Данный документ предназначен для того, чтобы предоставить возможность ознакомления с разработанной в ходе курсового проекта программой лицам, не принимавшим участия в разработке программы ранее либо же лицам, предполагающим возможность дальнейшей разработки данного программного продукта. Также этот документ предназначен для ознакомления лиц, планирующих использовать разработанную в ходе курсового проекта программу в промышленных целях. Его суть состоит в ознакомлении лиц, не принимавших прямого участия в разработке программы, с программой, её возможностями, функциями, а также с тонкостями реализации и функционирования вышеупомянутой программы и её систем.

**СОДЕРЖАНИЕ**

[**1 ВВЕДЕНИЕ** 1](#_Toc153496187)

[**2 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ** 2](#_Toc153496188)

[**3 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ** 2](#_Toc153496189)

[**4 ВЫПОЛНЕНИЕ ПРОГРАММЫ** 2](#_Toc153496190)

[**5 ЗАКЛЮЧЕНИЕ** 2](#_Toc153496191)

[**6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 2](#_Toc153496192)

[**7 ПРИЛОЖЕНИЕ** 2](#_Toc153496193)

## **1 ВВЕДЕНИЕ**

Целью данного курсового проекта является разработка программы “Обработка табличной информации”, которая сможет обеспечить получение оперативных точных данных о студентах, пропущенных студентами часах, оправданных студентами часах, а также остальных необходимых для пользователя сведений о студентах.

Разработка вышеупомянутой программы ведётся в соответствии с техническим заданием (8 вариант), полученным в организации “Севастопольский государственный университет”, утверждённым шестнадцатого декабря две тысячи двадцать третьего года.

Дефицит кадров испытывают 42% предприятий, показал июльский опрос Института Гайдара. На момент предыдущего опроса в апреле о нехватке работников заявляли 35% компаний. Эта неоспоримо пугающая тенденция продолжает своё неумолимое движение по всему миру. В настоящий момент времени особенно резко во всех странах мира стоит вопрос о наборе, обучении и выпуске квалифицированных кадров. Можно успешно провести корреляцию между посещаемостью студента, его успехами, достижениями, а также оценками, следовательно, и шансом выпуститься из высшего учебного заведения с дипломом. Однако, даже учитывая всё вышесказанное, во многих учебных заведениях до сих пор посещаемость проставляется и контролируется преподавателем, который, при всём своём опыте и при всех своих неоспоримых навыках и знаниях, всё же человек, и может ошибиться. Для решения подобной проблемы некоторые учебные заведения разрабатывают свою систему по контрою посещаемости занятий студентами. Но, как показала практика, далеко не каждое учебное заведение способно самостоятельно разработать программу с необходимым функционалом. Программа, разработанная в ходе данного курсового проекта, призвана помочь справиться с вышеупомянутым вопросом и дать возможность учебным организациям по всему миру контролировать прилежность их студентов или иных обучающихся.

В задачи, которые необходимо решить программой, разработанной в ходе данного курсового проекта, входят задачи хранения, обработки, добавления, редактирования, удаления, а также произведения иных действий с данными, которые были внесены в программу напрямую или же через файл какого-либо типа.

## **2 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ**

Разработанная в ходе данного курсового проекта программы предназначена для использования учебными заведениями, которым необходима помощь в хранении информации о студентах, отслеживании посещаемости студентов данного учебного заведения, загрузкой и выгрузкой разных наборов данных в виде файлов, хранящих информацию о студентах, а также в произведении иных операций над данными о студентах, содержащимися в том или ином виде. Однако данная программа не только предоставляет широкий перечень функциональных возможностей, но и удобный, дружелюбный к пользователю интерфейс, разработанный специально для консольных приложений подобного типа. Также данная программа предоставляет возможность интерактивной работы с меню посредством клавиш на клавиатуре пользователя. Единственным ограничением по использованию данного приложения является возможность его запуска только на операционных системах Windows от 7 и новее.

## **3 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ**

В качестве языка программирования для реализации программы, решающей поставленные задачи, был выбран язык Си. Язык Си был выбран из-за его легковесности, его быстродействия, а также его лёгкости в освоении и применении. Также язык Си предоставляет возможность работать с памятью напрямую, что решает проблему недетерминированности сборщика мусора в таких языках, как C#, Java и других языках, использующих подобный функционал.

В качестве среды программирования была выбрана интегрированная среда разработки под названием “Visual Studio Code”. Эта среда разработки была выбрана из-за практически нескончаемого количества вспомогательных, для разработки программного обеспечения, средств. Также данная среда разработки имеет небольшой, относительно других сред, вес, и, большую, опять же, относительно других сред, свободу в выборе инструментов разработчика.

В качестве компилятора был выбран GCC (Gnu Compiler Collection) из-за того, что при использовании GCC для компиляции кода под разные платформы будет использован один и тот же синтаксический анализатор. Поэтому, если удалось собрать программу для одной из целевых платформ, то велика вероятность, что программа нормально соберётся и для других платформ. Также GCC является свободным программным обеспечением и используется как стандартный компилятор для свободных UNIX-подобных операционных систем.

Рассмотрим следующую задачу. Даны сведения за месяц о пропусках занятий студентами групп. Структура записи: порядковый номер, шифр группы (6 символов), ФИО (30 символов), дата рождения, пол (булевская переменная), пропущено часов, оправдано часов. Описание соответствующей структуры данных о студенте на языке Си представим в виде:

struct student {

int number;

WCHAR group[6];

WCHAR FIO[30];

WCHAR birthday[11];

int gender;

int missedHours;

int justifiedHours;

};

Необходимо, чтобы программа могла выполнять следующие c функции с приведённой выше структурой данных:

а) подсчитать количество неоправданных часов по каждому студенту;

б) подсчитать суммарные показатели по каждому виду часов;

в) подсчитать процент неоправданных пропусков в целом;

г) добавление новых студентов в список;

д) удаление студента из списка;

е) корректировка записи о студенте в списке;

ж) сортировка данных;

з) поиск студентов в списке;

и) сохранение, загрузка студентов из бинарных и стандартных файлов;

к) выход из программы.

Многие из вышеперечисленных функций требуют возможность взаимодействия со студентами, поэтому необходимо разработать структуру, которая будет предоставлять интерфейс для работы со студентами. Эта структура должна иметь репозиторий (хранилище записей о студентах), а также некоторые вспомогательные поля. Представление данной структуры данных в коде:

struct studentService {

int lastId;

int studentsAmount;

int lastFoundStudentsAmount;

struct node\* studentRepository;

};

Также взаимодействие с программой должно строиться посредством интерактивного меню. Исходя из вышесказанного необходимо разработать структуру пользовательского интерфейса, которая будет хранить все входящие в него элементы, а также будет отвечать за взаимодействие с пользователем. Так как взаимодействие пользователя с программой происходит через эту структуру, она должна иметь доступ к функциям работы со студентами. Представление данной структуры на языке Си:

struct UI {

struct studentService\* studentService;

struct menu \*mainMenu;

struct scrollMenu \*scrollMenu;

struct menu \*fileMenu;

struct scrollMenu \*badStudentsMenu;

};

Далее необходимо реализовать стандартное меню, которое будет содержать выбранный пользователем элемент, количество пунктов в меню, координаты начала и конца меню, а также сами пункты меню, это всё понадобится в дальнейшем для реализации функций отображения меню, выбора пользователя, очистки меню и т.д. Описание соответствующей структуры данных на языке Си представим в виде:

struct menu {

int choice;

int pointsAmount;

COORD start, end;

WCHAR \*\*points;

};

Следующим шагом будет написание структур данных, которые также будут реализовывать функционал меню, но с некоторыми изменениями. Первой из тких является структура меню с возможностью скроллинга. Эта структура будет содержать в себе вышеупомянутую структуру menu и buffer, который будет хранить в себе все элементы меню, которые будут подгружаться или выгружаться в зависимости от действий пользователя. Также в данной структуре должны присутствовать поля, которые понадобятся для реализации её некоторых функций. Представление данной структуры на языке Си:

struct scrollMenu {

struct menu menu;

struct student\* buffer;

int bufferPointsAmount;

int maxPoints;

int page;

int printStudentsMode;

};

Второй подобной структурой будет studentMenu. Она станет отвечать за отображение студента на экране и взаимодействие пользователя со студентом. Кроме структуры menu она должна также содержать указатель на студента, подвергающегося взаимодействию в данный момент. Описание соответствующей структуры данных на языке Си:

struct studentMenu {

struct menu menu;

struct student\* student;

};

Также необходимо реализовать структуру, которая будет хранить все записи о студентах. В качестве структуры данных для этой задачи было выбрано АВЛ-дерево, которое, как известно, имеет сложность поиска . Описание данной структуры на языке Си:

struct node {

struct student student;

size\_t height;

struct node\* left;

struct node\* right;

};

Теперь, когда в наличие имеются все необходимые структуры данных, можно приступать к реализации функций, необходимых для работы программы. Функцией, которая отвечает за запуск пользовательского интерфейса будет runUI, чьей единственной задачей является запустить работу основного меню и дожидаться логического окончания его работы. За работу меню будет отвечать функция runMenu, которая будет принимать указатель на структуру данных типа menu. Структурна схема функции runMenu показана на рисунке 3.1. В данной функции также используется функция showMenu (Вывести меню), но её реализацию можно опустить, так как единственное, что она делает, это очищает область меню и выводит его построчно заново в определённом формате. Функция runMenu вернёт номер выбранного пользователем элемента. Эта функция будет применятся для главного меню, потому что для остальных меню будет разработана функция runSubMenu, чьё единственное отличие будет заключаться в замене проверки нажатия на клавишу ESC на проверку нажатия на клавишу стрелки влево.

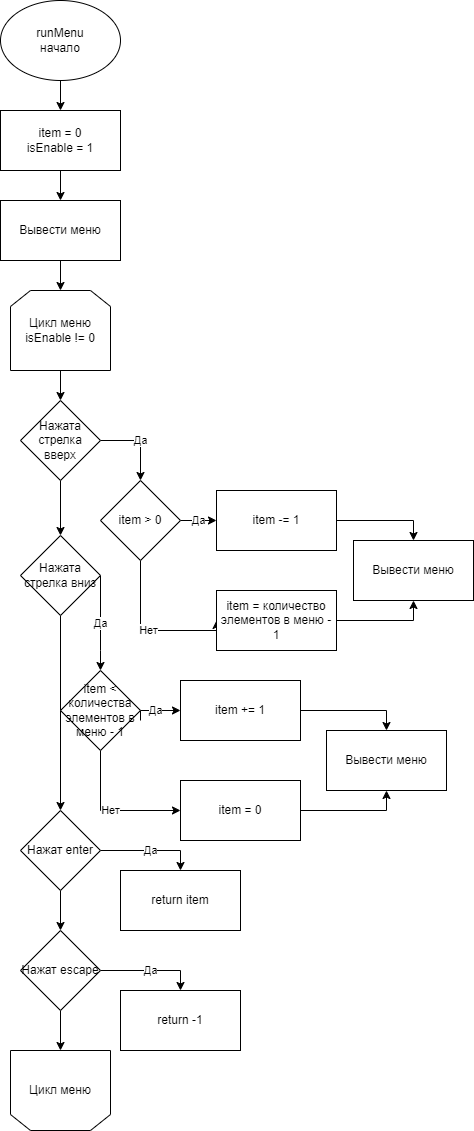


Рисунок 3.1 – runMenu

Итак, в основной программе нам требуется только создать структуры studentService и UI, после чего запустить UI функцией runUI.

## **4 ВЫПОЛНЕНИЕ ПРОГРАММЫ**

## **5 ЗАКЛЮЧЕНИЕ**

## **6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

## **7 ПРИЛОЖЕНИЕ**