**АННОТАЦИЯ**

Данная курсовая работа описывает разработку базы данных на языке C++ с элементами Си с использованием двусвязного списка для учета материальных ценностей в различных филиалах завода за отчетный квартал. Каждая запись содержит информацию о номере завода, номере филиала, ответственном лице, начальном остатке материальных ценностей, поступлении и выбытии за период. Целью работы является создание отчета о движении материальных ценностей, включающего подробную информацию о стоимости материальных ценностей по каждому филиалу и по всему заводу на конец отчетного периода, а также сводные показатели по видам ценностей и по заводу в целом. Также предусматривается контроль получения итоговой цифры о наличии материальных ценностей на конец отчетного периода как по строкам, так и по столбцам.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc156446637)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 5](#_Toc156446638)

[2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 6](#_Toc156446639)

[**2.1 Постановка задачи**…………………………………………………….**6**](#_Toc156446640)

[**2.2 Особенности работы программы**……………………………………**6**](#_Toc156446641)

[**2.3 Описание основных типов данных программы……………….......7**](#_Toc156446642)

[**2.4 Структура данных программы**……………………………………...**7**](#_Toc156446643)

[**2.5 Входные данные**……………………………………………………….**7**](#_Toc156446644)

[**2.6 Выходные данные**……………………………………………………..**8**](#_Toc156446645)

[**2.7** **Используемые функции**……………..………………………………..**8**](#_Toc156446646)

[**2.8 Алгоритм функций**…………………………………………………..**10**](#_Toc156446647)

[3. ВЫПОЛНЕНИЕ ПРОГРАММЫ 27](#_Toc156446648)

[**3.1 Условия выполнения программы**…………………………………**27**](#_Toc156446649)

[**3.2 Загрузка и запуск программы**………………………………………**27**](#_Toc156446650)

[**3.3 Проверка работоспособности программы**………………………..**27**](#_Toc156446651)

[ЗАКЛЮЧЕНИЕ 33](#_Toc156446652)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc156446653)

[ПРИЛОЖЕНИЕ А 35](#_Toc156446654)

# **ВВЕДЕНИЕ**

С каждым годом появляется все больше информации, которую становится всё менее рациональным хранить на бумажных носителях, тем более систематизировать такой объём информации. В данном случае используются возможности вычислительной техники, в том числе и компьютера.

Целью данного курсового проекта является разработка программы подсчёта количества просроченного товара и размер ущерба.

Для написания программы был выбран язык C++ с элементами С языка. Средой программирования была выбрана Visual Studio 2023.

Программа написана согласно заданию кафедры информационных систем Севастопольского Государственного Университета.

# **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**

Для представления таблицы в коде программы используются двунаправленные списки. Разработанная программа использует меню-ориентированный интерфейс со следующими пунктами: начальное создание таблицы, просмотр таблицы, добавление новой записи в таблицу, удаление, корректировка, сортировка, поиск, сохранение, обработка и чтение. Данная программа предназначена для хранения и обработки данных о ресурсах на заводе. Данные имеют следующий вид: номер завода, номер филиала, фамилия ответственного за материальные ценности, наличие материальных ценностей на начало периода, получено материальных ценностей на сумму, выбыло на сумму. Программа дает возможность проводить следующие операции с данными: ввод записей с клавиатуры, сохранение и чтение из типизированного файла, поиск, редактирование и удаление записи, сортировку, вывод отчетной таблицы и обработка данных.

Программа может быть применена для работы на любых производственных предприятиях.

# **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**

## **2.1 Постановка задачи**

Даны сведения за квартал о материальных ценностях в стоимостном выражении по филиалам завода. Структура записи: номер завода, номер филиала, фамилия ответственного за материальные ценности, наличие материальных ценностей на начало периода, получено материальных ценностей на сумму, выбыло на сумму. Получить ведомость движения материальных ценностей за отчетный период, содержащую: стоимость материальных ценностей по каждому филиалу на конец отчетного периода, по всему заводу на конец периода, а также итоговые цифры по каждому виду (на начало периода, получено, выбыло, на конец периода) и по всему заводу в целом. Проконтролировать двумя путями (по строке и столбцу) получение итоговой цифры: наличие материальных ценностей на конец отчетного периода в стоимостном выражении по заводу в целом

В программе необходимо организовать функции: открытия файла с данными, сортировки, добавления, редактирования и удаления данных, а также вывод информации по заданию.

Также были использованы вспомогательные функции, такие как интерфейсное меню, меню для выбора удаляемого элемента и меню для выбора редактируемого элемента.

## **2.2 Особенности работы программы**

Рассматриваемая программа написана на языке программирования C++. Программа может получать входные данные из файла, сохранять их и выводить в виде таблицы. Программа работает в режиме диалога. В меню пользователю предложены задачи, которые он может выполнить.

## **2.3 Описание основных типов данных программы**

Char, int, bool.

## **2.4 Структура данных программы**

Программа была написана в среде Visual Studio 2023. Данная среда была выбрана, так как в ней присутствует большое число подключаемых библиотек по умолчанию, а также удобный интерфейс для работы.

Для курсового проектирования была выбрана структура двунаправленного списка.

**Структуры данных, которые используются в программе:**

**struct Node**

**{**

**int plantNumber;**

**int branchNumber;**

**char surname[50];**

**int beginningAssets;**

**int receivedAssets;**

**int disposedAssets;**

**Node \*prev;**

**Node \*next;**

**};**

Структура содержит данные о дате, данные о заводе, двунаправленный список.

Цена на номер завода, номер ветви, начальные ресурсы, полученные и вычтенные представляют собой целочисленные числа. Фамилия – массив char типа.

## **2.5 Входные данные**

При запуске пользователь должен выбрать пункт меню «Refresh list» и ввести название файла, без указания расширения, с которого будут считываться данные, расположенные в папке с приложением. После чего уже можно будет добавлять к списку новые элементы.

## **2.6 Выходные данные**

Для просмотра выходных данных необходимо использовать функцию:

**void displayPage(Node \*node, Node \*hNode, int items, int page, int totalPages)**

**Для выхода с функции необходимо нажать Q (quit).**

## **Используемые функции**

1. **void addElementToEnd(Node \*head)**

Функция добавления в конец новый элемент списка. Передающиеся параметры – первый элемент списка.

1. **Node \*createDoublyLinkedList()**

Функция для удаления элемента списка. Передающийся параметр – указатель на элемент списка, указатель на адрес начала списка.

1. **void removeElement(Node \*&head, Node \*elem)**

Функция для отображения всех элементов, с одинаковым название продукта, и выбора определенного элемента списка для его дальнейшего удаления, поиск и вывод элементов осуществляется по полю названия продукта. Передающийся параметр – первый элемент списка.

1. **void output(product\* beg);**

Функция вывода всего списка на экран с возможностью перелистывания. Передающийся параметр – указатель на первый элемент списка.

1. **void displayDoublyLinkedList(Node \*head)**

Функция сортировки по уникальному полю Plant number с использованием bubble sort.

1. **void bubbleSort(Node \*\*head)**

Функция для редактирования элемента списка. Передающийся параметр – уникальное поле, начало списка.

1. **void updateField(Node \*&head, int plantNumber)**

Функция выравнивания вывода.

1. **void goToXY(short x, short y)**

Вспомогательная функция для перелистывания списка.

1. **void displayPage(Node \*node, Node \*hNode, int items, int page, int totalPages)**

Вспомогательная функция для обмена узлов.

1. **struct Node \*swap(struct Node \*ptr1, struct Node \*ptr2)**

Функция для подсвечивания активного пункта меню.

1. **void highlightMenuItem(const string &menuItem, bool isSelected)**

Функция поиска узла по любому полю.

1. **void searchNode(Node \*head)**

Функция для подсчёта длины массива.

1. **int getCharArrayLength(const char \*arr)**

Функция для копирования массива.

1. **void copyCharArray(const char \*source, char \*destination)**

Функция для конкатенации массива.

1. **void concatenateCharArray(const char \*str1, const char \*str2, char \*result)**

Функция для сохранения списка в тектовый и типизированный файл.

1. **void saveLinkedListToFile(Node \*head)**

Функция для загрузки типизированного списка.

1. **void refreshList(Node \*&head)**

Функция для обработки данных.

1. **void handleData(Node \*head)**

## **2.8 Алгоритм функций**

Блок 1 – проверка списка ну пустоту

Блок 2 – если пуст добавляемый элемент становится первым

Блок 3 – если нет устанавливаем добавляемый элемент последним

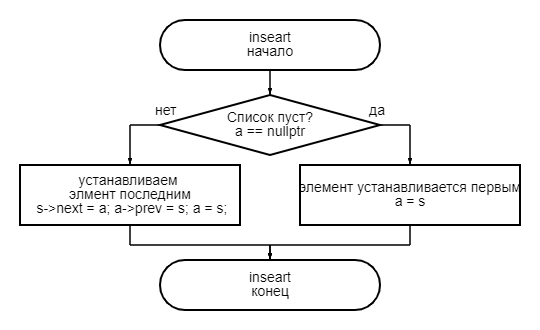


Рисунок 2.1 - Схема алгоритма функции вставки элемента в конец

Блок 1 – создание нового объекта и присвоение ему полей структуры

Блок 2 – установка указателей на следующий и предбудущий элемент, возвращения нового объекта

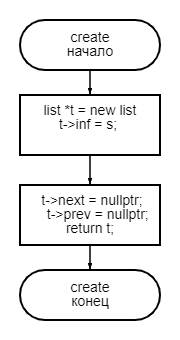


Рисунок 2.2 - Схема алгоритма функции создания списка

Блок 1 – создание указателя на текущий элемент

Блок 2 – проверка списка на пустоту

Блок 3 – если пуст, то выводится сообщение об этом

Блок 4 – если нет, напечатанные страницы устанавливаются в ноль, текущая страница устанавливается в ноль, вычисляет индекс, с которого будет начато печатание элементов списка на текущей странице

Блок 5 – выводится шапка меню

Блок 6 – начало цикла while с условием до тех пор пока, напечатанные элементы не достигнут максимума либо текущие элементы не закончатся

Блок 7 – проверка на то больше или равен текущий элемент индексу элемента списка

Блок 8 – если да, то выводится таблица

Блок 9 – увеличивается количество напечатанных элементов printed­\_elements++

Блок 10 – текущий элемент увеличивается и текущий элемент переходит на следующий

Блок 11 – закрытие цикла while

Блок 12 – выводится «Выход»

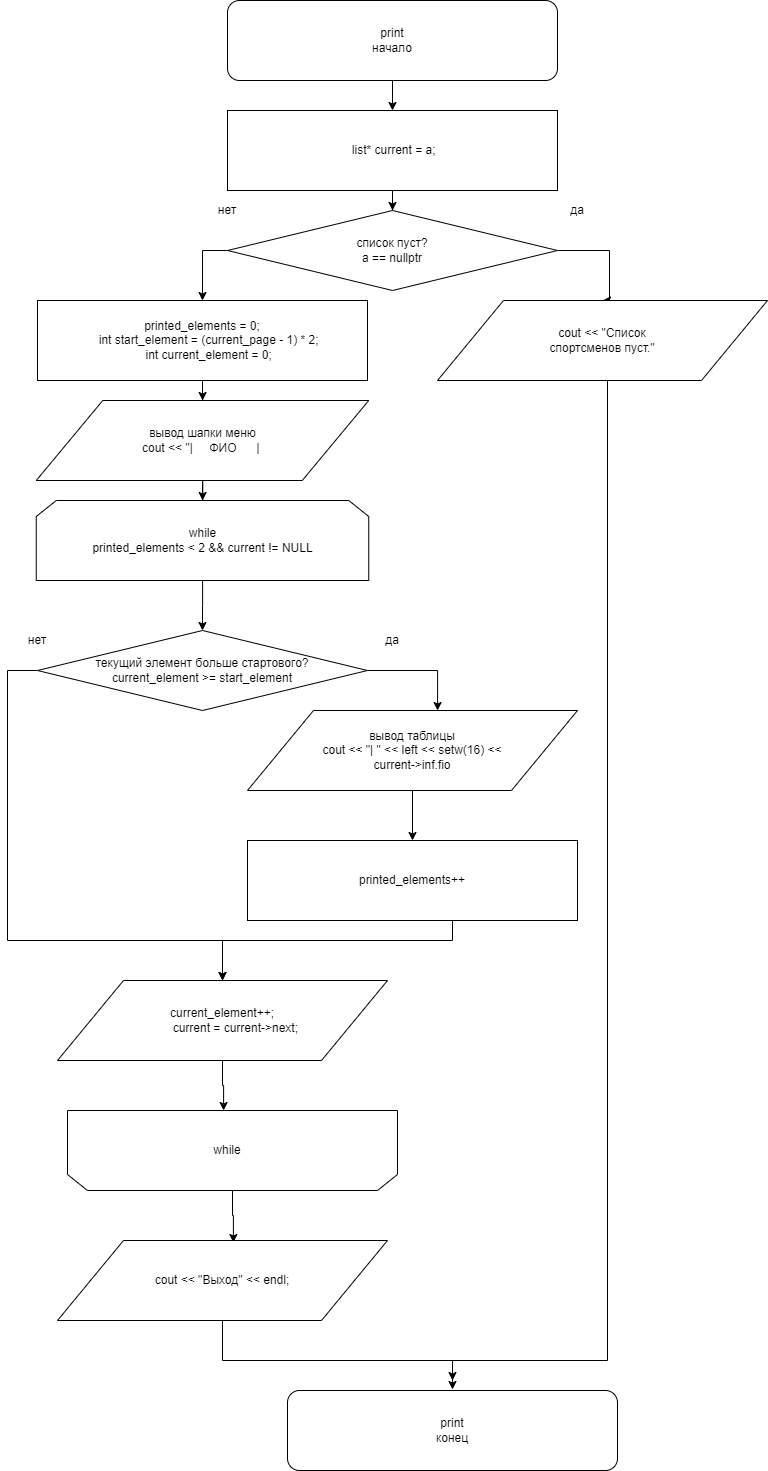


Рисунок 2.3 - Схема алгоритма функции печати элементов

Блок 1 – вывод проверки точно ли пользователь хочет удалить список

Блок 2 – ввод переменной vybor которая используется для подтверждения своего действия

Блок 3 – считывание переменной vybor

Блок 4 – проверка ввел ли пользователь 1

Блок 5 – если да то инициализируется переменная текущего элемента списка

Блок 6 – начало цикла while с условием до тех пор, пока, текущий элемент не достигнет конца списка

Блок 7 – инициализированние переменной next и присваивание ей значения следующий после текущего элемент (list\*next = current->next), удаление текущего элемента, присвоение значения следующего текущему (current =next)

Блок 8 – конец цикла while

Блок 9 – присвоение началу списка значения 0

Блок 10 – вывод информации о том, что список был удален

Блок 11 – если была введена не 1, то проверка был ли введен 0

Блок 12 – если да вывод «Действие было отменено»

Блок 13 – если нет тогда вывод «Ошибка ввода. Попробуйте еще раз»

Блок 14 – вызов функции deleteList

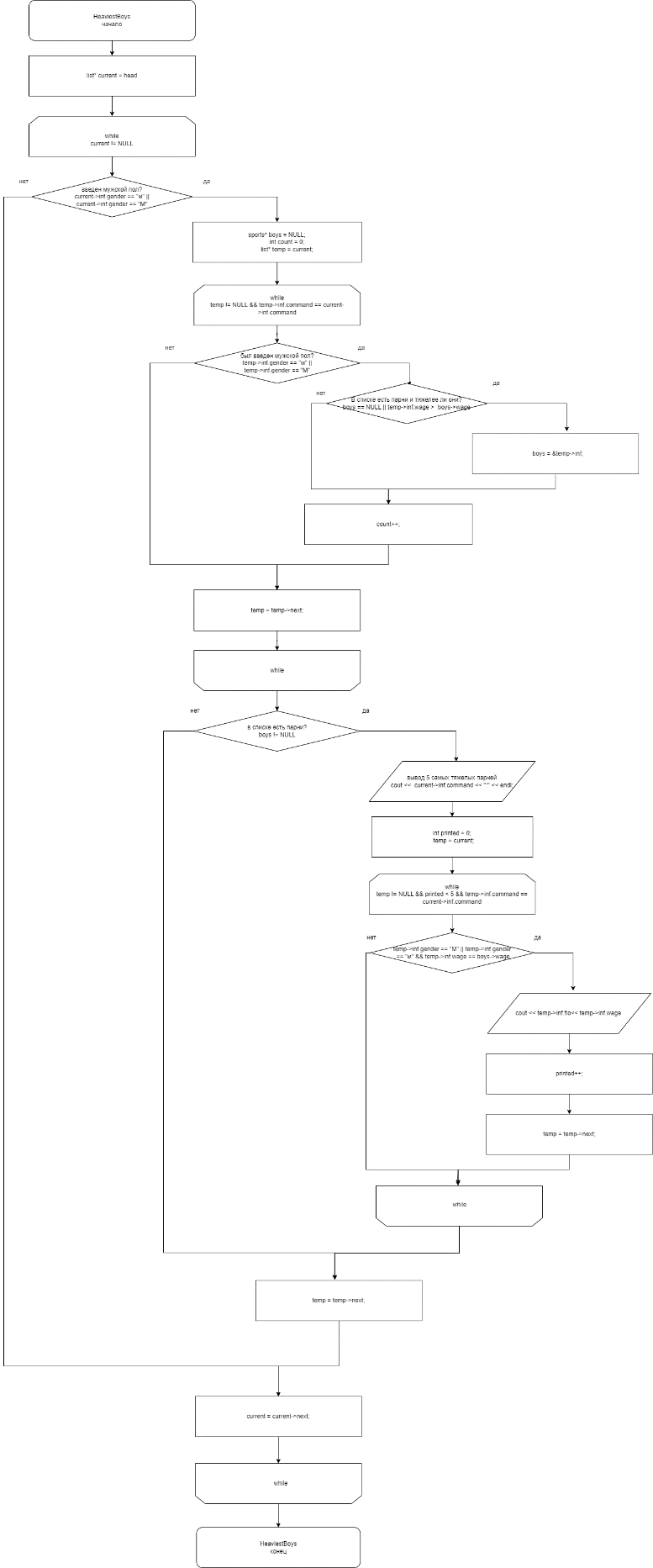


Рисунок 2.5 - Схема алгоритма функции поиска

Блок 1 – проверки пуст ли список

Блок 2 – если да, то происходит преждевременный выход из функции

Блок 3 – если нет, вводится переменная имени файла filename

Блок 4 – выводится сообщение «Введите имя файла для сохранения» и считывается значение переменной filename

Блок 5 – начало цикла while с условием до тех пор пока, пока имя файла не будет введено правильно (проверяется с помощью вызова функции !NameFile)

Блок 6 – вывод «Некорректное имя файла для сохранения» и считывание переменной filename

Блок 7 – конец цикла while

Блок 8 – открытие файла

Блок 9 – проверка на то не был ли файл открыт

Блок 10 – если да, то вывод «Ошибка при открытии файла»

Блок 11 – преждевременный выход из программы за счет использования return

Блок 12 – если нет, инициализируется текущий элемент

Блок 13 – начало цикла while с условием до тех пор, пока, текущий элемент не будет достигнут конца списка

Блок 14 – запись данных о спортсменах

Блок 15 –установка значения текущего элемента на следующий

Блок 16 – конец цикла while

Блок 17 – закрытие файла

Блок 18 – вывод информации «Данные успешно сохранены в файл»

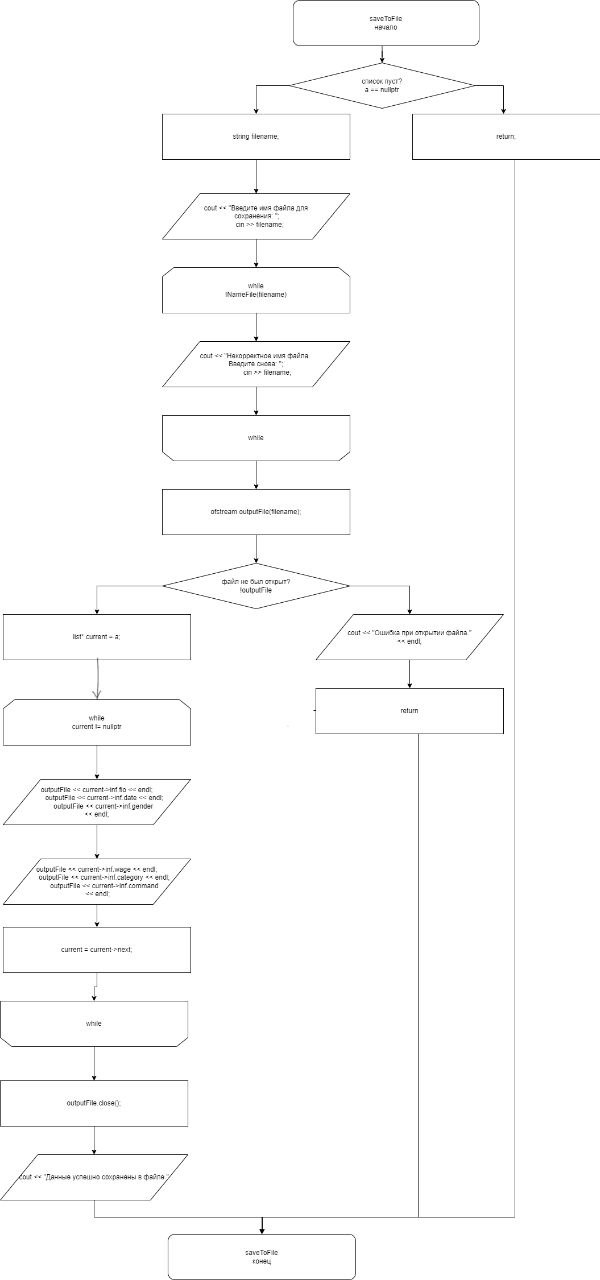


Рисунок 2.6 - Схема алгоритма функции сохранения в текстовый файл

Блок 1 – инициализация переменной filename

Блок 2 – вывод «Введите имя файла для загрузки» и считывание переменной filename

Блок 3 – открытие файла в режиме чтения

Блок 4 – проверка на то не был ли файл открыт

Блок 5 – если да, вывод «Ошибка при открытии файла»

Блок 6 – преждевременный выход из функции с помощью return и возвращение нуля

Блок 7 – если нет, создаем указатель на список

Блок 8 – начало цикла while с условием до тех пор, пока не будет достигнут конец файла

Блок 9 – инициализируем экземпляр структуры sports

Блок 10 – получение информации о спортсменах из файла

Блок 11 – вывозов функции create для создания списка

Блок 12 – вызов функции inseart для добавления элемента в конец

Блок 13 – конец цикла while

Блок 14 – закрытие файла и инициализация текущего элемента current

Блок 15 – начало цикла while с условием до тех пор, пока список не будет пуст

Блок 16 – вывод информации о спортсменах

Блок 17 – перестановка текущего элемента current на следующий current->next

Блок 18 – конец цикла while

Блок 19 – проверка пуст ли список

Блок 20 – если да, вывод «Файл пуст»

Блок 21 – если нет вывод «Данные из файла успешно загружены»

Блок 22 – выход из функции и возвращение указателя на список

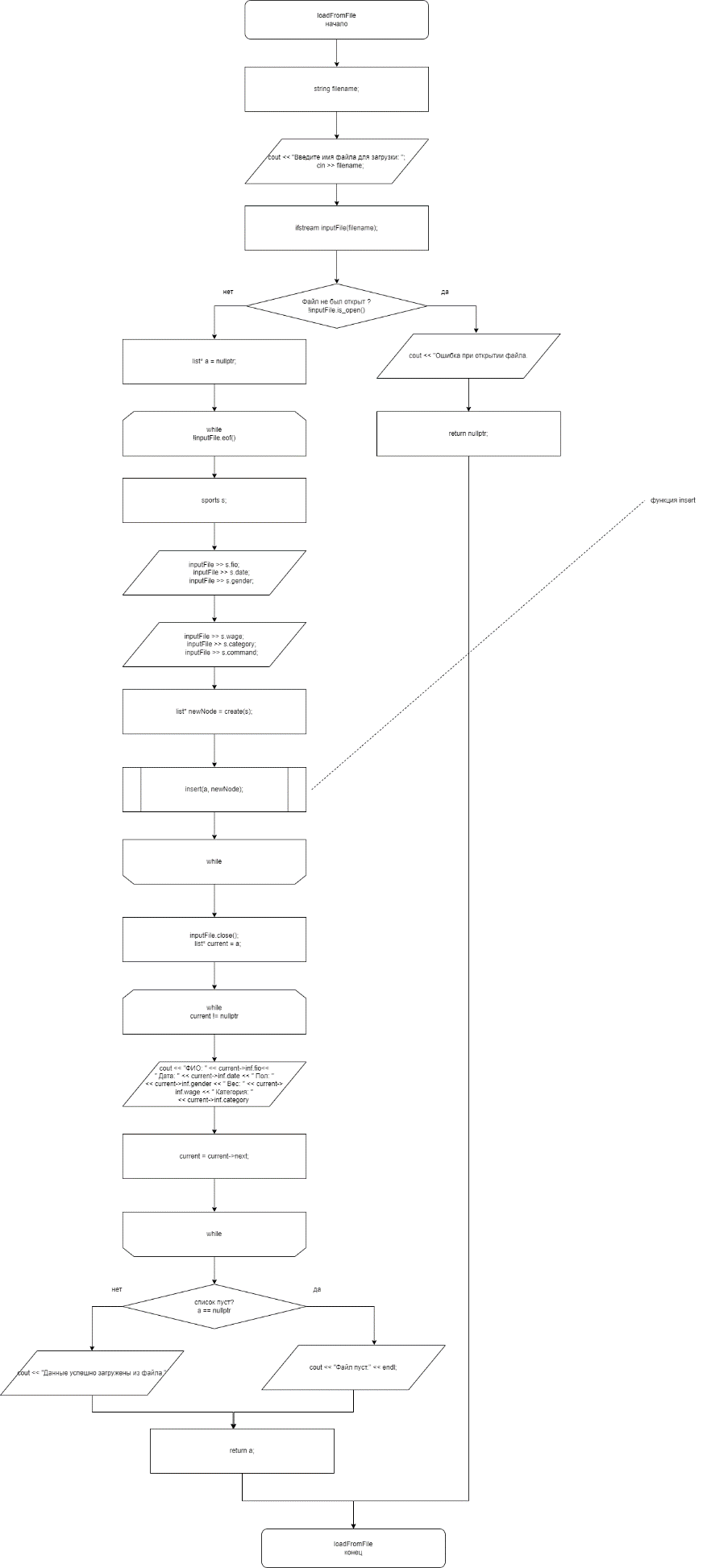


Рисунок 2.7 - Схема алгоритма функции загрузки из текстового файла

Блок 1 – инициализация переменной key

Блок 2 – вывод «Введите ФИО»

Блок 3 – считывание переменной key

Блок 4 – инициализация текущего элемента

Блок 5 – начало цикла while с условием до тех пор, пока текущий элемент не достигнет конца списка

Блок 6 – проверка на то соответствует ли ключ ФИО?

Блок 7 – если да, вывод информации о спортсменах

Блок 8 – возврат текущего элемента

Блок 9 – если нет, задать текущему элементу current значение следующего currenr->next

Блок 10 – закрытие цикла while

Блок 11 – вывод «Элемент не найден»

Блок 12 – возвращение 0

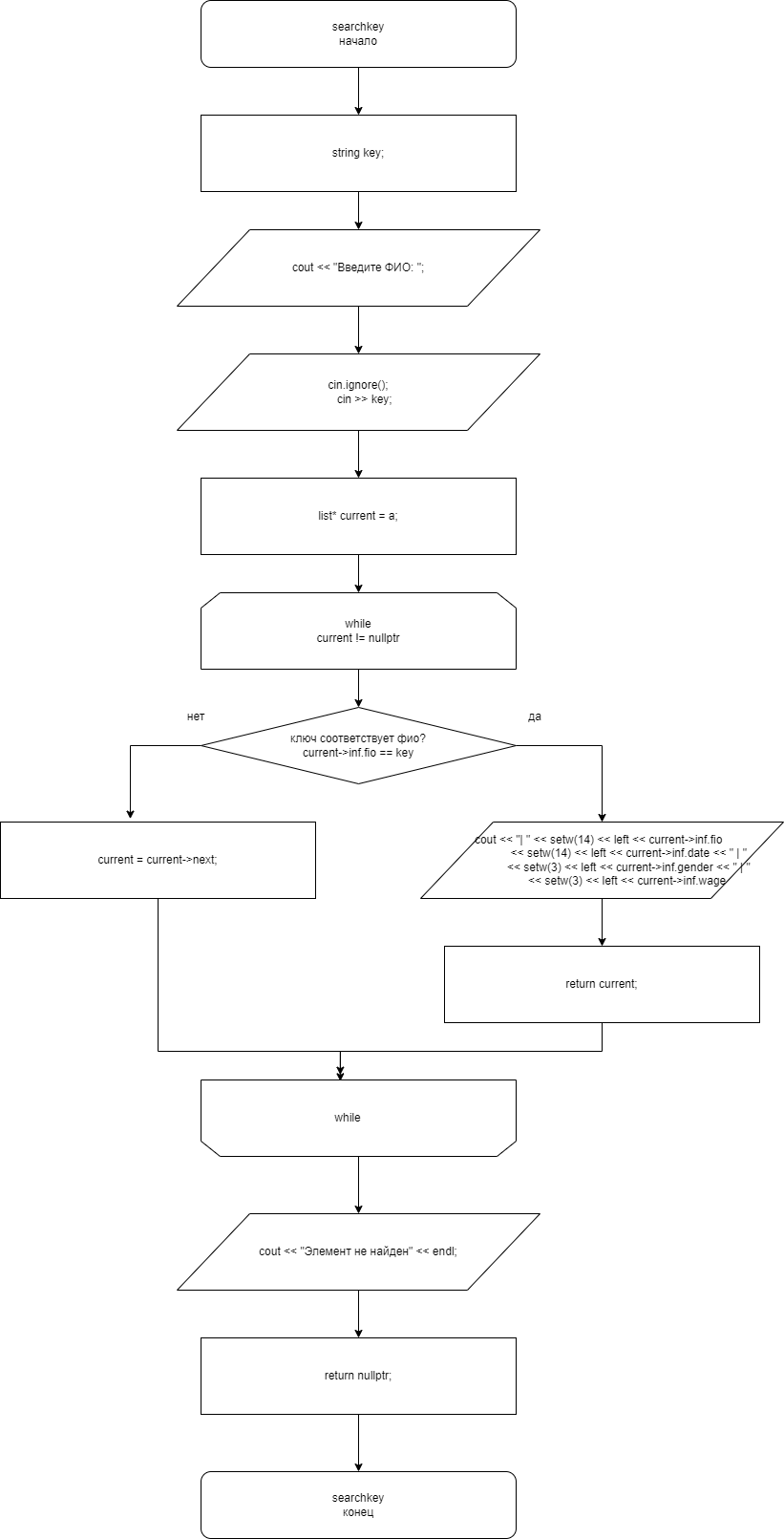


Рисунок 2.8 - Схема алгоритма функции поиска по заданному ключу

Блок 1 – инициализация переменных current и temp

Блок 2 – вывод «Введите фио» и считывание переменной key

Блок 3 – начало цикла while с условием до тех пор, пока не выполнится проверка !Symbol

Блок 4 – вывод «В строке содержатся символы. Введите снова» и считывание переменной key

Блок 5 – конец цикла while

Блок 6 – вывод переменной key

Блок 7 – инициализация переменной deleate со значением false

Блок 8 – начало цикла while с условием до тех пор, пока текущий элемент не достигнет конца списка

Блок 9 – проверка совпадает ли ФИО с ключом?

Блок 10 – если да, проверка равен ли текущий элемент 0?

Блок 11 – если да, установка переменной current значение следующего элемента

Блок 12 – если нет присвоение переменной a значения current->next

Блок 13 – проверка не достиг ли текущий элемент конца списка

Блок 14 – если да, установка переменной current значение пребедущего

Блок 15 – если нет, присвоение переменной temp значения current, переменной current значение следующего, переменной deleted значение true

Блок 16 – прерывание функции

Блок 17 – если нет, установка переменной current значения следующего

Блок 18 – проверка не совпала ли запись

Блок 19 – если да, вывод «Запись не найдена»

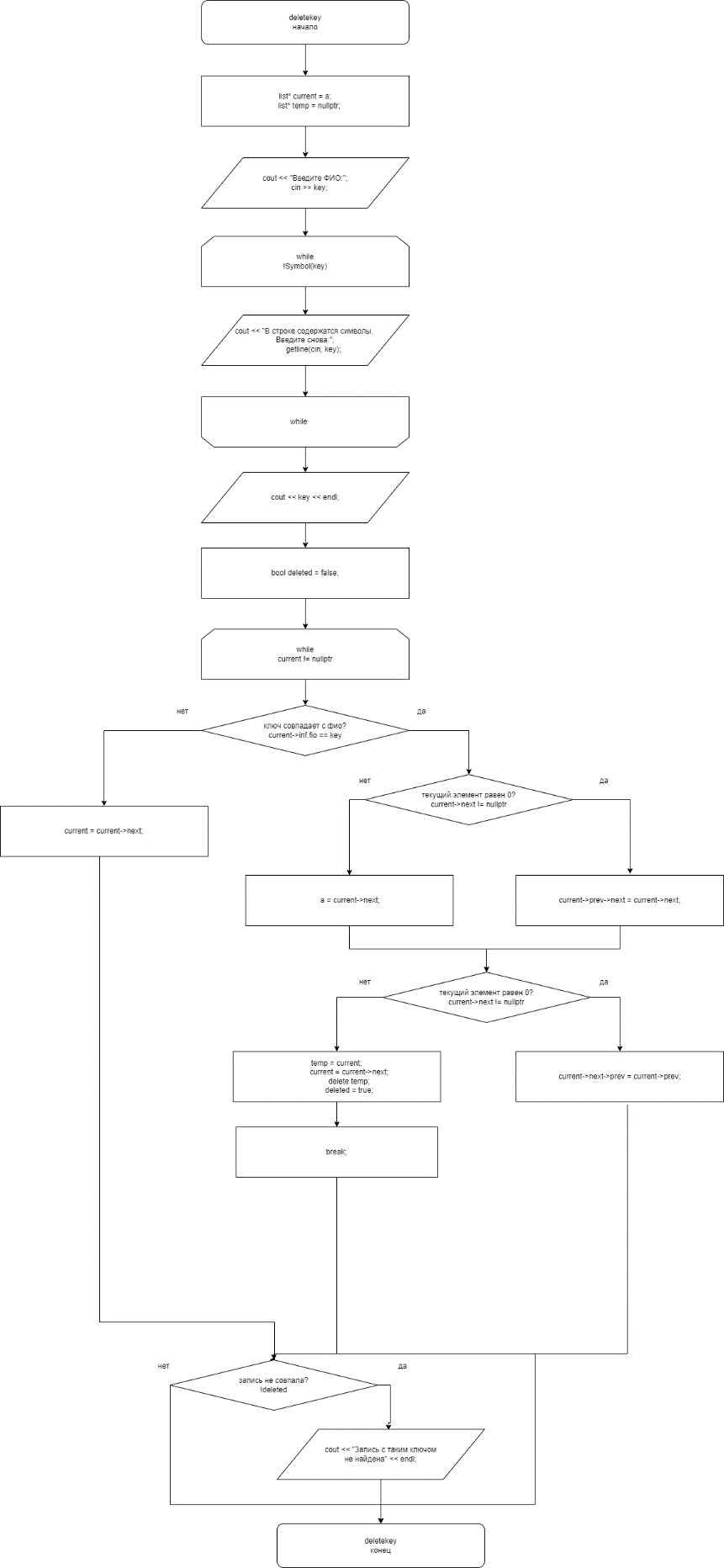


Рисунок 2.9 - Схема алгоритма функции удаления по заданному ключу

Блок 1 – инициализация переменной current

Блок 2 – начало цикла while с условием до тех пор, пока текущий элемент не достигнет конца списка

Блок 3 – проверка на то был ли введен женский пол

Блок 4 – если да, инициализация переменной girl, переменной cout со значением 0, переменной temp со значением current

Блок 5 – начло цикла while с условием до тех пор, пока текущий элемент не дойдет до конца списка и пока команды сравниваемых спортсменов не сравняются

Блок 6 – проверка есть ли женский пол в списке

Блок 7 – если да, проверить больше ли temp->inf.date чем girls->date

Блок 8 – если да, присвоить переменной girls значение temp->inf

Блок 9 – если нет, увеличить значение count

Блок 10 –присвоить переменной temp значение следующего

Блок 11 – конец цикла while

Блок 12 – проверка на наличие девушек в списке

Блок 13 – если да, то вывод «5 самых страших девушек»

Блок 14 – установить переменной printed значение 0

Блок 15 – начало цикла while с условием до тех пор, пока текущий элемент не достигнет конца списка, напечатанных элементов меньше 5, название команд совпадает

Блок 16 – проверка есть ли женский пол в списке

Блок 17 – если да, вывести данные о спортсменах

Блок 18 – увеличить переменную printed

Блок 19 – установить переменной temp значение следующего элемента

Блок 20 – конец цикла while

Блок 21 – установить переменной current значение следующего элемента

Блок 22 – конец цикла while

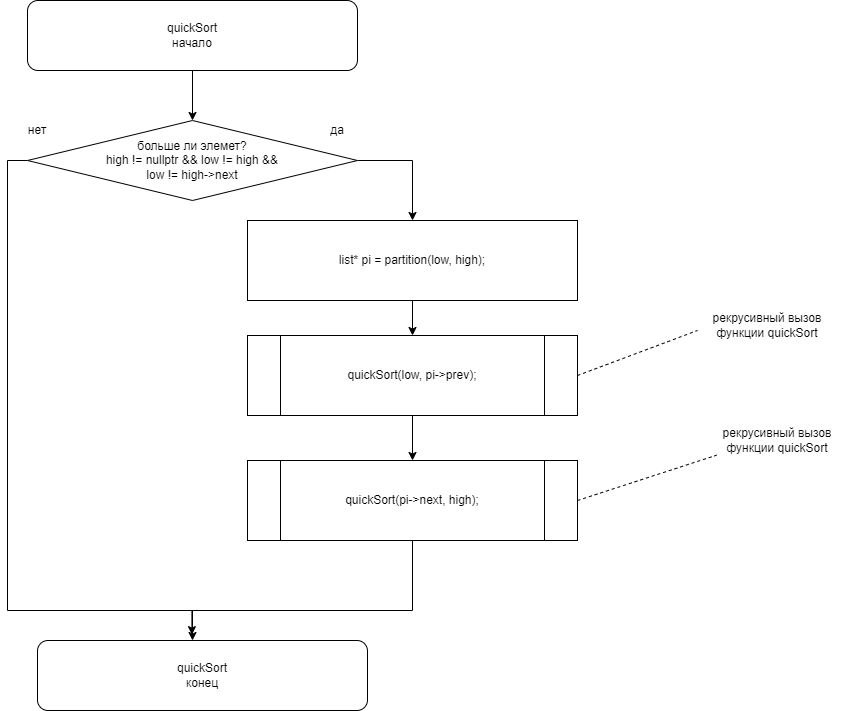


Рисунок 2.11 - Схема алгоритма функции быстрой сортировки

Блок 1 – инициализация булевой переменой exit\_requested со значением false, переменной printed\_elements со значением 0 и current\_page со значением 1

Блок 2 – проверка пуст ли список

Блок 3 – если да, преждевременный выход из функции с помощью return

Блок 4 – если нет, вывод информации об управление

Блок 5 – начало цикла while с условием да тех пор, пока exit\_requested не приймет значение true

Блок 6 – инициализация и считывание переменной key

Блок 7 – проверка на о был ли введен +

Блок 8 – если да, проверка меньше ли текущая страница чем максимальная

Блок 9 – если да, current\_page увеличиваем, printed element устанавливаем значение 0

Блок 10 – вызов функции print

Блок 11 – если нет, проверить был ли введен ­-

Блок 12 – если да, проверить меньше ли 1 текущая страница

Блок 13 – если да current\_page уменьшить, printed\_elements установить значение 0

Блок 14 – если нет, проверить был ли введен 0

Блок 15 – если да, установить переменной exit\_requested значение true

Блок 16 – конец цикла while

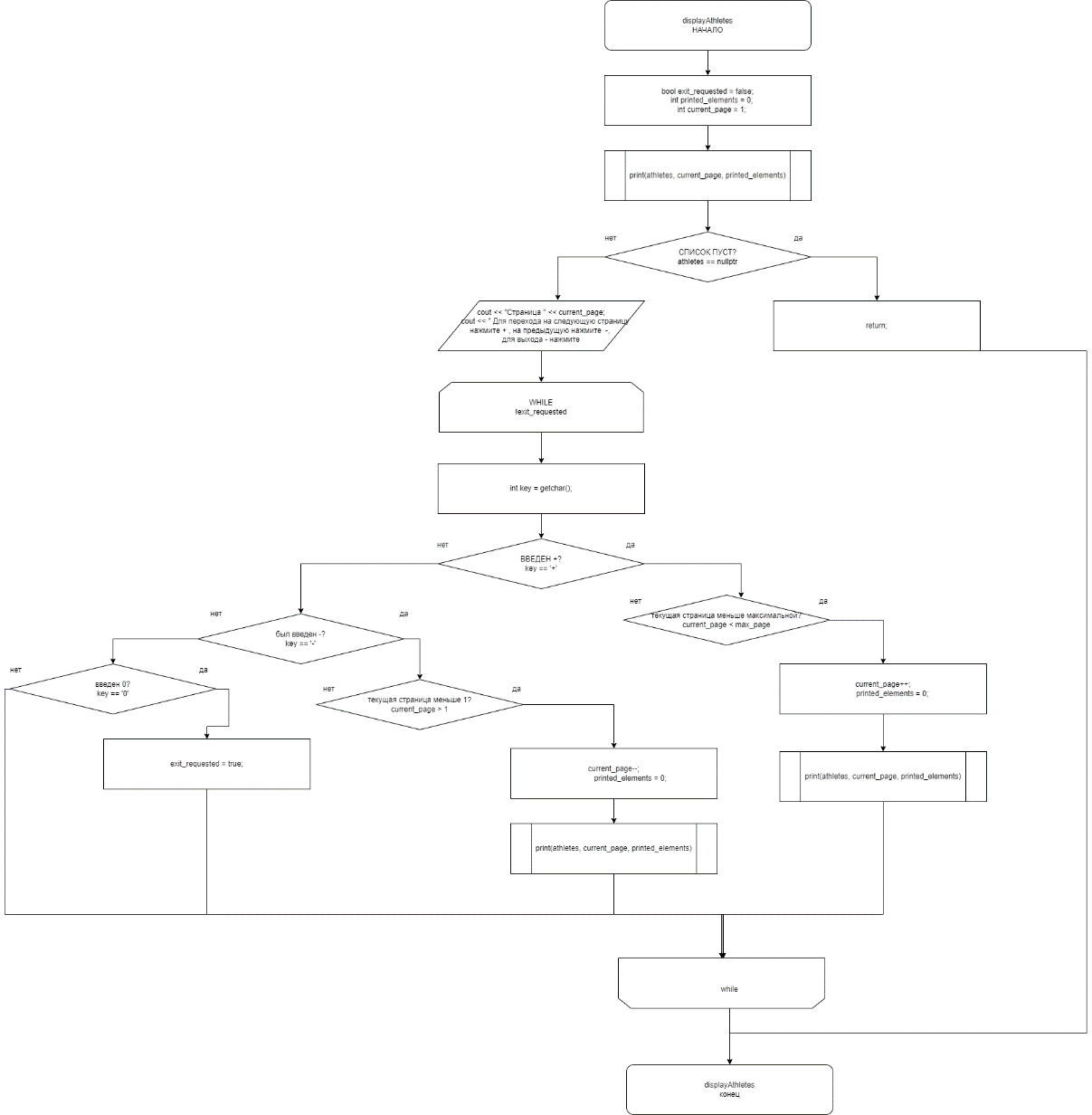


Рисунок 2.12 - Схема алгоритма функции скроллинга

# **ВЫПОЛНЕНИЕ ПРОГРАММЫ**

## **3.1 Условия выполнения программы**

В результате компиляции программы был получен исполняемый файл с расширением EXE размером 15364 Кбайт.

Для выполнения программы необходим минимальный состав программных и аппаратных средств:

-монитор;

-клавиатура;

-центральный процессор, с производительностью от 500 МГц;

-видеокарта, с объемом памяти от 128Мб.

## **3.2 Загрузка и запуск программы**

Запуск программы производится с любого носителя открытием файла firstDef.exe.

## **3.3 Проверка работоспособности программы**

Управление меню осуществляется с помощью стрелок вверх и вниз. Для выбора пункта меню необходимо нажать ENTER. Для выхода с программы ESC.

Меню представлено на рисунке 3.1.

****

Рисунок 3.1 – «Меню программы»

Чтобы открыть список нужно выбрать пункт «Display the list». Реализацию чего предоставлена на рисунке 3.2.

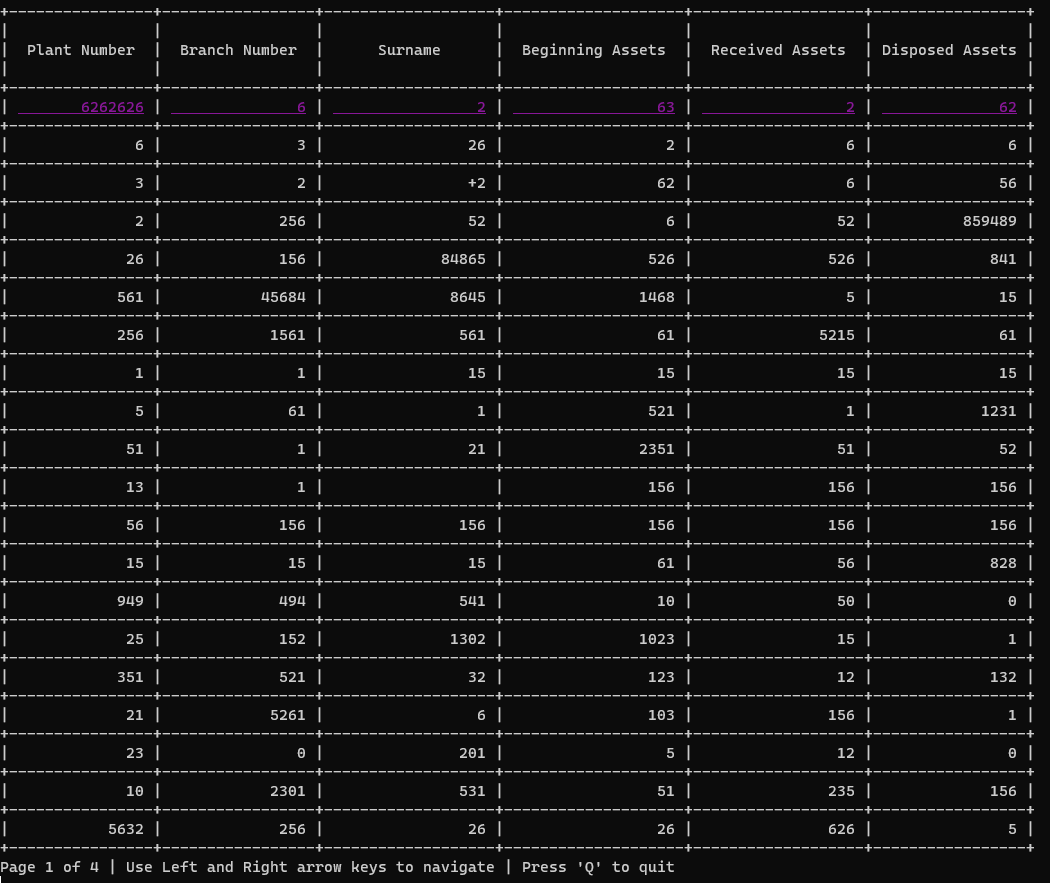
****

Рисунок 3.2 – «Считанный c файла список»

Навигация по списку осуществляется с помощью клавиш PageUp и PageDown. Переход на следующую страницу предоставлен на рисунке 3.4.

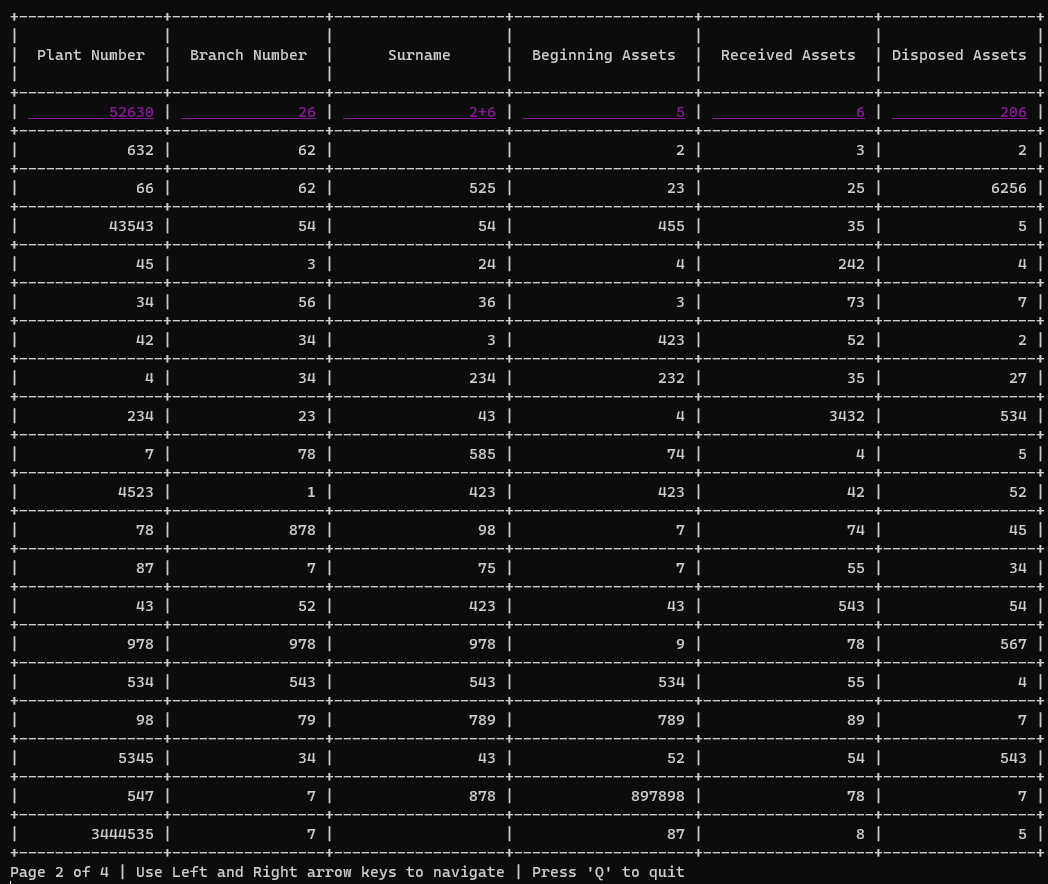
****

Рисунок 3.4 – «Вторая страница списка»

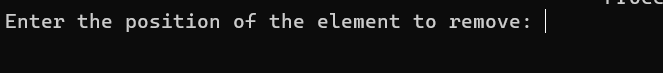


Рисунок 3.5 –«Удаление элемента списка»

Для редактирования элемента списка нужно выбрать пункт «Rewrite the field» или нажать Enter при перелистывании списка. После чего появится меню выбора элемента для редактирования, которое мы видим на рисунке 3.6.



Рисунок 3.6 –«Меню корректировки поля»

Для сортировки нужно выбрать пункт «Sort the list». Меню сортировки и её результат можем наблюдать на рисунке 3.7. Сортировка происходит с помощью bubble sort.

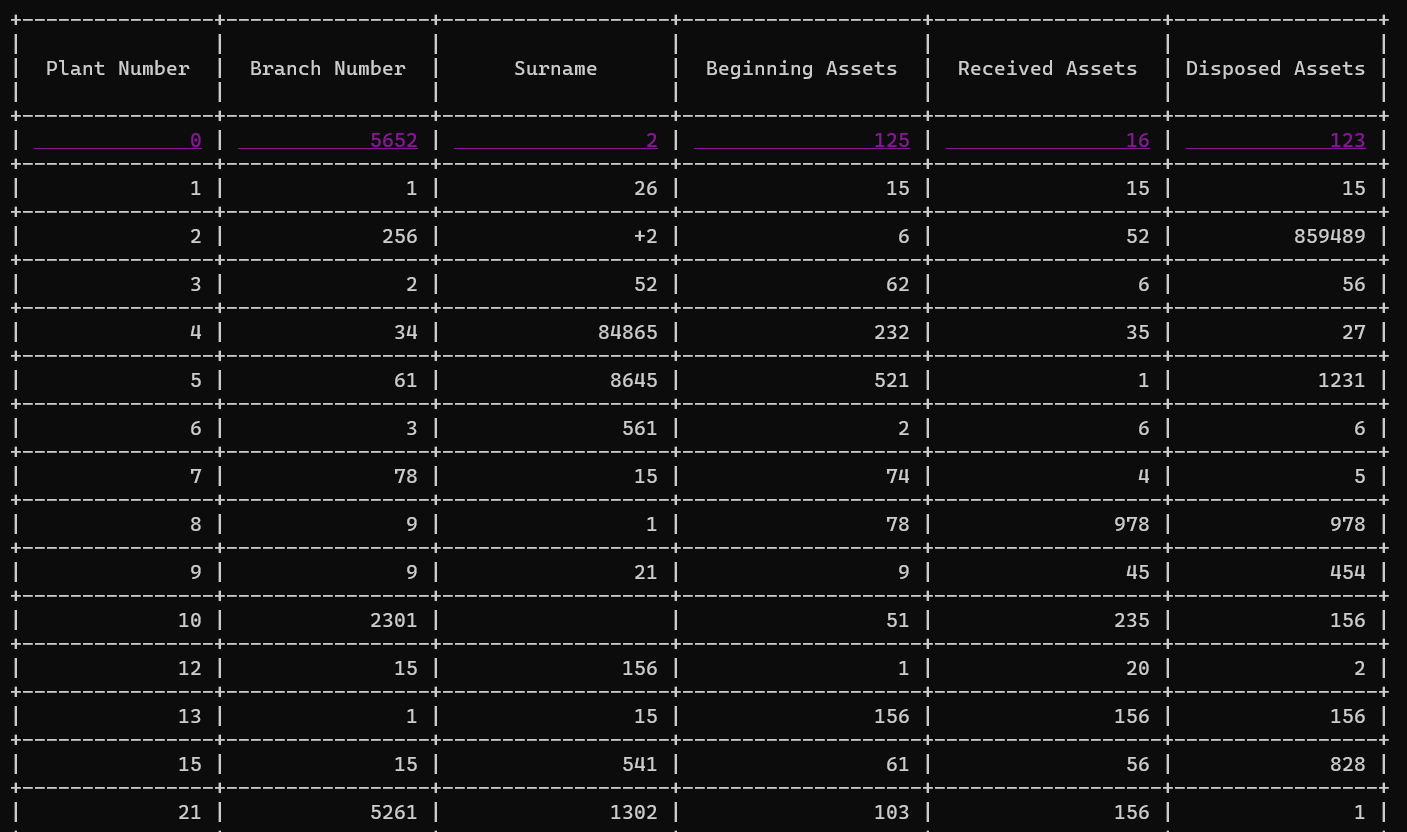


Рисунок 3.7 – «Результат сортировки сортировки»

Для поиска элемента нужно выбрать пункт «Find node». И ввести название продукта, по которому будет производиться поиск, который мы видим на рисунке 3.8.

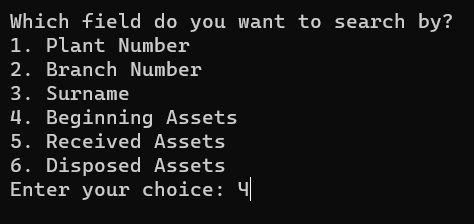


Рисунок 3.8 – «Меню поиска элемента»

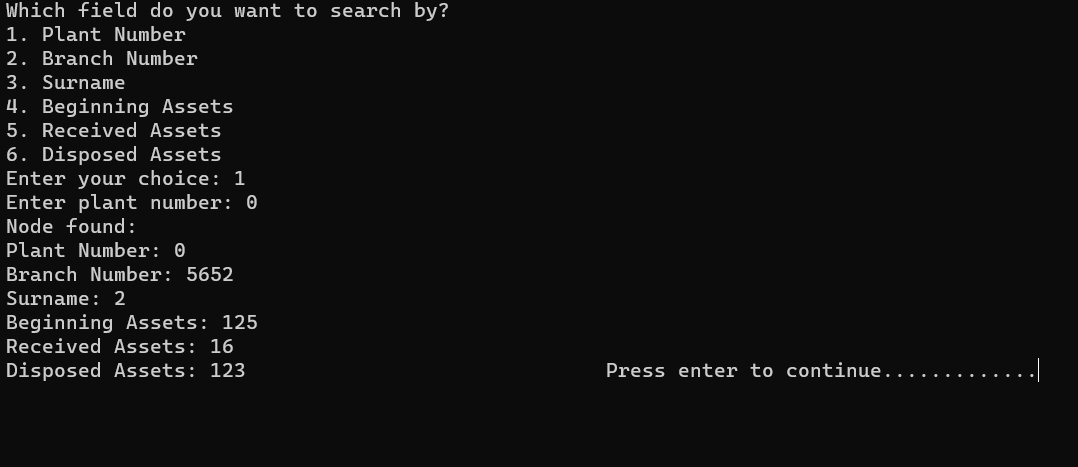


Рисунок 3.9 – «Результат поиска»

Чтобы добавить элемент, нужно выбрать пункт «Add an element», реализацию которого изображена на рисунке 3.10.

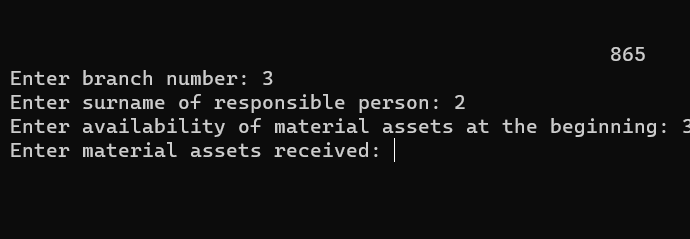


Рисунок 3.10 – Ввод данных

При вводе неуникального поля выводится ошибка, представленная на рисунке 3.11:

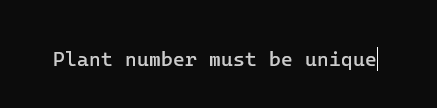


Рисунок 3.11 – Проверка при вводе номера фабрики

Для сохранения в текстовый файл нужно выбрать пункт «Save the database». (Рисунок 3.12)

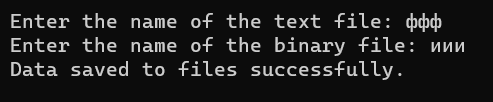


Рисунок 3.12 – Сохранение в типизированный и текстовые файлы

После этого можно просмотреть данные в любом текстовом редакторе, мы можем видеть на рисунке 3.13

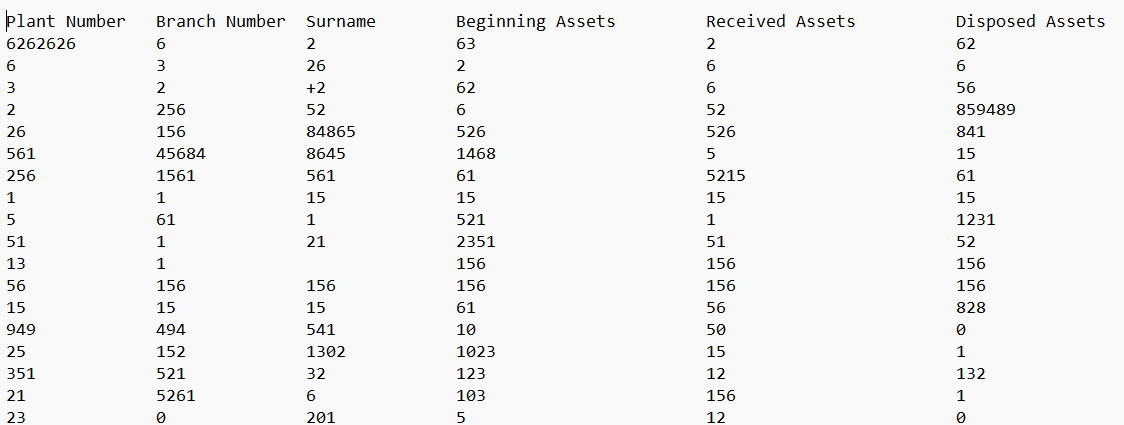


Рисунок 3.13 – Текстовыйфайл

# **ЗАКЛЮЧЕНИЕ**

Программа была составлена в соответствии с постановкой задачи на курсовое проектирование по дисциплине «Информатика и программирование».

Программа работает в виде интерактивного меню, что минимизирует сложности её использования для пользователя. В том числе, программа сама определяет, работать с текстовым или типизированным файлом, исходя из того считывался ли с этого текстового файла или нет. Проведены тестовые примеры для подтверждения работоспособности программы. Тестирование подтвердило, что программа работает корректно, верно выводит и подсчитывает значения, введенные пользователем. Пользователь может работать со списком, а именно добавлять/удалять элементы, редактировать их и получать информацию о просроченном товаре и размере ущерба.

Тестирование подтвердило, что программа работает корректно и готова к применению на производстве.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Алгоритмы и программы. Язык С++ [Электронный ресурс] / Конова Е. А., Поллак Г. А.– 3-е изд., стер. – : Лань, 2018 .– 384 с. Книга из коллекции Лань - Информатика .– ISBN 978-5-8114-2020-9 . – URL: https://e.lanbook.com/book/103905

2. Алгоритмы и программы. Язык С++ [Электронный ресурс]: учебное пособие / Конова Е. А., Поллак Г. А. – 4-е изд., стер. – Санкт-Петербург: Лань, 2019 .– 384 с. Книга из коллекции Лань - Информатика .– ISBN 978-5-8114-4039-9 .– URL: https://e.lanbook.com/book/114696

3. Искусство алгоритмизации [Электронный ресурс] / В. В. Потопахин. – Москва: ДМК Пресс, 2011 . – 320 с. : ил. + CD . – . – Библиогр. : с. 319. ISBN 978-5-94074-621-8

4. Язык Си: введение для программистов: пер. с англ. / Р. Берри, Б. Микинз .– М. : Финансы и статистика, 1988 .– 191 с .– ISBN 5-279-00280-1.

5. Программирование на языке Си: Учеб. пособие для студентов вузов, обучающихся по направлениям: "Прикладная математика и информатика", "Информатика и вычислительная техника". / В.В. Подбельский, С.С. Фомин. – 2-е изд., доп. – М.: Финансы и статистика, 2003. – 600с. – Библиогр.:с.578 .– ISBN 5-279-02180-6.

6. Ефимова, И.Ю. Лабораторный практикум по структурному программированию: учебное пособие / И.Ю. Ефимова, Т.Н. Варфоломеева. – 3-е изд., стер. – Москва: ФЛИНТА, 2019. – 112 с. – ISBN 978-5-9765-2041-7. – Текст: электронный // Электронно-библиотечная система «Лань»: [сайт]. – URL: https://e.lanbook.com/book/125416

7. ГОСТ 2.105-95 Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам (с Изменением N 1, с Поправками), ГОСТ от 08 августа 1995 года №2.105-95. – URL: http://docs.cntd.ru/document/1200001260

8. Методические указания к курсовому проектированию по дисциплине «Информатика и программирование» для студентов дневной и заочной форм обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика»

# **ПРИЛОЖЕНИЕ А**

**Код программы**

**#include <iostream>**

**#include <conio.h>**

**#include <limits>**

**#include <iomanip>**

**#include <fstream>**

**#include <windows.h>**

**using namespace std;**

**struct Node**

**{**

**int plantNumber;**

**int branchNumber;**

**char surname[50];**

**int beginningAssets;**

**int receivedAssets;**

**int disposedAssets;**

**Node \*prev;**

**Node \*next;**

**};**

**HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);**

**void goToXY(short x, short y)**

**{**

**SetConsoleCursorPosition(hStdOut, {x, y});**

**}**

**// Function to create a new doubly linked list**

**int checkIfPlantNumberIsUnique(Node \*head, int plantNumber)**

**{**

**while (head)**

**{**

**if (head->plantNumber == plantNumber)**

**{**

**return 0;**

**}**

**head = head->next;**

**}**

**return 1;**

**}**

**Node \*createDoublyLinkedList()**

**{**

**Node \*head = NULL;**

**Node \*temp = NULL;**

**Node \*newNode = NULL;**

**system("cls");**

**int choice;**

**do**

**{**

**system("cls");**

**goToXY(50, 15);**

**newNode = new Node();**

**cout << "Enter plant number: ";**

**while (!(cin >> newNode->plantNumber))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value. Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**while (!checkIfPlantNumberIsUnique(head, newNode->plantNumber))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Plant number must be unique";**

**cin.clear();**

**\_getch();**

**system("cls");**

**goToXY(50, 15);**

**while (!(cin >> newNode->plantNumber))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value. Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**}**

**system("cls");**

**goToXY(50, 15);**

**cout << "Enter branch number: ";**

**while (!(cin >> newNode->branchNumber))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value! Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**system("cls");**

**goToXY(50, 15);**

**cout << "Enter surname of responsible person: ";**

**cin.ignore();**

**cin.getline(newNode->surname, 50);**

**system("cls");**

**goToXY(50, 15);**

**cout << "Enter availability of material assets at the beginning: ";**

**while (!(cin >> newNode->beginningAssets))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value! Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**system("cls");**

**goToXY(50, 15);**

**cout << "Enter material assets received: ";**

**while (!(cin >> newNode->receivedAssets))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value! Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**system("cls");**

**goToXY(50, 15);**

**cout << "Enter material assets disposed: ";**

**while (!(cin >> newNode->disposedAssets))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value! Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**newNode->prev = NULL;**

**newNode->next = NULL;**

**if (head == NULL)**

**{**

**head = newNode;**

**temp = newNode;**

**}**

**else**

**{**

**temp->next = newNode;**

**newNode->prev = temp;**

**temp = newNode;**

**}**

**goToXY(50, 15);**

**cout << "Do you want to add another element? (1 for Yes, 0 for No): ";**

**cin >> choice;**

**} while (choice != 0);**

**system("cls");**

**return head;**

**}**

**void updateField(Node \*&head, int plantNumber)**

**{**

**system("cls");**

**bool found = false;**

**Node \*currNode = head;**

**while (currNode != nullptr)**

**{**

**if (currNode->plantNumber == plantNumber)**

**{**

**found = true;**

**int fieldChoice;**

**goToXY(50, 15);**

**cout << "Enter the field number you want to update:\n"**

**<< "1. Plant Number\n"**

**<< "2. Branch Number\n"**

**<< "3. Surname\n"**

**<< "4. Beginning Assets\n"**

**<< "5. Received Assets\n"**

**<< "6. Disposed Assets\n";**

**cin >> fieldChoice;**

**switch (fieldChoice)**

**{**

**case 1:**

**cout << "Enter new Plant Number: ";**

**cin >> currNode->plantNumber;**

**break;**

**case 2:**

**cout << "Enter new Branch Number: ";**

**cin >> currNode->branchNumber;**

**break;**

**case 3:**

**cout << "Enter new Surname: ";**

**cin.ignore(); // Ignore newline character**

**cin.getline(currNode->surname, 49);**

**break;**

**case 4:**

**cout << "Enter new Beginning Assets: ";**

**cin >> currNode->beginningAssets;**

**break;**

**case 5:**

**cout << "Enter new Received Assets: ";**

**cin >> currNode->receivedAssets;**

**break;**

**case 6:**

**cout << "Enter new Disposed Assets: ";**

**cin >> currNode->disposedAssets;**

**break;**

**default:**

**cout << "Invalid field number!";**

**break;**

**}**

**break;**

**}**

**currNode = currNode->next;**

**}**

**if (!found)**

**{**

**char choice;**

**cout << "Plant number is not found" << endl;**

**}**

**}**

**void displayPage(Node \*node, Node \*hNode, int items, int page, int totalPages)**

**{**

**Node \*current = node;**

**int itemsDisplayed = 0;**

**// display the head of a table**

**cout << "+----------------+-----------------+-------------------+--------------------+-------------------+-----------------+" << endl;**

**cout << "| | | | | | |" << endl;**

**cout << "| Plant Number | Branch Number | Surname | Beginning Assets | Received Assets | Disposed Assets |" << endl;**

**cout << "| | | | | | |" << endl;**

**// display the core of a list**

**while (current != nullptr && itemsDisplayed != items)**

**{**

**if (current->plantNumber != hNode->plantNumber)**

**{**

**cout << "+----------------+-----------------+-------------------+--------------------+-------------------+-----------------+" << endl;**

**cout << "| " << setw(14) << current->plantNumber**

**<< " | " << setw(15) << current->branchNumber**

**<< " | " << setw(17) << current->surname**

**<< " | " << setw(18) << current->beginningAssets**

**<< " | " << setw(17) << current->receivedAssets**

**<< " | " << setw(15) << current->disposedAssets << " |" << endl;**

**}**

**else**

**{**

**cout << "+----------------+-----------------+-------------------+--------------------+-------------------+-----------------+" << endl;**

**cout << "| "**

**<< "\e[4;35m" << setw(14) << current->plantNumber << "\033[0m"**

**<< " | "**

**<< "\e[4;35m" << setw(15) << current->branchNumber << "\033[0m"**

**<< " | "**

**<< "\e[4;35m" << setw(17) << current->surname << "\033[0m"**

**<< " | "**

**<< "\e[4;35m" << setw(18) << current->beginningAssets << "\033[0m"**

**<< " | "**

**<< "\e[4;35m" << setw(17) << current->receivedAssets << "\033[0m"**

**<< " | "**

**<< "\e[4;35m" << setw(15) << current->disposedAssets << "\033[0m"**

**<< " |" << endl;**

**}**

**current = current->next;**

**itemsDisplayed++;**

**}**

**cout << "+----------------+-----------------+-------------------+--------------------+-------------------+-----------------+" << endl;**

**// Display navigation instructions**

**cout << "Page " << page << " of " << totalPages << " | Use Left and Right arrow keys to navigate | Press 'Q' to quit" << endl;**

**}**

**void displayDoublyLinkedList(Node \*head)**

**{**

**system("cls");**

**if (head == NULL)**

**{**

**cout << "Doubly linked list is empty!" << endl;**

**return;**

**}**

**int page = 1;**

**int nodesPerPage = 20;**

**int highlightNum = 1;**

**Node \*highlightedNode = head;**

**Node \*temp = head;**

**Node \*itemsPtr = head;**

**int totalNodes = 0;**

**while (temp != nullptr)**

**{**

**totalNodes++;**

**temp = temp->next;**

**}**

**int totalPages = (totalNodes + nodesPerPage - 1) / nodesPerPage;**

**temp = head;**

**displayPage(itemsPtr, highlightedNode, nodesPerPage, page, totalPages);**

**while (1)**

**{**

**// getting arrow key code**

**char input = \_getch();**

**if (input == -32)**

**{**

**input = \_getch();**

**switch (input)**

**{**

**case 75:**

**if (page != max(1, page - 1))**

**{**

**page = max(1, page - 1);**

**for (int i = 0; i < nodesPerPage; i++)**

**{**

**itemsPtr = itemsPtr->prev;**

**}**

**highlightNum = 1;**

**highlightedNode = itemsPtr;**

**system("cls");**

**displayPage(itemsPtr, highlightedNode, nodesPerPage, page, totalPages);**

**}**

**break;**

**case 77:**

**if (page != min(totalPages, page + 1))**

**{**

**page = min(totalPages, page + 1);**

**for (int i = 0; i < nodesPerPage; i++)**

**{**

**itemsPtr = itemsPtr->next;**

**}**

**highlightNum = 1;**

**highlightedNode = itemsPtr;**

**system("cls");**

**displayPage(itemsPtr, highlightedNode, nodesPerPage, page, totalPages);**

**}**

**break;**

**case 72:**

**if (highlightNum > 1)**

**{**

**highlightedNode = highlightedNode->prev;**

**highlightNum--;**

**system("cls");**

**displayPage(itemsPtr, highlightedNode, nodesPerPage, page, totalPages);**

**}**

**break;**

**case 80:**

**if (highlightNum + 1 <= nodesPerPage)**

**{**

**if (highlightedNode->next != nullptr)**

**{**

**highlightedNode = highlightedNode->next;**

**highlightNum++;**

**system("cls");**

**displayPage(itemsPtr, highlightedNode, nodesPerPage, page, totalPages);**

**}**

**}**

**break;**

**}**

**}**

**else if (input == 13)**

**{**

**updateField(head, highlightedNode->plantNumber);**

**system("cls");**

**displayPage(itemsPtr, highlightedNode, nodesPerPage, page, totalPages);**

**}**

**else**

**{**

**break;**

**}**

**}**

**}**

**/\*Function to swap the nodes \*/**

**struct Node \*swap(struct Node \*ptr1, struct Node \*ptr2)**

**{**

**struct Node \*tmp = ptr2->next;**

**ptr2->next = ptr1;**

**ptr1->next = tmp;**

**return ptr2;**

**}**

**void bubbleSort(Node \*\*head)**

**{**

**if (\*head == nullptr || (\*head)->next == nullptr)**

**{**

**cout << "Cannot sort" << endl;**

**return;**

**}**

**bool swapped;**

**Node \*current;**

**Node \*lastSorted = nullptr;**

**do**

**{**

**swapped = false;**

**current = \*head;**

**while (current->next != lastSorted)**

**{**

**if (current->plantNumber > current->next->plantNumber)**

**{**

**std::swap(current->plantNumber, current->next->plantNumber);**

**std::swap(current->beginningAssets, current->next->beginningAssets);**

**std::swap(current->branchNumber, current->next->branchNumber);**

**std::swap(current->disposedAssets, current->next->disposedAssets);**

**std::swap(current->receivedAssets, current->next->receivedAssets);**

**swapped = true;**

**}**

**current = current->next;**

**}**

**lastSorted = current;**

**} while (swapped);**

**cout << "Swapping succeeded" << endl;**

**}**

**void addElementToEnd(Node \*head)**

**{**

**system("cls");**

**if (head == NULL)**

**{**

**cout << "Doubly linked list is not created. Please create a list first." << endl;**

**return;**

**}**

**Node \*newNode = new Node();**

**cout << "Enter plant number: ";**

**while (!(cin >> newNode->plantNumber))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value. Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**while (!checkIfPlantNumberIsUnique(head, newNode->plantNumber))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Plant number must be unique";**

**cin.clear();**

**\_getch();**

**system("cls");**

**goToXY(50, 15);**

**while (!(cin >> newNode->plantNumber))**

**{**

**system("cls");**

**goToXY(50, 15);**

**cout << "Bad value. Enter to rewrite.";**

**getch();**

**system("cls");**

**goToXY(50, 15);**

**cin.clear();**

**goToXY(50, 15);**

**cin.ignore(numeric\_limits<streamsize>::max(), '\n');**

**}**

**}**

**cout << "Enter branch number: ";**

**cin >> newNode->branchNumber;**

**cout << "Enter surname of responsible person: ";**

**cin.ignore();**

**cin.getline(newNode->surname, 50);**

**cout << "Enter availability of material assets at the beginning: ";**

**cin >> newNode->beginningAssets;**

**cout << "Enter material assets received: ";**

**cin >> newNode->receivedAssets;**

**cout << "Enter material assets disposed: ";**

**cin >> newNode->disposedAssets;**

**newNode->prev = NULL;**

**newNode->next = NULL;**

**Node \*temp = head;**

**while (temp->next != NULL)**

**{**

**temp = temp->next;**

**}**

**temp->next = newNode;**

**newNode->prev = temp;**

**}**

**// Function to remove any element from the doubly linked list**

**void removeElement(Node \*&head, Node \*elem)**

**{**

**system("cls");**

**if (head == NULL)**

**{**

**cout << "Doubly linked list is empty!" << endl;**

**return;**

**}**

**if (elem->prev != NULL)**

**{**

**elem->prev->next = elem->next;**

**}**

**else**

**{**

**head = elem->next;**

**}**

**if (elem->next != NULL)**

**{**

**elem->next->prev = elem->prev;**

**}**

**delete elem;**

**}**

**// Function to highlight the selected menu item**

**void highlightMenuItem(const string &menuItem, bool isSelected)**

**{**

**if (isSelected)**

**{**

**cout << "\e[4;35m" << menuItem << "\033[0m" << endl;**

**}**

**else**

**{**

**cout << menuItem << endl;**

**}**

**}**

**void searchNode(Node \*head)**

**{**

**system("cls");**

**if (head == NULL)**

**{**

**cout << "Doubly linked list is empty!" << endl;**

**return;**

**}**

**int fieldChoice;**

**cout << "Which field do you want to search by?" << endl;**

**cout << "1. Plant Number" << endl;**

**cout << "2. Branch Number" << endl;**

**cout << "3. Surname" << endl;**

**cout << "4. Beginning Assets" << endl;**

**cout << "5. Received Assets" << endl;**

**cout << "6. Disposed Assets" << endl;**

**cout << "Enter your choice: ";**

**cin >> fieldChoice;**

**Node \*current = head;**

**bool found = false;**

**switch (fieldChoice)**

**{**

**case 1:**

**{**

**int plantNumber;**

**cout << "Enter plant number: ";**

**cin >> plantNumber;**

**while (current != nullptr)**

**{**

**if (current->plantNumber == plantNumber)**

**{**

**found = true;**

**break;**

**}**

**current = current->next;**

**}**

**break;**

**}**

**case 2:**

**{**

**int branchNumber;**

**cout << "Enter branch number: ";**

**cin >> branchNumber;**

**while (current != nullptr)**

**{**

**if (current->branchNumber == branchNumber)**

**{**

**found = true;**

**break;**

**}**

**current = current->next;**

**}**

**break;**

**}**

**case 3:**

**{**

**char surname[50];**

**cout << "Enter surname: ";**

**cin.ignore(); // Ignore newline character left in the stream**

**cin.getline(surname, sizeof(surname));**

**while (current != nullptr)**

**{**

**bool match = true;**

**for (int i = 0; i < sizeof(current->surname); i++)**

**{**

**if (current->surname[i] != surname[i])**

**{**

**match = false;**

**break;**

**}**

**else if (current->surname[i] == '\0' && surname[i] == '\0')**

**{**

**break;**

**}**

**}**

**if (match)**

**{**

**found = true;**

**break;**

**}**

**current = current->next;**

**}**

**break;**

**}**

**case 4:**

**{**

**int beginningAssets;**

**cout << "Enter beginning assets: ";**

**cin >> beginningAssets;**

**while (current != nullptr)**

**{**

**if (current->beginningAssets == beginningAssets)**

**{**

**found = true;**

**break;**

**}**

**current = current->next;**

**}**

**break;**

**}**

**case 5:**

**{**

**int receivedAssets;**

**cout << "Enter received assets: ";**

**cin >> receivedAssets;**

**while (current != nullptr)**

**{**

**if (current->receivedAssets == receivedAssets)**

**{**

**found = true;**

**break;**

**}**

**current = current->next;**

**}**

**break;**

**}**

**case 6:**

**{**

**int disposedAssets;**

**cout << "Enter disposed assets: ";**

**cin >> disposedAssets;**

**while (current != nullptr)**

**{**

**if (current->disposedAssets == disposedAssets)**

**{**

**found = true;**

**break;**

**}**

**current = current->next;**

**}**

**break;**

**}**

**default:**

**cout << "Invalid choice." << endl;**

**}**

**if (found)**

**{**

**cout << "Node found:" << endl;**

**cout << "Plant Number: " << current->plantNumber << endl;**

**cout << "Branch Number: " << current->branchNumber << endl;**

**cout << "Surname: " << current->surname << endl;**

**cout << "Beginning Assets: " << current->beginningAssets << endl;**

**cout << "Received Assets: " << current->receivedAssets << endl;**

**cout << "Disposed Assets: " << current->disposedAssets << endl;**

**}**

**else**

**{**

**cout << "No node with the specified field exists." << endl;**

**}**

**}**

**// Function to calculate the length of a null-terminated char array**

**int getCharArrayLength(const char \*arr)**

**{**

**int count = 0;**

**while (arr[count] != '\0')**

**{**

**count++;**

**}**

**return count;**

**}**

**// Function to copy a null-terminated char array**

**void copyCharArray(const char \*source, char \*destination)**

**{**

**int i = 0;**

**while (source[i] != '\0')**

**{**

**destination[i] = source[i];**

**i++;**

**}**

**destination[i] = '\0';**

**}**

**// Function to concatenate two null-terminated char arrays**

**void concatenateCharArray(const char \*str1, const char \*str2, char \*result)**

**{**

**int len1 = getCharArrayLength(str1);**

**int len2 = getCharArrayLength(str2);**

**int i = 0;**

**for (int j = 0; j < len1; j++)**

**{**

**result[i++] = str1[j];**

**}**

**for (int j = 0; j < len2; j++)**

**{**

**result[i++] = str2[j];**

**}**

**result[i] = '\0';**

**}**

**// Function to save the doubly linked list to text and binary files**

**void saveLinkedListToFile(Node \*head)**

**{**

**system("cls");**

**const int MAX\_FILENAME\_LENGTH = 100;**

**char textFileName[MAX\_FILENAME\_LENGTH];**

**char binaryFileName[MAX\_FILENAME\_LENGTH];**

**cout << "Enter the name of the text file: ";**

**cin.getline(textFileName, MAX\_FILENAME\_LENGTH);**

**concatenateCharArray(textFileName, ".txt", textFileName);**

**cout << "Enter the name of the binary file: ";**

**cin.getline(binaryFileName, MAX\_FILENAME\_LENGTH);**

**concatenateCharArray(binaryFileName, ".bin", binaryFileName);**

**ofstream textFile(textFileName);**

**ofstream binaryFile(binaryFileName, ios::binary);**

**if (!textFile || !binaryFile)**

**{**

**cout << "Failed to create/open output files!" << endl;**

**return;**

**}**

**textFile << left;**

**textFile << setw(15) << "Plant Number";**

**textFile << setw(15) << "Branch Number";**

**textFile << setw(15) << "Surname";**

**textFile << setw(25) << "Beginning Assets";**

**textFile << setw(25) << "Received Assets";**

**textFile << setw(25) << "Disposed Assets";**

**textFile << endl;**

**Node \*current = head;**

**while (current != nullptr)**

**{**

**textFile << left << setw(15) << current->plantNumber**

**<< setw(15) << current->branchNumber**

**<< setw(15) << current->surname**

**<< setw(25) << current->beginningAssets**

**<< setw(25) << current->receivedAssets**

**<< setw(25) << current->disposedAssets << endl;**

**binaryFile.write(reinterpret\_cast<char \*>(current), sizeof(Node));**

**current = current->next;**

**}**

**textFile.close();**

**binaryFile.close();**

**cout << "Data saved to files successfully." << endl;**

**}**

**//-----------------------------------------------------------------------------------READ**

**void refreshList(Node \*&head)**

**{**

**system("cls");**

**const int MAX\_FILENAME\_LENGTH = 100;**

**char binaryFileName[MAX\_FILENAME\_LENGTH];**

**cout << "Enter the name of the binary file to refresh from: ";**

**cin.getline(binaryFileName, MAX\_FILENAME\_LENGTH);**

**concatenateCharArray(binaryFileName, ".bin", binaryFileName);**

**ifstream binaryFile(binaryFileName, ios::binary);**

**if (!binaryFile)**

**{**

**cout << "Failed to open the binary file!" << endl;**

**return;**

**}**

**while (head != nullptr)**

**{**

**Node \*temp = head;**

**head = head->next;**

**delete temp;**

**}**

**Node \*current = nullptr;**

**while (true)**

**{**

**Node \*newNode = new Node();**

**binaryFile.read(reinterpret\_cast<char \*>(newNode), sizeof(Node));**

**if (binaryFile.eof())**

**{**

**delete newNode;**

**break;**

**}**

**newNode->prev = current;**

**newNode->next = nullptr;**

**if (current)**

**{**

**current->next = newNode;**

**}**

**else**

**{**

**head = newNode;**

**}**

**current = newNode;**

**}**

**binaryFile.close();**

**cout << "List refreshed from the binary file successfully." << endl;**

**}**

**//-----------------------------------------PROCEED**

**void handleData(Node \*head)**

**{**

**system("cls");**

**int branchNumber;**

**Node \*temp = head;**

**cout << "Enter the branch number you want to proceed: ";**

**cin >> branchNumber;**

**while (temp != nullptr)**

**{**

**if (branchNumber != temp->branchNumber)**

**{**

**temp = temp->next;**

**}**

**else**

**{**

**break;**

**}**

**if (temp->next == nullptr)**

**{**

**cout << "There's no such branch. Try again" << endl;**

**return;**

**}**

**}**

**int sum1 = 0; // Sum of beginning assets, received assets, and disposed assets in a particular branch**

**int sum2 = 0; // Sum of beginning assets, received assets, and disposed assets across all branches**

**int sum3 = 0; // Sum of beginning assets across all nodes**

**int sum4 = 0; // Sum of received assets across all nodes**

**int sum5 = 0; // Sum of disposed assets across all nodes**

**Node \*current = head;**

**while (current != nullptr)**

**{**

**sum2 += (current->beginningAssets + current->receivedAssets) - current->disposedAssets;**

**if (current->branchNumber == branchNumber)**

**{**

**sum1 += (current->beginningAssets + current->receivedAssets) - current->disposedAssets;**

**}**

**sum3 += current->beginningAssets;**

**sum4 += current->receivedAssets;**

**sum5 += current->disposedAssets;**

**current = current->next;**

**}**

**cout << "Sum of beginning assets, received assets, and disposed assets in branch " << branchNumber << " = " << sum1 << endl;**

**cout << "Sum of assets along all branches = " << sum2 << endl;**

**cout << "Sum of assets along factory, including only beginning assets = " << sum3 << endl;**

**cout << "Sum of assets along factory, including only received assets = " << sum4 << endl;**

**cout << "Sum of assets along factory, including only disposed assets = " << sum5 << endl;**

**}**

**int main()**

**{**

**Node \*doublyLinkedList = NULL;**

**int plantNumber;**

**int choice = 0;**

**int menuIndex = 0;**

**char keyPressed;**

**bool shouldExit = false;**

**while (!shouldExit)**

**{**

**system("cls"); // Clear console screen**

**goToXY(48, 8);**

**cout << " ...::::::.," << endl;**

**goToXY(48, 9);**

**cout << " .::::::::::::::." << endl;**

**goToXY(48, 10);**

**cout << " .::.. .::::::::::." << endl;**

**goToXY(48, 11);**

**cout << " :::::. .:.::::. " << endl;**

**goToXY(48, 12);**

**cout << "::::::::. .:::: " << endl;**

**goToXY(48, 13);**

**cout << "::::::::::. .::: " << endl;**

**goToXY(48, 14);**

**cout << ".::::::::. .:::::: " << endl;**

**goToXY(48, 15);**

**cout << " :::::::: .:::::::: " << endl;**

**goToXY(48, 16);**

**cout << " ::::::..::::::::::. " << endl;**

**goToXY(48, 17);**

**cout << " .::::.:::::::. " << endl;**

**goToXY(48, 18);**

**cout << " ::.:::: " << endl;**

**goToXY(45, 19);**

**cout << ".----. ----..-. .-. .---. .-. .-." << endl;**

**goToXY(45, 20);**

**cout << "{ {\_\_ | {\_ | | | |: \_\_}| { } |" << endl;**

**goToXY(45, 21);**

**cout << ".-.\_} }| {\_\_ : :\_/ /: {\_ }| {\_} |" << endl;**

**goToXY(45, 22);**

**cout << "`----' `----' `---' `---' `-----'" << endl;**

**goToXY(50, 24);**

**highlightMenuItem("Create list", menuIndex == 0);**

**goToXY(50, 25);**

**highlightMenuItem("Add an element", menuIndex == 1);**

**goToXY(50, 26);**

**highlightMenuItem("Remove any element", menuIndex == 2);**

**goToXY(50, 27);**

**highlightMenuItem("Display the list", menuIndex == 3);**

**goToXY(50, 28);**

**highlightMenuItem("Rewrite the field", menuIndex == 4);**

**goToXY(50, 29);**

**highlightMenuItem("Sort the list", menuIndex == 5);**

**goToXY(50, 30);**

**highlightMenuItem("Find node", menuIndex == 6);**

**goToXY(50, 31);**

**highlightMenuItem("Save the database", menuIndex == 7);**

**goToXY(50, 32);**

**highlightMenuItem("Refresh list", menuIndex == 8);**

**goToXY(50, 33);**

**highlightMenuItem("Proceed data", menuIndex == 9);**

**// Handle up and down arrow key presses**

**keyPressed = \_getch();**

**if (keyPressed == 80)**

**{ // Down arrow key**

**menuIndex = (menuIndex + 1) % 10;**

**}**

**else if (keyPressed == 72)**

**{ // Up arrow key**

**menuIndex = (menuIndex - 1 + 10) % 10;**

**}**

**else if (keyPressed == 13)**

**{ // Enter key**

**switch (menuIndex)**

**{**

**case 0:**

**{**

**doublyLinkedList = createDoublyLinkedList();**

**goToXY(50, 15);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**}**

**case 1:**

**addElementToEnd(doublyLinkedList);**

**goToXY(50, 15);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**case 2:**

**{**

**int position;**

**cout << "Enter the position of the element to remove: ";**

**cin >> position;**

**Node \*temp = doublyLinkedList;**

**while (position > 1 && temp != NULL)**

**{**

**temp = temp->next;**

**position--;**

**}**

**if (temp != NULL)**

**{**

**removeElement(doublyLinkedList, temp);**

**}**

**else**

**{**

**cout << "Invalid position!" << endl;**

**}**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**}**

**case 3:**

**displayDoublyLinkedList(doublyLinkedList);**

**cout << "\n\nPress enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**case 4:**

**cout << "Enter the plant number you want to update: ";**

**cin >> plantNumber;**

**updateField(doublyLinkedList, plantNumber);**

**goToXY(50, 15);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**case 5:**

**bubbleSort(&doublyLinkedList);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**case 6:**

**searchNode(doublyLinkedList);**

**goToXY(50, 15);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**case 7:**

**saveLinkedListToFile(doublyLinkedList);**

**goToXY(50, 15);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**case 8:**

**refreshList(doublyLinkedList);**

**goToXY(50, 15);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**case 9:**

**handleData(doublyLinkedList);**

**goToXY(50, 15);**

**cout << "Press enter to continue.............";**

**cin.ignore();**

**cin.get();**

**break;**

**}**

**}**

**else if (keyPressed == 27)**

**{ // Escape key**

**shouldExit = true;**

**}**

**}**

**return 0;**

**}**