МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт информационных технологий

Кафедра/департамент «Информационные системы»

**КУРСОВАЯ РАБОТА / КУРСОВОЙ ПРОЕКТ**

по дисциплине

Алгоритмизация и программирование

на тему «Программа учета выданных книг в библиотеке»

Выполнил: обучающийся

группы: ИС/б-21-3-о

Пышногуб В.С.

« \_\_» \_\_\_\_\_\_\_20 22 г.

Научный руководитель:

Сметанина Т.И.

«\_\_» \_\_\_\_\_\_\_\_20 22 г.

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

« \_\_» \_\_\_\_\_\_\_\_\_ 20 22 г.

Севастополь 2022

# АННОТАЦИЯ

Пояснительная записка содержит описание программы для работы с табличными данными, а именно, записями о выданных книгах в библиотеке, разработанной в рамках курсового проектирования, целью которого является закрепление и углубление знаний в области основ структурного и модульного программирования. Также целью является приобретение практических навыков разработки приложений с использованием сторонних открытых библиотек и примитивной терминальной графики.

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc120451273)

[1 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ 6](#_Toc120451274)

[2 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ 7](#_Toc120451275)

[2.1 Постановка задачи на разработку программы 7](#_Toc120451276)

[2.2 Описание и обоснование выбора метода организации входных, выходных и промежуточных данных 8](#_Toc120451277)

[2.3 Обоснование выбора языка и среды программирования 9](#_Toc120451278)

[2.4 Разработка модульной структуры программы 10](#_Toc120451279)

[2.4.1 Дополнительные константы, массивы и перечисления 10](#_Toc120451280)

[2.5 Описание алгоритмов функционирования программы 11](#_Toc120451281)

[2.6 Обоснование состава технических и программных средств 12](#_Toc120451282)

[3 ВЫПОЛНЕНИЕ ПРОГРАММЫ 13](#_Toc120451283)

[3.1 Условия выполнения программы 13](#_Toc120451284)

[3.2 Загрузка и запуск программы 13](#_Toc120451285)

[3.3 Проверка работоспособности программы 14](#_Toc120451286)

[ЗАКЛЮЧЕНИЕ 15](#_Toc120451287)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 16](#_Toc120451288)

[А. ПРИЛОЖЕНИЕ А 18](#_Toc120451289)

[Б. ПРИЛОЖЕНИЕ Б Листинг программы 19](#_Toc120451290)

# ВВЕДЕНИЕ

В рамках настоящего курсового проектирования ведется разработка программы по теме «Программа учета выданных книг в библиотеке» на основании документа – техническое задание – и в рамках организации – Севастопольский государственный университет. Дата выдачи задания: 08.09.2022.

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

Целью курсового проектирования является систематизация, закрепление и углубление знаний в области основ процедурного программирования и совершенствование практических навыков разработки программ на языке Си на примере разработки программы «Программа учета выданных книг в библиотеке», представляющей собой упрощённое подобие базы данных и позволяющей выполнять различные операции над записями.

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;

Также должны быть разработаны программные документы в соответствии с действующими стандартами.

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ

Программа предназначена для организации, хранения, поиска и модификации данных о выданных в библиотеке книгах через пользователь-ориентированный терминальный интерфейс.

Области применения программы: библиотеки, находящиеся в городе Севастополь и в его округе. Одной из возможностей программы является импорт данных, сохранённых в формате «txt», что позволяет работать с данными как из других программ, так и с данными, изначально созданными с помощью текстовых редакторов.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ

## Постановка задачи на разработку программы

Был выдан 14 вариант задания.

Входной файл имеет следующую структуру:

* ид;
* ФИО абонента;
* автор книги;
* название книги;
* издательство;
* дата выдачи;
* стоимость книги;

Программа должна предоставлять меню-ориентированный интерфейс, позволяющий выполнять следующий минимально необходимый набор действий:

* создание и добавление элементов в таблицу;
* просмотр существующих записей;
* удаление записи по ключевому полю;
* редактирование записи;
* сортировка записей в таблице по выбранному полю;
* сохранение данных в бинарный/текстовый файл с заданным именем файла (по выбору пользователя);
* загрузка данных из бинарного/текстового файла с заданным именем файла (по выбору пользователя);
* реализовать выполнение требуемого по варианту задания;
* корректное завершение работы программы.

В случае обработки записей по варианту программа должна выбрать ФИО и количество книг задолжников, которые не сдали книги обратно в течении 90 дней с момента выдачи.

## Описание и обоснование выбора метода организации входных, выходных и промежуточных данных

Для хранения данных в оперативной памяти было принято решение использовать бинарное дерево, потому что бинарное дерево позволяет ускорить поиск требуемых элементов и постраничный вывод данных на экран.

Были определены дополнительные структуры данных.

Структура хранения ФИО:

typedef struct { //Структура для хранения фио

char name[80]; //Имя

char surname[80];//Фамилия

char secondname[80];//отчество

}fio\_t;

Структура хранения пользовательских данных:

typedef struct { // основная структура информационного поля

unsigned int id; //идентификатор поля

fio\_t fio; //поле структуры фио

struct { // структура данных для автора книги

char surname[60];

char inicial[20];

}autor;

char book\_name[160]; //поле для названия книги

char izd[70]; // поле для издания

struct { // структура, хранит дату выдачи книги

int d;

int m;

int y;

}date\_out;

float cost; // цена книги

} abonent\_t;

Элемент дерева:

typedef struct abonent\_l {

abonent\_t info; // информационное поле

struct abonent\_l\* right; //правая нода

struct abonent\_l\* left; // левая нода

} abonent;

Структура для хранения должников:

typedef struct { // структура для хранения должников

unsigned int id; //идентификатор

fio\_t fio; // поле фио

int count\_dolg\_books; // количество книг в задолжности

}dolgi\_pers\_t;

Структура для хранения массива должников и его размера:

typedef struct { // структура-обертка для хранения данных должников

dolgi\_pers\_t\* info\_mass; // массив должников

int count; // количество должников

}dolgi\_pers\_t\_obr;

Структура для хранения меню.

typedef struct { // структура одного элемента меню

int \_menu\_size; //размер меню

char \_name[80]; //наименование пункта меню

char\*\* \_sub\_menu; //указатель на массив сабменю

int\* \_sub\_menu\_lenght; //количество элементов сабменю

int \_max\_sub\_lenght; //длина самой большого элемента в сабменю

int \_menu\_name\_lenght; //длина имени меню

} \_menu\_item;

Структура для хранения информации о строке таблицы.

typedef struct { // структура хранящая данные столбца таблицы

char\* name; //имя столбца

int size; //его размер

int resizebl; //флаг, можно ли изменять ее размер ?

}\_table\_col;

Структура для хранения общей информации о таблице.

typedef struct { // структура для хранения метаданных таблицы

\_table\_col\* \_cols; //массив столбцов

int \_col\_count; //количество столбцов

}\_tabel\_metadata;

## Обоснование выбора языка и среды программирования

Исходный код программы был написан на языке программирования C, стандарт С99. Причина выбора:

* скорость выполнения функций;
* эффективное потребление памяти;

Для разработки программы была создана следующая среда разработки:

* компилятор: Microsoft C ++. Основные причины выбора: совместимость с многими версия ос Windows, быстрая скорость работы, отслеживание множества ошибок;
* текстовый редактор: Microsoft Visual Studio. Основные причины выбора: наличие огромного функционала по работе с кодом и ревакторингу кода; возможность установки дополнений из предоставленной библиотеки; наличие встроенного гитхаба;
* система сборки проекта: Windwos 11. Причина использования: самая популярная операционная среда среди пользователей.
* ситема контроля версий кода: git.

В качестве основной операционной системы была выбрана Windows .

## Разработка модульной структуры программы

В основу организации программы был положен принцип событийного управления. До начала разработки было принято решение разделить программу на 5 частей:

* главный файл, содержащий инициализатор меню, инициализатор таблицы и реализующий вызовы других модулей;
* ядро интерфейса, содержащее основные функции для работы с пользовательским интерфейсом;
* ядро программы, содержащий функции работы с данными.
* вспомогательные утилиты ввода, содержит в себе утилиты форматированного ввода данных
* вспомогательые утилиты для работы с идентификатором, содержащая функции создания хеш-кода для ид

### Дополнительные константы, массивы и перечисления

Перечисление кодов для обработки вызовов меню.

enum MenuItemCodes //Коды для обработки вызовов меню

{

ADD\_NEW\_RECORD = 1,

LOAD\_FROM\_FILE\_TYPE = 2,

LOAD\_FROM\_FILE = 3,

SAVE\_TO\_FILE\_TYPE = 4,

SAVE\_TO\_FILE = 5,

TREE\_SIZE = 6,

PRINT\_TREE\_STRUCT = 7,

CLEAN\_TREE = 8,

DOLGNIKI\_WINDWO = 9,

PRINT\_HORRIBLE\_ANIMATION = 10,

PROGRAM\_EXIT = 11

};

Перечисление кодов кнопок для обработки вызовов нажатий.

enum KeyboardCodes

{

KEY\_ARROW\_UP = 72,

KEY\_ARROW\_DOWN = 80,

KEY\_ARROW\_LEFT = 75,

KEY\_ARROW\_RIGHT = 77,

KEY\_TAB = 9,

KEY\_HOME = 71,

KEY\_END = 79,

KEY\_ENTER = 13,

KEY\_ESC = 27,

KEY\_DEL = 83,

KEY\_BACKSPACE = 8,

KEY\_PGUP = 73,

KEY\_PGDOWN = 81

};

Перечисление кодов для обработки сортировки.

enum sort\_fild {

DEF,

FIO,

AUTHOR,

BOOK\_NAME,

IZD,

DATE\_OUT,

COST,

ZADANIE

};

Перечисление кодов для обработки ввода.

enum WorkingMode

{

NORMAL = 0,

PERSONAL = 1,

INICIAL = 2,

DATA = 3,

};

## Описание алгоритмов функционирования программы

Рекурсивная функция добавления объекта в дерево. В качестве параметров требует указатель на указатель корня и указатель на структуру данных для записи.

void tree\_add(abonent\*\* root, const abonent\_t\* info);

Рекурсивная функция получения количества объектов в дерево. Возвращает количество элементов, либо 0 если их нет. В качестве параметров требует указатель на корень дерева, и текущее количество просчетов

int tree\_getNodeCount(const abonent\* root, const int accum);

Рекурсивная функция получения объекта в дереве по его ид. Возвращает указатель на объект дерева, либо NULL если объект не найден. В качестве параметров требует указатель на корень дерева, и ключь для поиска.

abonent\* tree\_getLeafById(abonent\* root, const int id);

Рекурсивная функция удаления объекта в дереве по его ид. В качестве параметров требует указатель на корень дерева, и ключь для поиска.

void tree\_deleteNodeById(abonent\*\* root, const int id);

Рекурсивная функция отрисовки структуры дерева. В качестве параметров требует указатель на корень дерева, и параметр оступа.

void tree\_deleteNodeById(abonent\*\* root, const int id);

Рекурсивная функция удаления дерева. В качестве параметров требует указатель на корень дерева. Вернет пустой указатель.

abonent\* tree\_delete(abonent\* root);

Рекурсивная функция записи дерева в бинарный файл. В качестве параметров требует указатель на файл и корень дерева.

void printToFile(FILE\* f, abonent\* root);

Рекурсивная функция записи дерева в тектовый файл. В качестве параметров требует указатель на файл и корень дерева.

void printToFile\_Text(FILE\* f, abonent\* root);

Рекурсивная функция получения данных из дерева для дальнейшей работы с ними. В качестве параметров требует указатель на корень дерева, указатель на область хранения данных для вывода, указатель на индекс. Вернет указатель на массив данных.

abonent\_t\* \_get\_output\_info(abonent\* root, abonent\_t\* \_output\_memory, int\* index);

Рекурсивная функция получения данных должников из дерева для дальнейшей работы с ними. В качестве параметров требует указатель на корень дерева, указатель на область хранения данных для вывода, день, месяц и год для сравнения. Вернет указатель на струтктуру данных.

dolgi\_pers\_t\_obr\* \_get\_dolgi\_info(abonent\* root, dolgi\_pers\_t\_obr\* \_output\_memory, int d, int m, int y)

Функция циклического добавления элементов в дерево. В качестве параметров передается указатель на указатель на корень дерева и указатель на данные таблицы. Возвращает EXIST\_SUCCESS.

int аddNewElement(abonent\*\* st, \_tabel\_metadata \*table);

Функция-обертка для вызова печати дерева в файл. В качестве параметров передается указатель на файл, указатель на корень дерева. Возвращает 0 в случае успеха, 666 если возникла ошибка.

int create\_file\_type(FILE\* f, abonent\* St);

Функция считывания данных из типизированного файла. В качестве параметров передается указатель на файл. Возвращает указатель на корень дерева.

abonent\* loadFromFile\_new\_type(FILE\* f);

Функция считывания данных из текстового файла. В качестве параметров передается указатель на файл. Возвращает указатель на корень дерева.

abonent\* loadFromFile\_new\_text(FILE\* f);

Функция вызова функций программы согласно выбраного индекса. В качестве параметров передается номер выбранного пункта, указатель на файл, указатель на структуру с данными таблицы.

void MenuSelect(int selector, FILE\* f, \_tabel\_metadata\* table);

Функция инициализации меню программы. В качестве параметров передается указатель на пустую структуру меню.

\_menu\_item\* \_init\_menu(\_menu\_item\* menu);

Функция инициализации таблицы программы. В качестве параметров передается указатель на пустую структуру таблицы.

\_tabel\_metadata\* \_init\_table(\_tabel\_metadata\* table);

Функция получения хеш кода из ФИО. В качестве параметра, передается указатель на информационное поле. Возвращает беззнаковое значение.

unsigned int util\_hashCodeFromAbonentStruct(const abonent\_t\* a);

Функция получения хеш кода из ФИО. В качестве параметра, передается указатель на поле структуры fio. Возвращает беззнаковое значение.

unsigned int util\_hashCodeFromFio(const fio\_t\* fio);

Функция получения хеш кода из строки. В качестве параметра, передается указатель на поле структуры fio. Возвращает беззнаковое значение.

unsigned int util\_hashCodeFromString(const char\* string);

Функция получения информации о консоли. В качестве параметра, передается указатель на пустой указатель на буфер консоли. Возвращает 1 или 0 в зависимости от успешности.

BOOL \_get\_con\_info\_local(CONSOLE\_SCREEN\_BUFFER\_INFO\* x);

Функция установки курсора в консоли на определенные координаты. В качестве параметра, структура COORD. Возвращает 1 или 0 в зависимости от успешности.

BOOL \_set\_cur\_to\_pos\_local(COORD cor);

Функция перевода строки из utf-8 в 1251. В качестве параметра передается один элемент типа char. Возвращает элемент типа char.

char convert\_u8\_to\_1251(int c);

Функция форматированного ввода данных с плавающей точкой. В качестве параметра передается указатель на поле, в которое необходимо записать значение. Возвращает номер нажатого спецсимвола.

int input\_float(float \* info);

Функция форматированного ввода данных. В качестве параметра передается указатель на буфер, в которое необходимо записать значение, размер буфера, а так же режим работы: 0 - разрешение на ввод всего, кроме спец символов, 1- для реализации ввода фамилии\имени\отчества\издательства (не допускаются цифры и т.д), 2 - ввод инициалов. Возвращает номер нажатого спецсимвола.

int input\_string(char\* input\_buff, int buff\_size, int mode);

Функция форматированного ввода даты. В качестве параметров передается указатели на значения, в которое необходимо записать данные. Возвращает номер нажатого спецсимвола.

int in\_date(int\* d, int\*m,int \*y);

Функция очистки внутреннего поля окна.

void clear();

Функция очистки поля табличного окна для отрисовки таблицы.

void clear\_table();

Функция очистки поля окна программы без очистки меню.

void clear\_for\_info();

Функция получения новой позиции курсора. В качестве параметров получает символ, который ввел пользователь, указатель на двумерный массив текущей позиции, максимальное число строк, максимальное число столбцов и флаг разрешения на изменение x. Вернет указатель на двумерный массив.

int\* \_get\_curent\_selection(char c // Символ клавиатуры

,int \* position // Массив в котором хранятся x и y

,int MaxY // Максимальный количество строк

, int Colums // Количество столбцов, по умолчанию - 1

,int \_flag\_x\_readonly //можно ли менять x?

)

Функция отрисовки меню программы и работы с окнами. В качестве параметров получает указатель на массив объектов меню, указатель на двумерный массив текущей позиции, количество элементов в меню, количество кнопок в меню, указатель на массив информации для вывода, количесвто данных для вывода, указатель на структуру, хранящую данные о полях таблицы, двойной указатель на корень дерева и структуру, содержащую текущие параметры сортировки. Вернет индекс выбраного элемента.

int \_print\_menu(\_menu\_item\* \_menu //Массив объектов меню

, int\* position //Массив текущей позиции x и y

, int \_menu\_size //Колличество элементов в массиве объектов меню

, int \_menu\_buttons,//Количество кнопок меню

abonent\_t \* \_output\_mas, //массив для вывода информации

int \_output\_colcount, // количество данных для вывода

\_tabel\_metadata \* table, //структура хранящая информацию о полях таблицы

abonent\*\* root, //указатель на указатель на корень дерева

sort\_struct\* sort // структура, хранящая текущую сортировку

)

Функция печати заднего фона и рабочей области окна. В качестве параметров требует ширину и высоту окна.

void \_print\_bakground(int \_window\_w,int \_window\_h);

Функция печати рабочей области программы. В качестве параметров принимает ширину и высоту окна.

void \_print\_border(int \_window\_w, int \_window\_h);

Функция получения текущего размера окна. В качестве параметра передается указатель на двумерный массив. Возвращает указатель на двумерный массив.

int\* \_get\_window\_size(int\* size);

Функция отрисовки маленького окна. В качестве параметров требует текущюю ширину и высоту окна, а так же указатель на строку названия файла.

void \_window(int \_window\_w, int \_window\_h, char\* title);

Функция отрисовки окна выбора. В качестве параметра требует указатель на отображаемое сообщение. Возвращает 1 в случае подтверждения , иначе 0.

int \_confirm\_window(char \* message);

Функция отрисовки окна ввода значения.

void \_in\_window();

Функция отрисовки окна с сообщением. В качестве параметра требует указатель на сообщение.

void \_message\_window(char\* message);

Функция отрисовки окна таблицы с выводом данных и возможностью управления и работы с записями. В качестве параметров принимает указатель на данные о таблице, указатель на массив данных для вывода, указатель на количество данных для отрисовки, указатель на номер текущей страницы, указатель на флаг активности фокуса на таблице, двойной указатель на дерево и указатель на структуру, хранящуюю сортировку. В качестве результата возвращает EXIT\_SUCCESS.

int \_table\_window(\_tabel\_metadata \* table, abonent\_t \* \_output\_mass, int \* \_info\_count, int\* page, int \* \_table\_focus\_flag, abonent\*\* root, sort\_struct\* sort);

Функция отрисовки большого окна. В качестве параметра получает указатель на массив, хранящий наименование окна.

void \_big\_window(char\* title);

Функция отрисовки окна ввода/редактирования данных о записи. В качестве параметров получет указатель на данные таблицы, указатель на структуру для хранения данных или NULL если нужно создать новую запись, флаг цикличности.

abonent\_t\* \_in\_info\_window(\_tabel\_metadata\* table, abonent\_t \*\_output\_info,int \_cycle\_in\_flag);

Функция печати помощи в главном окне программы. В качестве параметра получает указатель на сообщение, которое нужно вывести в помощь. Вернет 0 в случае успеха.

int print\_help(char \* help\_message);

Функция сортировки данных по различным полям. В качестве параметров получает указатель на массив данных, указатель на количество данных, указатель на структуру с типами сортировки. Возвращает указатель на массив данных для вывода.

abonent\_t\* \_sort\_output(abonent\_t\* \_output\_mass, int\* filds\_count, sort\_struct\* sorts);

Функция отрисовки окна с выводом таблицы с должниками и вохможностью просмотра нескольких страниц. В качестве параметра передается двойной указатель на корень дерева.

void dolgiWindow(abonent \*\* root);

Рисунок А.1 – структурная схема функции получения элемента из дерева по его ID tree\_getLeafById

Поблочное описание:

* блок 1 – получение двойного указателя на корень дерева и ключа ID для поиска;
* блок 2 – если указатель пуст;
* блок 3 – вернем пустой указатель на данные
* блок 4 – если ключь поиска ID совпал с ключем ID в данных;
* блок 5 – вернуть указатель на текущий элемент;
* блок 6 – если ID текущего элемента меньше ключа ID;
* блок 7 – вызов рекурсивной проверки в правом поддереве, вернуть ее результат;
* блок 8 – вызов рекурсивной проверки в левом поддереве, вернуть ее результат;

Рисунок А.2 – структурная схема функции добавления элемента в дерево Tree\_\_add.

Поблочное описание:

* блок 1 – получение двойного указателя на корень дерева и структуры с данными;
* блок 2 – если указатели пусты;
* блок 3 – вернем ничего;
* блок 4 – если указатель на корень дерева пуст;
* блок 5 – выделим память под элемент;
* блок 6 – если выделение успешно;
* блок 7 – запоминаем информационное поле;
* блок 8 – если id записи которую хотим добавить меньшен id текущей записи;
* блок 9 – рекурсивный вызов добавления в левое поддерево;
* блок 10 – рекурсивный вызов добавления в правое поддерево;

Рисунок А.3 – структурная схема функции отображения структуры дерева View.

Поблочное описание:

* блок 1 – получение указателя на корень дерева и размер отступа;
* блок 2 – если указатель на корень не пуст;
* блок 3 – увеличим отступ на 3;
* блок 4 – рекурсивный вызов отрисовки структуры в правом поддереве;
* блок 5 – цикл работающий до размера отступа;
* блок 6 – выводим пробел;
* блок 7 – выводим id в соответствующей позиции;
* блок 8 – рекурсивный вызов отрисовки структуры в левом поддереве;

Рисунок А.4 – структурная схема функции преобразования строки в хеш-код util\_hashCodeFromString.

Поблочное описание:

* блок 1 – получение указателя на массив строки;
* блок 2 – создаем беззнаковую переменную для хранения результата;
* блок 3 – цикл прохода от начала до конца строки;
* блок 4 – к результату добовляем значение кода символа строки;
* блок 5 – вернем беззнаковую переменную;

Рисунок А.5 – структурная схема функции получения данных из дерева для вывода таблицы с данными \_get\_output\_info.

Поблочное описание:

* блок 1 – если указатель на корень дерева существует;
* блок 2 – если существует левое поддерево;
* блок 3 – рекурсивный вызов функции получения даннх из дерева в левом поддереве;
* блок 4 – запоминаем данные в массив с соответствующим индексом;
* блок 5 – увеличиваем текущюю позицию в массиве;
* блок 6 – если существует правое поддерево;
* блок 7 – рекурсивный вызов функции получения данных из дерева в правом поддереве;

Рисунок А.6 – структурная схема функции конвертирования символа из UTF8 в Windows 1251 convert\_u8\_to\_1251.

Поблочное описание:

* блок 1 – получаем символ;
* блок 2 – создаем временные переменные, запоминаем туда символ, явно преобразуя его;
* блок 3 – вызов подпрограмм преобразования символа в многобайтовую кодировку;
* блок 4 – запоминаем необходимые данные;
* блок 5 – вызов подпрограммы конвертирования из многобайтного символа в однобайтовый;
* блок 6 – создание второго массива для данных;
* блок 7 – вызов преобразования однобайтового символа в многобайтовый в нутри переменной wstr;
* блок 8 – вызов преобразования однобайтового символа в многобайтовый в нутри переменной s1251;
* блок 9 – вернем значение полученного символа.

## Обоснование состава технических и программных средств

Для работы программы требуется использование компьютера с дисплеем для вывода данных и клавиатурой для приёма команд от пользователя. Должна быть предустановлена операционная система семейства Windows версии не ниже Windows 7. Требования относительно жёсткого диска: 160 килобайта для хранения программы, дополнительное пространство для хранения файлов базы данных. Для запуска программы требуется не менее 1024 килобайт оперативной памяти для 64 битных систем, а также дополнительная память для обработки базы данных.

# ВЫПОЛНЕНИЕ ПРОГРАММЫ

## Условия выполнения программы

Таблица 3.1 демонстрирует минимально необходимые системные требования для запуска приложения.

Таблица 3.1 – Системные требования

|  |  |
| --- | --- |
| Операционная система | Windows 7 и выше |
| Процессор | Одноядерный процессор, x86\_64, не менее 1ГГц |
| ОЗУ | Не менее 1024 КБ |
| Место на внешнем носителе | Не менее 160 КБ |
| Требования к терминалу | поддержка 256bit цветов;  размеры окна – не менее 107x24 символов;  поддержка кодировки UTF-8 |

## Загрузка и запуск программы

Запустить программу можно двумя способами: …..

## Проверка работоспособности программы

Для проверки работоспособности программы был создан файл в формате «txt». Содержимое файла представлено ниже: ……

Проверка работоспособности программы была проведена успешно: программа корректно считала данные из одного файла, изменила их и записала изменения в указанный пользователем другой файл.

# ЗАКЛЮЧЕНИЕ

В соответствии с вариантом задания разработана программа, в основу алгоритма которой положена структура данных в виде бинарного дерева, позволяющего выполнять просмотр данных. Особенностями программы являются: сортировка элементов конкретной таблицы по всем полям как по возрастанию (в алфавитном порядке), так и по убыванию (не в алфавитном порядке); удобный, ориентированный на пользователя интерфейс программы, схожий с стандартным интерфейсом некоторых текстовых редакторов.

Разработка велась на базе Windows 11 в редакторе Visual Studio с использованием Microsoft C++ для компиляции и git для контроля версий кода.

В процессе разработки возникало несколько проблем.

Во-первых, для хранения строк было принято решение использовать массив символов char. С одной стороны, это дало возможность хранить строки в кодировке 1251, а также корректное отображение нелатинские символов. С другой стороны, был усложнён процесс разработки ввиду того, что для отрисовки меню приложения используется кодировка UTF-8. Соответственно пришлось использовать отдельные функции конвертации данных из UTF-8 в 1251, а так же использовать стороний модуль utf\_8, который позволял работать с кодировкой в UTF-8 в обычном массиве типа char.

Во-вторых, ввиду громосткости и наполненности графическими элементами отлаживать программу было тяжелее.

Таким образом были достигнуты цели курсового проектирования: углублены знания языка Си; получен навык разработки программ с использованием методологии структурного программирования, а также получены практические навыки разработки приложений с использованием стронних открытых библиотек. Полученные навыки помогут разрабатывать пользователь-ориентированные приложения в терминальной среде.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Г. Шилдт C++ для начинающих. Серия "Шаг за шагом" / Г. Шилдт; пер. с англ. - М.: ЭКОМ Паблишерз, 2013. - 643 с.
2. Методические указания к лабораторным работам по дисциплине «Информатика» для студентов дневной и заочной форм обучения направления 09.03.02 – «Информационные системы и технологии», часть 1 / Сост. В.Н.Бондарев, Т.И. Сметанина. – Севастополь: Изд-во СевГУ, 2014. – 44 с.
3. Методические указания к лабораторным работам по дисциплине «Информатика» для студентов дневной и заочной форм обучения направления 09.03.02 – «Информационные системы и технологии», часть 2 / Сост. В.Н.Бондарев, Т.И. Сметанина – Севастополь: Изд-во СевГУ, 2014. – 64с.
4. Структурное программирование на языке С/С++: методические указания клабораторным работам по дисциплине «Основы программирования и алгоритмические языки» для студентов дневной и заочной форм обучения направления 09.03.02 – «Информационные системы и технологии», часть 1 / Сост. В.Н. Бондарев, Т.И. Сметанина.– Севастополь: Изд-во СевГУ, 2015. – 60 с.
5. Павловская Т. А. Паскаль. Программирование на языке высокого уровня: практикум / Т. А. Павловская. – СПб.: Питер, 2006. – 317 с.
6. Павловская Т. А. Паскаль. Программирование на языке высокого уровня: учеб. для вузов / Т. А. Павловская. – СПб.: Питер, 2008. – 393 с.
7. Керниган Б., Ритчи Д. Язык программирования СИ: пер. с англ./Под ред. и спредисл. В.С. Штаркмана. –2-е изд., перераб. и доп. – М.; СПб. ; К. : Вильямс, 2006. –272с.
8. Вирт Н. Алгоритмы и структуры данных / Н. Вирт; пер. с англ. – М.: Мир,1989. – 360 с.
9. Белецкий Я. Энциклопедия языка Си / Я. Белецкий; пер. с польск. – М.: Мир, 1992. – 687 с.
10. Разработка САПР: в 10 кн. Кн. 3. Проектирование программногообеспечения САПР: практ. пособие / Б. С. Федоров, Н. Б. Гуляев; под ред. А.В. Петрова. – М.: Высш. шк., 1990. – 159с.

# ПРИЛОЖЕНИЕ А

Структурные схемы подпрограмм



Рисунок А.1 – Подпрограмма получения элемента из дерева по его ID



Рисунок А.2 – Подпрограмма добавления элемента в дерево



Рисунок А.3 – Подпрограмма отображения структуры дерева



Рисунок А.4 – Подпрограмма получения хеш кода на основе строки



Рисунок А.5 – Подпрограмма получения данных из дерева для вывода в таблицу



Рисунок А.6 – Подпрограмма преобразования из UTF8 в Windows 1251



Рисунок А.7 – Подпрограмма отрисовки окна с полем для ввода



Рисунок А.8 – Подпрограмма очискти области таблицы



Рисунок А.9 – Подпрограмма получения данных о должниках, часть 1.



Рисунок А.10 – Подпрограмма получения данных о должниках, часть 2.



Рисунок А.11 – Подпрограмма отрисовки окна с сообщением



Рисунок А.12 – Подпрограмма удаления элемента в дереве



Рисунок А.13 – Подпрограмма получения новой позиции указателя на основе нажатой кнопки, часть 1



Рисунок А.14 – Подпрограмма получения новой позиции указателя на основе нажатой кнопки, часть 3

# ПРИЛОЖЕНИЕ Б Листинг программы

Листинг Б.1 – Исходный код программы

**Исходный код header-файла «data\_utils.h»**

#include <stdlib.h>

#include "../ThreeStruct.h"

#ifndef \_DATAUTILS\_H

#define \_DATAUTILS\_H

extern unsigned int util\_hashCodeFromAbonentStruct(const abonent\_t\*);

extern unsigned int util\_hashCodeFromFio(const fio\_t\*);

extern unsigned int util\_hashCodeFromString(const char\*);

#endif //! \_DATAUTILS\_H

**Исходный код header-файла «input\_utils.h»**

#ifndef INPUT\_UTILS

#define INPUT\_UTILS

/// <summary>

/// Ввод любых строковых данных

/// </summary>

/// <param name="input\_buff">Буфер для ввода</param>

/// <param name="buff\_size">Размер буфера</param>

/// <param name="mode">Режим работы, от которого зависят допустимые символы. 0 - разрешение на ввод всего, кроме спец символов, 1- для реализации ввода Фамилии\имени\Отчества\Издательства(не допускаются цифры и т.д),2 - Ввод инициалов</param>

/// <returns></returns>

int input\_string(char\* input\_buff, int buff\_size, int mode);

int in\_date(int\* d, int\* m, int\* y);

int input\_float(float\* info);

typedef struct {

int d;

int m;

int y;

}date;

enum WorkingMode

{

NORMAL = 0,

PERSONAL = 1,

INICIAL = 2,

DATA = 3,

};

#endif // !input\_utils

#ifndef KEYCODE

#define KEYCODE

enum KeyboardCodes

{

KEY\_ARROW\_UP = 72,

KEY\_ARROW\_DOWN = 80,

KEY\_ARROW\_LEFT = 75,

KEY\_ARROW\_RIGHT = 77,

KEY\_TAB = 9,

KEY\_HOME = 71,

KEY\_END = 79,

KEY\_ENTER = 13,

KEY\_ESC = 27,

KEY\_DEL = 83,

KEY\_BACKSPACE = 8

};

#endif // !1

**Исходный код header-файла «MenuStruct.h»**

#ifndef MenuStruct

#define MenuStruct

typedef struct { // структура одного элемента меню

int \_menu\_size; //размер меню

char \_name[80]; //наименование пункта меню

char\*\* \_sub\_menu; //указатель на массив сабменю

int\* \_sub\_menu\_lenght; //количество элементов сабменю

int \_max\_sub\_lenght; //длина самой большого элемента в сабменю

int \_menu\_name\_lenght; //длина имени меню

} \_menu\_item;

typedef struct { // структура хранящая данные столбца таблицы

char\* name; //имя столбца

int size; //его размер

int resizebl; //флаг, можно ли изменять ее размер ?

}\_table\_col;

typedef struct { // структура для хранения метаданных таблицы

\_table\_col\* \_cols; //массив столбцов

int \_col\_count; //количество столбцов

}\_tabel\_metadata;

enum MenuItemCodes //Коды для обработки вызовов меню

{

ADD\_NEW\_RECORD = 1,

LOAD\_FROM\_FILE\_TYPE = 2,

LOAD\_FROM\_FILE = 3,

SAVE\_TO\_FILE\_TYPE = 4,

SAVE\_TO\_FILE = 5,

TREE\_SIZE = 6,

PRINT\_TREE\_STRUCT = 7,

CLEAN\_TREE = 8,

DOLGNIKI\_WINDWO = 9,

PRINT\_HORRIBLE\_ANIMATION = 10,

PROGRAM\_EXIT = 11

};

#endif //! MenuStruct

**Исходный код header-файла «ThreeStruct.h»**

#pragma once

typedef struct { //Структура для хранения фио

char name[80]; //Имя

char surname[80];//Фамилия

char secondname[80];//отчество

}fio\_t;

typedef struct { // основная структура информационного поля

unsigned int id; //идентификатор поля

fio\_t fio; //поле структуры фио

struct { // структура данных для автора книги

char surname[60];

char inicial[20];

}autor;

char book\_name[160]; //поле для названия книги

char izd[70]; // поле для издания

struct { // структура, хранит дату выдачи книги

int d;

int m;

int y;

}date\_out;

float cost; // цена книги

} abonent\_t;

typedef struct { // структура для хранения должников

unsigned int id; //идентификатор

fio\_t fio; // поле фио

int count\_dolg\_books; // количество книг в задолжности

}dolgi\_pers\_t;

typedef struct { // структура-обертка для хранения данных должников

dolgi\_pers\_t\* info\_mass; // массив должников

int count; // количество должников

}dolgi\_pers\_t\_obr;

/// <summary>

/// Структура ноды дерева

/// </summary>

typedef struct abonent\_l {

abonent\_t info; // информационное поле

struct abonent\_l\* right; //правая нода

struct abonent\_l\* left; // левая нода

} abonent;

const int size\_abonent\_t; //константа, хранит размер информационного поля

**Исходный код header-файла «tree\_operation.h»**

#include "ThreeStruct.h"

#include <stdio.h>

#ifndef TREE\_OPERATION

#define TREE\_OPERATION

/// <summary>

/// Вспомогательная рекурсивная функция поиска необходимого листка/ноды

/// </summary>

/// <param name="root">Корень дерева или текущаяя нода</param>

/// <param name="indexToSerch">Ид который необходимо найти</param>

/// <returns>Возвращает указатель на элемент с соответствующим ид, иначе NULL</returns>

abonent\* tree\_getLeafById(abonent\* root, const int id);

/// <summary>

/// Удалить дерево или часть дерева. Также удалить все поддеревья, относящиеся к данному узлу.

/// </summary>

/// <param name="st">Корень дерева/текущаяя нода</param>

/// <returns>Пустой указатель на структуру</returns>

abonent\* tree\_delete(abonent\* root);

/// <summary>

/// Удалить узел дерева по id.

/// </summary>

/// <param name="root">Узел дерева, с которого начинать поиск.</param>

/// <param name="id">ID искомого узла.</param>

/// <returns>Корень дерева после изменения.</returns>

//abonent\*\* tree\_deleteNodeById(abonent\*\* root, const int id);

void tree\_deleteNodeById(abonent\*\* root, const int id);

/// <summary>

/// Подсчитать количество узлов в дереве.

/// </summary>

/// <param name="root">Узел дерева, с которого начинать подсчёт.</param>

/// <param name="count">Стартовое значение счётчика.</param>

/// <returns>Количество элементво в дереве.</returns>

int tree\_getNodeCount(const abonent\* root, const int accum);

/// <summary>

/// Вставить элемент в дерево.

/// </summary>

/// <param name="head">Указатель на указатель дерева.</param>

/// <param name="info">Вставляемый элемент.</param>

void tree\_add(abonent\*\* head, const abonent\_t\* info);

void printToFile(FILE\* f, abonent\* St);

void printToFile\_Text(FILE\* f, abonent\* St);

void View(abonent\* top, int offset);

abonent\_t\* \_get\_output\_info(abonent\* root, abonent\_t\* \_output\_memory, int\* index);

dolgi\_pers\_t\_obr\* \_get\_dolgi\_info(abonent\* root, dolgi\_pers\_t\_obr\* \_output\_memory, int d, int m, int y);

#endif // !TREE\_OPERATION

**Исходный код header-файла «VicMenuDLL.h»**

#include "MenuStruct.h"

#include "ThreeStruct.h"

#ifndef VicMenuDLL

#define VicMenuDLL

typedef struct {

int sort\_f;

int sort\_t;

}sort\_struct;

int\* \_get\_curent\_selection(char c // Символ клавиатуры

, int\* position // Массив в котором хранятся x и y

, int MaxY // Максимальный количество строк

, int Colums // Количество столбцов, по умолчанию - 1

,int \_flag\_x\_readonly

);

/// <summary>

/// Функция печати основного окна меню. Предназначена для создания внутренней области для работы с данными и реализует возможность

/// работы с самим меню используя стрелки.

/// </summary>

/// <param name="\_menu"> структура данных содержащая в себе меню </param>

/// <param name="position">массив текущей позиции x,y</param>

/// <param name="\_menu\_size">Количество основных пунктов меню</param>

/// <param name="Colums">Количество кнопок меню</param>

/// <param name="\_output\_mas">Массив, необходимый для вывода информации</param>

/// <param name="\_output\_colcount">Количество данных в массиве вывода </param>

/// <param name="table">Структура хранящая информациию о таблице</param>

/// <param name="root">Корень дерева, предназначен для вызова функций из внутренностей меню</param>

/// <param name="sort">Структура, хранящая настройки сортировки</param>

/// <returns>Индекс выбранного элемента</returns>

int \_print\_menu(\_menu\_item\* \_menu, int\* position, int \_menu\_size, int Colums, abonent\_t\* \_output\_mas,

int \_output\_colcount, \_tabel\_metadata\* table, abonent\*\* root, sort\_struct\* sort);

/// <summary>

/// Анимация(над сделать)

/// </summary>

void animatedNeko();

/// <summary>

/// Рисует задний фон согласно параметрам, делая область для работы

/// </summary>

/// <param name="\_window\_w">ширина окна</param>

/// <param name="\_window\_h">высота окна</param>

void \_print\_bakground(int \_window\_w, int \_window\_h);

/// <summary>

/// чистит экран

/// </summary>

void clear();

/// <summary>

/// Печать границ окна с очисткой в нутри

/// </summary>

/// <param name="\_window\_w">ширина окна</param>

/// <param name="\_window\_h">высота окна</param>

void \_print\_border(int \_window\_w, int \_window\_h);

/// <summary>

/// Возвращает текущий размер окна

/// </summary>

/// <returns>Двумерный массив, ширина и высота</returns>

int\* \_get\_window\_size(int\* size);

/// <summary>

/// Функция отричовки окна подверждения с кастомным сообщением

/// </summary>

/// <param name="msg">Указатель на строку сообщения, если NULL то выведет стандартное окно</param>

/// <returns>0 либо 1 в зависимости от того что выбрал пользователь</returns>

int \_confirm\_window(char \*);

/// <summary>

/// Функция печати окна для работы в дальнейшем

/// </summary>

/// <param name="\_window\_w">Текущая ширина</param>

/// <param name="\_window\_h">Текущая высота </param>

/// <param name="title">Название окна </param>

void \_window(int \_window\_w, int \_window\_h,char \* title);

/// <summary>

/// Очистка только внутренней части окна

/// </summary>

void clear\_for\_info();

/// <summary>

/// Окно ввода данных. Считывать данные можно чем угодно вызвав после окна функцию считывания

/// </summary>

void \_in\_window();

/// <summary>

/// Окно сообщения.

/// </summary>

/// <param name="message">Сообщение</param>

void \_message\_window( char\* message);

/// <summary>

/// Табличная форма вывода данных с поддержкой управления в нутри таблицы

/// </summary>

/// <param name="table">структура данных, содержащая информацию о полях таблицы</param>

/// <param name="\_output\_mass">Массив данных для вывода</param>

/// <param name="\_info\_count">Количество данных в массиве для вывода</param>

/// <param name="page">Указатель, хранящий текущюю страницу</param>

/// <param name="\_table\_focus\_flag">Флаг работы с таблицей</param>

/// <param name="root">Корень дерева</param>

/// <param name="sort">Структура сортировки </param>

/// <returns></returns>

int \_table\_window(\_tabel\_metadata\* table, abonent\_t\* \_output\_mass, int\* \_info\_count, int\* page, int\* \_table\_focus\_flag,abonent \*\* root, sort\_struct\* sort);

/// <summary>

/// Большое окно для работы с данными

/// </summary>

/// <param name="title">Название окна </param>

void \_big\_window(char\* title);

/// <summary>

/// Окно ввода/редактирования информации

/// </summary>

/// <param name="table">Поля из таблицы, используются для отрисовки пунктов</param>

/// <param name="\_output\_mass">указатель на элемент данных, null для ввода</param>

/// <param name="flag">Флаг цикличного ввода. Для формирования дерева в цикле</param>

/// <returns>Возвращает элемент</returns>

abonent\_t\* \_in\_info\_window(\_tabel\_metadata\* table, abonent\_t\* \_output\_mass,int);

/// <summary>

/// Печать информационной помощи

/// </summary>

/// <param name="help\_message">Сообщение помощи. Поддерживает изменение цвета</param>

/// <returns></returns>

int print\_help(char\* help\_message);

/// <summary>

/// Сортирует данные согласно указанной структуре

/// </summary>

/// <param name="">Указатель на данные (массив)</param>

/// <param name="">Количество данных в массиве</param>

/// <param name="">Структура хранящаяя текущюю сортировку</param>

/// <returns>Указатель на новые отсортированные данные</returns>

abonent\_t\* \_sort\_output(abonent\_t\*, int \*,sort\_struct \*);

/// <summary>

/// Окно просмотра должников

/// </summary>

/// <param name="root">Корень дерева для получения информации</param>

void dolgiWindow(abonent\*\* root);

#endif

#ifndef KEYCODE

#define KEYCODE

/// <summary>

/// Енум хранящий коды кнопок

/// </summary>

enum KeyboardCodes

{

KEY\_ARROW\_UP = 72,

KEY\_ARROW\_DOWN = 80,

KEY\_ARROW\_LEFT = 75,

KEY\_ARROW\_RIGHT = 77,

KEY\_TAB = 9,

KEY\_HOME = 71,

KEY\_END = 79,

KEY\_ENTER = 13,

KEY\_ESC = 27,

KEY\_DEL = 83,

KEY\_BACKSPACE = 8,

KEY\_PGUP = 73,

KEY\_PGDOWN = 81

};

#endif

/// <summary>

/// Енум , хранящий поля сортировок

/// </summary>

enum sort\_fild {

DEF,

FIO,

AUTHOR,

BOOK\_NAME,

IZD,

DATE\_OUT,

COST,

ZADANIE

};

enum sort\_type {

UP,

DOWN

};

**Исходный код файла «Cursach\_main.cpp»**

#pragma warning(disable : 4996);

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#include <Windows.h>

#include "utf8.h"

#include "include/data\_utils.h"

#include "ThreeStruct.h"

#include "VicMenuDLL.h"

#include "MenuStruct.h"

#include "tree\_operation.h"

#define clearf() system("cls");

const int size\_abonent\_t = sizeof(abonent\_t);

const int menu\_size = 5;

HANDLE hConsole;

int аddNewElement(abonent\*\* st, \_tabel\_metadata \*table);

int create\_file\_type(FILE\* f, abonent\* St);

abonent\* loadFromFile\_new\_type(FILE\* f);

abonent\* loadFromFile\_new\_text(FILE\* f);

int correctInfo(abonent\* st);

void MenuSelect(int selector, FILE\* f, \_tabel\_metadata\* table);

int correctInfo(abonent\* st);

\_menu\_item\* \_init\_menu(\_menu\_item\* menu);

\_tabel\_metadata\* \_init\_table(\_tabel\_metadata\* table);

abonent\* abonents = NULL;

int position[] = { 1,1 };

int\* size;

int main(void) {

hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO structCursorInfo;

GetConsoleCursorInfo(hConsole, &structCursorInfo);

structCursorInfo.bVisible = FALSE;

SetConsoleCursorInfo(hConsole, &structCursorInfo);

system("chcp 1251");

SetConsoleCP(1251); // Задаем таблицу символов для консоли.

SetConsoleOutputCP(65001);

clearf();

\_menu\_item\* menu = NULL;

\_tabel\_metadata\* table = NULL;

menu = \_init\_menu(menu);

table = \_init\_table(table);

abonent\_t\* \_output\_info;

// system("color F0");

FILE\* f = fopen("data.dat", "rb+");//Открытие существующего файла для чтения и записи в конец

if (!f) {

f = fopen("data.dat", "wb+"); //Создание нового файла для обновления

if (!f) {

puts("Не могу открыть (создать) файл\n");

return 1;

}

}

while (1) {//вывод меню и запуск соответствующих функций

clear();

int leafCount = tree\_getNodeCount(abonents, 0);

\_output\_info = (abonent\_t\*)calloc(leafCount, sizeof(abonent\_t));

int temp = 0;

\_output\_info = \_get\_output\_info(abonents, \_output\_info,&temp);

if (leafCount == 0)

\_output\_info = NULL;

static sort\_struct sort;

sort.sort\_f = DEF;

sort.sort\_t = UP;

MenuSelect(\_print\_menu(menu, position, menu\_size, 5,\_output\_info,leafCount,table, &abonents, &sort), f,table);

}

}

int Posid = 1;

void MenuSelect(int selector, FILE\* f,\_tabel\_metadata \*table )

{

char\* a = (char\*)calloc(200, sizeof(char));

switch (selector) {

case ADD\_NEW\_RECORD:

аddNewElement(&abonents,table);

break;

case LOAD\_FROM\_FILE\_TYPE:

if (\_confirm\_window(NULL)) {

f = fopen("data.dat", "rb+");

abonents = loadFromFile\_new\_type(f);

}

break;

case SAVE\_TO\_FILE\_TYPE:

create\_file\_type(f, abonents);

break;

case CLEAN\_TREE:

size = \_get\_window\_size(size);

if (\_confirm\_window(NULL)) {

abonents = tree\_delete(abonents);

sprintf(a, "Дерево очищено");

\_message\_window( a);

getch();

}

break;

case PRINT\_TREE\_STRUCT:

clear();

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

View(abonents, 1);

printf("-------------------------------------------- Конец струк. дерева ----------------------------------------\n");

getch();

break;

case TREE\_SIZE:

if (\_confirm\_window(NULL)) {

sprintf(a, "Дерево содержит %d записей.",tree\_getNodeCount(abonents, 0));

\_message\_window(a);

getch();

}

break;

case PRINT\_HORRIBLE\_ANIMATION:

animatedNeko();

break;

case LOAD\_FROM\_FILE:

if (\_confirm\_window(NULL)) {

\_in\_window();

char a[90]; scanf("%s", a);

sprintf(a, "%s.txt", a);

FILE\* text = fopen(a, "rt");

if (!text) break;

abonents = loadFromFile\_new\_text(text);

fclose(text);

}

break;

case SAVE\_TO\_FILE:

if (\_confirm\_window(NULL)) {

\_in\_window();

char a[90]; scanf("%s", a);

sprintf(a, "%s.txt", a);

FILE\* text = fopen(a, "wt");

if (!text) break;

printToFile\_Text(text, abonents);

fclose(text);

}

break;

case DOLGNIKI\_WINDWO:

if (\_confirm\_window(NULL)) {

dolgiWindow(&abonents);

}

break;

case PROGRAM\_EXIT:

if (\_confirm\_window(NULL)) {

if (!abonents) {

tree\_delete(abonents);

}

exit(666);

}

break;

}

}

int create\_file\_type(FILE\* f, abonent\* root)

{

if (!root)

return 666;

fclose(f);

f = fopen("data.dat", "wb+");

if (f == NULL)

{

\_message\_window("Не удалось открть файл для записи");

Sleep(2000);

return 666;

}

fseek(f, 0, SEEK\_SET);

printToFile(f, root);

fclose(f);

\_message\_window("Данные успешно сохранены...");

Sleep(2000);

return 0;

}

int аddNewElement(abonent\*\* st,\_tabel\_metadata\* table)

{

abonent\_t\* d;

while (1) {

d = \_in\_info\_window(table, NULL, 1);

if (!d) return EXIT\_SUCCESS;

d->id = util\_hashCodeFromFio(&d->fio);

tree\_add(st, d);

}

return EXIT\_SUCCESS;

}

abonent\* loadFromFile\_new\_text(FILE\* f)

{

abonent\_t tmp;

abonent\* head = NULL;

int count = 1;

while (1) {

fscanf(f, "%s %s %s", tmp.fio.surname, tmp.fio.name, tmp.fio.secondname);

fgets(tmp.book\_name, 160, f);

fscanf(f, "%s %s", tmp.autor.surname, tmp.autor.inicial);

fgets(tmp.izd, 70, f);

fscanf(f, "%d %d %d", &(tmp.date\_out.d), &(tmp.date\_out.m), &(tmp.date\_out.y));

fscanf(f, "%f", &(tmp.cost));

if (feof(f)) break;

for (int i = 0; i < strlen(tmp.book\_name); i++) {

if (tmp.book\_name[i] == '\n') tmp.book\_name[i] = '\0';

}

for (int i = 0; i < strlen(tmp.izd); i++) {

if (tmp.izd[i] == '\n') tmp.izd[i] = '\0';

}

tmp.id = util\_hashCodeFromFio(&tmp.fio);

tree\_add(&head, &tmp);

count++;

}

\_message\_window("Данные считаны");

Sleep(2000);

return head;

}

abonent\* loadFromFile\_new\_type(FILE\* f)

{

abonent\_t tmp;

abonent\* head = NULL;

int count = 1;

fseek(f, 0, SEEK\_SET);

while (fread(&tmp, sizeof(abonent\_t), 1, f))

{

tree\_add(&head, &tmp);

count++;

}

\_message\_window("Данные считаны");

Sleep(2000);

return head;

}

int correctInfo(abonent\* st)

{

return EXIT\_SUCCESS;

}

\_menu\_item\* \_init\_menu(\_menu\_item\* menu) {

menu = (\_menu\_item \*) calloc(1, sizeof(\_menu\_item) \* 5);

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

strcpy(menu[0].\_name, "Внести данные");

menu[0].\_menu\_name\_lenght = 14;

menu[0].\_menu\_size = 3;

menu[0].\_sub\_menu = (char\*\*)calloc(menu[0].\_menu\_size, sizeof(char\*) );

for (int i = 0; i < menu[0].\_menu\_size; i++) {

menu[0].\_sub\_menu[i] = (char\*)calloc(90, sizeof(char));

}

menu[0].\_sub\_menu\_lenght = (int\*)calloc(3, sizeof(int));

strcpy(menu[0].\_sub\_menu[0], "Добавить новый элемент c клавиатуры"); menu[0].\_sub\_menu\_lenght[0] = 36;

strcpy(menu[0].\_sub\_menu[1], "Загрузить из типизированного файла"); menu[0].\_sub\_menu\_lenght[1] = 35;

strcpy(menu[0].\_sub\_menu[2], "Загрузить из текстового файла"); menu[0].\_sub\_menu\_lenght[2] = 30;

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

strcpy(menu[1].\_name, "Сохранить");

menu[1].\_menu\_name\_lenght = 10;

menu[1].\_menu\_size = 2;

menu[1].\_sub\_menu = (char\*\*)calloc(menu[1].\_menu\_size, sizeof(char\*) );

for (int i = 0; i < menu[1].\_menu\_size; i++) {

menu[1].\_sub\_menu[i] = (char\*)calloc(90, sizeof(char));

}

menu[1].\_sub\_menu\_lenght = (int\*)calloc(4, sizeof(int));

strcpy(menu[1].\_sub\_menu[0], "Сохранить в типизированный файл"); menu[1].\_sub\_menu\_lenght[0] = 32;

strcpy(menu[1].\_sub\_menu[1], "Сохранить в текстовый файл"); menu[1].\_sub\_menu\_lenght[1] = 27;

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

strcpy(menu[2].\_name, "Работа с деревом");

menu[2].\_menu\_name\_lenght = 17;

menu[2].\_menu\_size = 4;

menu[2].\_sub\_menu = (char\*\*)calloc(menu[2].\_menu\_size, sizeof(char\*));

for (int i = 0; i < menu[2].\_menu\_size; i++) {

menu[2].\_sub\_menu[i] = (char\*)calloc(90, sizeof(char));

}

menu[2].\_sub\_menu\_lenght = (int\*)calloc(5, sizeof(int));

strcpy(menu[2].\_sub\_menu[0], "Отобразить структуру дерева"); menu[2].\_sub\_menu\_lenght[0] = 28;

strcpy(menu[2].\_sub\_menu[1], "Колличество элементов в дереве"); menu[2].\_sub\_menu\_lenght[1] = 31;

strcpy(menu[2].\_sub\_menu[2], "Очистить дерево"); menu[2].\_sub\_menu\_lenght[2] = 16;

strcpy(menu[2].\_sub\_menu[3], "Список должников"); menu[2].\_sub\_menu\_lenght[3] = 17;

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

strcpy(menu[3].\_name, "Прочее");

menu[3].\_menu\_name\_lenght = 7;

menu[3].\_menu\_size = 2;

//"Анимация",

//"Картинка",

menu[3].\_sub\_menu = (char\*\*)calloc(menu[3].\_menu\_size, sizeof(char\*) );

for (int i = 0; i < menu[3].\_menu\_size; i++) {

menu[3].\_sub\_menu[i] = (char\*)calloc(90, sizeof(char));

}

menu[3].\_sub\_menu\_lenght = (int\*)calloc(2, sizeof(int));

strcpy(menu[3].\_sub\_menu[0], "Анимация"); menu[3].\_sub\_menu\_lenght[0] = 9;

strcpy(menu[3].\_sub\_menu[1], "Картинка"); menu[3].\_sub\_menu\_lenght[1] = 9;

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

strcpy(menu[4].\_name, "Выход");

menu[4].\_menu\_name\_lenght = 6;

menu[4].\_menu\_size = 0;

//"Выход"

menu[4].\_sub\_menu = NULL;

return menu;

}

\_tabel\_metadata\* \_init\_table(\_tabel\_metadata\* table) {

table = (\_tabel\_metadata\*)calloc(sizeof(\_tabel\_metadata), 1);

table->\_col\_count = 7;

table->\_cols = (\_table\_col\*)calloc(sizeof(\_table\_col) , table->\_col\_count);

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

table->\_cols[0].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[0].name, "№");

table->\_cols[0].resizebl = 0;

table->\_cols[0].size = u8\_strlen(table->\_cols[0].name);

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

table->\_cols[1].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[1].name, "ФИО абонента");

table->\_cols[1].resizebl = 1;

table->\_cols[1].size = u8\_strlen(table->\_cols[1].name);

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

table->\_cols[2].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[2].name, "Автор книги");

table->\_cols[2].resizebl = 0;

table->\_cols[2].size = u8\_strlen(table->\_cols[2].name); table->\_cols[2].size =18;

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

table->\_cols[3].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[3].name, "Название книги");

table->\_cols[3].resizebl = 1;

table->\_cols[3].size = u8\_strlen(table->\_cols[3].name);

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

table->\_cols[4].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[4].name, "Издательство");

table->\_cols[4].resizebl = 0;

table->\_cols[4].size = u8\_strlen(table->\_cols[4].name);

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

table->\_cols[5].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[5].name, "Дата выдачи");

table->\_cols[5].resizebl = 0;

table->\_cols[5].size = u8\_strlen(table->\_cols[5].name);

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

table->\_cols[6].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[6].name, "Стоимость,р.");

table->\_cols[6].resizebl = 0;

table->\_cols[6].size = u8\_strlen(table->\_cols[6].name);

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

return table;

}

**Исходный код файла «data\_utils.c»**

#include "include/data\_utils.h"

unsigned int util\_hashCodeFromAbonentStruct(const abonent\_t\* a)

{

return util\_hashCodeFromFio(&a->fio);

}

unsigned int util\_hashCodeFromFio(const fio\_t\* fio)

{

return util\_hashCodeFromString(fio->name) + util\_hashCodeFromString(fio->secondname) + util\_hashCodeFromString(fio->surname);

}

unsigned int util\_hashCodeFromString(const char\* string)

{

unsigned int ret = 0;

for (int position = 0; string[position] != '\0'; ++position)

{

ret += string[position];

}

return ret;

}

**Исходный код файла «input\_utils.cpp»**

#pragma warning(disable : 4996);

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <conio.h>

#include <Windows.h>

#include <time.h>

#include "utf8.h"

#include "input\_utils.h"

BOOL \_get\_con\_info\_local(CONSOLE\_SCREEN\_BUFFER\_INFO\* x)

{

return GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), x);

}

BOOL \_set\_cur\_to\_pos\_local(COORD cor) {

return SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), cor);

}

char convert\_u8\_to\_1251(int c) {

wchar\_t s[2];

s[0] = (wchar\_t)c; s[1] = '\0';

char utf8[20];

wchar\_t wstr[20];

char s1251[20];

WideCharToMultiByte(CP\_UTF8, 0, s, -1, utf8, 100, NULL, NULL);

utf8[WideCharToMultiByte(CP\_UTF8, 0, s, -1, utf8, 0, NULL, NULL)] = '\0';

// Подготовили строку UTF8 дальше идет ее преобразование в 1251

MultiByteToWideChar(CP\_UTF8, 0, utf8, -1, wstr, 100);

wstr[MultiByteToWideChar(CP\_UTF8, 0, utf8, -1, wstr, 0)];

WideCharToMultiByte(1251, 0, wstr, -1, s1251, 100, NULL, NULL);

s1251[WideCharToMultiByte(1251, 0, wstr, -1, s1251, 0, NULL, NULL)] = '\0';

return s1251[0];

}

int input\_float(float \* info) {

int flag\_point = 0;

char correct\_sym[] = { '0','1','2','3' ,'4','5' ,'6','7' ,'8','9','.' };

SetConsoleCP(65001);

SetConsoleOutputCP(1251);

CONSOLE\_SCREEN\_BUFFER\_INFO con\_inf; \_get\_con\_info\_local(&con\_inf);

int current\_pos = 0;

int in\_sym\_count = 0;

char input\_buff[400] = {""};

float old\_info = \*info;

COORD positionCur = { 0,0 };

COORD positionCur\_start = { 0,0 };

positionCur = con\_inf.dwCursorPosition;

positionCur\_start = con\_inf.dwCursorPosition;

if (u8\_strlen(input\_buff) > 0) {

int k = u8\_strlen(input\_buff);

current\_pos = k;

in\_sym\_count = k;

// positionCur.X = positionCur.X - k;

\_set\_cur\_to\_pos\_local(positionCur);

printf("%s", input\_buff);

positionCur = con\_inf.dwCursorPosition;

}

int c;

while (1) {

c = \_getwch();

switch (c)

{

case KEY\_ENTER:

if (in\_sym\_count > 0) {

printf(" ");

\*info = atof(input\_buff);

return KEY\_ENTER;

}

break;

case KEY\_ESC:

//Вернуть старую инфу

return KEY\_ESC;

break;

case KEY\_BACKSPACE:

{

if (current\_pos > 0 && in\_sym\_count > 0) {

if (current\_pos > 0)

{

if (input\_buff[current\_pos-1] == '.') flag\_point = 0;

input\_buff[current\_pos-1] = '\0';

current\_pos--; in\_sym\_count--;

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

}

}

}

break;

default:

c = convert\_u8\_to\_1251(c);

{

int flag = 1;

for (int i = 0; i < 11; i++)

{

if (c == correct\_sym[i]) {

flag = 0;

break;

}

}

if (c == '.' && flag\_point) flag = 1;

if (c == '.') {

flag\_point = 1;

}

if (c == ' ' )

{

if (in\_sym\_count > 0) {

printf(" ");

return KEY\_ENTER;

}

}

if (!flag) {

if (in\_sym\_count < 400) {

input\_buff[in\_sym\_count] = c;

printf("%c", c);

in\_sym\_count++;

current\_pos++;

}

}

}

break;

}

}

}

/// <summary>

/// Ввод любых строковых данных

/// </summary>

/// <param name="input\_buff">Буфер для ввода</param>

/// <param name="buff\_size">Размер буфера</param>

/// <param name="mode">Режим работы, от которого зависят допустимые символы. 0 - разрешение на ввод всего, кроме спец символов, 1- для реализации ввода Фамилии\имени\Отчества\Издательства(не допускаются цифры и т.д),2 - Ввод инициалов</param>

/// <returns></returns>

int input\_string(char\* input\_buff, int buff\_size, int mode)

{

int n\_count;

char\* invalid\_sym;

int temp = 0;

switch (mode)

{

case NORMAL:

{

n\_count = 5;

invalid\_sym = (char\*)calloc(n\_count, sizeof(char));

for (int i = 35; i < 40; i++) {

invalid\_sym[temp] = i;

temp++;

}

break;

}

case PERSONAL: {

n\_count = 42;

invalid\_sym = (char\*)calloc(n\_count, sizeof(char));

for (int i = 32; i < 65; i++) {

invalid\_sym[temp] = i;

temp++;

}

for (int i = 91; i < 97; i++) {

invalid\_sym[temp] = i;

temp++;

}

for (int i = 123; i < 127; i++) {

invalid\_sym[temp] = i;

temp++;

}

break; }

case INICIAL: {

n\_count = 42;

invalid\_sym = (char\*)calloc(n\_count, sizeof(char));

for (int i = 32; i < 65; i++) {

invalid\_sym[temp] = i;

temp++;

}

for (int i = 91; i < 97; i++) {

invalid\_sym[temp] = i;

temp++;

}

for (int i = 123; i < 127; i++) {

invalid\_sym[temp] = i;

temp++;

}

break; }

default:

{

n\_count = 5;

invalid\_sym = (char\*)calloc(n\_count, sizeof(char));

for (int i = 35; i < 40; i++) {

invalid\_sym[temp] = i;

temp++;

}

break;

}

}

SetConsoleCP(65001);

SetConsoleOutputCP(1251);

CONSOLE\_SCREEN\_BUFFER\_INFO con\_inf; \_get\_con\_info\_local(&con\_inf);

int current\_pos = 0;

int in\_sym\_count = 0;

char\* old\_info = (char\*)calloc(buff\_size, sizeof(char));

strcpy(old\_info, input\_buff);

//old\_info = \*input\_buff;

COORD positionCur = { 0,0 };

COORD positionCur\_start = { 0,0 };

positionCur = con\_inf.dwCursorPosition;

positionCur\_start = con\_inf.dwCursorPosition;

if (u8\_strlen(input\_buff) > 0) {

int k = u8\_strlen(input\_buff);

current\_pos = k;

in\_sym\_count = k;

// positionCur.X = positionCur.X - k;

\_set\_cur\_to\_pos\_local(positionCur);

printf("%s", input\_buff);

positionCur = con\_inf.dwCursorPosition;

}

int c;

while (1) {

c = \_getwch();

switch (c)

{

case KEY\_ENTER:

if (in\_sym\_count > 0) {

printf(" ");

return KEY\_ENTER;

}

break;

case KEY\_ESC:

strcpy(input\_buff, old\_info);

return KEY\_ESC;

break;

case KEY\_ARROW\_UP:

if (in\_sym\_count > 0) {

strcpy(input\_buff, old\_info);

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

return KEY\_ARROW\_UP;

}

break;

case KEY\_ARROW\_DOWN:

if (in\_sym\_count > 0) {

strcpy(input\_buff, old\_info);

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

return KEY\_ARROW\_DOWN;

}

break;

case KEY\_ARROW\_LEFT:

case KEY\_ARROW\_RIGHT:

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

break;

case KEY\_BACKSPACE:

if (mode != INICIAL) {

if (current\_pos > 0 && in\_sym\_count > 0)

{

input\_buff[current\_pos] = '\0';

current\_pos--; in\_sym\_count--;

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

}

}

else

{

if (current\_pos > 0 && in\_sym\_count > 0) {

if (current\_pos>2)

{

input\_buff[current\_pos] = '\0';

current\_pos--; in\_sym\_count--;

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

}

else {

input\_buff[current\_pos] = '\0';

current\_pos--;

input\_buff[current\_pos] = '\0';

current\_pos--;

in\_sym\_count--; in\_sym\_count--;

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" "); printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

}

}

}

break;

default:

c = convert\_u8\_to\_1251(c);

if (mode == INICIAL) {

int flag = 0;

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

{

if (c == invalid\_sym[i]) {

flag = 1;

break;

}

}

if (!flag) {

if (in\_sym\_count < buff\_size) {

input\_buff[in\_sym\_count] = c;

if (in\_sym\_count == 0) {

in\_sym\_count++;

current\_pos++;

input\_buff[in\_sym\_count] = '.';

printf("%c.", c);

}

else printf("%c", c);

in\_sym\_count++;

current\_pos++;

}

}

}

else {

int flag = 0;

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

{

if (c == invalid\_sym[i]) {

flag = 1;

break;

}

}

if (c == ' ' && mode == PERSONAL)

{

if (in\_sym\_count > 0) {

printf(" ");

return KEY\_ENTER;

}

}

if (!flag) {

if (in\_sym\_count < buff\_size) {

input\_buff[in\_sym\_count] = c;

printf("%c", c);

in\_sym\_count++;

current\_pos++;

}

}

}

break;

}

}

}

int in\_date(int\* d, int\*m,int \*y) {

int n\_count;

char\* correct\_sym;

int temp = 0;

n\_count = 10;

char buff[10] = "";

correct\_sym = (char\*)calloc(n\_count, sizeof(char));

for (int i = 48; i < 59; i++) {

correct\_sym[temp] = i;

temp++;

}

SetConsoleCP(65001);

SetConsoleOutputCP(1251);

CONSOLE\_SCREEN\_BUFFER\_INFO con\_inf; \_get\_con\_info\_local(&con\_inf);

int current\_pos = 0;

int in\_sym\_count = 0;

int cur\_selector = 1;

int old\_d, old\_m, old\_y;

if (d) old\_d = \*d;

if(m)old\_m = \*m;

if (y) old\_y = \*y;

COORD positionCur = { 0,0 };

COORD positionCur\_start = { 0,0 };

positionCur = con\_inf.dwCursorPosition;

printf("\_\_\_\_\_\_\_\_\_\_\_");

\_set\_cur\_to\_pos\_local(positionCur);

positionCur\_start = con\_inf.dwCursorPosition;

int c;

while (1)

{

c = \_getwch();

switch (c)

{

case KEY\_ENTER:

if (cur\_selector > 3) {

printf(" ");

return KEY\_ENTER;

}

break;

case KEY\_ESC:

return KEY\_ESC;

break;

case KEY\_ARROW\_UP:

if (in\_sym\_count > 0) {

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

return KEY\_ARROW\_UP;

}

break;

case KEY\_ARROW\_DOWN:

if (in\_sym\_count > 0) {

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

return KEY\_ARROW\_DOWN;

}

break;

case KEY\_ARROW\_LEFT:

case KEY\_ARROW\_RIGHT:

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

break;

case KEY\_BACKSPACE:

{

if (current\_pos > 0 && in\_sym\_count > 0) {

{

buff[current\_pos] = '\0';

current\_pos--;

in\_sym\_count--; int flag = 0;

if (current\_pos <= 4 && current\_pos > 2) {

if (cur\_selector > 2) {

cur\_selector--;

flag = 1;

}

}

else if (current\_pos < 2) {

if (cur\_selector > 1) {

cur\_selector--;

flag = 1;

}

}

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

if (flag) {

positionCur.X--; positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" "); printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

}

else {

positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

}

}

}

}

break;

default: {

c = convert\_u8\_to\_1251(c);

int flag = 0;

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

{

if (c == correct\_sym[i]) {

flag = 1;

break;

}

}

if (flag) {

if (cur\_selector < 4) {

if (in\_sym\_count >= 2 && in\_sym\_count < 4) {

buff[in\_sym\_count-2] = c;

}

else if (in\_sym\_count >= 4){

buff[in\_sym\_count-4] = c;

}

else

buff[in\_sym\_count] = c;

printf("%c", c);

in\_sym\_count++;

if (in\_sym\_count == 2) {

\*d = atoi(buff);

if (\*d > 31) {

\*d = 31;

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" "); printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

printf("%d", \*d);

}

printf(".");

memset(buff, 0, strlen(buff));

cur\_selector++;

}

else if (in\_sym\_count == 4) {

\*m = atoi(buff);

if (\*m > 12) { \*m = 12;

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X--; positionCur.X--; \_set\_cur\_to\_pos\_local(positionCur);

printf(" "); printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

printf("%d", \*m);

}

printf(".");

memset(buff, 0, strlen(buff));

cur\_selector++;

}

else if (in\_sym\_count == 8) {

// printf(".");

\*y = atoi(buff);

time\_t now = time(0);

struct tm\* ltm = localtime(&now);

if (\*y > ltm->tm\_year + 1900) { \*y = ltm->tm\_year + 1900;

\_get\_con\_info\_local(&con\_inf);

positionCur.X = con\_inf.dwCursorPosition.X;

positionCur.X-=4; \_set\_cur\_to\_pos\_local(positionCur);

printf(" ");

\_set\_cur\_to\_pos\_local(positionCur);

printf("%d", \*y);

}

return KEY\_ENTER;

}

current\_pos++;

}

}

}

break;

}

}

}

**Исходный код файла «tree\_operation.cpp»**

#include "tree\_operation.h"

#include "VicMenuDLL.h"

#include <conio.h>

#include <time.h>

#include <Windows.h>

#include <stdlib.h>

#include <stdio.h>

abonent\* tree\_getLeafById(abonent\* root, const int id)

{

if (root == NULL)

return NULL;

else if (root->info.id == id)

return root;

else if (root->info.id < id)

return tree\_getLeafById(root->right, id);

else

return tree\_getLeafById(root->left, id);

}

void tree\_deleteNodeById(abonent\*\* root, const int id)

{

if (root == NULL || \*root == NULL)

return; // выход если пустой узел

if ((\*root)->info.id == id)

{

if ((\*root)->right == NULL)

{

abonent\* t = \*root;

\*root = (\*root)->left;

free(t);

}

else

{

if ((\*root)->right->left == NULL)

{

abonent\* tmp = (\*root)->right;

(\*root)->right->left = (\*root)->left;

free(\*root);

\*root = tmp;

}

else

{

abonent\*\* candidate\_prev = &((\*root)->right);

for (; (\*candidate\_prev)->left->left != NULL; candidate\_prev = &((\*candidate\_prev)->left));

abonent\* candidate = (\*candidate\_prev)->left;

(\*candidate\_prev)->left = candidate->right;

candidate->left = (\*root)->left;

candidate->right = (\*root)->right;

free(\*root);

\*root = candidate;

}

}

}

else

{

//бинарный поиск в левом или правом поддереве

tree\_deleteNodeById

(

(\*root)->info.id > id ?

&((\*root)->left) :

&((\*root)->right),

id

);

}

}

int tree\_getNodeCount(const abonent\* root, const int accum) {

if (root != NULL)

return tree\_getNodeCount(root->left, accum) + 1 + tree\_getNodeCount(root->right, accum);

else

return 0;

}

void View(abonent\* top, int offset) {

if (top)

{

offset += 3; //отступ от края экрана

View(top->right, offset); //обход правого поддерева

for (int i = 0; i < offset; i++) printf(" ");

printf("|%d\n", top->info.id);

View(top->left, offset); //обход левого поддерева

}

}

void tree\_add(abonent\*\* root, const abonent\_t\* info)

{

if (root == NULL || info == NULL)

{

return;

}

if (\*root == NULL)

{

\*root = (abonent\*)calloc(sizeof(abonent), 1);

if (\*root != NULL)

{

(\*root)->info = \*info;

}

}

else

{

/\*if (((\*root)->info.id == info->id)) {

\_message\_window("Запись с таким id уже существует\0");

Sleep(2000);

return;

}

else \*/

if (((\*root)->info.id > info->id))

tree\_add(&((\*root)->left), info);

else

tree\_add(&((\*root)->right), info);

}

}

abonent\* tree\_delete(abonent\* root) {

if (root) {

if (root->left) {

tree\_delete(root->left);

}

if (root->right) {

tree\_delete(root->right);

}

free(root);

}

return NULL;

}

void printToFile(FILE\* f, abonent\* root)

{

if (root) {

abonent\_t te = root->info;

fwrite(&te, size\_abonent\_t, 1, f);

if (root->left) {

printToFile(f, root->left);

}

if (root->right) {

printToFile(f, root->right);

}

}

}

void printToFile\_Text(FILE\* f, abonent\* root)

{

if (root) {

abonent\_t te = root->info;

fprintf(f, "%s %s %s %s\n%s %s %s\n%d %d %d %f\n", te.fio.surname, te.fio.name, te.fio.secondname, te.book\_name, te.autor.surname, te.autor.inicial, te.izd, te.date\_out.d, te.date\_out.m, te.date\_out.y, te.cost);

if (root->left) {

printToFile\_Text(f, root->left);

}

if (root->right) {

printToFile\_Text(f, root->right);

}

}

}

abonent\_t\* \_get\_output\_info(abonent\* root, abonent\_t\* \_output\_memory, int\* index) {

{

if (root) {

if (root->left) {

\_get\_output\_info(root->left, \_output\_memory, index);

}

\_output\_memory[\*index] = root->info;

(\*index)++;

if (root->right) {

\_get\_output\_info(root->right, \_output\_memory, index);

}

return \_output\_memory;

}

}

}

dolgi\_pers\_t\_obr\* \_get\_dolgi\_info(abonent\* root, dolgi\_pers\_t\_obr\* \_output\_memory, int d, int m, int y) {

{

if (root) {

if (root->left) {

\_get\_dolgi\_info(root->left, \_output\_memory, d, m, y);

}

if ((((y \* 365) + (m \* 31) + d) - ((root->info.date\_out.y) \* 365) - ((root->info.date\_out.m) \* 31) - (root->info.date\_out.d)) > 90 ){

if (!\_output\_memory) {

\_output\_memory->info\_mass = (dolgi\_pers\_t\*)calloc(sizeof(dolgi\_pers\_t), 1);

\_output\_memory->info\_mass[0].id = root->info.id;

\_output\_memory->info\_mass[0].fio = root->info.fio;

\_output\_memory->info\_mass[0].count\_dolg\_books++;

\_output\_memory->count++;

}

else {

int flag = 0;

for (int i = 0; i < \_output\_memory->count; i++ ) {

if (\_output\_memory->info\_mass[i].id == root->info.id) {

\_output\_memory->info\_mass[i].count\_dolg\_books++;

flag = 1;

}

}

if (!flag) {

\_output\_memory->info\_mass = (dolgi\_pers\_t\*)realloc(\_output\_memory->info\_mass, sizeof(dolgi\_pers\_t) \* ((\_output\_memory->count) + 1));

\_output\_memory->count++;

\_output\_memory->info\_mass[(\_output\_memory->count) - 1].count\_dolg\_books = 1;

\_output\_memory->info\_mass[(\_output\_memory->count) - 1].fio = root->info.fio;

\_output\_memory->info\_mass[(\_output\_memory->count) - 1].id = root->info.id;

}

}

}

if (root->right) {

\_get\_dolgi\_info(root->right, \_output\_memory, d,m,y);

}

//return \_output\_memory;

}

return \_output\_memory;

}

}

**Исходный код файла «VicMenuDLL.cpp»**

#pragma warning(disable : 4996)

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#include <Windows.h>

#include "utf8.h"

#include "VicMenuDLL.h"

#include "MenuStruct.h"

#include "tree\_operation.h"

#include "input\_utils.h"

const int \_otstup = 3; //Хранит отступ слева и справа в окне

const int \_interval = 3; //хранит интервал сверху и снизу окна

int\* \_window\_size; //указатель на массив размера

int \_first\_start = 1; //флаг первого запуска.

COORD positionCur = {4,4}; //Хранит текущую позицию курсора

#define clearf() system("cls"); // Полная очистка экрана

#define KEY\_ENTER 13 // дефайны для кнопок

#define KEY\_ESC 27

int page = 1; // хранит текущую страницу.

BOOL \_get\_con\_info(CONSOLE\_SCREEN\_BUFFER\_INFO\* x) //Внутренняя функция, получает информацию из буфера экрана.

{

return GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), x);

}

BOOL \_set\_cur\_to\_pos(HANDLE h, COORD cor) { //Устанавливает курсор на необходимую позицию

return SetConsoleCursorPosition(h, cor);

}

void clear() { //функция кастомной очистки , чистит только окно

positionCur.X = \_otstup + 1; positionCur.Y = \_interval + 1; //установить стартовые позиции

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE); //получить хендлер консоли

int\* temp\_window\_size = NULL; //получить размеры окна

temp\_window\_size = \_get\_window\_size(temp\_window\_size);

\_set\_cur\_to\_pos(hConsole, positionCur); // установить курсор в позицию

\_print\_border(temp\_window\_size[0], temp\_window\_size[1]); // напечатать внутренний бордер

positionCur.X = \_otstup + 1; positionCur.Y = \_interval + 1; //данные курсора обнулить в исходное состояние

\_set\_cur\_to\_pos(hConsole, positionCur); // установить курсор в исходное состояние

}

void clear\_table() { //Внутренняя функция очистки таблицы

positionCur.X = \_otstup + 2; positionCur.Y = \_interval + 3; //ставим курсор согласно параметрам положения таблицы

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE); //получаем хендлер консоли

int\* temp\_window\_size = NULL; // получаем размер кона

temp\_window\_size = \_get\_window\_size(temp\_window\_size);

\_set\_cur\_to\_pos(hConsole, positionCur); int \_window\_h = temp\_window\_size[1];

int \_window\_w = temp\_window\_size[0]; //запомнинаем рахмеры , ставим курсор в нужную позицию

for (int y = \_interval+3; y <= \_window\_h - \_interval-3; y++) { // цикл очистки таблицы

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int x = \_otstup + 2; x < \_window\_w - \_otstup-1; x++) {

printf(" "); //\_otstup+2,\_interval+3

}

positionCur.Y++;

}

}

void clear\_for\_info() { //установка курсора в позицию для вывода информации

positionCur.X = \_otstup + 4; positionCur.Y = \_interval + 4;

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

\_set\_cur\_to\_pos(hConsole, positionCur);

}

int\* \_get\_curent\_selection(char c // Символ клавиатуры

,int \* position // Массив в котором хранятся x и y

,int MaxY // Максимальный количество строк

, int Colums // Количество столбцов, по умолчанию - 1

,int \_flag\_x\_readonly //можно ли менять x?

)

{

int x = position[0]; int y = position[1];

switch (c)

{

case KEY\_ARROW\_UP://вверх

if (y > 1) y--;

break;

case KEY\_ARROW\_DOWN://вниз

if (y < MaxY) y++;

break;

case KEY\_ARROW\_LEFT://лево

if (x > 1) x--;

if(\_flag\_x\_readonly) if (y > 1) y--;

break;

case KEY\_ARROW\_RIGHT://право

if (x < Colums) x++;

if (\_flag\_x\_readonly) if (y < MaxY) y++;

break;

}

if (!\_flag\_x\_readonly)

position[0] = x; //запоминаем позицию по столбцу

position[1] = y;//позиция по строке

return position;

}

int \_print\_menu(\_menu\_item\* \_menu //Массив объектов меню

, int\* position //Массив текущей позиции x и y

, int \_menu\_size //Колличество элементов в массиве объектов меню

, int \_menu\_buttons,//Количество кнопок меню

abonent\_t \* \_output\_mas, //массив для вывода информации

int \_output\_colcount, // количество данных для вывода

\_tabel\_metadata \* table, //структура хранящая информацию о полях таблицы

abonent\*\* root, //указатель на указатель на корень дерева

sort\_struct\* sort // структура, хранящая текущую сортировку

)

{

int table\_focus\_flag = 0; // флаг работы с таблицей

int \_padding; //оступ

int \_new\_padding; //новый отступ

int\* \_size\_now = NULL; //текущий размер окна

if (\_first\_start) { // Если у нас первый запуск

\_window\_size = \_get\_window\_size(\_window\_size); //порлучаем размер окна

\_print\_bakground(\_window\_size[0], \_window\_size[1]); // вызываем отрисовку заднего фона

\_first\_start = 0; // первый запуск закончен

}

COORD positionCur = { \_otstup + 1, \_interval + 1 }; //позиция x и y у курсора. Стартовый отступ

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE); //получаем хендлер консоли

\_set\_cur\_to\_pos(hConsole, positionCur);// установим курсор в позици.

CONSOLE\_SCREEN\_BUFFER\_INFO info\_x; //необходимо для получения размера консоли

int \_max\_subm\_lenght = 0; //поиск максимальной длины записи в сабменю

for (int i = 0; i < \_menu\_size; i++) { //цикл прохода по всем записям в меню

if (\_menu[i].\_menu\_size > 0) {

\_max\_subm\_lenght = \_menu[i].\_sub\_menu\_lenght[0];

for (int j = 1; j < \_menu[i].\_menu\_size; j++) {

if (\_menu[i].\_sub\_menu\_lenght[j] > \_max\_subm\_lenght) {

\_max\_subm\_lenght = \_menu[i].\_sub\_menu\_lenght[j];

}

\_menu[i].\_max\_sub\_lenght = \_max\_subm\_lenght; //получаем максимальный размер в сабменю, который будет использован для его вывода в дальнейшем

}

}

}

while (1) //цикл отрисовки меню

{

\_size\_now = \_get\_window\_size(\_size\_now); //получение текущего размера окна

if ((\_size\_now[0] != \_window\_size[0]) || (\_size\_now[1] != \_window\_size[1])) //ели текузий размер не совпал с стартовым

{

\_window\_size[0] = \_size\_now[0]; \_window\_size[1] = \_size\_now[1]; // запоминаем текущий размер

clearf(); // очистка экрана

\_print\_bakground(\_window\_size[0], \_window\_size[1]); //по новой отрисовываем задний фон и рабочую область

positionCur.X = \_otstup + 1; positionCur.Y = \_interval + 1; // задаем базовые стартовые параметры курсора

\_set\_cur\_to\_pos(hConsole, positionCur); //устанавливаем курсор в начальное положение

page = 1;

}

positionCur.X = \_otstup + 1; positionCur.Y = \_interval + 1;

\_set\_cur\_to\_pos(hConsole, positionCur);//устанавливаем курсор в начальное положение

for (int i = 0; i < \_menu\_size; i++) { //Пока у нас есть элементы в меню

\_get\_con\_info(&info\_x); // получаем текущее положение

\_padding = info\_x.dwCursorPosition.X - 1; //задаем текущий падинг

if (position[0] == i + 1) { //если у нас позиция курсора-указателя совпадает с текущей позицией

printf("\x1b[43m %s \x1b[0m", \_menu[i].\_name); // выделяем цветом

}

else printf(" %s ", \_menu[i].\_name); // иначе просто выводим на экран

printf("│"); //разделитель

\_get\_con\_info(&info\_x);

positionCur.X = info\_x.dwCursorPosition.X - 1; // получаем текущую позицию курсора, сместив его на один символ назад

\_new\_padding = info\_x.dwCursorPosition.X - 1; //запоминаем новый падинг(оступ)

positionCur.Y -= 1; // подымаем курсор на 1

\_set\_cur\_to\_pos(hConsole, positionCur); //Установили курсор

printf("┬"); // выводим разделитель между двумя

\_get\_con\_info(&info\_x); //получаем снова информацию о курсоре

positionCur.Y += 2; positionCur.X = \_padding; //смещаем курсор вниз на две ячейки, установив его на оступ

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_padding == \_otstup) // печатаем разделитель

{

printf("├"); positionCur.X++;

}

else {

positionCur.X++; //иначе увеличели X

\_set\_cur\_to\_pos(hConsole, positionCur);

}

for (int j = positionCur.X; j < \_new\_padding; j++) { //отрисовка нижнего отделителя

printf("─");

}

if (i == \_menu\_size - 1) {

printf("┘"); // если элемент последний , отрисовать уголок

}

else

printf("┴"); //иначе отрисовка отделителя

positionCur.Y -= 1; positionCur.X = \_new\_padding + 1;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

if (!table\_focus\_flag) //если у нас флаг не активен

print\_help("\x1b[45mESC\x1b[0m:Выход \x1b[45mENTER\x1b[0m:Ввод \x1b[45mСТРЕЛКИ\x1b[0m: Переключение селектора меню \x1b[45mTAB\x1b[0m:Переключить фокус на таблицу "); //печатаем один хелп

else print\_help("\x1b[45mESC\x1b[0m:Выход \x1b[45mENTER\x1b[0m:Редактировать \x1b[45mСТРЕЛКИ\x1b[0m:Навигация \x1b[45mTAB\x1b[0m:Фокус на меню \x1b[45mDEL\x1b[0m: Удалить запись \x1b[45mHOME|END|PgUp|PgDown\x1b[0m:Сортировка");// иначе другой

\_table\_window(table,\_output\_mas,&\_output\_colcount,&page,&table\_focus\_flag,root,sort);// вызываем печать окна таблицы

char c = getch(); // получаем селектор меню

if (c == KEY\_TAB) { // если таб то переход на таблицу с проверкой

if (\_output\_mas && \_output\_colcount != 0)

table\_focus\_flag = 1; else {

print\_help("\x1b[41mДанных для вывода нет. Невозможно переключиться на таблицу.\x1b[0m");

Sleep(1000);

}

} else

if (c == KEY\_ENTER) { // если ввели enter

if (\_menu[position[0] - 1].\_menu\_size > 0) { //если в текущем пункте меню есть сабменю , то

position[1] = 1;

while (1) {//бесконечный цикл работы с сабменю

positionCur.X = \_menu[position[0] - 1].\_menu\_name\_lenght + 2;//устанавливаем текущие позиции для курсора

positionCur.Y = \_interval + 2;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int i = 0; i <= \_menu[position[0] - 1].\_menu\_size; i++) { //цикл отрисовки каждого окошка меню

for (int j = 0; j <= \_menu[position[0] - 1].\_max\_sub\_lenght + 2; j++) { //цикл отрисовки бордера

if (j == 0) {

if (i == 0) {//если позиция 0 , то у нас верхние ограничитель

printf("┌");

}

else

if (i == \_menu[position[0] - 1].\_menu\_size) {

printf("└"); //если конечная позиция , то нижний

}

else printf("├"); // иначе соеденитель

}

else

if (j == \_menu[position[0] - 1].\_max\_sub\_lenght + 2) {

if (i == 0) {

printf("┐"); //печать верхнего ограничителя

}

else

if (i == \_menu[position[0] - 1].\_menu\_size) {

printf("┘"); //печать нижнего ограничителя , если достигли конца

\_get\_con\_info(&info\_x);

positionCur.X = info\_x.dwCursorPosition.X;

positionCur.Y = info\_x.dwCursorPosition.Y;

positionCur.Y--; positionCur.X--;

\_set\_cur\_to\_pos(hConsole, positionCur); // ставим курсор на один назад и на один вверх

printf("│"); //печать разделителя

positionCur.Y++;

\_set\_cur\_to\_pos(hConsole, positionCur);//возврат на текущую позциию

}

else {

printf("┤"); //печать соеденителя

\_get\_con\_info(&info\_x);

positionCur.X = info\_x.dwCursorPosition.X;

positionCur.Y = info\_x.dwCursorPosition.Y;

positionCur.Y--; positionCur.X--;

\_set\_cur\_to\_pos(hConsole, positionCur);// ставим курсор на один назад и на один вверх

printf("│");// печать разделителя

positionCur.Y++;

\_set\_cur\_to\_pos(hConsole, positionCur);//возврат на текущую позциию

}

}

else (printf("─")); //печать промежуточного разделителя

}

positionCur.Y++; positionCur.X = \_menu[position[0] - 1].\_menu\_name\_lenght + 2; //стартовые параметры для печати данных

\_set\_cur\_to\_pos(hConsole, positionCur); // установим курсор на данную позицию

if (i == \_menu[position[0] - 1].\_menu\_size) break; //если текущий номер совпал с размером , то ничего не рисуем дальше

int \_margin = \_menu[position[0] - 1].\_max\_sub\_lenght + 2 - \_menu[position[0] - 1].\_sub\_menu\_lenght[i]; //высчитываем отступ

\_margin = \_margin / 2;

printf("│");//печатаем разделитель

for (int l = 0; l < \_margin; l++) { printf(" "); } //делаем отступ слева

if (position[1] - 1 == i) {

printf("\x1b[43m%s\x1b[0m", \_menu[position[0] - 1].\_sub\_menu[i]); // если текущий выбор сабменю совпал , то выделяем его

}

else { printf("%s ", \_menu[position[0] - 1].\_sub\_menu[i]); } // иначе печатаем обычно

for (int l = 0; l < \_margin; l++) { printf(" "); } // печать отступа справа

positionCur.Y++;

\_set\_cur\_to\_pos(hConsole, positionCur); // увеличиваем y , и ставим курсор на след позицию

}

c = getch(); //получаем нажатую кнопку

if (c == KEY\_ENTER) { // если у нас нажат enter

int result = 0; //возвращаемое значение

for (int l = 0; l < position[0] - 1; l++) {

result += \_menu[l].\_menu\_size; //до текущей позиции высчитываем результат выбора

}

result += position[1]; // добавляем текущее смещение

return result; // вернем результат

}

if (c == KEY\_ESC) { clear(); break; } // если escape , то очистим экран меню , выйдем с цикла

position = \_get\_curent\_selection(c, position, \_menu[position[0] - 1].\_menu\_size, \_menu\_buttons, 1); //получение новой позиции курсора согласно c

}

}

else return PROGRAM\_EXIT; //Вернте выход из программы если в текущем меню 0 сабменю

}

else { //внутренний цикл работы будет получать новую позицию курсора

position = \_get\_curent\_selection(c, position, 1, \_menu\_buttons, 0);

}

}

return EXIT\_SUCCESS; //вернет успешный выход

}

/// <summary>

/// Анимированная картинка в консоли

/// </summary>

void animatedNeko() {

FILE\* f; int count = 1;

while (1) {

for (int i = 0; i < 11; i++) {

char url[256] = { 0 };

sprintf(url, "bakemonogatari-monogatari/banner (%d).txt", i);

f = fopen(url, "r"); char a[210];

while (fgets(a,210,f) != NULL)

{

printf(" %s", a);

}

Sleep(30);

COORD positionCur = { 0,0 }; //позиция x и y

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

\_set\_cur\_to\_pos(hConsole, positionCur);

}

count++;

if (count >= 20) return;

}

}

void \_print\_bakground(int \_window\_w,int \_window\_h)

{

clearf();

char c = rand() % (47 - 33 + 1) + 33;

srand(time(NULL));

for (int i = 0; i < \_window\_h; i++)

{

for (int j = 0; j < \_window\_w; j++) {

printf("\x1b[44m%c\x1b[0m",c);

c= rand() % (47 - 33 + 1) + 33;

}

printf("\n");

}

\_print\_border(\_window\_w, \_window\_h);

}

void \_print\_border(int \_window\_w, int \_window\_h) {

COORD positionCur = { \_otstup,\_interval }; //позиция x и y

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int y = \_interval; y <= \_window\_h - \_interval; y++) {

\_set\_cur\_to\_pos(hConsole, positionCur);

if(y == \_interval) printf("┌");

else if (y == \_window\_h - \_interval)

printf("└");

else

printf("│");

for (int x = \_otstup+1; x < \_window\_w - \_otstup; x++) {

if ((y == \_interval) || (y == \_window\_h - \_interval)) printf("─"); else printf(" ");

}

if (y == \_interval) printf("┐");

else if (y == \_window\_h - \_interval)

printf("┘");

else

printf("│");

positionCur.Y++;

}

}

int\* \_get\_window\_size(int\* size) {

HANDLE hWndConsole;

size = (int\*)calloc(2, sizeof(int));

if (hWndConsole = GetStdHandle(-12))

{

CONSOLE\_SCREEN\_BUFFER\_INFO consoleInfo;

if (GetConsoleScreenBufferInfo(hWndConsole, &consoleInfo))

{

size[0] = consoleInfo.srWindow.Right - consoleInfo.srWindow.Left + 1;

size[1] = consoleInfo.srWindow.Bottom - consoleInfo.srWindow.Top + 1;

}

else

printf("Error: %d\n", GetLastError());

}

return size;

}

void \_window(int \_window\_w, int \_window\_h, char\* title) {

int height = \_window\_h / 4; int width = \_window\_w / 4;

COORD positionCur = { \_otstup,\_interval }; //позиция x и y

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int \_center\_x, \_center\_y;

\_center\_x = \_window\_w / 2;

\_center\_y = \_window\_h / 2;

positionCur.X = \_center\_x - width / 2;

positionCur.Y = \_center\_y - height / 2;

for (int y = 0; y < height; y++) {

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int x = 0; x < width; x++) {

if (y == 0) {

if (x == 0) {

printf("┌");

}

else if (x == width - 1) { printf("┐"); }

else printf("─");

}

else

if (y == height - 1)

{

if (x == 0) {

printf("└");

}

else if (x == width - 1) {

printf("┘");

}

else printf("─");

}

else {

if (x == 0) {

printf("│");

}

else if (x == width - 1) { printf("│"); }

else printf(" ");

}

}

positionCur.Y++;

}

positionCur.X = \_center\_x - width / 2 + 1;

positionCur.Y = \_center\_y - height / 2 + 1;

\_set\_cur\_to\_pos(hConsole, positionCur);

int \_temp\_ots=0;

if (title) \_temp\_ots = (width - 2 - u8\_strlen(title)) / 2;

for (int j = 0; j < \_temp\_ots; j++) {

printf(" ");

}

if (title) printf("%s", title);

for (int j = 0; j < \_temp\_ots; j++) {

printf(" ");

}

positionCur.X = \_center\_x - width / 2;

positionCur.Y = \_center\_y - height / 2 + 2;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < width; j++) {

if (j == 0) printf("├");

else if (j == width - 1) {

printf("┤");

}

else printf("─");

}

}

int \_confirm\_window(char \* message)

{

SetConsoleOutputCP(65001);

if (!message) message = "Выполнить операцию ?";

int\* \_size\_n = NULL;

\_size\_n = \_get\_window\_size(\_size\_n);

int \_window\_w = \_size\_n[0]; int \_window\_h = \_size\_n[1];

\_window(\_window\_w, \_window\_h, "Подтверждение");

CONSOLE\_SCREEN\_BUFFER\_INFO info\_x;

int height = \_window\_h / 4; int width = \_window\_w / 4;

COORD positionCur = { \_otstup,\_interval }; //позиция x и y

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int \_center\_x = \_window\_w / 2;

int \_center\_y = \_window\_h / 2;

positionCur.X = \_center\_x - width / 2;

positionCur.Y = \_center\_y - height / 2 + height / 4+2;

int m\_lenght = u8\_strlen(message);

int margin = 0;

/\*margin = width - 21;

margin = margin / 2;

positionCur.X += margin;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("%s");\*/

if (m\_lenght < width - 2) {

int margin = width - m\_lenght;

margin = margin / 2;

positionCur.X += margin;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("%s", message);

}

else {

positionCur.X++;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int i = 0; i < (width - 1) \* 2; i++) {

printf("%c", message[i]);

\_get\_con\_info(&info\_x);

if (info\_x.dwCursorPosition.X - positionCur.X >= width - 5) {

printf("..."); break;

}

}

}

int \_selection = 1;

positionCur.Y = \_center\_y - height / 2 + height / 4 + height / 3+2;

while (1) {

positionCur.X = \_center\_x - width / 2 + 3;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_selection) {

printf(" \x1b[43mДА\x1b[0m ");

positionCur.X = \_center\_x + width / 2-8;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf(" НЕТ ");

}

else {

printf(" ДА ");

positionCur.X = \_center\_x + width / 2 - 8;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf(" \x1b[43mНЕТ\x1b[0m ");

}

char c = getch();

switch (c) {

case 75://лево

if (\_selection != 1) \_selection++;

break;

case 77://право

if (\_selection == 1) \_selection--;

break;

case KEY\_ENTER:

return \_selection;

break;

case KEY\_ESC:

return 0;

break;

}

}

return EXIT\_SUCCESS;

}

void \_in\_window() {

int\* \_size\_n = NULL;

\_size\_n = \_get\_window\_size(\_size\_n);

int \_window\_w = \_size\_n[0]; int \_window\_h = \_size\_n[1];

\_window(\_window\_w, \_window\_h,"Окно ввода");

int height = \_window\_h / 4; int width = \_window\_w / 4;

COORD positionCur = { \_otstup,\_interval }; //позиция x и y

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int \_center\_x = \_window\_w / 2;

int \_center\_y = \_window\_h / 2;

positionCur.X = \_center\_x - width / 2;

positionCur.Y = \_center\_y;

int margin = width - 21;

margin = margin / 2;

positionCur.X += margin;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("Введите данные -> ");

}

void \_message\_window(char\* message) {

int\* \_size\_n = NULL;

\_size\_n = \_get\_window\_size(\_size\_n);

int \_window\_w = \_size\_n[0]; int \_window\_h = \_size\_n[1];

\_window(\_window\_w, \_window\_h,"Сообщение");

CONSOLE\_SCREEN\_BUFFER\_INFO info\_x; HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int m\_lenght = u8\_strlen(message);

int height = \_window\_h / 4; int width = \_window\_w / 4;

COORD positionCur = { \_otstup,\_interval }; //позиция x и y

int \_center\_x = \_window\_w / 2;

int \_center\_y = \_window\_h / 2;

positionCur.X = \_center\_x - width / 2;

positionCur.Y = \_center\_y;

if (m\_lenght < width - 2) {

int margin = width - m\_lenght;

margin = margin / 2;

positionCur.X += margin;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("%s", message);

}

else {

positionCur.X++;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int i = 0; i < (width - 1)\*2; i++) {

printf("%c",message[i]);

\_get\_con\_info(&info\_x);

if (info\_x.dwCursorPosition.X - positionCur.X >= width - 5) {

printf("..."); break;

}

}

}

}

int \_table\_window(\_tabel\_metadata \* table, abonent\_t \* \_output\_mass, int \* \_info\_count, int\* page, int \* \_table\_focus\_flag, abonent\*\* root, sort\_struct\* sort) {

CONSOLE\_SCREEN\_BUFFER\_INFO info\_x; HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO structCursorInfo;

GetConsoleCursorInfo(hConsole, &structCursorInfo);

structCursorInfo.bVisible = FALSE;

SetConsoleCursorInfo(hConsole, &structCursorInfo);

static int row\_selection[] = {1,1};

int\* \_size\_n = NULL;

\_size\_n = \_get\_window\_size(\_size\_n);

int \_window\_w = \_size\_n[0]; int \_window\_h = \_size\_n[1];

int \_padding, \_new\_padding; int \_mn\_size\_flag = 0; int \_size\_temp = 0;

int \_size\_delta = 0; static int last\_sel = 0;

for (int i = 0; i < table->\_col\_count; i++) {

\_size\_delta += table->\_cols[i].size;

}

int height = \_window\_h - \_interval \* 2 - 10 ; int width = \_window\_w - \_otstup \* 2 - 4;

\_size\_delta = (width -2) - \_size\_delta - table->\_col\_count\*2-5 ;

do {

COORD positionCur = { \_otstup+2,\_interval+3}; //позиция x и y

\_set\_cur\_to\_pos(hConsole, positionCur);

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

if (i == 0) {

printf("┌");

}

if (i == width - 1) {

printf("┐");

} else

printf("─");

}

positionCur.Y++; //positionCur.X++;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("│");

for (int i = 0; i < table->\_col\_count; i++) {

\_get\_con\_info(&info\_x);

\_padding = info\_x.dwCursorPosition.X - 1;

if (i == sort->sort\_f && i!=0)

if (sort->sort\_t == UP)

printf(" \x1b[44m%s\x1b[0m ", table->\_cols[i].name);

else printf(" \x1b[43m%s\x1b[0m ", table->\_cols[i].name);

else printf(" %s ", table->\_cols[i].name);

if (table->\_cols[i].resizebl) {

if (!\_mn\_size\_flag) {

int \_tmp\_padding = \_size\_delta / 3;

\_size\_delta -= \_tmp\_padding;

\_mn\_size\_flag = 1;

if ((u8\_strlen(table->\_cols[i].name) + \_tmp\_padding) != table->\_cols[i].size) {

table->\_cols[i].size += \_tmp\_padding;

}

}

else {

if ((u8\_strlen(table->\_cols[i].name) + \_size\_delta) != table->\_cols[i].size) {

table->\_cols[i].size += \_size\_delta;

}

}

}

if (u8\_strlen(table->\_cols[i].name) != table->\_cols[i].size) {

int \_sim\_padding = table->\_cols[i].size - u8\_strlen(table->\_cols[i].name);

for (int i = 0; i < \_sim\_padding; i++) {

printf(" ");

}

}

printf("│");

\_get\_con\_info(&info\_x);

positionCur.X = info\_x.dwCursorPosition.X - 1;

\_new\_padding = info\_x.dwCursorPosition.X - 1;

positionCur.Y -= 1;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (i == table->\_col\_count - 1) {

printf("┐");

}

else

printf("┬");

\_get\_con\_info(&info\_x);

positionCur.Y += 2; positionCur.X = \_padding;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_padding == \_otstup+2)

{

printf("├"); positionCur.X++;

}

else {

positionCur.X++;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

for (int j = positionCur.X; j < \_new\_padding; j++) {

printf("─");

}

if (i == table->\_col\_count - 1) {

printf("┤");

}

else

printf("┼");

positionCur.Y -= 1; positionCur.X = \_new\_padding + 1;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

positionCur.Y += 2;

positionCur.X = \_otstup + 2;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_output\_mass && \*\_info\_count!=0) {

int \_col\_inpage = height;

int \_diap[2] = { 0,0 };

\_diap[0] = ((\*page) - 1) \* \_col\_inpage;

\_diap[1] = \_diap[0] + \_col\_inpage;

// if (row\_selection[1]== last\_sel)

for (int j = \_diap[0]; j < \_diap[1]; j++)

{

if (\*\_info\_count <= j) break;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\*\_table\_focus\_flag)

if (j == ((\*page) - 1) \* \_col\_inpage + row\_selection[1]-1)

printf("\x1b[42m");

printf("│");

char buff[400] = { "" };

sprintf(buff, "%d", j+1);

SetConsoleOutputCP(1251); //-------

printf("%s", buff);

if (u8\_strlen(buff) < 3)

{

for (int l = 0; l < 3 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

sprintf(buff, "%s %s %s", \_output\_mass[j].fio.surname, \_output\_mass[j].fio.name, \_output\_mass[j].fio.secondname);

if (u8\_strlen(buff) > table->\_cols[1].size + 2) {

sprintf(buff, "%s %c.%c", \_output\_mass[j].fio.surname, \_output\_mass[j].fio.name[0], \_output\_mass[j].fio.secondname[0]);

}

SetConsoleOutputCP(1251); //-------

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[1].size + 2)

{

for (int l = 0; l < table->\_cols[1].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

SetConsoleOutputCP(1251); //-------

sprintf(buff, "%s %s", \_output\_mass[j].autor.surname, \_output\_mass[j].autor.inicial);

if (u8\_strlen(buff) > table->\_cols[2].size + 2) {

for (int l = 0; l < table->\_cols[2].size - 1; l++) {

printf("%c", buff[l]);

}

printf("...");

}

else

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[2].size + 2)

{

for (int l = 0; l < table->\_cols[2].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

SetConsoleOutputCP(1251); //-------

sprintf(buff, "%s", \_output\_mass[j].book\_name);

if (u8\_strlen(buff) > table->\_cols[3].size + 2) {

for (int l = 0; l < table->\_cols[3].size - 1; l++) {

printf("%c", \_output\_mass[j].book\_name[l]);

}

printf("...");

}

else

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[3].size + 2)

{

for (int l = 0; l < table->\_cols[3].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

SetConsoleOutputCP(1251); //-------

sprintf(buff, "%s", \_output\_mass[j].izd);

if (u8\_strlen(buff) > table->\_cols[4].size + 2) {

for (int l = 0; l < table->\_cols[4].size - 1; l++) {

printf("%c", \_output\_mass[j].izd[l]);

}

printf("...");

}

else

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[4].size + 2)

{

for (int l = 0; l < table->\_cols[4].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

SetConsoleOutputCP(1251); //-------

sprintf(buff, "%d.%d.%d", \_output\_mass[j].date\_out.d, \_output\_mass[j].date\_out.m, \_output\_mass[j].date\_out.y);

if (u8\_strlen(buff) > table->\_cols[5].size + 2) {

for (int l = 0; l < table->\_cols[5].size - 1; l++) {

printf("%c", buff[l]);

}

printf("...");

}

else

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[5].size + 2)

{

for (int l = 0; l < table->\_cols[5].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

SetConsoleOutputCP(1251); //-------

sprintf(buff, "%.2f", \_output\_mass[j].cost);

if (u8\_strlen(buff) > table->\_cols[6].size + 2) {

for (int l = 0; l < table->\_cols[6].size - 1; l++) {

printf("%c", buff[l]);

}

printf("...");

}

else

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[6].size + 2)

{

for (int l = 0; l < table->\_cols[6].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

if (j == ((\*page) - 1) \* \_col\_inpage + row\_selection[1] - 1)

printf("\x1b[0m");

positionCur.Y++;

}

}

else {

printf("│");

for (int i = 0; i < (width - 12) / 2; i++)

printf("-");

SetConsoleOutputCP(65001); //-------

printf(" Данных нет ");

SetConsoleOutputCP(65001); //-------

for (int i = 0; i < (width - 12) / 2 - 1; i++)

printf("-");

if ((width - 12) % 2 == 1) printf("-");

printf("│");

positionCur.Y++;

\*\_table\_focus\_flag = 0;

}

\_set\_cur\_to\_pos(hConsole, positionCur);

int \_row\_num = 0; int \_padd\_border = table->\_cols[\_row\_num].size + 2;

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

if (i == 0) {

printf("└");

}

if (i == width - 1) {

printf("┘");

}

else

if (i == \_padd\_border) {

\_row\_num++;

if (\_row\_num < table->\_col\_count)

\_padd\_border += table->\_cols[\_row\_num].size + 3;

printf("┴");

}

else

printf("─");

}

COORD temp\_cord = positionCur;

temp\_cord.Y++;

\_set\_cur\_to\_pos(hConsole,temp\_cord);

char message[200] = "";

//if(( \* \_info\_count) != 0)

sprintf(message, "Страница %d из %d",\*page,(\*\_info\_count/height)+1);

printf(message);

if (\*\_table\_focus\_flag)

{

char c = getch();

if (c == KEY\_ENTER) {

\_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1] = \*\_in\_info\_window(table, &\_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1], 0);

tree\_deleteNodeById(root, \_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1].id);

\_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1].id = util\_hashCodeFromFio(&\_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1].fio);

tree\_add(root, &\_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1]);

clear\_table();

}

if (c == KEY\_ESC || c == KEY\_TAB)

{

\*\_table\_focus\_flag = 0;

break;

}

if (c == KEY\_DEL) {

if (\_confirm\_window("Вы действительно хотите удалить запись ?")) {

// root = tree\_deleteNodeById(root, \_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1].id);

tree\_deleteNodeById(root, \_output\_mass[((\*page) - 1) \* height + row\_selection[1] - 1].id);

\_message\_window("Запись успешно удалена");

int leafcount = tree\_getNodeCount(\*root, 0);

int temp = 0;

\_output\_mass = \_get\_output\_info(\*root, \_output\_mass, &temp);

if (leafcount == 0)

\_output\_mass = NULL;

\*\_info\_count = leafcount;

if (\*\_info\_count < ((\*page) \* height)) {

clear\_table();

if ((((\*page) - 1) \* height + row\_selection[1] - 1) > \*\_info\_count) {

row\_selection[1] = \*\_info\_count - ((\*page) - 1) \* height;

}

if (((\*\_info\_count) \* (\*page)) % height == 0){

if (\*page > 1) { (\*page)--; clear\_table(); }

}

}

}

}

if (c == KEY\_HOME)

{

if (sort->sort\_f > 0) {

sort->sort\_f--;

\_output\_mass = \_sort\_output(\_output\_mass, \_info\_count, sort);

}

}

if (c == KEY\_END)

{

if (sort->sort\_f < ZADANIE-1) {

sort->sort\_f++;

\_output\_mass = \_sort\_output(\_output\_mass, \_info\_count, sort);

}

}

if (c == KEY\_PGUP)

{

sort->sort\_t = UP;

\_output\_mass = \_sort\_output(\_output\_mass, \_info\_count, sort);

}

if (c == KEY\_PGDOWN)

{

sort->sort\_t = DOWN;

\_output\_mass = \_sort\_output(\_output\_mass, \_info\_count, sort);

}

if (c == KEY\_ARROW\_LEFT)

{

if (\*page > 1) { (\*page)--; clear\_table(); }

}

if (c == KEY\_ARROW\_RIGHT)

{

if (\*\_info\_count > ((\*page) \* height)) {

(\*page)++;

clear\_table();

if ((((\*page) - 1) \* height + row\_selection[1] - 1) > \*\_info\_count) {

row\_selection[1] = \*\_info\_count - ((\*page) - 1) \* height;

}

}

}

int\* temp = (int\*)calloc(2, sizeof(int));

temp = \_get\_curent\_selection(c, row\_selection, height, 1, 0);

row\_selection[1] = temp[1];

if ((((\*page) - 1) \* height + row\_selection[1] - 1) == \*\_info\_count) {

row\_selection[1] = row\_selection[1] - 1;

}

}

}while (\*\_table\_focus\_flag);

return EXIT\_SUCCESS;

}

void \_big\_window(char\* title) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int\* \_size\_n = NULL;

\_size\_n = \_get\_window\_size(\_size\_n);

int \_window\_w = \_size\_n[0]; int \_window\_h = \_size\_n[1];

int \_padding, \_new\_padding; int \_mn\_size\_flag = 0; int \_size\_temp = 0; char buff[200];

int height = \_window\_h / 2;

int width = \_window\_w / 2;

int \_center\_x = \_window\_w / 2;

int \_center\_y = \_window\_h / 2;

positionCur.X = \_center\_x - width / 2;

positionCur.Y = \_center\_y - height / 2;

for (int y = 0; y < height; y++) {

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int x = 0; x < width; x++) {

if (y == 0) {

if (x == 0) {

printf("┌");

}

else if (x == width - 1) { printf("┐"); }

else printf("─");

}

else

if (y == height - 1)

{

if (x == 0) {

printf("└");

}

else if (x == width - 1) {

printf("┘");

}

else printf("─");

}

else {

if (x == 0) {

printf("│");

}

else if (x == width - 1) { printf("│"); }

else printf(" ");

}

}

positionCur.Y++;

}

positionCur.X = \_center\_x - width / 2 + 1;

positionCur.Y = \_center\_y - height / 2 + 1;

\_set\_cur\_to\_pos(hConsole, positionCur);

int \_temp\_ots = 0;

if (title) \_temp\_ots = (width - 2 - u8\_strlen(title)) / 2;

for (int j = 0; j < \_temp\_ots; j++) {

printf(" ");

}

if(title) printf("%s", title);

for (int j = 0; j < \_temp\_ots; j++) {

printf(" ");

}

positionCur.X = \_center\_x - width / 2;

positionCur.Y = \_center\_y - height / 2 + 2;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < width; j++) {

if (j == 0) printf("├");

else if (j == width - 1) {

printf("┤");

}

else printf("─");

}

}

abonent\_t\* \_in\_info\_window(\_tabel\_metadata\* table, abonent\_t \*\_output\_info,int \_cycle\_in\_flag) {

CONSOLE\_SCREEN\_BUFFER\_INFO info\_x; HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int\* \_size\_n = NULL; abonent\_t \* \_temp\_info;

if (\_output\_info ) {

\_temp\_info = (abonent\_t\*)calloc(1, sizeof(abonent\_t));

\*\_temp\_info = \*\_output\_info;

}

else \_temp\_info = (abonent\_t\*)calloc(1, sizeof(abonent\_t));

\_size\_n = \_get\_window\_size(\_size\_n);

int \_window\_w = \_size\_n[0]; int \_window\_h = \_size\_n[1];

int \_padding, \_new\_padding; int \_mn\_size\_flag = 0; int \_size\_temp = 0; char buff[200];

int height = \_window\_h / 2; int width = \_window\_w / 2;

int \_center\_x = \_window\_w / 2; int flag\_clear = 0;

int \_center\_y = \_window\_h / 2; char\* title = NULL;

start:

if (\_cycle\_in\_flag) title = "Окно ввода информации"; else title = "Окно редактирования информации";

\_big\_window(title);

int max\_lenght = 0; int y\_modifire = 1;

if (height >= 20) y\_modifire = 2;

for (int i = 1; i < table->\_col\_count; i++) {

if (u8\_strlen(table->\_cols[i].name) > max\_lenght)

max\_lenght = u8\_strlen(table->\_cols[i].name);

}

char c;

int\* \_men\_position = (int\*)calloc(2, sizeof(int));

\_men\_position[0] = 1; \_men\_position[1] = 1;

while (1) {

if (flag\_clear) { \_big\_window(title); flag\_clear = 0; }

positionCur.X = \_center\_x - width / 2 + 4;

positionCur.Y = \_center\_y - height / 2 + 4;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int i = 1; i < table->\_col\_count; i++) {

if (\_men\_position[1] == i) {

// printf("\x1b[43m%s\x1b[0m", table->\_cols[i].name);

printf("%s", table->\_cols[i].name);

}

else {

printf("%s", table->\_cols[i].name);

}

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

positionCur.X = \_center\_x - width / 2 + 4 + max\_lenght + 1;

positionCur.Y = \_center\_y - height / 2 + 4;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int i = 1; i < table->\_col\_count; i++) {

if (\_men\_position[1] == i) {

if(!\_cycle\_in\_flag) printf("\x1b[43m --> \x1b[0m"); else printf(" --> ");

}

else {

printf(" --> ");

}

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

\_get\_con\_info(&info\_x);

int padding = info\_x.dwCursorPosition.X; // получаем текущую позицию курсора, сместив его на один символ назад

padding -= \_center\_x;

positionCur.X = \_center\_x - width / 2 + 4 + max\_lenght + 6;

positionCur.Y = \_center\_y - height / 2 + 4;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int i = 1; i < table->\_col\_count; i++) {

for (int j = 0; j < (width / 2 - padding - 8); j++) {

printf("\_");

}

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

positionCur.X = \_center\_x - width / 2 + 4 + max\_lenght + 6;

positionCur.Y = \_center\_y - height / 2 + 4;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (!\_cycle\_in\_flag) {

COORD fild\_cords[6];

SetConsoleOutputCP(1251); //-------

if (u8\_strlen(\_temp\_info->fio.surname) > 0)

printf(" %s ", \_temp\_info->fio.surname);

if (u8\_strlen(\_temp\_info->fio.name) > 0)

printf(" %s ", \_temp\_info->fio.name);

if (u8\_strlen(\_temp\_info->fio.secondname) > 0)

printf(" %s ", \_temp\_info->fio.secondname);

fild\_cords[0] = positionCur;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (u8\_strlen(\_temp\_info->autor.surname) > 0)

printf(" %s ", \_temp\_info->autor.surname);

if (u8\_strlen(\_temp\_info->autor.inicial) > 0)

printf(" %s ", \_temp\_info->autor.inicial);

fild\_cords[1] = positionCur;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (u8\_strlen(\_temp\_info->book\_name) > 0) {

printf(" %s ", \_temp\_info->book\_name);

}

fild\_cords[2] = positionCur;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (u8\_strlen(\_temp\_info->izd) > 0) {

printf(" %s ", \_temp\_info->izd);

}

fild\_cords[3] = positionCur;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_temp\_info->date\_out.d > 0) {

printf(" %d.", \_temp\_info->date\_out.d);

printf("%d.", \_temp\_info->date\_out.m);

printf("%d ", \_temp\_info->date\_out.y);

}

fild\_cords[4] = positionCur;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_temp\_info->cost > 0) {

printf(" %f ", \_temp\_info->cost);

}

fild\_cords[5] = positionCur;

positionCur.X = \_center\_x - width / 2 + 4;

positionCur.Y = \_center\_y + height / 2 - 2;

SetConsoleOutputCP(65001); //-------

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_men\_position[1] == table->\_col\_count) {

printf("\x1b[43mСохранить\x1b[0m");

}

else printf("Сохранить");

positionCur.X = \_center\_x + (width / 2 - u8\_strlen("Отмена") - 3);

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_men\_position[1] == table->\_col\_count + 1) {

printf("\x1b[43mОтмена\x1b[0m");

}

else printf("Отмена");

c = getch();

if (c == KEY\_ENTER)

{

if (\_men\_position[1] == table->\_col\_count + 1)

{

if (\_confirm\_window("Отменить операцию ?"))

{

return \_output\_info;

}

else

{

flag\_clear = 1;

}

}else if (\_men\_position[1] == table->\_col\_count)

{

if (\_confirm\_window("Сохранить данные ?"))

{

return \_temp\_info;

}

else

{

flag\_clear = 1;

}

}

else

{

switch (\_men\_position[1])

{

case 1: {

\_set\_cur\_to\_pos(hConsole, fild\_cords[0]);

for (int i = 0; i < width / 2 - padding - 8; i++) { printf("\_"); }

\_set\_cur\_to\_pos(hConsole, fild\_cords[0]);

input\_string(\_temp\_info->fio.surname, 40, PERSONAL);

input\_string(\_temp\_info->fio.name, 40, PERSONAL);

input\_string(\_temp\_info->fio.secondname, 40, PERSONAL);

SetConsoleOutputCP(65001); //-------

break;

}

case 2: {

\_set\_cur\_to\_pos(hConsole, fild\_cords[1]);

for (int i = 0; i < width / 2 - padding - 8; i++) { printf("\_"); }

\_set\_cur\_to\_pos(hConsole, fild\_cords[1]);

input\_string(\_temp\_info->autor.surname, 40, PERSONAL);

input\_string(\_temp\_info->autor.inicial, 3, INICIAL);

SetConsoleOutputCP(65001); //-------

break;

}

case 3: {

\_set\_cur\_to\_pos(hConsole, fild\_cords[2]);

for (int i = 0; i < width / 2 - padding - 8; i++) { printf("\_"); }

\_set\_cur\_to\_pos(hConsole, fild\_cords[2]);

input\_string(\_temp\_info->book\_name, width / 2 - padding - 8, NORMAL);

SetConsoleOutputCP(65001); //-------

break;

}

case 4: {

\_set\_cur\_to\_pos(hConsole, fild\_cords[3]);

for (int i = 0; i < width / 2 - padding - 8; i++) { printf("\_"); }

\_set\_cur\_to\_pos(hConsole, fild\_cords[3]);

input\_string(\_temp\_info->izd, width / 2 - padding - 8, NORMAL);

SetConsoleOutputCP(65001); //-------

break;

}

case 5: {

\_set\_cur\_to\_pos(hConsole, fild\_cords[4]);

for (int i = 0; i < width / 2 - padding - 8; i++) { printf("\_"); }

\_set\_cur\_to\_pos(hConsole, fild\_cords[4]);

in\_date(&\_temp\_info->date\_out.d, &\_temp\_info->date\_out.m, &\_temp\_info->date\_out.y);

SetConsoleOutputCP(65001); //-------

break;

}

case 6: {

\_set\_cur\_to\_pos(hConsole, fild\_cords[5]);

for (int i = 0; i < width / 2 - padding - 8; i++) { printf("\_"); }

\_set\_cur\_to\_pos(hConsole, fild\_cords[5]);

input\_float(&\_temp\_info->cost);

//scanf("%f", &\_temp\_info->cost);

SetConsoleOutputCP(65001); //-------

break;

}

default:

break;

}

}

}

if (c == KEY\_ESC)

{

return \_output\_info;

break;

}

\_men\_position = \_get\_curent\_selection(c, \_men\_position, table->\_col\_count + 1, 1, 1);

}

else {

//

char mes[] = "\x1b[45mESC\x1b[0m:Выход \x1b[45mENTER\x1b[0m:Ввод \x1b[45mСТРЕЛКИ\x1b[0m: Переключение поля"; //51

positionCur.Y = \_center\_y + height / 2 - 2;

positionCur.X = \_center\_x - 50/2;

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("%s", mes);

int cur\_step = 1; int cur\_key = 0; int step\_compl = 0;

CONSOLE\_SCREEN\_BUFFER\_INFO con\_inf;

CONSOLE\_CURSOR\_INFO structCursorInfo;

GetConsoleCursorInfo(hConsole, &structCursorInfo);

structCursorInfo.bVisible = TRUE;

SetConsoleCursorInfo(hConsole, &structCursorInfo);

positionCur.X = \_center\_x - width / 2 + 4 + max\_lenght + 6;

positionCur.Y = \_center\_y - height / 2 + 4;

\_set\_cur\_to\_pos(hConsole, positionCur);

COORD lastcord = {0,0};

SetConsoleOutputCP(1251); //-------

while (cur\_step <= 9) {

\_get\_con\_info(&con\_inf);

lastcord = con\_inf.dwCursorPosition;

switch (cur\_step)

{

case 1: {

cur\_key = input\_string(\_temp\_info->fio.surname, 40, PERSONAL);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

break;

case KEY\_ARROW\_UP:

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < (width / 2 - padding - 8); j++) {

printf("\_");

}

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

cur\_step++;

printf(" ");

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 2: {

cur\_key = input\_string(\_temp\_info->fio.name, 40, PERSONAL);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

break;

case KEY\_ARROW\_UP:

if (cur\_step > 1) {

cur\_step=1;

}

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < (width / 2 - padding - 8); j++) {

printf("\_");

}

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

cur\_step++;

printf(" ");

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 3: {

cur\_key = input\_string(\_temp\_info->fio.secondname, 40, PERSONAL);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_UP:

if (cur\_step > 1) {

cur\_step = 1;

}

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < (width / 2 - padding - 8); j++) {

printf("\_");

}

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

cur\_step++;

printf(" ");

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 4: {

cur\_key = input\_string(\_temp\_info->autor.surname, 40, PERSONAL);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

break;

case KEY\_ARROW\_UP:

if (cur\_step > 1) {

cur\_step=1;

positionCur.Y -= y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < (width / 2 - padding - 8); j++) {

printf("\_");

}

\_set\_cur\_to\_pos(hConsole, positionCur);

}

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

printf(" ");

cur\_step++;

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 5: {

cur\_key = input\_string(\_temp\_info->autor.inicial, 3, INICIAL);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_UP:

if (cur\_step > 1) {

cur\_step=4;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < (width / 2 - padding - 8); j++) {

printf("\_");

}

\_set\_cur\_to\_pos(hConsole, positionCur);

}

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

cur\_step++; printf(" ");

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 6: {

cur\_key = input\_string(\_temp\_info->book\_name, 100, NORMAL);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

\_get\_con\_info(&con\_inf);

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_UP:

if (cur\_step > 1) {

cur\_step=4;

positionCur.Y -= y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int j = 0; j < (width / 2 - padding - 8); j++) {

printf("\_");

}

\_set\_cur\_to\_pos(hConsole, positionCur);

}

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

cur\_step++;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 7: {

cur\_key = input\_string(\_temp\_info->izd, 60, NORMAL);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

\_get\_con\_info(&con\_inf);

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_UP:

if (cur\_step > 1) {

cur\_step--;

positionCur.Y -= y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

cur\_step++;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 8: {

cur\_key = in\_date(&\_temp\_info->date\_out.d, &\_temp\_info->date\_out.m, &\_temp\_info->date\_out.y);

switch (cur\_key)

{

case KEY\_ENTER:

cur\_step++;

step\_compl++;

\_get\_con\_info(&con\_inf);

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

break;

case KEY\_ARROW\_UP:

if (cur\_step > 1) {

cur\_step--;

positionCur.Y -= y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

break;

case KEY\_ARROW\_DOWN:

if (cur\_step < step\_compl) {

cur\_step++;

positionCur.Y += y\_modifire;

\_set\_cur\_to\_pos(hConsole, positionCur);

}

else {

\_set\_cur\_to\_pos(hConsole, lastcord);

}

break;

case KEY\_ESC:

if (\_confirm\_window("Вы действительно хотите отменить ввод?")) {

SetConsoleOutputCP(65001); //-------

return NULL;

}

else { SetConsoleOutputCP(65001); goto start; }

break;

default:

break;

}

break;

}

case 9: {

cur\_step++;

step\_compl++;

input\_float(&\_temp\_info->cost);

break;

}

default:

break;

}

}

SetConsoleOutputCP(65001);

\_message\_window("Запись успешно добавлена");

Sleep(2000);

structCursorInfo.bVisible = FALSE;

SetConsoleCursorInfo(hConsole, &structCursorInfo);

return \_temp\_info;

}

}

}

int print\_help(char \* help\_message) {

int\* \_size\_now = NULL; //текущий размер окна

\_size\_now = \_get\_window\_size(\_size\_now);

CONSOLE\_SCREEN\_BUFFER\_INFO info\_x; HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

\_get\_con\_info(&info\_x);

COORD positionCur = { \_otstup,\_size\_now[1] - \_interval - 2 };

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("├");

for (int i = 0; i < \_size\_now[0] - \_otstup\*2 - 1; i++) {

printf("─");

}

printf("┤");

positionCur.Y++; positionCur.X++;

\_set\_cur\_to\_pos(hConsole, positionCur);

for (int i = 0; i < \_size\_now[0] - \_otstup \* 2 - 1; i++)

printf(" ");

\_set\_cur\_to\_pos(hConsole, positionCur);

printf("%s", help\_message);

return 0;

}

abonent\_t\* \_sort\_output(abonent\_t\* \_output\_mass, int\* filds\_count, sort\_struct\* sorts)

{

if (!sorts) return \_output\_mass;

switch (sorts->sort\_f)

{

case DEF:

return \_output\_mass;

break;

case FIO:

if (sorts->sort\_t == UP){

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (\_output\_mass[sort].id > \_output\_mass[sort + 1].id) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

else {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (\_output\_mass[sort].id < \_output\_mass[sort + 1].id) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

break;

case AUTHOR:

if (sorts->sort\_t == UP) {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (strcmp(\_output\_mass[sort].autor.surname, \_output\_mass[sort + 1].autor.surname) == 1) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

else {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (strcmp(\_output\_mass[sort].autor.surname, \_output\_mass[sort + 1].autor.surname) == -1) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

break;

case BOOK\_NAME:

if (sorts->sort\_t == UP) {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (strcmp(\_output\_mass[sort].book\_name, \_output\_mass[sort + 1].book\_name) == 1) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

else {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (strcmp(\_output\_mass[sort].book\_name, \_output\_mass[sort + 1].book\_name) == -1) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

break;

case DATE\_OUT:

if (sorts->sort\_t == UP) {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (\_output\_mass[sort].date\_out.y > \_output\_mass[sort + 1].date\_out.y) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

else if (\_output\_mass[sort].date\_out.m > \_output\_mass[sort + 1].date\_out.m && \_output\_mass[sort].date\_out.y == \_output\_mass[sort + 1].date\_out.y)

{

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

else if (\_output\_mass[sort].date\_out.m == \_output\_mass[sort + 1].date\_out.m && \_output\_mass[sort].date\_out.y == \_output\_mass[sort + 1].date\_out.y && \_output\_mass[sort].date\_out.d > \_output\_mass[sort + 1].date\_out.d) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

else {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (\_output\_mass[sort].date\_out.y < \_output\_mass[sort + 1].date\_out.y) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

else if (\_output\_mass[sort].date\_out.m < \_output\_mass[sort + 1].date\_out.m && \_output\_mass[sort].date\_out.y == \_output\_mass[sort + 1].date\_out.y)

{

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

else if (\_output\_mass[sort].date\_out.m == \_output\_mass[sort + 1].date\_out.m && \_output\_mass[sort].date\_out.y == \_output\_mass[sort + 1].date\_out.y && \_output\_mass[sort].date\_out.d < \_output\_mass[sort + 1].date\_out.d) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

break;

case IZD:

if (sorts->sort\_t == UP) {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (strcmp(\_output\_mass[sort].izd, \_output\_mass[sort + 1].izd) == 1) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

else {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (strcmp(\_output\_mass[sort].izd, \_output\_mass[sort + 1].izd) == -1) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

break;

case COST:

if (sorts->sort\_t == UP) {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (\_output\_mass[sort].cost > \_output\_mass[sort + 1].cost) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

else {

for (int write = 0; write < \*filds\_count; write++) {

for (int sort = 0; sort < \*filds\_count - 1; sort++) {

if (\_output\_mass[sort].cost < \_output\_mass[sort + 1].cost) {

abonent\_t temp = \_output\_mass[sort + 1];

\_output\_mass[sort + 1] = \_output\_mass[sort];

\_output\_mass[sort] = temp;

}

}

}

}

break;

case ZADANIE:

if (sorts->sort\_t == DEF) return \_output\_mass;

break;

default:

break;

}

return \_output\_mass;

}

void dolgiWindow(abonent \*\* root) {

\_big\_window("Должники");

CONSOLE\_SCREEN\_BUFFER\_INFO info\_x; HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

int\* \_size\_n = NULL;

\_size\_n = \_get\_window\_size(\_size\_n);

int \_window\_w = \_size\_n[0]; int \_window\_h = \_size\_n[1];

int \_padding, \_new\_padding; int \_mn\_size\_flag = 0; int \_size\_temp = 0; char buff[200];

int height = \_window\_h / 2; int width = \_window\_w / 2;

int \_center\_x = \_window\_w / 2; int flag\_clear = 0;

int \_center\_y = \_window\_h / 2;

int refresh\_flag = 1;

time\_t now = time(0);

struct tm\* ltm = localtime(&now);

int d, y, m;

y = ltm->tm\_year + 1900;

m = ltm->tm\_mon+1;

d = ltm->tm\_mday;

dolgi\_pers\_t\_obr\* dolgniki = (dolgi\_pers\_t\_obr\*)calloc(sizeof(dolgi\_pers\_t\_obr), 1);

dolgniki->count = 0;

dolgniki = \_get\_dolgi\_info(\*root, dolgniki, d, m, y);

COORD positionCur;

positionCur.X = \_center\_x - width / 2 + 4;

positionCur.Y = \_center\_y - height / 2 + 4;

int\* \_men\_position = (int\*)calloc(2, sizeof(int));

\_men\_position[0] = 1; \_men\_position[1] = 1;

\_set\_cur\_to\_pos(hConsole, positionCur);

\_tabel\_metadata \* table = (\_tabel\_metadata\*)calloc(sizeof(\_tabel\_metadata), 1);

table->\_col\_count = 2;

table->\_cols = (\_table\_col\*)calloc(sizeof(\_table\_col), table->\_col\_count);

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

table->\_cols[0].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[0].name, "ФИО абонента-должника");

table->\_cols[0].resizebl = 1;

table->\_cols[0].size = u8\_strlen(table->\_cols[0].name);

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

table->\_cols[1].name = (char\*)calloc(sizeof(char), 60);

strcpy(table->\_cols[1].name, "Кол-во книг");

table->\_cols[1].resizebl = 0;

table->\_cols[1].size = u8\_strlen(table->\_cols[1].name);

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

COORD PosCur = { \_center\_x - width / 2 + 4,\_center\_y - height / 2 + 3 }; //позиция x и y

int \_size\_delta = 0;

for (int i = 0; i < table->\_col\_count; i++) {

\_size\_delta += table->\_cols[i].size;

}

\_size\_delta = (width - 2) - \_size\_delta - ((table->\_col\_count )\* 2) - 10;

\_set\_cur\_to\_pos(hConsole, PosCur);

int size\_w = 0;

for (int i = 0; i < table->\_col\_count; i++) {

size\_w += table->\_cols[i].size;

}

for (int i = 0; i < size\_w+ (table->\_col\_count)+3 + \_size\_delta; i++) {

if (i == 0) {

printf("┌");

}

if (i == width - 1) {

printf("┐");

}

else

printf("─");

}

PosCur.Y++; //positionCur.X++;

\_set\_cur\_to\_pos(hConsole, PosCur);

printf("│");

for (int i = 0; i < table->\_col\_count; i++) {

\_get\_con\_info(&info\_x);

\_padding = info\_x.dwCursorPosition.X - 1;

printf(" %s ", table->\_cols[i].name);

if (table->\_cols[i].resizebl) {

{

if ((u8\_strlen(table->\_cols[i].name) + \_size\_delta) != table->\_cols[i].size) {

table->\_cols[i].size += \_size\_delta;

}

}

}

if (u8\_strlen(table->\_cols[i].name) != table->\_cols[i].size) {

int \_sim\_padding = table->\_cols[i].size - u8\_strlen(table->\_cols[i].name);

for (int i = 0; i < \_sim\_padding; i++) {

printf(" ");

}

}

printf("│");

\_get\_con\_info(&info\_x);

PosCur.X = info\_x.dwCursorPosition.X - 1;

\_new\_padding = info\_x.dwCursorPosition.X - 1;

PosCur.Y -= 1;

\_set\_cur\_to\_pos(hConsole, PosCur);

if (i == table->\_col\_count - 1) {

printf("┐");

}

else

printf("┬");

\_get\_con\_info(&info\_x);

PosCur.Y += 2; PosCur.X = \_padding;

\_set\_cur\_to\_pos(hConsole, PosCur);

if (\_padding == \_center\_x - width / 2 + 4 )

{

printf("├"); PosCur.X++;

}

else {

PosCur.X++;

\_set\_cur\_to\_pos(hConsole, PosCur);

}

for (int j = PosCur.X; j < \_new\_padding; j++) {

printf("─");

}

if (i == table->\_col\_count - 1) {

printf("┤");

}

else

printf("┼");

PosCur.Y -= 1; PosCur.X = \_new\_padding + 1;

\_set\_cur\_to\_pos(hConsole, PosCur);

}

PosCur.Y += 2;

PosCur.X = \_center\_x - width / 2 + 4;

int page = 1;

int cor\_X, cor\_Y; cor\_X = PosCur.X; cor\_Y = PosCur.Y;

SetConsoleOutputCP(65001); //-------

while (1) {

PosCur.X = cor\_X; PosCur.Y = cor\_Y;

if (refresh\_flag) {

if ((dolgniki->count)) {

int \_col\_inpage = height - 10;

int \_diap[2] = { 0,0 };

\_diap[0] = ((page)-1) \* \_col\_inpage;

\_diap[1] = \_diap[0] + \_col\_inpage;

for (int j = \_diap[0]; j < \_diap[1]; j++)

{

if (dolgniki->count <= j) {

break;

}

\_set\_cur\_to\_pos(hConsole, PosCur);

char buff[400] = { "" };

SetConsoleOutputCP(65001); //-------

printf("│");

sprintf(buff, "%s %s %s", dolgniki->info\_mass[j].fio.surname, dolgniki->info\_mass[j].fio.name, dolgniki->info\_mass[j].fio.secondname);

if (u8\_strlen(buff) > table->\_cols[0].size + 2) {

sprintf(buff, "%s %c.%c", dolgniki->info\_mass[j].fio.surname, dolgniki->info\_mass[j].fio.name[0], dolgniki->info\_mass[j].fio.secondname[0]);

}

SetConsoleOutputCP(1251); //-------

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[0].size + 2)

{

for (int l = 0; l < table->\_cols[0].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

SetConsoleOutputCP(1251); //-------

sprintf(buff, "%d", dolgniki->info\_mass[j].count\_dolg\_books);

if (u8\_strlen(buff) > table->\_cols[1].size + 2) {

for (int l = 0; l < table->\_cols[1].size - 1; l++) {

printf("%c", buff[l]);

}

printf("...");

}

else

printf("%s", buff);

if (u8\_strlen(buff) < table->\_cols[1].size + 2)

{

for (int l = 0; l < table->\_cols[1].size + 2 - u8\_strlen(buff); l++)

printf(" ");

}

SetConsoleOutputCP(65001); //-------

printf("│");

SetConsoleOutputCP(1251); //-------

PosCur.Y++;

}

refresh\_flag = 0;

}

else

{

\_set\_cur\_to\_pos(hConsole, PosCur);

printf("│");

for (int i = 0; i < (width - 21) / 2; i++)

printf("-");

SetConsoleOutputCP(65001); //-------

printf(" Данных нет ");

SetConsoleOutputCP(65001); //-------

for (int i = 0; i < (width - 21) / 2 - 1; i++)

printf("-");

if ((width - 6) % 2 == 1) printf("-");

printf("│");

PosCur.Y++;

refresh\_flag = 0;

}

SetConsoleOutputCP(65001); //-------

\_set\_cur\_to\_pos(hConsole, PosCur);

int \_row\_num = 0; int \_padd\_border = table->\_cols[\_row\_num].size + 2;

for (int i = 0; i < width - 10; i++) {

if (i == 0) {

printf("└");

}

if (i == width - 11) {

printf("┘");

}

else

if (i == \_padd\_border) {

\_row\_num++;

if (\_row\_num < table->\_col\_count)

\_padd\_border += table->\_cols[\_row\_num].size + 3;

printf("┴");

}

else

printf("─");

}

if (\_center\_y + height / 2 - 2 - PosCur.Y > 1) {

for (; \_center\_y + height / 2 - 2 - PosCur.Y > 1;) {

PosCur.Y++; \_set\_cur\_to\_pos(hConsole, PosCur);

for (int i = PosCur.X; i < \_center\_x + width / 2 - 4; i++) {

printf(" ");

}

}

}

}

positionCur.X = \_center\_x - width / 2 + 4;

positionCur.Y = \_center\_y + height / 2 - 2;

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_men\_position[0] == 1) {

printf("\x1b[43mПред.страница\x1b[0m");

}

else printf("Пред.страница");

positionCur.X = 3+ width / 2 + 4 + u8\_strlen("Пред.страница");

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_men\_position[0] == 2) {

printf("\x1b[43mСлед.страница\x1b[0m");

}

else printf("След.страница");

positionCur.X = \_center\_x + (width / 2 - u8\_strlen("Выход") - 3);

\_set\_cur\_to\_pos(hConsole, positionCur);

if (\_men\_position[0] == 3) {

printf("\x1b[43mВыход\x1b[0m");

}

else printf("Выход");

char c = getch();

if (c == KEY\_ENTER)

{

if (\_men\_position[0] == 1)

{

if (page > 1) { page--; refresh\_flag = 1; }

}

if (\_men\_position[0] == 2)

{

if (dolgniki->count>((page) \* (height - 10))) {

page++;

refresh\_flag = 1;

}

}

if (\_men\_position[0] == 3)

{

return;

}

}

if (c == KEY\_ESC)

return;

\_men\_position = \_get\_curent\_selection(c, \_men\_position, 2, 3, 0);

}

}