Рисунок 3.26 – Выбор обрабатываемого поля записи

При выборе седьмого пункта меню “Сортировка таблицы” вызывается контекстное меню, которое спрашивает пользователя сортировать записи по возрастанию или убыванию (рис. 3.27). Далее выводится сообщение об успешном выполнении сортировки, если так и было (рис. 3.28). После выполнения сортировки записи в таблице становятся отсортированными (рис.3.29).

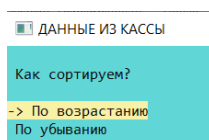


Рисунок 3.27 – Выбор направления сортировки

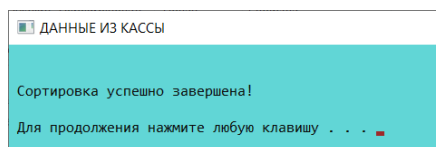


Рисунок 3.28 – Сообщение об успешной сортировке

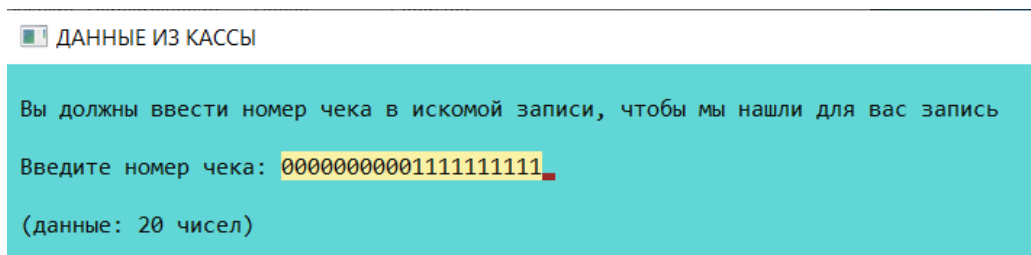
ДАННЫЕ ИЗ КАССЫ

Навигация: стрелочками вверх/вниз; Esc - выход

#МАГ	#СЕК	НОМЕР ЧЕКА	НАИМЕНОВАНИЕ ТОВАРА	Артикул товара	Цена товара	кол-во товара	Дата продажи
1	4	75849524629729284311	Робот-пылесос iRobot iRobot Roomba 694	H0G1D10WXA	12999	1	14 / 7 / 2019
1	9	81220369717651106507	Наушники Apple AirPods Pro with Wireless Case	WYRDS8LL9G	18999	1	2 / 8 / 2019
1	1	00000000001111111111	Видеокарта ASUS RTX 3090	D6Q09ZEL8I	359990	6	16 / 4 / 2019
2	3	21209878451100390716	Смартфон realme GT Master Edition 8+256GB Daybreak	R3103QFEJU	32999	9	17 / 5 / 2021
2	7	28597550613333358943	Электробритва Braun 50	1JHUM9QKAX	7499	1	21 / 7 / 2020
2	5	31134049104808351202	Умная колонка Новая Яндекс .Станция Мини	1KY3VE7QX7	6999	1	1 / 11 / 2020
2	4	71904952631864216937	Телевизор Huawei Vision S	JGBA347RLO	52599	20	10 / 12 / 2021
2	5	60272370996514907565	Ноутбук Honor MagicBook X 15 15/8/512 Gray	NOSM2WXC8E	59999	1	16 / 1 / 2020
54	2	83922422913761516077	Смартфон realme 8 PRO 6+128GB Infinite Blue	TRUGGGL3K0	24999	68	13 / 4 / 2020
54	6	10118456708564495796	Наушники True Wireless Huawei Freebuds 4	J17TAMDDOY	10999	2	20 / 12 / 2021

Рисунок 3.29 – Отсортированная таблица

При выборе восьмого пункта меню “Поиск записи в таблице”, требуется ввести номер чека, так как номера чеков в таблице уникальны (рис. 3.30). Если пользователь укажет не правильный номер чека, то будет предложено повторить попытку ввода (рис. 3.31). На рисунке 3.32 отображена выходная форма найденной записи с подсчётом итоговой суммы по продукту (количество продуктов, умноженное на его цену).



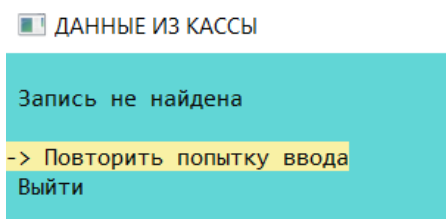
ДАННЫЕ ИЗ КАССЫ

Вы должны ввести номер чека в искомой записи, чтобы мы нашли для вас запись

Введите номер чека: 00000000001111111111

(данные: 20 чисел)

Рисунок 3.30 – Ввод номера чека



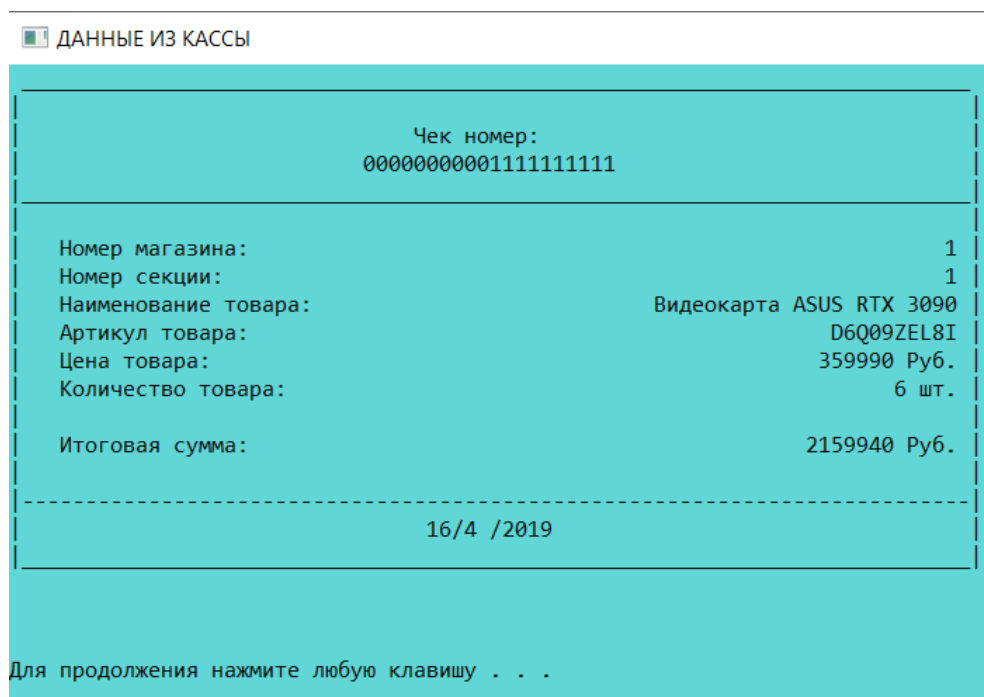
ДАННЫЕ ИЗ КАССЫ

Запись не найдена

-> Повторить попытку ввода

Выйти

Рисунок 3.31 – Меню при вводе несуществующего номера чека



ДАННЫЕ ИЗ КАССЫ

Чек номер:		00000000001111111111
Номер магазина:		1
Номер секции:		1
Наименование товара:	Видеокарта ASUS RTX 3090	
Артикул товара:	D6Q09ZEL8I	
Цена товара:	359990	Руб.
Количество товара:	6	шт.
Итоговая сумма:	2159940	Руб.

16/4 /2019

Для продолжения нажмите любую клавишу . . .

Рисунок 3.32 – Выходная форма одного элемента

При вызове сохранения таблицы (пункт 9) выводится контекстное меню подтверждения, и затем ещё одно контекстное меню о выборе куда сохранять базу данных, в текстовый файл или бинарный (типизированный) (рис. 3.33). Затем вводится путь к файлу сохранения с проверками (рис. 3.34). Затем выводится сообщение об успешном сохранении таблицы.

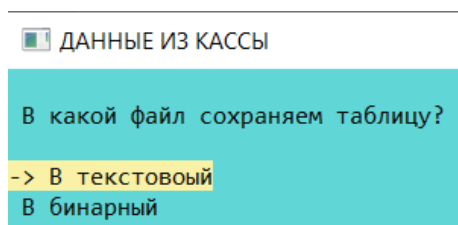


Рисунок 3.33 – Контекстное меню “куда сохранить базу данных”

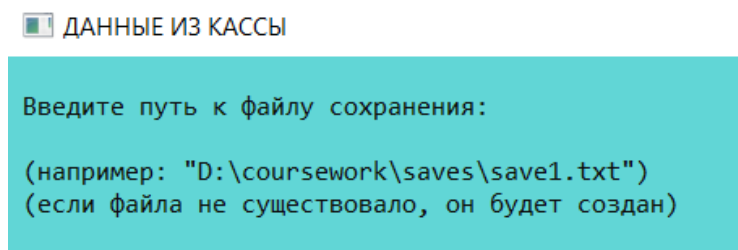


Рисунок 3.34 – Ввод пути к файлу сохранения

При выборе одиннадцатого пункта главного меню вызывается контекстное меню подтверждения определения объёма товарооборота по каждому магазину (рис. 3.35). Затем у пользователя спрашивается выводить ли информацию о товарообороте дополнительно в текстовый файл или не надо (рис. 3.36). Затем если подтвержден вывод в текстовый файл, то осуществляется ввод пути к текстовому файлу и далее выводится информация о товарообороте в каждом магазине (рис. 3.37). На рисунке 3.38 изображена информация, которая вывелась в текстовый файл.

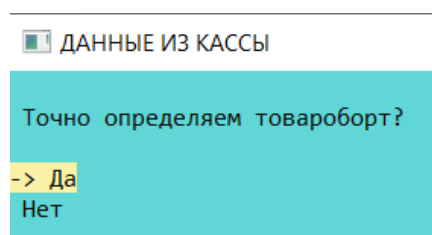


Рисунок 3.35 – Контекстное меню подтверждения определения товарооборота

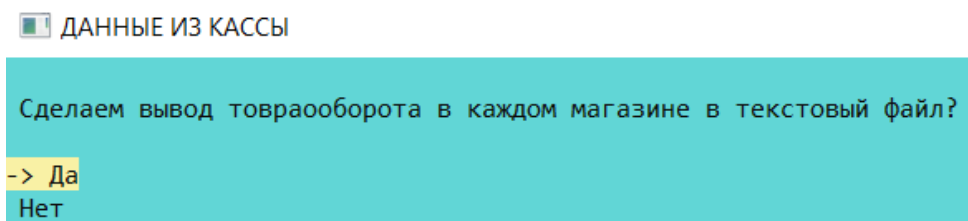


Рисунок 3.36 – Контекстное меню вывода товарооборота в текстовый файл

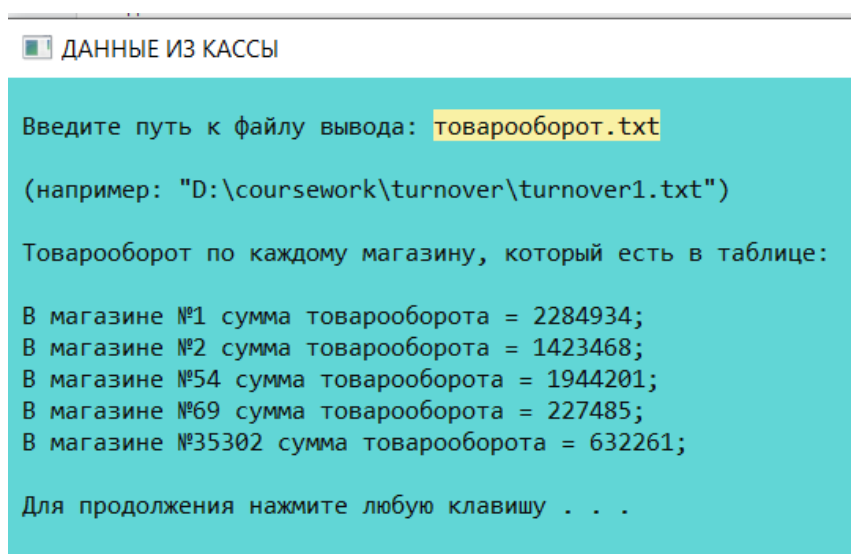


Рисунок 3.37 – Ввод пути к файлу вывода товарооборота и вывод товарооборота

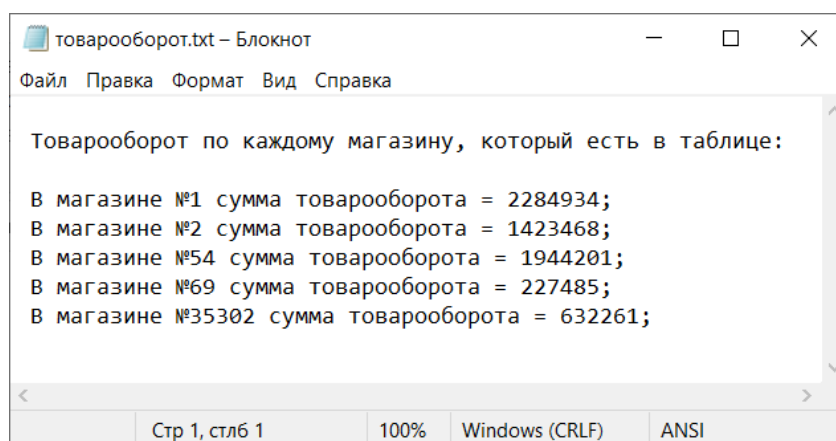


Рисунок 3.38 – Информация в текстовом файле

Для выхода из программы надо воспользоваться пунктом 12 «Выход». При выборе этого пункта вызывается контекстное меню подтверждения выхода из программы (рис. 3.39). При выходе осуществляется очистка памяти и автоматическое сохранение таблицы в бинарный файл автоматического сохранения.

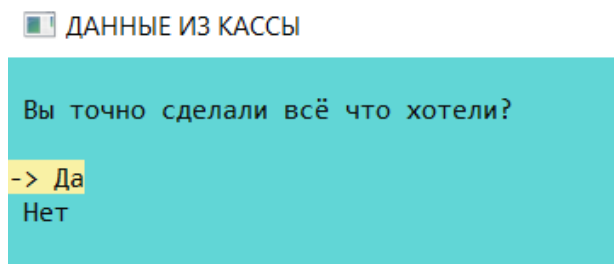


Рисунок 3.39 – Контекстное меню выхода

Результаты проверки программы полностью соответствуют ожиданиям. Все пункты меню работают корректно. Все проверки работают правильно и позволяют пользователю удобно работать с программой.

ВЫВОДЫ

В соответствии с вариантом задания разработана программа, в основу алгоритма которой положена структура данных в виде двунаправленного списка, позволяющая выполнять просмотр данных в двух направлениях. Программа позволяет вносить, хранить и обрабатывать данные о продуктах в магазинах. За счёт удобного интерфейса пользователь может легко и понятно работать с программой.

Просмотр данных был осуществлен при помощи таблицы, с возможностью скроллинга. В любой момент времени в таблице отображается только десять элементов. Данный вид отображения данных позволяет удобно лицезреть содержимое базы данных. Ввод данных в базу данных был осуществлен как непосредственно пользователем, так и с возможностью считать данные из текстового или бинарного файла, сохраненного заранее.

У пользователя есть возможность редактирования любых полей элемента. Был осуществлен поиск по номеру чека. Так же была разработана возможность отсортировать список в двух направлениях по полю “номер магазина”. В случае, если ошибочно были введены все поля элемента или элемент больше не нужен, была осуществлена возможность удалить элемент из таблицы. Так же дополнительно для удобства пользователя была разработана возможность удалить все записи сразу.

Так же для корректной работы программы при вводе любых данных были осуществлены системы отслеживания ошибок, что позволяет пользователю работать с программой менее аккуратно.

Таким образом, была достигнута задача курсового проектирования – углублены, пусть и не значительно, познания языка Си, получен опыт разработки программ с внедрением методологии структурного программирования. Итоги проектирования применять не рекомендуется по основанию существования значительно более продвинутых программных продуктов, надёжность которых была

неоднократно испытана на различных входных данных и вследствие этого не вызывает сомнений в работе.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Do.sevsu.ru – Основы алгоритмизации и технологии программирования [Электронный курс]: <https://do.sevsu.ru/>
2. Методические указания к курсовому проектированию по дисциплине «Алгоритмизация и программирование» для студентов дневной и заочной форм обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика»;
3. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Настоящий стандарт устанавливает требования к выполнению текстовых документов [Электронный ресурс]: <https://docs.cntd.ru/document/1200001260>
4. ГОСТ 19.002-80. Единая система программной документации (ЕСПД). Схемы алгоритмов и программ. Правила выполнения [Электронный ресурс]: <https://www.swrit.ru/doc/espd/19.002-80.pdf>
5. КиберФорум – форум программистов и сисадминов [Электронный ресурс]: <https://www.cyberforum.ru/>
6. Stack Overflow – Система вопросов и ответов о программировании [Электронный ресурс]: <https://ru.stackoverflow.com/>
7. Андрианова, А. А. Алгоритмизация и программирование. Практикум [Электронный ресурс]: учебное пособие / А. А. Андрианова, Л.Н. Исмагилов, Т.М. Мухтарова. - Электрон, дан. - Санкт-Петербург: Лань, 2019. - 240 с. – Режим доступа: <https://e.lanbook.com/book/113933>.
8. Павловская Т.А. С/ С++. Программирование на языке высокого уровня: учеб. для студ. вузов, обуч. по напр. «Информатика и вычислительная техника» / Т. А. Павловская. – СПб.: Питер, 2009. – 461 с.

ПРИЛОЖЕНИЕ А

```

/*-----Strutures.h-----*/

#ifndef TEST_STRUCTURES_H
#define TEST_STRUCTURES_H

#include <stdio.h>
#include <windows.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

#define SetBacklightYellow SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
224);

#define SetBacklightOff SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
176);

#define UI unsigned int
#define UL unsigned long

#define PRODUCT_NAME_MAX_LEN 50 // Максимальная длина наименования товара
#define PRODUCT_CODE_MAX_LEN 20 // Максимальная длина кода товара (артикул)
#define RECEIPT_MAX_LEN 20      // Максимальная длина чека

//=====
//                                  СТРУКТУРЫ
//=====

typedef struct when {
    UI day;
    UI month;
    UI year;
}When;

typedef struct info {
    UI num_shop;           // Номер магазина
    UI num_section;       // Номер секции
    char num_receipt[RECEIPT_MAX_LEN + 1]; // Номер чека

```

```

    char product_name[PRODUCT_NAME_MAX_LEN + 1]; // Наименование товара
    char product_code[PRODUCT_CODE_MAX_LEN + 1]; // Артикул товара
    UL product_price; // Цена товара
    UI product_quantity; // Количество товара
    When date; // Дата продажи
}Info;

typedef struct element {
    Info data; //данные
    struct element *next;
    struct element *previous;
}Element;

#endif //TEST_STRUCTURES_H

/*-----main.c-----*/

// Информация о продаже товаров подготовлена по следующему макету: номер магазина;
// номер секции; номер чека; наименование товара; артикул товара;
// цена товара; количество товара; дата продажи. Составить программу определения
// объема товарооборота по каждому магазину, т.е. общую сумму, вырученную от
// продажи всех товаров в данном магазине (данные содержат записи по нескольким
// магазинам). Разработать форму выходного документа.

#include "headers/structures.h"
#include "headers/specialFunctions.h"
#include "headers/workWithTable.h"
#include "headers/setData.h"
#include "headers/saveAndDownloadTable.h"

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

int main() {
    // Поддержка русского языка
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

```

```

// Ставим фон консоли светло-голубым, текст черный
SetConsoleTitle("ДАННЫЕ ИЗ КАССЫ");
system("color B0");
system ("mode con cols=161 lines=30");

Element *head = NULL;

const char *firstMenu[] = {
    " Создание таблицы вручную",
    " Загрузка таблицы из текстового файла",
    " Загрузка таблицы из бинарного файла",
    " Загрузка автосохранения",
    " Продолжить без таблицы",
    " Выход"
};

switch (menu(firstMenu, 6, " Выберите способ первичного создания таблицы:") + 1){
    case 1: {
        do {
            addNewElement(&head, setInfo(head));
        } while (yesOrNo(" Продолжаем добавление элементов?") == 0);
        break;
    }

    case 2: {
        readTable(&head);
        break;
    }

    case 3: {
        readTableBIN(&head);
        break;
    }

    case 4: {
        autoDownloudTableBin(&head);
        break;
    }
}

```



```

    case 5: {
        break;
    }

    case 6: {
        return 0;
    }
}

const char *menuItems[] = {
    " Создание таблицы",
    " Просмотр таблицы",
    " Добавление новой записи в таблицу",
    " Удаление записи",
    " Удаление таблицы",
    " Корректировка записи в таблице",
    " Сортировка таблицы (по №магазина)",
    " Поиск записи в таблице",
    " Сохранение таблицы в файле",
    " Чтение данных из файла",
    " Определить объём товарооборота по каждому магазину",
    " Выход"
};

const int NUM_MENU_ITEMS = 12; // Кол-во пунктов меню

while (1) {
    switch (menu(menuItems, NUM_MENU_ITEMS, " ГЛАВНОЕ МЕНЮ:") + 1) {

        // проверки: зайти с существующей таблицей,
        case 1: {
            if (yesOrNo(" Точно создаём таблицу?") == 1) break;

            // Если таблица уже есть
            if (head) {
                const char *optionsForCreatingTable[] = {

```

```

        " Удалить текущую таблицу без сохранения и создавать новую",
        " Сохранить текущую таблицу, затем создать новую",
        " Добавить элементы в существующую таблицу",
        " Хочу выйти, не хочу создавать новую таблицу"
    };

    UI exit = 0;

    switch (menu(optionsForCreatingTable, 4, " У вас уже есть таблица!
Выберите действие:")) {
        case 0: {
            deleteTable(&head);
            break;
        }
        case 1: {
            saveTable(head);
            deleteTable(&head);
            break;
        }
        case 2: {
            break;
        }
        case 3: {
            exit = 1;
            break;
        }
    }

    if (exit == 1) break; //выход из "case 1"
}

// Ввод элементов
do {
    addNewElement(&head, setInfo(head));
} while (yesOrNo(" Продолжаем добавление элементов?") == 0);
break;
}

case 2: {
    if (yesOrNo(" Точно показать таблицу?") == 1) break;

```

```

        showTable(head);
        break;
    }

    case 3: {
        if (yesOrNo(" Точно добавляем новую запись?") == 1) break;
        addNewElement(&head, setInfo(head));
        break;
    }

    case 4: {
        if (yesOrNo(" Точно удаляем запись?") == 1) break;
        deleteElement(&head);
        break;
    }

    case 5: {
        if (yesOrNo(" Точно удаляем таблицу?") == 1) break;
        deleteTable(&head);
        break;
    }

    case 6: {
        if (yesOrNo(" Точно корректируем запись в таблице?") == 1) break;
        correctRecord(selectElement(head));
        break;
    }

    case 7: {
        if (yesOrNo(" Точно сортируем записи?") == 1) break;
        if (head != NULL) {
            const char *ascendingOrDescending[] = {

```

```

        " По возрастанию",
        " По убыванию"
    };

    if (menu(ascendingOrDescending, 2, " Как сортируем?") == 0) {
        sortTableAscending(&head);
    }
    else {
        sortTableDescending(&head);
    }

    system("cls");
    printf("\n\n Сортировка успешно завершена!\n\n ");
    system("pause");
}
else {
    system("cls");
    printf("\n\n \a Записей нет, сортировать нечего! \n\n");
    system("pause");
}
break;
}

case 8: {
    if (yesOrNo(" Точно ищем запись в таблице?") == 1) break;
    if (head != NULL) {
        showOneElement(searchElement(head));
    }
    else {
        system("cls");
        printf("\n\n \a А таблицы то нет! Где искать собрались?
\n\n");

        system("pause");
    }
    break;
}

case 9: {

```

```

    if (yesOrNo(" Точно сохраняем таблицу в файл?") == 1) break;
    const char *where[] = {
        " В текстовый",
        " В бинарный",
    };
    if (menu(where, 2, " В какой файл сохраняем таблицу?") == 0) {
        saveTable(head);
    }
    else {
        saveTableBIN(head);
    }
    break;
}

case 10: {
    if (yesOrNo(" Точно читаем таблицу из файла?") == 1) break;
    // Если таблица уже существует
    if (head != NULL) {
        const char *tableAlreadyExists[] = {
            " Удалить текущую таблицу без сохранения и считать новую из
            текстового файла",
            " Сохранить текущую таблицу, затем считать новую из файла",
            " Хочу выйти, передумал считывать таблицу из файла"
        };
        UI exit = 0;
        switch (menu(tableAlreadyExists, 3, " У вас уже есть таблица!
        Выберите действие:")) {
            case 0: {
                deleteTable(&head);
                break;
            }
            case 1: {
                saveTable(head);
                deleteTable(&head);
                break;
            }
        }
    }
}

```

```

        case 2: {
            exit = 1;
            break;
        }
    }
    if (exit == 1) break; //выход из "case 9"
}

// Если же таблицы нет, то смело её читаем
const char *where[] = {
    " Из текстового",
    " Из бинарного",
};

if (menu(where, 2, " Из какого файла читаем таблицу?") == 0) {
    readTable(&head);
}
else{
    readTableBIN(&head);
}
break;
}

case 11: {
    if (yesOrNo(" Точно определяем товароборт?") == 1) break;
    determineTurnoverForEachStore(head);
    break;
}

case 12: {
    if (yesOrNo(" Вы точно сделали всё что хотели?") == 1) break;

    //Автоматическое сохранение таблицы в формате .bin (типизированный файл)
    autoSaveTableBin(head);

    deleteTable(&head);
}

```

```

        return 0;

    }

} //switch

} //while(1)

}

/*-----specialFunctions.h-----*/

#ifndef TEST_SPECIALFUNCTIONS_H
#define TEST_SPECIALFUNCTIONS_H

#include "structures.h"

//=====
//
//                СПЕЦИАЛЬНЫЕ ФУНКЦИИ
//
//                ПРОТОТИПЫ
//=====

void gotoxy(int x, int y); // Установить координаты курсора
void setCursor(int i); // Показать/скрыть курсор
void showMenuItems(const char *menuItems[], const int NUM_MENU_ITEMS, const char *title); // Вывод пунктов меню
int menu(const char *menuItems[], int numPunct, const char *title);
// Меню интерактивное
// Параметры: список пунктов, кол-во пунктов, заглавие меню

int yesOrNo(const char *title); // Вы уверены в своём выборе?

#endif //TEST_SPECIALFUNCTIONS_H

/*-----specialFunctions.c-----*/

#include "../headers/specialFunctions.h"

//=====
//
//                СПЕЦИАЛЬНЫЕ ФУНКЦИИ
//
//=====

```

```

// Установить координаты курсора
void gotoxy(int x, int y) {
    COORD coord = {x, y}; // Координаты
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

// Показать/скрыть курсор
void setCursor(int i) {
    CONSOLE_CURSOR_INFO tmp;

    tmp.bVisible = i; // Видимость=1/невидимость=0 курсора
    tmp.dwSize = 1; // Размер курсора
    SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &tmp);
}

// Вывод пунктов меню
void showMenuItems(const char *menuItems[], const int NUM_MENU_ITEMS, const char
 *title) {
    system("cls");

    int i;

    // Напечатать заглавие
    gotoxy(0, 1);
    printf("%s", title);

    for (i=0; i != NUM_MENU_ITEMS; i++) {
        gotoxy(0, i+3);
        printf("%s\n", menuItems[i]);
    }
}

// Меню интерактивное
// Параметры: список пунктов, кол-во пунктов, заглавие меню
int menu(const char *menuItems[], int numPunct, const char *title) {
    const int NUM_MENU_ITEMS = numPunct; // Кол-во пунктов меню
    int activeMenuItem = 0; // Активный элемент
    int ch = 0;

```



```

setCursor(0); // удалить курсор

while (ch != 13) {
    // Если Esc, то вернуть номер последнего пункта меню
    if (ch == 27) {
        setCursor(1);
        system("cls");
        return NUM_MENU_ITEMS-1;
    }

    showMenuItems(menuItems, NUM_MENU_ITEMS, title); // Показать пункты меню
    gotoxy(0, activeMenuItem+3);

    // Выделить текущий пункт ярко жёлтым цветом
    SetBacklightYellow;
    printf("->%s", menuItems[activeMenuItem]);

    ch = getch();
    // так как у стрелок первый код 224 и только второй 72 или 80, то делаем
2 getch
    if (ch == 224)
        ch = getch();

    switch (ch) {
        case 81: activeMenuItem++; break; // page down
        case 80: activeMenuItem++; break; // стрелка вниз
        case 73: activeMenuItem--; break; // page up
        case 72: activeMenuItem--; break; // стрелка вверх
    }

    // Переход по пунктам меню с 1 на последний и на оборот
    if (activeMenuItem < 0)
        activeMenuItem = NUM_MENU_ITEMS-1;
    if (activeMenuItem > NUM_MENU_ITEMS-1)
        activeMenuItem = 0;

    SetBacklightOff;
}

```

```

        setCursor(1);
        system("cls");
        return activeMenuItem;
    }

// Вы уверены в своём выборе?
int yesOrNo(const char *title) {
    const char *yesORno [] = {
        " Да",
        " Нет"
    };
    int kol_selections = 2;
    return menu(yesORno, kol_selections, title);
}

/*-----setData.h-----*/

#ifndef TEST_SETDATA_H
#define TEST_SETDATA_H

#include "structures.h"
#include "specialFunctions.h"

//=====
//
//          ФУНКЦИИ ВВОДА ДАННЫХ С ПРОВЕРКАМИ
//
//          ПРОТОТИПЫ
//=====

UI setNumShop(); // Ввод номера магазина
UI setNumSection(); // Ввод номера секции
int checkingForIdenticalReceipts(Element *head, char buf[25]); // Проверка наличия
одинаковых чеков
void setNumReceipt(char *buf, Element *head); // Ввод номера чека
void setProductName(char *buf); // Ввод наименования продукта
void setProductCode(char *buf); // Ввод артикула продукта
UL setProductPrice(); // Ввод цены товара
UI setProductQuantity(); // Ввод количества продуктов

```

```

When setDate(); // Ввод даты: день/месяц/год
#endif //TEST_SETDATA_H
/*-----setData.c-----*/

#include "../headers/setData.h"

//=====
//
//          ФУНКЦИИ ВВОДА ДАННЫХ С ПРОВЕРКАМИ
//=====

// Ввод номера магазина
UI setNumShop() {
    UI numShop;
    char buf[255];
    int i, kol;
    while(1){
        i = 0;
        kol = 0;
        SetBacklightOff;
        system("cls");
        gotoxy(0,0);
        printf("\n Введите номер магазина:\n\n");
        printf(" (данные: число от 1 до 50к)");
        gotoxy(25 ,1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);
        while(1) {
            while(buf[i]) {
                // если число увеличиваем счетчик
                if ((buf[i] == '1') || (buf[i] == '2') || (buf[i] == '3') || (buf[i] == '4') ||
(buf[i] == '5') || (buf[i] == '6') || (buf[i] == '7') || (buf[i] == '8') || (buf[i]
== '9') || (buf[i] == '0')) {
                    kol++;
                }
                i++;
            }
            if (strlen(buf) == kol) {

```



```

        // если число увеличиваем счетчик
        if ((buf[i] == '1') || (buf[i] == '2') || (buf[i] == '3') || (buf[i] == '4') ||
            (buf[i] == '5') || (buf[i] == '6') || (buf[i] == '7') || (buf[i] == '8') || (buf[i]
            == '9') || (buf[i] == '0')) {

            kol++;

        }

        i++;

    }

    if (strlen(buf) == kol) {
        numSection = atoi(buf);
        if ((numSection > 0) && (numSection < 1001)) {
            SetBacklightOff;
            return numSection;
        }
        else {
            printf("\a");
            break;
        }
    }

    else {
        printf("\a");
        break;
    }

}

}
}

```

// Проверка наличия одинаковых чеков

```

int checkingForIdenticalReceipts(Element *head, char buf[25]){
    system("cls");
    char num_receipt[22];
    int i;
    int mismatch;
    int kol;
    while (head != NULL) {
        kol = 0;
        memset(num_receipt, 0, sizeof(num_receipt));
        strcpy(num_receipt, head->data.num_receipt);
    }
}

```

```

for (i=0; i<20; i++) {
    if (num_receipt[i] == buf[i]) {
        mismatch = 1;
    }
    else {
        mismatch = 0;
    }
    kol += mismatch;
}
if (kol == 20){
    printf("\n\a\n ОШИБКА! Такой чек уже есть!\n\n ");
    system("pause");
    return 1;
}
head = head->next;
}
return 0;
}

```

//Ввод номер чека

```

void setNumReceipt(char *buf, Element *head) {
    int i, kol;
    while(1) {
        i = 0;
        kol = 0;
        SetBacklightOff;
        system("cls");
        gotoxy(0,0);
        printf("\n Введите номер чека:\n\n");
        printf(" (данные: 20 чисел)");
        gotoxy(21 ,1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);
        SetBacklightOff;
        while(1) {
            while(buf[i]) {

```

```

        // если число увеличиваем счетчик
if ((buf[i] == '1') || (buf[i] == '2') || (buf[i] == '3') || (buf[i] == '4') ||
    (buf[i] == '5') || (buf[i] == '6') || (buf[i] == '7') || (buf[i] == '8') || (buf[i]
    == '9') || (buf[i] == '0')) {

            kol++;

        }

        i++;

    }

    if (strlen(buf) == kol) {
        if (strlen(buf) == 20) {
            // Если такого чека ещё нет в таблице
            if (checkingForIdenticalReceipts(head, buf) == 0) {
                return;
            }
            else {
                break;
            }
        }
        else {
            printf("\a");
            break;
        }
    }
    else {
        printf("\a");
        break;
    }
}

}

//Ввод наименования продукта
void setProductName(char *buf) {
    while(1) {
        memset(buf, 0, sizeof(buf));
        SetBacklightOff;
        system("cls");
        gotoxy(0,0);
    }
}

```

```

printf("\n Введите наименование товара:\n\n");
printf(" (данные: любые до 50 символов)");
gotoxy(30 ,1);
SetBacklightYellow;
fflush(stdin);
gets(buf);
if ((strlen(buf) < 51) && (strlen(buf) > 0)) {
    SetBacklightOff;
    break;
}
else {
    printf("\a");
}
}
}

```

//Ввод артикула продукта

```

void setProductCode(char *buf) {
    while(1) {
        memset(buf, 0, sizeof(buf));
        SetBacklightOff;
        system("cls");
        gotoxy(0,0);
        printf("\n Введите артикул товара:\n\n");
        printf(" (данные: любые до 20 символов)");
        gotoxy(25 ,1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);
        if ((strlen(buf) < 21) && (strlen(buf) > 0)) {
            SetBacklightOff;
            break;
        }
        else {
            printf("\a");
        }
    }
}

```



```

}

// Ввод цена товара
UL setProductPrice() {
    UL numSection;
    char buf[255];
    int i, kol;
    while(1) {
        i = 0;
        kol = 0;
        SetBacklightOff;
        system("cls");
        gotoxy(0,0);
        printf("\n Введите цену товара:\n\n");
        printf(" (данные: целое число от 1 до 1млрд.)");
        gotoxy(22 ,1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);
        while(1) {
            while(buf[i]) {
                // если число увеличиваем счетчик
                if ((buf[i] == '1') || (buf[i] == '2') || (buf[i] == '3') || (buf[i] == '4') ||
                    (buf[i] == '5') || (buf[i] == '6') || (buf[i] == '7') || (buf[i] == '8') ||
                    (buf[i] == '9') || (buf[i] == '0')) {
                    kol++;
                }
                i++;
            }
            if (strlen(buf) == kol) {
                numSection = atoi(buf);
                if ((numSection > 0) && (numSection < 1000000001)) {
                    SetBacklightOff;
                    return numSection;
                }
            }
            else {
                printf("\a");
                break;
            }
        }
    }
}

```

```

        }
    }
    else {
        printf("\a");
        break;
    }
}

}

}

// Ввод количества продуктов
UI setProductQuantity() {
    UI productQuantity;
    char buf[255];
    int i, kol;
    while(1){
        i = 0;
        kol = 0;
        SetBacklightOff;
        system("cls");
        gotoxy(0,0);
        printf("\n Введите количество товаров:\n\n");
        printf(" (данные: число от 1 до 50к)");
        gotoxy(29 ,1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);
        while(1) {
            while(buf[i]) {
                // если число увеличиваем счетчик
                if ((buf[i] == '1') || (buf[i] == '2') || (buf[i] == '3') || (buf[i] == '4') ||
                    (buf[i] == '5') || (buf[i] == '6') || (buf[i] == '7') || (buf[i] == '8') ||
                    (buf[i] == '9') || (buf[i] == '0')) {
                    kol++;
                }
                i++;
            }
            if (strlen(buf) == kol) {

```



```

        SetBacklightYellow;
        printf("%d", i);
        ch = getch();
        // так как у стрелок первый код 224 и только второй 72 или 80, то делаем
2 getch
        if (ch == 224) ch = getch();
        switch (ch) {
            case 81: i--; break; // page down
            case 80: i--; break; // стрелка вниз
            case 73: i++; break; // page up
            case 72: i++; break; // стрелка вверх
        }

        // Переход по пунктам меню с 1 на последний и на оборот
        if (i < 1) i = 31;
        if (i > 31) i = 1;
    }
    date.day = i;

    //-----МЕСЯЦ
    ch = 0;
    i = 1;
    char *month[] = {"январь", "Февраль", "Март", "Апрель", "Май", "Июнь", "Июль",
"Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"};
    while(ch != 13) {
        SetBacklightOff;
        system("cls");
        gotoxy(0,0);
        printf("\n Месяц:\n\n");
        printf(" (используйте стрелки вверх/вниз и Enter)");
        gotoxy(8 ,1);
        SetBacklightYellow;
        printf("%s", month[i-1]);
        ch = getch();
        // так как у стрелок первый код 224 и только второй 72 или 80, то делаем 2 getch
        if (ch == 224) ch = getch();
        switch (ch) {
            case 81: i--; break; // page down

```

```

        case 80: i--; break; // стрелка вниз
        case 73: i++; break; // page up
        case 72: i++; break; // стрелка вверх
    }

    // Переход по пунктам меню с 1 на последний и на оборот
    if (i < 1) i = 12;
    if (i > 12) i = 1;
}
date.month = i;

//-----ГОД
ch = 0;
i = 2021;
while(ch != 13) {
    SetBacklightOff;
    system("cls");
    gotoxy(0,0);
    printf("\n Год:\n\n");
    printf(" (используйте стрелки вверх/вниз и Enter)");
    gotoxy(6 ,1);
    SetBacklightYellow;
    printf("%d", i);
    ch = getch();
}
// так как у стрелок первый код 224 и только второй 72 или 80, то делаем 2 getch
if (ch == 224) ch = getch();
switch (ch) {
    case 81: i--; break; // page down
    case 80: i--; break; // стрелка вниз
    case 73: i++; break; // page up
    case 72: i++; break; // стрелка вверх
}

// Переход по пунктам меню с 1 на последний и на оборот
if (i < 1990) i = 2023;
if (i > 2023) i = 1990;
}

```

```

date.year = i;

// Проверка на совпадение числа и месяца (и високостного года)
i = 0;
switch(date.month) {
    case 4:
    case 6:
    case 9:
    case 11: {
        if (date.day == 31) {
            printf("\a");
            i = 1;
        }
        break;
    }
    case 2: {
        // Если год високостный, то
        if ((date.year % 4) == 0) {
            if (date.day > 29) {
                printf("\a");
                i = 1;
            }
        }
        // Если год не високостный, то
        if ((date.year % 4) != 0) {
            if (date.day > 28) {
                printf("\a");
                i = 1;
            }
        }
        break;
    }
}
if (i == 0)
    break;

} //while(1)

```

```

        setCursor(1);
        SetBacklightOff;
        return date;
    }

/*-----saveAndDownload.h-----*/

#ifndef TEST_SAVEANDDOWNLOADTABLE_H
#define TEST_SAVEANDDOWNLOADTABLE_H

#include "structures.h"
#include "specialFunctions.h"
#include "setData.h"
#include "workWithTable.h"

//=====
//                      ФУНКЦИИ СОХРАНЕНИЯ И ЗАГРУЗКИ ТАБЛИЦЫ
//                      ПРОТОТИПЫ
//=====

void readTable(Element **head); // Считать таблицу из файла
void saveTable(Element *head); // Сохранить таблицу в файл

void readTableBIN(Element **head); // Считать таблицу из бинарного файла
void saveTableBIN(Element *head); // Сохранить таблицу в бинарный файл

void autoDownloadTableBin(Element **head); //Автоматическая загрузка последнего
сохранения в формате .bin (типизированный файл)
void autoSaveTableBin(Element *head); //Автоматическое сохранение таблицы в формате
.bin (типизированный файл)

#endif //TEST_SAVEANDDOWNLOADTABLE_H

/*-----saveAndDownload.c-----*/

#include "../headers/saveAndDownloadTable.h"

//=====

```

```
//
//          ФУНКЦИИ СОХРАНЕНИЯ И ЗАГРУЗКИ ТАБЛИЦЫ
//=====

// Сохранить таблицу в файл
void saveTable(Element *head) {
    system("cls");

    if (head == NULL) {
        printf("\n\n\a СОХРАНЯТЬ НЕЧЕГО! \n\n\n ");
        system("pause");
        return;
    }

    FILE *fdownload;
    int exitFromWhile = 0;
    int lenBuf = 0;

    //Ввод пути к файлу где будем сохранять
    char buf[255];
    do {
        memset(buf, 0, sizeof(buf));
        system("cls");
        SetBacklightOff;
        gotoxy(0, 0);
        printf("\n Введите путь к файлу сохранения:\n\n");
        printf(" (например: \"D:\\\\coursework\\\\saves\\\\save1.txt\")\n");
        printf(" (если файла не существовало, он будет создан)");
        gotoxy(34, 1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);

        // проверить текстовый файл или нет, и если нет то файл идет на }{*#
        lenBuf = strlen(buf);
        if (lenBuf > 4) {
            if ((buf[lenBuf-4] == '.') && (buf[lenBuf-3] == 't') && (buf[lenBuf-2] ==
            'x') && (buf[lenBuf-1] == 't')) {
                SetBacklightOff;
            }
        }
    } while (lenBuf < 4 || !((buf[lenBuf-4] == '.') && (buf[lenBuf-3] == 't') && (buf[lenBuf-2] ==
    'x') && (buf[lenBuf-1] == 't')));
}
```



```

    }
    else {
        SetBacklightOff;
        printf("\a");
        continue;
    }
}
else {
    SetBacklightOff;
    printf("\a");
    continue;
}

//Если файл уже существует перезаписать его?
if ((fdownload = fopen(buf, "r")) != NULL) {
    const char *overwrite[] = {
        " Да, давай перезапишем",
        " Неа, повторим попытку ввода",
        " Я хочу выйти, передумал сохранять таблицу"
    };
    fclose(fdownload);
    switch(menu(overwrite, 3, " Указанный файл уже существует. Перезапишем его?")) {
        case 0: {
            //выходим из ввода
            fdownload = fopen(buf, "w");
            exitFromWhile = 1;
            break;
        }
        case 1: {
            exitFromWhile = 2;
            break;
        }
        case 2: {
            return;
        }
    }
    if (exitFromWhile == 1) break;
}

```

```

        if (exitFromWhile == 2) continue;
    }

    if ((fdownload = fopen(buf, "w")) != NULL) {
        exitFromWhile = 1;
    }
    else {
        printf("\a");
        const char *error[] = {
            " Повторить попытку ввода",
            " Выйти"
        };
        if (menu(error, 2, " Не удалось получить доступ к файлу!") == 1) {
            return;
        }
    }
} while (exitFromWhile != 1);

fseek(fdownload, 0, SEEK_SET);
while (head != NULL) {
    memset(buf, 0, sizeof(buf));
    itoa(head->data.num_shop, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    itoa(head->data.num_section, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    strcpy(buf, head->data.num_receipt);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    strcpy(buf, head->data.product_name);

```

```

    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    strcpy(buf, head->data.product_code);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    itoa(head->data.product_price, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    itoa(head->data.product_quantity, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    itoa(head->data.date.day, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    itoa(head->data.date.month, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    itoa(head->data.date.year, buf, 10);
    fputs(buf, fdownload);
    if (head->next != NULL){
        fputc('\n', fdownload);
    }

    head = head->next;
}

```

```

fclose(fdownload);

system("cls");
printf("\n\n Таблица успешно сохранена! \n\n\n ");
system("pause");
return;
}

// Считать таблицу из файла
void readTable(Element **head) {
    FILE *fread;

    //Ввод пути к файлу
    char buf[255];
    while(1) {
        memset(buf, 0, sizeof(buf));
        system("cls");
        SetBacklightOff;
        gotoxy(0,0);
        printf("\n Введите путь к файлу загрузки:\n\n");
        printf(" (например: \"D:\\\\coursework\\\\saves\\\\save1.txt\\\")");
        gotoxy(32 ,1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);
        SetBacklightOff;

        //Если файл не открылся
        if ((fread = fopen(buf, "r")) == NULL) {
            printf("\a");
            const char *error[] = {
                " Повторить попытку ввода",
                " Выйти"
            };
            if (menu(error, 2, " Файл не найден!") == 1) {

```

```

        break;
    }
}
else {
    Info info;

    fseek(fread, 0, SEEK_SET);
    while(!feof(fread)) {
        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);
        info.num_shop = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);
        info.num_section = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 21, fread); // считать 20 символов (\n не считается)
        strcpy(info.num_receipt, buf);
        fgets(buf, 21, fread); //символ перехода новой строки игнорировать

        memset(buf, 0, sizeof(buf));
        memset(info.product_name, 0, sizeof(info.product_name));
        fgets(buf, 55, fread);
        strncpy(info.product_name, buf, (strlen(buf)-1));

        memset(buf, 0, sizeof(buf));
        memset(info.product_code, 0, sizeof(info.product_code));
        fgets(buf, 25, fread);
        strncpy(info.product_code, buf, (strlen(buf)-1));

        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);
        info.product_price = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);

```

```

        info.product_quantity = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 10, fread);
        info.date.day = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 10, fread);
        info.date.month = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 10, fread);
        info.date.year = atoi(buf);

        addNewElement(head, info);
    }

    fclose(fread);
    system("cls");
    printf("\n\n ИНФОРМАЦИЯ БЫЛА УСПЕШНО СЧИТАНА ИЗ ФАЙЛА! \n\n ");
    system("pause");
    break;
}

}

}

//=====

// Считать таблицу из бинарного файла
void readTableBIN(Element **head) {
    FILE *fread;

    //Ввод пути к файлу
    char buf[255];
    while(1) {
        memset(buf, 0, sizeof(buf));
        system("cls");

```

```

SetBacklightOff;

gotoxy(0,0);

printf("\n Введите путь к файлу загрузки:\n\n");
printf(" (например: \"D:\\\\coursework\\\\saves\\\\save1.bin\\\")");
gotoxy(32 ,1);

SetBacklightYellow;

fflush(stdin);

gets(buf);

SetBacklightOff;


//Если файл не открылся
if ((fread = fopen(buf, "rb")) == NULL) {
    printf("\a");
    const char *error[] = {
        " Повторить попытку ввода",
        " Выйти"
    };
    if (menu(error, 2, " Файл не найден!") == 1) {
        break;
    }
}
else {
    Info info;

    fseek(fread, 0, SEEK_SET);
    while(!feof(fread)) {
        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);
        info.num_shop = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);
        info.num_section = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 21, fread); // считать 20 символов (\n не считается)
        strcpy(info.num_receipt, buf);
    }
}

```

```

fgets(buf, 21, fread);//символ перехода новой строки игнорировать

memset(buf, 0, sizeof(buf));
memset(info.product_name, 0, sizeof(info.product_name));
fgets(buf, 55, fread);
strncpy(info.product_name, buf, (strlen(buf)-1));

memset(buf, 0, sizeof(buf));
memset(info.product_code, 0, sizeof(info.product_code));
fgets(buf, 25, fread);
strncpy(info.product_code, buf, (strlen(buf)-1));

memset(buf, 0, sizeof(buf));
fgets(buf, 15, fread);
info.product_price = atoi(buf);

memset(buf, 0, sizeof(buf));
fgets(buf, 15, fread);
info.product_quantity = atoi(buf);

memset(buf, 0, sizeof(buf));
fgets(buf, 10, fread);
info.date.day = atoi(buf);

memset(buf, 0, sizeof(buf));
fgets(buf, 10, fread);
info.date.month = atoi(buf);

memset(buf, 0, sizeof(buf));
fgets(buf, 10, fread);
info.date.year = atoi(buf);

addNewElement(head, info);
}

fclose(fread);
system("cls");

```



```

        printf("\n\n ИНФОРМАЦИЯ БЫЛА УСПЕШНО СЧИТАНА ИЗ ФАЙЛА! \n\n ");
        system("pause");
        break;
    }
}

// Сохранить таблицу в бинарный файл
void saveTableBIN(Element *head) {
    system("cls");

    if (head == NULL) {
        printf("\n\n а СОХРАНЯТЬ НЕЧЕГО! \n\n\n ");
        system("pause");
        return;
    }

    FILE *fdownload;
    int exitFromWhile = 0;
    int lenBuf = 0;

    //Ввод пути к файлу где будем сохранять
    char buf[255];
    do {
        memset(buf, 0, sizeof(buf));
        system("cls");
        SetBacklightOff;
        gotoxy(0, 0);
        printf("\n Введите путь к файлу сохранения:\n\n");
        printf(" (например: \"D:\\\\coursework\\\\saves\\\\save1.bin\\\")\n");
        printf(" (если файла не существовало, он будет создан)\n");
        gotoxy(34, 1);
        SetBacklightYellow;
        fflush(stdin);
        gets(buf);

        // проверить текстовый файл или нет, и если нет то файл идет на {*#

```

```

    lenBuf = strlen(buf);

    if (lenBuf > 4) {
        if ((buf[lenBuf-4] == '.') && (buf[lenBuf-3] == 'b') && (buf[lenBuf-2] == 'i')
        && (buf[lenBuf-1] == 'n')) {
            SetBacklightOff;
        }
        else {
            SetBacklightOff;
            printf("\a");
            continue;
        }
    }
    else {
        SetBacklightOff;
        printf("\a");
        continue;
    }

    //Если файл уже существует перезаписать его?
    if ((fdownload = fopen(buf, "rb")) != NULL) {
        const char *overwrite[] = {
            " Да, давай перезапишем",
            " Неа, повторим попытку ввода",
            " Я хочу выйти, передумал сохранять таблицу"
        };
        fclose(fdownload);
        switch(menu(overwrite, 3, " Указанный файл уже существует. Перезапишем его?")) {
            case 0: {
                //выходим из ввода
                fdownload = fopen(buf, "wb");
                exitFromWhile = 1;
                break;
            }
            case 1: {
                exitFromWhile = 2;
                break;
            }
            case 2: {

```

```

        return;
    }

    }

    if (exitFromWhile == 1) break;
    if (exitFromWhile == 2) continue;
}

if ((fdownload = fopen(buf, "wb")) != NULL) {
    exitFromWhile = 1;
}
else {
    printf("\a");
    const char *error[] = {
        " Повторить попытку ввода",
        " Выйти"
    };
    if (menu(error, 2, " Не удалось получить доступ к файлу!") == 1) {
        return;
    }
}

} while (exitFromWhile != 1);

fseek(fdownload, 0, SEEK_SET);
while (head != NULL) {
    memset(buf, 0, sizeof(buf));
    itoa(head->data.num_shop, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    itoa(head->data.num_section, buf, 10);
    fputs(buf, fdownload);
    fputc('\n', fdownload);

    memset(buf, 0, sizeof(buf));
    strcpy(buf, head->data.num_receipt);
    fputs(buf, fdownload);
}

```

```

fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
strcpy(buf, head->data.product_name);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
strcpy(buf, head->data.product_code);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.product_price, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.product_quantity, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.date.day, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.date.month, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.date.year, buf, 10);
fputs(buf, fdownload);
if (head->next != NULL){
    fputc('\n', fdownload);
}

```

```

    }

    head = head->next;
}

fclose(fdownload);

system("cls");
printf("\n\n Таблица успешно сохранена! \n\n\n ");
system("pause");
return;
}

//=====

//Автоматическое сохранение таблицы в формате .bin (типизированный файл)
void autoSaveTableBin(Element *head) {
    FILE *fdownload;
    int exitFromWhile = 0;
    int lenBuf = 0;

    //Путь к файлу и его открытие
    char buf[255] = "autoSave.bin";
    if ((fdownload = fopen(buf, "wb")) == NULL) {
        return;
    }

    fseek(fdownload, 0, SEEK_SET);
    while (head != NULL) {
        memset(buf, 0, sizeof(buf));
        itoa(head->data.num_shop, buf, 10);
        fputs(buf, fdownload);
        fputc('\n', fdownload);

        memset(buf, 0, sizeof(buf));
        itoa(head->data.num_section, buf, 10);
        fputs(buf, fdownload);
    }
}

```

```
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
strcpy(buf, head->data.num_receipt);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
strcpy(buf, head->data.product_name);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
strcpy(buf, head->data.product_code);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.product_price, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.product_quantity, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.date.day, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);

memset(buf, 0, sizeof(buf));
itoa(head->data.date.month, buf, 10);
fputs(buf, fdownload);
fputc('\n', fdownload);
```

```

        memset(buf, 0, sizeof(buf));
        itoa(head->data.date.year, buf, 10);
        fputs(buf, fdownload);
        if (head->next != NULL) {
            fputc('\n', fdownload);
        }

        head = head->next;
    }

    fclose(fdownload);
    return;
}

// Автоматическая загрузка последнего сохранения в формате .bin (типизированный файл)
void autoDownloadTableBin(Element **head) {
    FILE *fread;

    //Путь к файлу и его открытие
    char buf[255] = "autoSave.bin";
    if ((fread = fopen(buf, "rb")) == NULL) {
        return;
    }

    Info info;

    fseek(fread, 0, SEEK_SET);
    while(!feof(fread)) {
        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);
        info.num_shop = atoi(buf);

        memset(buf, 0, sizeof(buf));
        fgets(buf, 15, fread);
        info.num_section = atoi(buf);
    }
}

```

```
memset(buf, 0, sizeof(buf));
fgets(buf, 21, fread); // считать 20 символов (\n не считается)
strcpy(info.num_receipt, buf);
fgets(buf, 21, fread); //символ перехода новой строки игнорировать
```

```
memset(buf, 0, sizeof(buf));
memset(info.product_name, 0, sizeof(info.product_name));
fgets(buf, 55, fread);
strncpy(info.product_name, buf, (strlen(buf)-1));
```

```
memset(buf, 0, sizeof(buf));
memset(info.product_code, 0, sizeof(info.product_code));
fgets(buf, 25, fread);
strncpy(info.product_code, buf, (strlen(buf)-1));
```

```
memset(buf, 0, sizeof(buf));
fgets(buf, 15, fread);
info.product_price = atoi(buf);
```

```
memset(buf, 0, sizeof(buf));
fgets(buf, 15, fread);
info.product_quantity = atoi(buf);
```

```
memset(buf, 0, sizeof(buf));
fgets(buf, 10, fread);
info.date.day = atoi(buf);
```

```
memset(buf, 0, sizeof(buf));
fgets(buf, 10, fread);
info.date.month = atoi(buf);
```

```
memset(buf, 0, sizeof(buf));
fgets(buf, 10, fread);
info.date.year = atoi(buf);
```

```
addNewElement(head, info);
```

```
}
```



```

        fclose(fread);
        return;
    }

/*-----workWithTable.h-----*/

#ifndef TEST_WORKWITHTABLE_H
#define TEST_WORKWITHTABLE_H

#include "structures.h"
#include "specialFunctions.h"
#include "setData.h"

//=====
//
//          "ГЛАВНЫЕ" ФУНКЦИИ
//
//          ПРОТОТИПЫ
//=====

Info setInfo(Element *head); // Ввод информации
void addNewElement(Element **head, Info info); // Добавить элемент в таблицу
int getNumElements(Element *head); // Взять количество элементов таблицы
void showTable(Element *head); // Демонстрация таблицы
Element *selectElement(Element *head); // Вернуть указатель на выбранный элемент
void correctRecord(Element *elem); // Выбрать и скорректировать запись
void deleteElement(Element **head); // Удалить элемент в таблице
void deleteTable(Element **head); // Удалить все элементы таблицы
Element *searchElement(Element *head); // Ищем элемент по номеру чека
void showOneElement(Element *head); // Вывести инфу об одном элементе
void sortTableAscending(Element **head); // Сортируем таблицу по возрастанию
(1..2..3..4....)
void sortTableDescending(Element **head); // Сортируем таблицу по убыванию
(....69..54..3..1)
void determineTurnoverForEachStore(Element *head); // Определить товарооборот для
каждого магазина

#endif //TEST_WORKWITHTABLE_H

/*-----workWithTable.c-----*/

```

```

#include "../headers/workWithTable.h"

//=====
//                                     "ГЛАВНЫЕ" ФУНКЦИИ
//=====

//Ввод информации
Info setInfo(Element *head) {
    Info info;
    info.num_shop = setNumShop();
    info.num_section = setNumSection();
    setNumReceipt(info.num_receipt, head);
    setProductName(info.product_name);
    setProductCode(info.product_code);
    info.product_price = setProductPrice();
    info.product_quantity = setProductQuantity();
    info.date = setDate();
    return info;
}

// Добавить элемент в таблицу
void addNewElement(Element **head, Info info) {
    Element *temp = (Element*)malloc(sizeof(Element));
    temp->data = info;
    temp->next = NULL;
    temp->previous = NULL;

    // Если таблица пуста
    if (*head == NULL) {
        *head = temp;
    } else {
        Element *p = *head;
        while(p->next != NULL)
            p = p->next;
        p->next = temp;
    }
}

```

```

        temp->previous = p;
    }
}

// Взять количество элементов таблицы
int getNumElements(Element *head) {
    int num = 0;
    while(head != NULL) {
        num++;
        head = head->next;
    }
    return num;
}

// Демонстрация таблицы
void showTable(Element *head) {
    setCursor(0);
    system("cls");
    if (head == NULL) {
        printf("\n\n\a ТАБЛИЦА ПУСТА! \n\n ");
        setCursor(1);
        system("pause");
        return;
    }

    int ch = 0;
    int i;
    const int numElemToShow = 10;
    const int numElementsOnTable = getNumElements(head);
    Element *temp;

    while(ch != 27) {
        system("cls");

printf("=====
=====
\n");

        printf("\n");
        printf("Навигация: стрелочками вверх/вниз; Esc - выход\n");
    }

```

```

printf("\n");

printf("=====
=====
\n");

printf(" #МАГ | #ЧЕК |          НОМЕР ЧЕКА          |
НАИМЕНОВАНИЕ ТОВАРА |          АРТИКУЛ ТОВАРА          | ЦЕНА ТОВАРА | КОЛ-ВО
ТОВАРА |   ДАТА ПРОДАЖИ   \n");

printf("=====
=====
\n");

temp = head;

for (i=0; i != numElemToShow; i++) {
    if (temp != NULL) {
printf(" %-5d | %-4d | %s |", temp->data.num_shop, temp->data.num_section, temp-
>data.num_receipt);

printf(" %-50s | %-20s | %-11d |", temp->data.product_name, temp->data.product_code,
temp->data.product_price);

printf(" %-13d | %-2d / %-2d / %-4d\n", temp->data.product_quantity, temp-
>data.date.day, temp->data.date.month, temp->data.date.year);

printf("-----
-----
\n");

temp = temp->next;
    }
}

do {
    ch = getch();
    if (ch == 27) break;
    if (ch == 224) break;
} while (1);

if (ch == 224) ch = getch(); // так как у стрелок первый код 224 и только
второй 72 или 80, то делаем 2 getch

if (numElementsOnTable > numElemToShow) {
    switch (ch) {
        // 81 - page down; 80 - стрелка вниз
        case 81: {
            if (head->next->next->next->next->next->next->next-
>next->next->next->next->next->next->next->next->next->next != NULL) {
                head = head->next->next->next->next->next->next->next->next->next->next;
                break;
            }
        }
    }
}

```

```

    }

    printf("\a");
    break;
}
case 80: {
if (head->next->next->next->next->next->next->next->next->next->next != NULL)
    head = head->next;
else
    printf("\a");
    break;
}
// 73 - page up; 72 - стрелка вверх
case 73: {
    if (head->previous->previous->previous->previous->previous->previous->previous->previous->previous->previous != NULL) {
        head = head->previous->previous->previous->previous->previous->previous->previous->previous->previous->previous;
        break;
    }
    printf("\a");
    break;
}
case 72: {
    if (head->previous != NULL)
        head = head->previous;
    else
        printf("\a");
    break;
}
}
}
else {
    switch (ch) {
        case 81: case 80: case 73: case 72: printf("\a");
    }
}
}
setCursor(1);

```

```

}

// Вернуть указатель на выбранный элемент
Element *selectElement(Element *head) {
    SetBacklightOff;
    system("cls");
    if (head == NULL) {
        printf("\n\n\a ТАБЛИЧКИ НЕТ! \n\n\n ");
        system("pause");
        return NULL;
    }

    int ch = 0;

    while(ch != 13) {
        system("cls");

printf("=====
=====
\n");

printf("\n");

printf("ДЛЯ ВЫБОРА ЗАПИСИ НУЖНО: \n");

printf("С помощью я сосу пенисы строку, с которой будем работать (Enter -
подтверждение выбора) \n");

printf("если передумали обрабатывать запись нажмите \"Esc\" для выхода\n");
printf("\n");

printf("=====
=====
\n");

printf(" #МАГ | #СЕК |          НОМЕР ЧЕКА          |          НАИМЕНОВАНИЕ ТОВАРА
|   АРТИКУЛ ТОВАРА   | ЦЕНА ТОВАРА | КОЛ-ВО ТОВАРА |   ДАТА ПРОДАЖИ   \n");

printf("=====
=====
\n");

        SetBacklightYellow;

        printf(" %-5d | %-4d | %s |", head->data.num_shop, head-
>data.num_section, head->data.num_receipt);

        printf(" %-50s | %-20s | %-11d |", head->data.product_name, head-
>data.product_code, head->data.product_price);

        printf(" %-13d | %-2d / %-2d / %-4d\n", head->data.product_quantity,
head->data.date.day, head->data.date.month, head->data.date.year);

        SetBacklightOff;

        printf("-----
-----
----- \n");

```

```

do {
    ch = getch();
    if (ch == 13) break; //Enter
    if (ch == 224) { ch = getch(); break;}
    if (ch == 27) return NULL; //Esc
} while (1);

switch (ch) {
    // 81 - page down; 80 - стрелка вниз
    case 81:
    case 80: {
        if (head->next != NULL)
            head = head->next;
        else
            printf("\a");
        break;
    }
    // 73 - page up; 72 - стрелка вверх
    case 73:
    case 72: {
        if (head->previous != NULL)
            head = head->previous;
        else
            printf("\a");
        break;
    }
}

return head;
}

// Выбрать и скорректировать запись
void correctRecord(Element *elem) {
    SetBacklightOff;
    system("cls");
}

```

```

    if (elem == NULL) {
        return;
    }

    int ch = 0;
    const int numRecords = 8;
    int selectedRecord = 1;

    while(ch != 13) {
        system("cls");

        printf("=====
===== \n");

        printf("
\n");

        printf("
КОРРЕКТИРОВКИ ЗАПИСИ НУЖНО:
\n");

        printf("
С помощью стрелок влево/вправо
выбрать корректируемую запись (Enter - подтверждение выбора)
\n");

        printf("
если передумали
корректировать запись нажмите \"Esc\" для выхода
\n");

        printf("
\n");

        printf("=====
===== \n");

        printf(" #МАГ | #ЧЕК |      НОМЕР ЧЕКА      |
НАИМЕНОВАНИЕ ТОВАРА |      АРТИКУЛ ТОВАРА      | ЦЕНА ТОВАРА | КОЛ-ВО
ТОВАРА |   ДАТА ПРОДАЖИ   \n");

        printf("=====
===== \n");

        printf(" %-5d | %-4d | %s |", elem->data.num_shop, elem-
>data.num_section, elem->data.num_receipt);

        printf(" %-50s | %-20s | %-11d |", elem->data.product_name, elem-
>data.product_code, elem->data.product_price);

        printf(" %-13d | %-2d / %-2d / %-4d\n", elem->data.product_quantity,
elem->data.date.day, elem->data.date.month, elem->data.date.year);

        SetBacklightYellow;

        switch(selectedRecord) {

```



```
case 1: {
    gotoxy(1, 9);
    printf("%-5d", elem->data.num_shop);
    break;
}
case 2: {
    gotoxy(9, 9);
    printf("%-4d", elem->data.num_section);
    break;
}
case 3: {
    gotoxy(16, 9);
    printf("%-20s", elem->data.num_receipt);
    break;
}
case 4: {
    gotoxy(39, 9);
    printf("%-50s", elem->data.product_name);
    break;
}
case 5: {
    gotoxy(92, 9);
    printf("%-20s", elem->data.product_code);
    break;
}
case 6: {
    gotoxy(115, 9);
    printf("%-11d", elem->data.product_price);
    break;
}
case 7: {
    gotoxy(129, 9);
    printf("%-13d", elem->data.product_quantity);
    break;
}
case 8: {
    gotoxy(145, 9);
```

```

        printf("%-2d / %-2d / %-4d", elem->data.date.day, elem-
>data.date.month, elem->data.date.year);

```

```

        break;

```

```

    }

```

```

}

```

```

SetBacklightOff;

```

```

printf("\n-----
----- \n");

```

```

do {

```

```

    ch = getch();

```

```

    if (ch == 13) break; //Enter

```

```

    if (ch == 224) { ch = getch(); break;}

```

```

    if (ch == 27) return; //Esc

```

```

} while (1);

```

```

switch (ch) {

```

```

    // <-JIEBO | ПРАВО->

```

```

    case 77: { // 77 - стрелка вправо

```

```

        selectedRecord++;

```

```

        if (selectedRecord > numRecords)

```

```

            selectedRecord = 1;

```

```

        break;

```

```

    }

```

```

    case 75: { // 75 - стрелка влево

```

```

        selectedRecord--;

```

```

        if (selectedRecord < 1)

```

```

            selectedRecord = numRecords;

```

```

        break;

```

```

    }

```

```

}

```

```

}

```

```

switch(selectedRecord) {

```

```

    case 1: {

```

```

        elem->data.num_shop = setNumShop();

```

```

        break;

```

```

    }

```

```

        case 2: {
            elem->data.num_section = setNumSection();
            break;
        }
        case 3: {
            setNumReceipt(elem->data.num_receipt, elem);
            break;
        }
        case 4: {
            setProductName(elem->data.product_name);
            break;
        }
        case 5: {
            setProductCode(elem->data.product_code);
            break;
        }
        case 6: {
            elem->data.product_price = setProductPrice();
            break;
        }
        case 7: {
            elem->data.product_quantity = setProductQuantity();
            break;
        }
        case 8: {
            elem->data.date = setDate();
            break;
        }
    }
}

// Удалить элемент в таблице
void deleteElement(Element **head) {
    system("cls");
    if ((*head) == NULL) {
        printf("\n\n\a Таблицы нет, удалять нечего! \n\n\n ");
        system("pause");
    }
}

```

```

        return;
    }

    Element *deletedElem = *head;
    deletedElem = selectElement(deletedElem);
    while(1) {
        // Если это head
        if (deletedElem == (*head)) {
            if ((*head)->next != NULL) {
                *head = (*head)->next;
                (*head)->previous = NULL;

                //чистим память
                deletedElem->next = NULL;
                deletedElem->previous = NULL;
            }
            free(deletedElem);
            break;
        }

        // Если элемент последний
        if (deletedElem->next == NULL) {
            deletedElem->previous->next = NULL;
            //чистим память
            deletedElem->next = NULL;
            deletedElem->previous = NULL;
            free(deletedElem);
            break;
        }

        // Если элемент не первый и не последний
        deletedElem->previous->next = deletedElem->next;
        deletedElem->next->previous = deletedElem->previous;
        //чистим память
        deletedElem->next = NULL;
        deletedElem->previous = NULL;
        free(deletedElem);
    }
}

```

```

        break;
    }
    printf("\n\n Запись удалена успешно! \n\n");
    system("pause");
}

// Удалить все элементы таблицы
void deleteTable(Element **head) {
    if ((*head) == NULL) {
        system("cls");
        // printf("\n\n а Таблицы нет, поэтому удалять её не надо! \n\n\n ");
        // system("pause");
        return;
    }

    Element *temp;
    while ((*head) != NULL) {
        temp = (*head);
        *head = (*head)->next;
        if ((*head) != NULL)
            (*head)->previous = NULL;

        // чистим память
        temp->next = NULL;
        free(temp);
    }
}

// Ищем элемент по номеру чека
Element *searchElement(Element *head) {
    int i, kol;
    char searchableNumReceipt[21];
    while(1) {
        i = 0;
        kol = 0;
        SetBacklightOff;
        system("cls");

```

```

printf("\n Вы должны ввести номер чека в искомой записи, чтобы мы нашли
для вас запись\n");

printf("\n Введите номер чека:\n");
printf("\n (данные: 20 чисел)");
gotoxy(21 ,3);
SetBacklightYellow;
fflush(stdin);
gets(searchableNumReceipt);
while(searchableNumReceipt[i]) {
    // если число увеличиваем счетчик
    if ((searchableNumReceipt[i] == '1') || (searchableNumReceipt[i] ==
'2') || (searchableNumReceipt[i] == '3') ||
        (searchableNumReceipt[i] == '4') || (searchableNumReceipt[i]
== '5') || (searchableNumReceipt[i] == '6') ||
        (searchableNumReceipt[i] == '7') || (searchableNumReceipt[i]
== '8') || (searchableNumReceipt[i] == '9') || (searchableNumReceipt[i] == '0')) {
        kol++;
    }
    i++;
}
if (strlen(searchableNumReceipt) == kol) {
    if (strlen(searchableNumReceipt) == 20) {
        SetBacklightOff;
        break;
    }
}
SetBacklightOff;
printf("\a");
const char *numReceiptNotFound[] = {
    " Повторить попытку ввода",
    " Выйти"
};
if (menu(numReceiptNotFound, 2, " Запись не найдена") == 1) return NULL;
}

while (head != NULL) {
    if (strcmp(searchableNumReceipt, head->data.num_receipt) == 0) {
        return head;
    }
}

```

```

        head = head->next;
    }
    system("cls");
    printf("\n\n Запись не найдена\n\n\a ");
    system("pause");
    return NULL;
}

// Вывести инфу об одном элементе
void showOneElement(Element *head) {
    if (head == NULL) {
        return;
    }

    unsigned long long sum = ((head->data.product_price) * (head->data.product_quantity));

    system("cls");
    printf("
_____\n");
    printf("|
|\n");
    printf("|
|\n");
    printf("|
|\n", head->data.num_receipt);
    printf("|
|\n");
    printf("|
|\n");
    printf("|
|\n", head->data.num_shop);
    printf("|
|\n", head->data.num_section);
    printf("|
|\n", head->data.product_name);
    printf("|
|\n", head->data.product_code);
    printf("|
|\n", head->data.product_price);
    printf("|
шт. |
|\n", head->data.product_quantity);
    printf("|
|\n");

```

```

        printf("|      Итоговая сумма:                %19llu Руб. |
\n", sum);

        printf("|
|\n");

        printf("|-----
-----|\n");

        printf("|                                %-2d/%-2d/%d
|\n", head->data.date.day, head->data.date.month, head->data.date.year);

        printf("|_____
_____| \n\n\n\n");

        system("pause");
}

// Сортируем таблицу по возрастанию (1..2..3..4....)
void sortTableAscending(Element **head) {
    Element *temp = *head;
    if (temp->next == NULL) {
        return;
    }

    UI num_shop;
    UI num_section;
    char num_receipt[RECEIPT_MAX_LEN + 1];
    char product_name[PRODUCT_NAME_MAX_LEN + 1];
    char product_code[PRODUCT_CODE_MAX_LEN + 1];
    UL product_price;
    UI product_quantity;
    UI day;
    UI month;
    UI year;

    Element *buf;
    UI numTemp = 0;
    UI kolElementsInTable = getNumElements(temp);

    UI i;
    for (i = kolElementsInTable; i > 1; i--) {
        numTemp = 1;

```



```

temp = (*head);
while(numTemp < i) {
    if (temp->data.num_shop > temp->next->data.num_shop) {
        num_shop = temp->data.num_shop;
        num_section = temp->data.num_section;
        strcpy(num_receipt, temp->data.num_receipt);
        strcpy(product_name, temp->data.product_name);
        strcpy(product_code, temp->data.product_code);
        product_price = temp->data.product_price;
        product_quantity = temp->data.product_quantity;
        day = temp->data.date.day;
        month = temp->data.date.month;
        year = temp->data.date.year;

        temp->data.num_shop = temp->next->data.num_shop;
        temp->data.num_section = temp->next->data.num_section;
        strcpy(temp->data.num_receipt, temp->next->data.num_receipt);
        strcpy(temp->data.product_name, temp->next-
>data.product_name);
        strcpy(temp->data.product_code, temp->next-
>data.product_code);
        temp->data.product_price = temp->next->data.product_price;
        temp->data.product_quantity = temp->next-
>data.product_quantity;
        temp->data.date.day = temp->next->data.date.day;
        temp->data.date.month = temp->next->data.date.month;
        temp->data.date.year = temp->next->data.date.year;

        temp->next->data.num_shop = num_shop;
        temp->next->data.num_section = num_section;
        strcpy(temp->next->data.num_receipt, num_receipt);
        strcpy(temp->next->data.product_name, product_name);
        strcpy(temp->next->data.product_code, product_code);
        temp->next->data.product_price = product_price;
        temp->next->data.product_quantity = product_quantity;
        temp->next->data.date.day = day;
        temp->next->data.date.month = month;
        temp->next->data.date.year = year;
    }
}

```

```

        numTemp++;
        temp = temp->next;
    }
}
temp = temp->previous;
(*head) = temp;
}

// Сортируем таблицу по убыванию (...69..54..3..1)
void sortTableDescending(Element **head) {
    Element *temp = *head;
    if (temp->next == NULL) {
        return;
    }

    UI num_shop;
    UI num_section;
    char num_receipt[RECEIPT_MAX_LEN + 1];
    char product_name[PRODUCT_NAME_MAX_LEN + 1];
    char product_code[PRODUCT_CODE_MAX_LEN + 1];
    UL product_price;
    UI product_quantity;
    UI day;
    UI month;
    UI year;

    Element *buf;
    UI numTemp = 0;
    UI kolElementsInTable = getNumElements(temp);

    UI i;
    for (i = kolElementsInTable; i > 1; i--) {
        numTemp = 1;
        temp = (*head);
        while(numTemp < i) {
            if (temp->data.num_shop < temp->next->data.num_shop) {
                num_shop = temp->data.num_shop;

```

```

        num_section = temp->data.num_section;
        strcpy(num_receipt, temp->data.num_receipt);
        strcpy(product_name, temp->data.product_name);
        strcpy(product_code, temp->data.product_code);
        product_price = temp->data.product_price;
        product_quantity = temp->data.product_quantity;
        day = temp->data.date.day;
        month = temp->data.date.month;
        year = temp->data.date.year;

        temp->data.num_shop = temp->next->data.num_shop;
        temp->data.num_section = temp->next->data.num_section;
        strcpy(temp->data.num_receipt, temp->next->data.num_receipt);
        strcpy(temp->data.product_name, temp->next->
>data.product_name);
        strcpy(temp->data.product_code, temp->next->
>data.product_code);
        temp->data.product_price = temp->next->data.product_price;
        temp->data.product_quantity = temp->next->
>data.product_quantity;
        temp->data.date.day = temp->next->data.date.day;
        temp->data.date.month = temp->next->data.date.month;
        temp->data.date.year = temp->next->data.date.year;

        temp->next->data.num_shop = num_shop;
        temp->next->data.num_section = num_section;
        strcpy(temp->next->data.num_receipt, num_receipt);
        strcpy(temp->next->data.product_name, product_name);
        strcpy(temp->next->data.product_code, product_code);
        temp->next->data.product_price = product_price;
        temp->next->data.product_quantity = product_quantity;
        temp->next->data.date.day = day;
        temp->next->data.date.month = month;
        temp->next->data.date.year = year;
    }
    numTemp++;
    temp = temp->next;
}
}

```

```

temp = temp->previous;
(*head) = temp;
}

// Определить товарооборот для каждого магазина
void determineTurnoverForEachStore(Element *head) {
    system("cls");
    if (head == NULL) {
        printf("\a\n\n Таблицы нет, товарооборот не может быть определен!\n\n ");
        system("pause");
        return;
    }

    //=====
    FILE *fdownload;
    // переменная флаг - "сохраняем файл или нет"
    int saveOrNo = 0;
    if (yesOrNo(" Сделаем вывод товарооборота в каждом магазине в текстовый файл?")
== 0) {
        // Вводим путь к файлу сохранения
        int exitFromWhile = 0;
        int lenBuf = 0;

        //Ввод пути к файлу где будем сохранять
        char buf[255];
        do {
            memset(buf, 0, sizeof(buf));
            system("cls");
            SetBacklightOff;
            printf("\n Введите путь к файлу вывода:\n\n");
            printf(" (например: \"D:\\\\coursework\\\\turnover\\\\turnover1.txt\")");
            gotoxy(30, 1);
            SetBacklightYellow;
            fflush(stdin);
            gets(buf);
            SetBacklightOff;

```

```

// проверить текстовый файл или нет, и если нет то повторяем
попытку ввода

lenBuf = strlen(buf);

if (lenBuf > 4) {
    if (((buf[lenBuf-4] == '.') && (buf[lenBuf-3] == 't') &&
(buf[lenBuf-2] == 'x') && (buf[lenBuf-1] == 't')) == 0) {
        printf("\a");
        continue;
    }
}
else {
    printf("\a");
    continue;
}

//Если файл уже существует перезаписать его?
if ((fdownload = fopen(buf, "r")) != NULL) {
    const char *overwrite[] = {
        " Да, давай перезапишем",
        " Неа, повторим попытку ввода",
        " Я хочу выйти, передумал делать вывод в текстовый
файл"

    };
    fclose(fdownload);
    switch(menu(overwrite, 3, " Указанный файл уже существует.
Перезапишем его?")) {
        case 0: {
            //выходим из ввода
            fdownload = fopen(buf, "w");
            saveOrNo = 1;
            exitFromWhile = 1;
            break;
        }
        case 1: {
            exitFromWhile = 2;
            break;
        }
        case 2: {
            exitFromWhile = 1;

```

```

        }

    }

    if (exitFromWhile == 1) break;
    if (exitFromWhile == 2) continue;
}

if ((fdownload = fopen(buf, "w")) != NULL) {
    saveOrNo = 1;
    exitFromWhile = 1;
}
else {
    printf("\a");
    const char *error[] = {
        " Повторить попытку ввода",
        " Я хочу выйти, передумал делать вывод в текстовый
файл"

    };
    if (menu(error, 2, " Не удалось получить доступ к файлу!") ==
1) {
        exitFromWhile = 1;
    }
}

} while (exitFromWhile != 1);
} //делаем вывод товарооборота в каждом магазине в текстовый файл?

//=====
//отсортируем список но не через функцию, потому что функция меняет оригинальный
список
while(1) {
    Element *firstElem = head;
    if (head->next == NULL) {
        break;
    }

    UI num_shop;
    UI num_section;
    char num_receipt[RECEIPT_MAX_LEN + 1];
    char product_name[PRODUCT_NAME_MAX_LEN + 1];
    char product_code[PRODUCT_CODE_MAX_LEN + 1];

```

```

UL product_price;
UI product_quantity;
UI day;
UI month;
UI year;

Element *buf;
UI numTemp = 0;
UI kolElementsInTable = getNumElements(head);

UI i;
for (i = kolElementsInTable; i > 1; i--) {
    numTemp = 1;
    head = firstElem;
while(numTemp < i) {
    if (head->data.num_shop > head->next->data.num_shop) {
        num_shop = head->data.num_shop;
        num_section = head->data.num_section;
        strcpy(num_receipt, head->data.num_receipt);
        strcpy(product_name, head->data.product_name);
        strcpy(product_code, head->data.product_code);
        product_price = head->data.product_price;
        product_quantity = head->data.product_quantity;
        day = head->data.date.day;
        month = head->data.date.month;
        year = head->data.date.year;

        head->data.num_shop = head->next->data.num_shop;
        head->data.num_section = head->next->data.num_section;
        strcpy(head->data.num_receipt, head->next->data.num_receipt);
        strcpy(head->data.product_name, head->next->data.product_name);
        strcpy(head->data.product_code, head->next->data.product_code);
        head->data.product_price = head->next->data.product_price;
        head->data.product_quantity = head->next->data.product_quantity;
        head->data.date.day = head->next->data.date.day;
        head->data.date.month = head->next->data.date.month;
        head->data.date.year = head->next->data.date.year;
    }
}
}

```

```

        head->next->data.num_shop = num_shop;
        head->next->data.num_section = num_section;
        strcpy(head->next->data.num_receipt, num_receipt);
        strcpy(head->next->data.product_name, product_name);
        strcpy(head->next->data.product_code, product_code);
        head->next->data.product_price = product_price;
        head->next->data.product_quantity = product_quantity;
        head->next->data.date.day = day;
        head->next->data.date.month = month;
        head->next->data.date.year = year;
    }
    numTemp++;
    head = head->next;
}
}
head = head->previous;
break;
}

//=====
printf("\n");
UI lastnumShop = head->data.num_shop; // последний магазин
unsigned long long turnoverSum = 0; // счетчик суммы товарооборота

char numberSTR[25];
char message[80];
if(saveOrNo == 1) {
    memset(message, 0, sizeof(message));
    strcat(message, "\n Товарооборот по каждому магазину, который есть в
таблице:\n\n");
    fputs(message, fdownload);
}

printf("\n\n Товарооборот по каждому магазину, который есть в таблице:\n\n");
//Выводим на экран (и в файл если saveOrNo == 1) инфу о товарообороте в каждом
магазине
while(head != NULL) {

```



```

    if (head->data.num_shop == lastnumShop) {
        //копим сумму товарооборота по одному магазину
        turnoverSum += (head->data.product_price * head-
>data.product_quantity);
    }
    else {
        //Вывод в консоль
        printf(" В магазине №%d сумма товарооборота = %llu;\n",
lastnumShop, turnoverSum);

        //запись в файл
        if(saveOrNo == 1) {
            memset(message, 0, sizeof(message));
            strcat(message, " В магазине №");
            memset(numberSTR, 0, sizeof(numberSTR));
            itoa(lastnumShop, numberSTR, 10);
            strcat(message, numberSTR);
            strcat(message, " сумма товарооборота = ");
            memset(numberSTR, 0, sizeof(numberSTR));
            itoa(turnoverSum, numberSTR, 10);
            strcat(message, numberSTR);
            strcat(message, ";\n");
            fputs(message, fdownload);
        }

        turnoverSum = 0; // обнуляем сумму для нового магазина
        turnoverSum += (head->data.product_price * head-
>data.product_quantity);

        lastnumShop = head->data.num_shop;
    }

    head = head->next;
}

//Вывод в консоль
printf(" В магазине №%d сумма товарооборота = %llu;\n", lastnumShop,
turnoverSum);

//запись в файл
if(saveOrNo == 1) {
    memset(message, 0, sizeof(message));
    strcat(message, " В магазине №");
    memset(numberSTR, 0, sizeof(numberSTR));

```

```
    itoa(lastnumShop, numberSTR, 10);
    strcat(message, numberSTR);
    strcat(message, " сумма товарооборота = ");
    memset(numberSTR, 0, sizeof(numberSTR));
    itoa(turnoverSum, numberSTR, 10);
    strcat(message, numberSTR);
    strcat(message, ";\n");
    fputs(message, fdownload);

    //и не забываем закрыть файл если он был
    fclose(fdownload);
}
printf("\n ");
system("pause");
}
```