Балаковский инженерно-технологический институт - филиал

федерального государственного автономного образовательного учреждения

высшего образования

«Национальный исследовательский ядерный университет «МИФИ»

Факультет атомной энергетики и технологий

Кафедра «Информационные системы и технологии»

**КУРСОВАЯ РАБОТА**

по дисциплине

«Языки программирования»

на тему

«Разработка программы на языке программирования СИ»

Выполнил: студент группы ИФСТ-11

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Науменко А. Р.

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г.

|  |  |
| --- | --- |
| Допущен к защите  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Несытых М.А.  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. | Защитил с оценкой \_\_\_\_\_\_\_\_\_\_\_  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Несытых М.А.  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Балаково 2022

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| ВВЕДЕНИЕ | 3 |
| 1 АНАЛИЗ ЗАДАЧИ | 5 |
| 1.1 Анализ предметной области «Выставка» |  |
| 1.2 Разработка интерфейса пользователя с программным продуктом и алгоритм выполнения данной задачи |  |
| 2 РАЗРАБОТКА КОНСОЛЬНОГО ПРИЛОЖЕНИЯ | 16 |
| 2.1 Описание применяемых функций |  |
| 2.2 Тестирование программного решения |  |
| ЗАКЛЮЧЕНИЕ | 28 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 29 |
| Приложение А | 30 |

ВВЕДЕНИЕ

*БИТИ ИФСТ-11*

*Лит.*

*2*

*Лист*

*Н.контр.*

*Утв.*

*Пров.*

*№ докум.*

*Лист*

*Изм.*

*Подпись*

*Дата*

*Разраб.*

ИФСТ.467239.032 ПЗ

*Листов*

*47*

*Разработка программы на*

*языке программирования СИ*

*Несытых*

*Несытых*

Современную жизнь невозможно представить без компьютеров. Технологии пронизывают весь цивилизованный мир и сейчас прочно вошли в окружающую нас действительность. В них чувствуются устойчивые тенденции к совершенствованию, что безусловно отражается на жизни человека. Сейчас каждый более или менее часто, но сталкивается с компьютерными технологиями. А некоторые люди не могут вообразить ни дня без мобильного телефона, ноутбука и интернета. Ведь там и учёба, и работа, и досуг.

За последние десятилетия мощность вычислительных систем в сочетании с установленными математическими алгоритмами позволила получить интересные программные решения в области искусственного интеллекта и обработки больших данных. Актуальность темы исследования определяется цифровой трансформацией, т.е. преобразованием деятельности организаций, а также автоматизацией и многих бытовых процессов. На базе комплекса передовых технологий, в число которых входит программирование, возникают инновационные продукты и услуги, упрощающие жизнь человека, требуя минимум его времени.

Пост-пандемическая реальность открывает новые виртуальные возможности, к которым мир искусства продолжает свою адаптацию. Одной из областей по внедрению и эксплуатации информационных технологий является сфера культуры, где сосредоточено немало передовых знаний. Структуризация данных становится неотъемлемой составляющей рынка искусств, делая ставку на помощь в его развитии. Для решения специализированных задач по настройке программного обеспечения вместе с оптимизацией работы с массивом информации и создаются новые утилиты.

Предмет исследования — реализация консольного программного приложения на языке программирования С.

Цель курсовой работы: закрепление знаний, полученных по дисциплине «Языки программирования» и их применение в ходе разработки программного

средства для предметной области «Выставка».

Задачи, которые необходимо решить для достижения поставленной цели:

– сформировать требования к создаваемому программному решению. Определить набор входных данных и ограничения на их использование вместе с форматом вывода результатов работы программы;

– разработать алгоритм работы программного продукта с учетом требований к организации интерфейса взаимодействия пользователя с ним;

– разработать и описать процедуры программного средства, выполняющие функционал приложения (реализация CRUD: функции создания, чтения, добавления, удаления элементов; поиск по текстовым параметрам; сортировка по заданному полю: числовое значение и символьное). Организовать работу программы, осуществив вызов функций в соответствии с навигационной схемой интерфейса пользователя;

– реализовать хранений данных, аккумулируемых в результате функционирования приложения, в файле;

– протестировать работу созданной программы на всех возможных вариантах входных значений.

Главное назначение разработанного программного проекта заключается в обеспечении комфортного взаимодействия пользователя с ним. А именно удобства во время обращения к перечню предоставляемой информации, в котором без труда можно сориентироваться. Приложение разделено на 2 кабинета: для пользователя и администратора. Их отличие состоит в том, что в последний можно зайти только по паролю – это сделано для защиты данных, которые редактируются исключительно администратором.

1 АНАЛИЗ ЗАДАЧИ

1.1 Анализ предметной области «Выставка»

В культуру сейчас можно погрузиться, не выходя из дома и не отрываясь от гаджета. Платформы цифрового искусства доступны круглосуточно и без выходных, а также охватывают огромное количество людей независимо от местонахождения. Однако с индивидуальным визитом они не сравнимы. Посещение выставки отвечает сразу трём верхним ярусам знаменитой Пирамиды Маслоу: потребностям познавательным, эстетическим и в самоактуализации [1]. Выставки нужны тем, у кого нет больших возможностей, но кто любит прекрасное. Имеет место быть и примечание о просветительской роли и образовательной функции выставочной работы, о воспитании художественного вкуса [2].

Многообразие экспонатов и сопутствующих им данных наряду со сведениями о категориях приобретаемых билетов для ознакомления с картинами представляют собой большой поток информации. В связи с чем требуется модифицировать процесс работы с такой информационной базой. С целью повышения эффективности выставочной деятельности и разработано программное обеспечение, которое обладает рядом функций, позволяющих выполнять основные задачи не только управляющих, но и посетителей. В частности, описание выставки, её каталога, графика работы и цен на билеты.

В данной курсовой выделены две главные сущности: «Билет» и «Экспонат». Сущность — это абстрактный объект определённого вида, которому отвечает набор атрибутов, т.е. свойств, которые его описывают [3]. В результате исследования предметной области были получены следующие атрибуты:

1) «Экспонат» содержит:

* код – номер экспоната типа данных «целое число»;
* год картины – целое число, указывающее, когда был создан экспонат;
* наименование картины – название экспоната типа данных «строка».

2) «Билет» содержит:

* число – номер экскурсии типа данных «целое число»;
* цена – целое число, характеризующее стоимость билета;
* тип билета – категории лиц строкового типа данных, для которых предназначен билет на выставку;
* вид экскурсии – «строка», для описания типа проводимых экскурсий.

Второстепенными сущностями, напрямую зависящими от вышеупомянутых, можно назвать корзину и чек. В первой хранятся данные о выборе пользователя, т.е. информация о нужном ему билете. В последней же находится своеобразная форма договора, подтверждающая факт приобретения покупателем услуги, иначе говоря – доказательство совершения сделки.

Любой объект можно конкретизировать как структуру, которая даёт человеку представление о его возможностях. Структура — это объединённое в единое целое множество поименованных компонентов данных [4]. Для её объявления в языке программирования С используют служебное слово «struct». В отличие от массивов, всегда состоящих из однотипных элементов, структуры в Си нужны как раз для того, чтобы формировать конструкции с различными важными свойствами объекта [5].

Для нашего случая дважды был введён производный тип, называемый структурным (о чём свидетельствует спецификатор struct), таким образом:

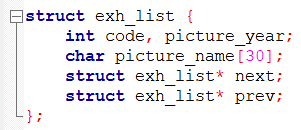


Рисунок 1 – Программный код на языке C для структуры сущности

«Экспонат» с соответствующим идентификатором exh\_list

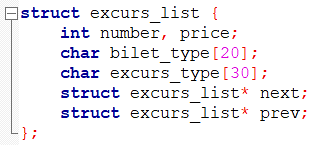


Рисунок 2 – Программный код на языке C для структуры «Билет на

выставку» с соответствующим идентификатором excurs\_list

В фигурных скобках размещаются описания элементов, каждый из которых служит прототипом для объектов структур вводимого типа [4]. На рисунке 1 зафиксированы следующие компоненты данных, которые уже назывались ранее. Это 2 целочисленные (int) переменные code и picture\_year, а также picture\_name типа char с числом 30 – общим и достаточным количеством символов, выделенных для массива, куда будут записаны названия выставочных картин.

Рисунок 2 демонстрирует составляющие последней имеющейся структуры, о которых тоже упоминалось выше, в атрибутах сущности «Билет». Такие переменные, как number и price целого типа int, bilet\_type[20] и excurs\_type[30] символьного типа char. В квадратных скобках обозначены размеры массивов для заполнения информацией об экскурсионных билетах.

Оставшиеся несколько строк кода, на которые стоит обратить внимание: struct exh\_list\* next и struct exh\_list\* prev (рис. 1) с struct excurs\_list\* next и struct excurs\_list\* prev (рис. 2). Благодаря им реализовано упорядочение сведений о выставке в виде двусвязного списка – структуры, которая состоит из узлов, хранящих полезные данные, указатели на предыдущий и следующий узлы [6]. Следовательно, наиболее важное из преимуществ этого решения состоит возможности перемещения по списку в обоих направлениях, что упрощает работу с ним.

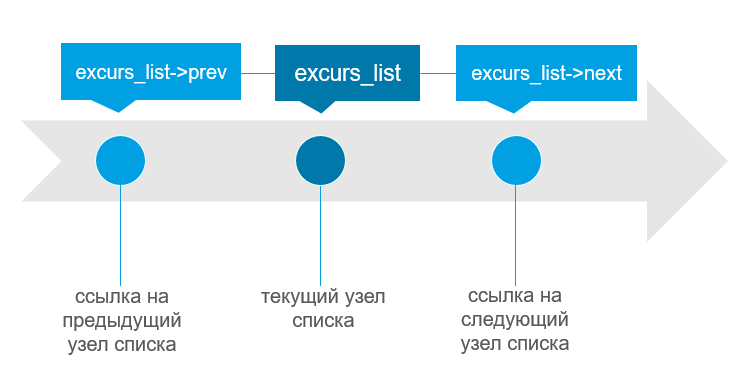


Рисунок 3 – Схема организации двунаправленного списка на примере

объявленной в курсовой работе структуры excurs\_list

Сформированная структура представления двусвязного списка наглядно показана на рисунке 3.

В данном пункте были рассмотрены основные сущности предметной области, на основе анализа которой сформулированы требования к продукту. А также организованы две структуры, необходимые для работы программы.

1.2 Разработка интерфейса пользователя с программным продуктом и алгоритм выполнения данной задачи

Прикладное решение для автоматизации выставочной деятельности было создано в свободной кроссплатформенной среде разработки Code::Blocks 20.03. Для корректного функционирования программы к ней произведено подключение необходимых стандартных библиотек языка Си с помощью директивы препроцессора #include: <stdio.h>, <stdlib.h>, <windows.h> и <string.h> [7].

Таблица 1 – Библиотеки языка С и набор их функций, реализованных в

программном коде

|  |  |
| --- | --- |
| Библиотеки | Используемые составляющие |
| stdio.h | Функции для файловых операций – fopen, fclose, rewind; функции для операций ввода-вывода – feof, fflush, fgetc, fgets, ftell, fseek, getchar, gets, printf, fprintf, scanf, fscanf. |
| stdlib.h | Вспомогательные функции – atoi, malloc, free, system, exit. |
| windows.h | Включает в себя всю подноготную Windows – SetConsoleCP() и SetConsoleOutputCP() устанавливают кодировки ввода/вывода. |
| string.h | Для работы с различными видами строк – strcat, strncat, strcmp, strcpy, strlen. |

Консольное приложение — программа, действующая через текстовый интерфейс и не требующая длительной настройки. В роли «проводника» между ней и пользователем выступает консоль: человек обычно взаимодействует с подобного рода приложением через клавиатуру и монитор, работающий в режиме отображения символьной информации (буквы, цифры и специальные знаки).

В результате анализа предметной области и данных, получаемых на входе

и выходе, был подготовлен алгоритм работы программного продукта. На его основе построена несложная схема пользовательского интерфейса, обеспечивающая беспрепятственную систему коммуникации (рис. 4).

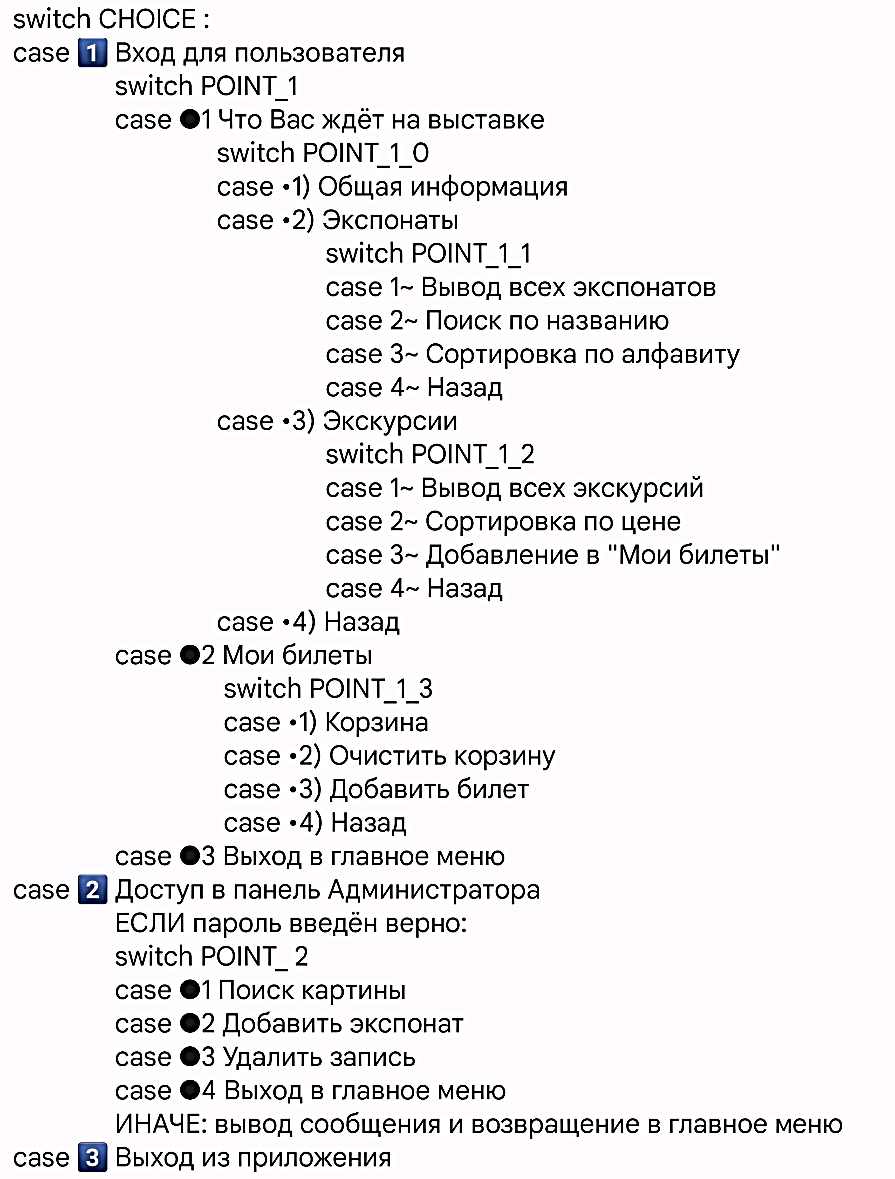


Рисунок 4 – Иерархическая схема интерфейса

Удобство работы приложения состоит в организации навигационного Меню. Для выполнения действия в зависимости от введённого человеком пункта использован оператор switch-case. Чтобы реализовать завершение программы в целом или же возврат к предыдущему разделу лишь при выборе цифры, обозначающей "выход/назад", применяется цикл while с надлежащим условием и оператор break. При этом вызывается функция задержки экрана system("pause"), чтобы после окончания работы с выбранным разделом, нажатия любой клавиши, все остальные пункты меню выводились только тогда, когда пользователь самостоятельно решит туда вернуться.

Комфортабельность работы пользователя с рассматриваемым программ-

ным обеспечением заключается и в свойствах, связанных с экранным буфером консоли. В их числе: шрифт фиксированной ширины Lucida Console, крупный размер курсора и характеристики прямоугольной области экранного буфера, которая отображается в консольном окне (рис. 5).

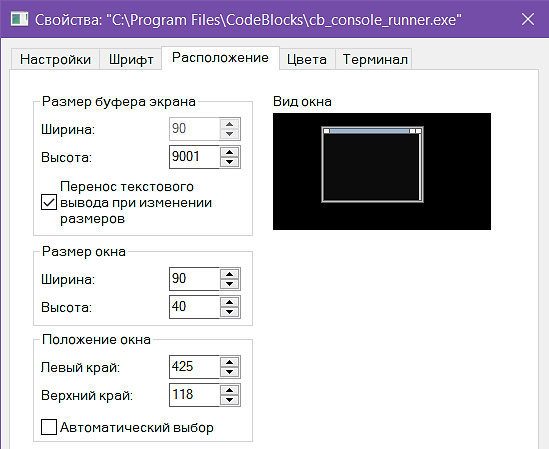


Рисунок 5 – Свойства: размеры буфера экрана и окна в строках и

столбцах символов, расположение окна

Считается, что консольная программа может выглядеть исключительно чёрно-белой и невзрачной, однако в данном случае вследствие управления цветами приложение обрело красивый внешний вид (рис. 6). Сделан этот шаг при помощи функции SetConsoleTextAttribute, которая позволяет установить цвета и другие атрибуты текста (например, подчёркивание). Перед её вызовом нужно извлечь дескриптор стандартного вывода STD\_OUTPUT\_HANDLE, возвращённый функцией GetStdHandle. Когда консоль создана, таковым и является требуемый дескриптор активного экранного буфера консоли [8].

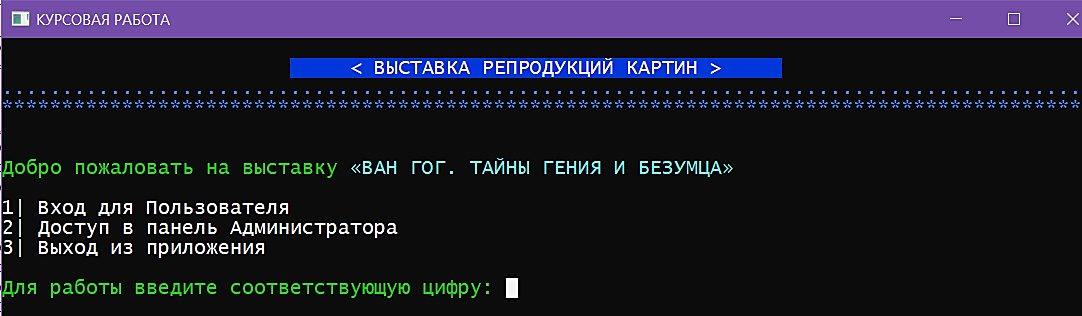


Рисунок 6 – Стартовая страница авторизации, видимая при запуске

Попадая в главное меню, пользователь делает выбор: войти как клиент, зайти как администратор или выйти из приложения.

После входа в систему в качестве гостя открывается соответствующее меню, где пользователь может узнать всё о данной выставке, посмотреть свои билеты и вернуться в главное меню программы.

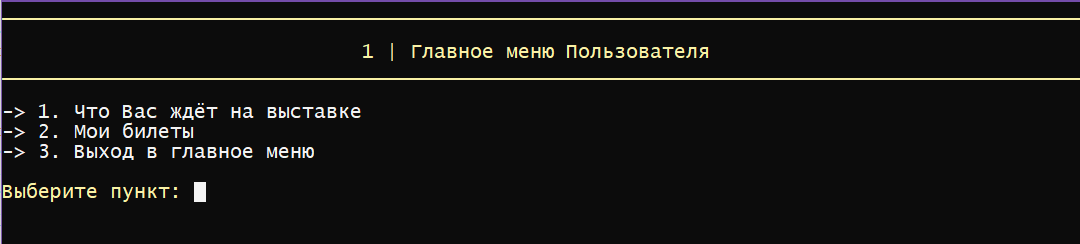


Рисунок 7 – Меню пользователя

В пункте 1 представлен список следующих разделов: общая информация, экспонаты и экскурсии выставки, выход назад (рис. 8).

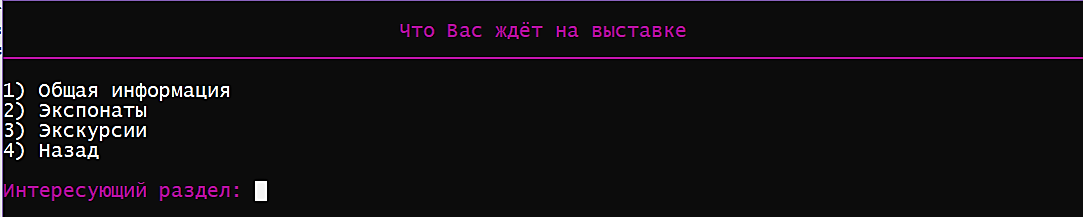


Рисунок 8 – Меню пользователя –> пункт 1

Раздел 1 выводит основные сведения о выставке. В «Экспонатах» у пользователя есть выбор действий: вывод картин, поиск по названию, сортировка по алфавиту и перейти назад (рис. 9).



Рисунок 9 – Раздел «Экспонаты»

Вывод экспонатов осуществляется в виде отформатированной таблицы для лёгкого восприятия человеком (рис. 10).

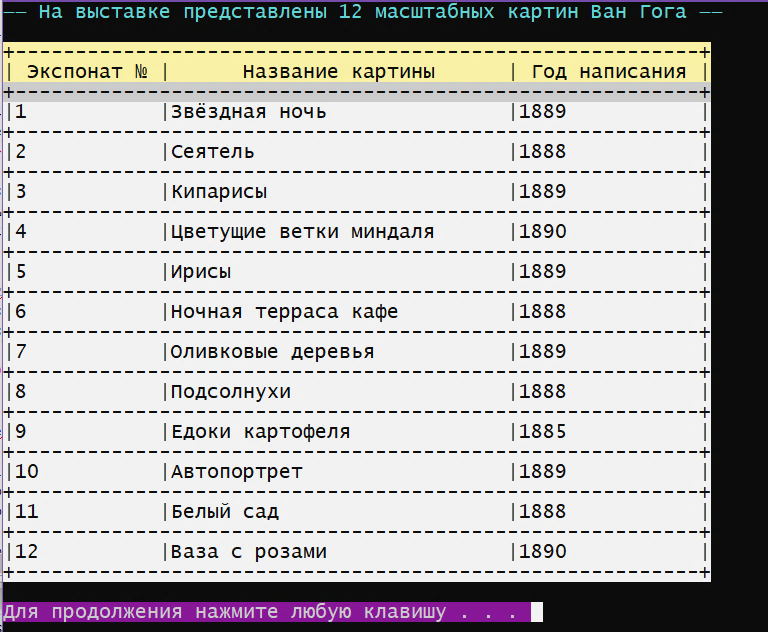


Рисунок 10 – Вывод списка экспонатов в консоль

Выбрав «Поиск по названию», пользователь видит текстовую строку, побуждающую ввести картину, которую он желает найти. После нажатия Enter, программа выдаёт результат: сообщение об отсутствии либо ему противоположное вместе с нужным пунктом из таблицы всех экспонатов.

«Сортировка по алфавиту» – на экран выводится отсортированный список экспонатов (также в виде таблицы). А действие «Назад» позволяет вернуться к первому пункту меню пользователя.

В третьем разделе пункта «Что вас ждёт на выставке» содержатся данные об экскурсиях. Пользователю предлагается ряд действий: вывод экскурсий, сортировка по цене, добавление в «Мои билеты» и вернуться назад, аналогичное описанному выше для экспонатов (рис. 11).

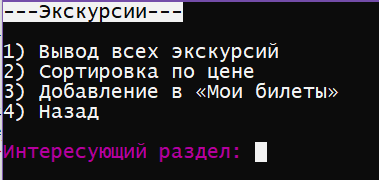


Рисунок 11 – Раздел «Экскурсии»

Все экскурсии выводятся в консоль подобно списку экспонатов, изображённому на рисунке 10. «Сортировка по цене» билетов реализована по числовому значению, хранящемуся в price и в результате её выполнения на экране появляется отсортированная таблица экскурсий. Добавление в «Мои билеты» позволяет пользователю поместить в корзину выбранный тип билета с целью его дальнейшего приобретения напрямую из раздела «Экскурсии».

После этого человек может перейти в сам пункт «Мои билеты» (рис. 12), где уже оформит заказ на добавленную услугу, выберет ещё один билет для покупки или же очистит корзину. Выход в меню пользователя будет произведён после нажатия на клавиатуре цифры 4.

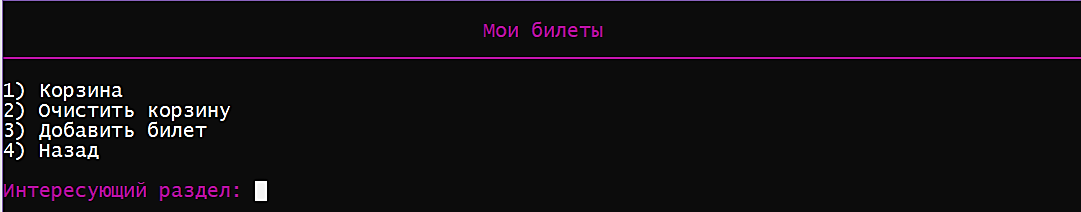


Рисунок 12 – Меню пользователя –> пункт 2

Доступ в панель управления администратором (2 пункт главного меню приложения) возможен исключительно по паролю, что гарантирует полную конфиденциальность. Точность поступившего на вход пароля проверяется с помощью операторов if...else. Условием является запись strcmp(str, s)==0, т.е. если сравниваемые строки искомого пароля str и введённого пользователем s идентичны, то человек, как администратор, попадает в меню редактирования записей (рис. 13). Иначе в консоли появляется сообщение, что пароль неверный и происходит возврат к основному меню (рис. 14).

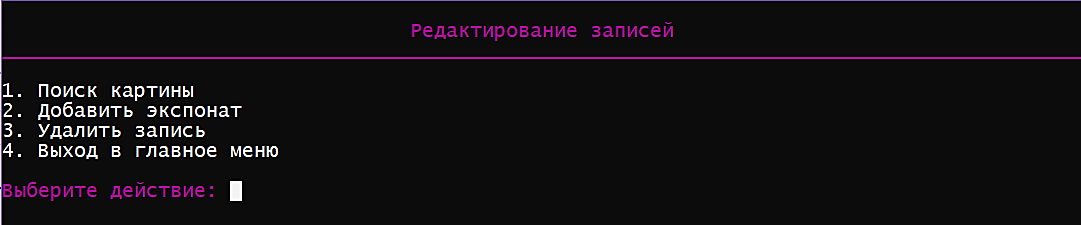


Рисунок 13 – Меню администратора

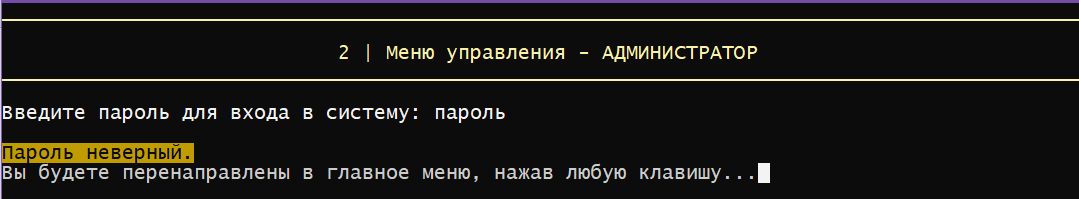


Рисунок 14 – Ответное действие на ошибочный ввод пароля

«Поиск картины» уже описывался ранее. Выбрав пункт 2 «Добавить экспонат», пользователю будет предложено заполнить все необходимые данные о картине. «Удаление записи» происходит по номеру экспоната, указанному в списке-таблице, отображаемой на экране. Выход в главное меню – по той же схеме, что и в режиме обычного пользователя.

Кроме того, в программе были обработаны исключительные ситуации, когда человек ввёл неверное слово или случайно нажал не ту клавишу. При наличии неправильного выбора раздела предусмотрено сбрасывание индикатора ошибок потока данных благодаря функции rewind. Она очищает буфер клавиатуры, устанавливая внутренний указатель стандартного файла ввода stdin в начальное положение. Тем самым осуществляется устранение программного сбоя – приложение не будет прекращать работу или зацикливаться.

Человеческий фактор учтён и при добавлении нового экспоната или экскурсии в корзину, а также удалении администратором записи. Если пользователь нечаянно вводит букву, отсутствующий в списке номер билета/картины или неподлинный год создания, то система оповещает об этом и позволяет записать числовое значение снова. На протяжении всей работы с программным средством человек ориентируется по заготовленным подсказкам. К примеру, его на каждом шаге встречает сообщение, какую клавишу нажать, чтобы продолжить. А во избежание произвольного «Выхода из приложения» пользователю прежде надо ответить на уточняющий вопрос (рис. 15).

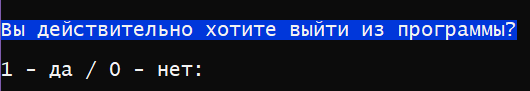


Рисунок 15 – Диалоговое окно выхода из программы

Опираясь на выявленные ранее сущности struct exh\_list\* ExhibitionList и struct excurs\_list\* ExcursionList, в курсовой работе используются два главных текстовых файла, хранящих данные об экспонатах и билетах на экскурсию: FILE\* pictures и FILE\* excursions соответственно. Для предотвращения внезапного закрытия программы реализована проверка, связанная с открытием файлов, и если результат неудачный, то система даст об этом знать (рис. 16).

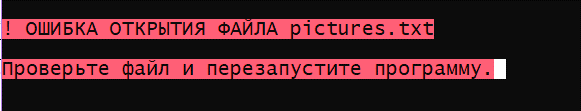


Рисунок 16 – Поведение приложения при ошибке открытия файла

Итак, была разобрана навигация по программе и пользовательский интерфейс вместе с обязательными для его реализации опциями. Выстроен принцип удобного диалога при работе человека с программным обеспечением. Полный код для исполнения описанных действий находится в Приложении А.

2 РАЗРАБОТКА КОНСОЛЬНОГО ПРИЛОЖЕНИЯ

2.1 Описание применяемых функций

В процессе разработки программный код был разделён на отдельные логические части. Это сделано с целью обхода дублирования и повышения его читаемости путём создания необходимых функций: инициализации, заполнения из файла, вывода и сортировки списков, добавления и удаления записи, поиска элемента и сохранения данных в файлах.

Стремясь не допустить падение производительности до низкого уровня, функциям передаются указатели на структуры, в результате чего в стек попадают именно их адреса и вызов выполняется очень быстро. Такой подход считается наиболее эффективным.

Первая вышеназванная функция ориентирована на создание списка. Её параметрами являются поля структуры exh\_list в первом случае (рис. 17) и excurs\_list во втором (рис. 18) за исключением порядковых номеров элементов их обеих (определяются самостоятельным числом). Т.к. сама функция служит только для инициализации первых объектов списка, под которые и выделяется динамически память при помощи malloc, то она применяется непосредственно внутри другой функции, а именно – заполнения структуры из файла. Возвращаемое значение – указатель на объявленную struct.

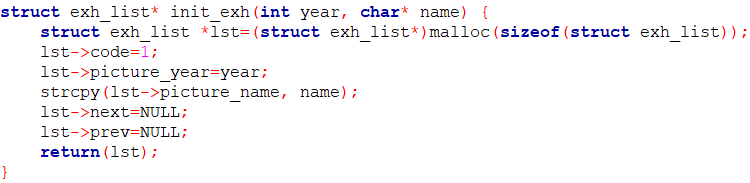


Рисунок 17 – Функция init\_exh для создания списка экспонатов

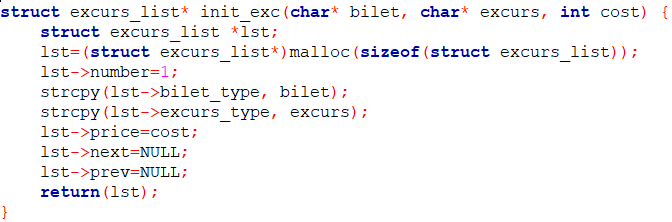


Рисунок 18 – Функция init\_exc для создания списка экскурсий

Там же можно встретить ещё одну функцию, которая возвращает указатель на новый узел списка, для чего собственно и реализована тоже в двух вариантах. Различие функции при struct excurs\_list и struct exh\_list состоит лишь в параметрах, которые уникальны для каждой структуры. Рассмотрим работу последней (рис. 19), т.к. она как раз понадобится и при добавлении записи администратором.

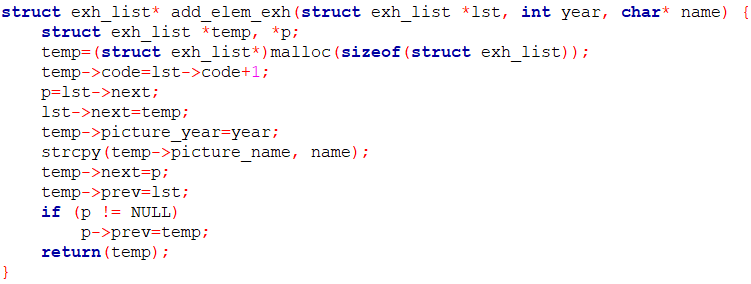


Рисунок 19 – Программный код функции add\_elem\_exh добавления

записи в список с экспонатами

Её параметры – указатель на структуру exh\_list, целочисленная переменная year и массив типа char, куда будет занесено название картины. Сначала в функции создаются два указателя на искомый список: \*temp и \*p. Далее выделяется память под переменную temp, содержащую адрес нового объекта структуры с экспонатами, а полю code, номеру узла списка, присваивается значение на 1 большее, чем было у прошлого элемента lst. Указатель на следующий узел сохраняется в «p»: p=lst->next, а на создаваемый узел – в предыдущий: lst->next =temp. Потом заполняются поля нового узла temp значениями year и name, которые передавались функции в качестве параметров. А полю новосозданного узла temp->next, отвечающему за указатель на следующий элемент структуры, присваивается нулевой указатель NULL – «p». Крайние строки кода написаны для визуализации двусвязного списка:

* temp->prev=lst (temp указывает на предыдущий узел);
* if (p!=NULL) p->prev=temp (сохранение ссылки на прошлый элемент для возможности перехода к нему при обращении к prev).

Добавление экспоната в режиме администратора осуществляется следующим образом. В первую очередь человек должен передать функции данные для заполнения. Часть кода – диалог:

printf("\nВведите название картины: ");

fflush(stdin); //очищаем поток ввода от \n

gets(picture\_name); //считываем введённую пользователем строку (scanf не подходит, т.к. считывает только до первого пробела)

strcat(picture\_name, "\n"); //присоединяем символ переноса строки к picture\_name, т.к. в файле все записи с новой строки

printf("\nВведите год создания: ");

scanf("%d", &year); //считываем число с клавиатуры

Затем объявляются два указателя struct exh\_list \*ptr, \*current на структуру «Экспонат» и ptr присваивается указатель на конкретный список ExhibitionList. В цикле – do ptr=ptr->next; while (ptr->next != NULL); – доходим до последнего объекта и вызываем current=add\_elem\_exh(ptr, year, picture\_name). Такая запись означает, что в переменную current возвращается память под узел списка, выделяемая в функции add\_elem\_exh. Т.е. current указывает на добавленный узел. Далее открывается файл с данными о картинах для дополнения в конец: FILE\* file=fopen("pictures.txt", "a"), туда заносится вся информация о последнем элементе структуры: fprintf(file, "%s\n", current->picture\_name); fprintf(file, "%d\n", current->picture\_year) и поток закрывается: fclose(file).

Итак, вторая из перечисленных функций включает в себя обе ранее описанные: инициализации и добавления. Она необходима для занесения из файла в список сведений об экспонатах в одной версии и об экскурсиях в другой. Открытие потока данных в режиме чтения происходит в main(): FILE\* pictures=fopen("pictures.txt", "r") и FILE\* excursions=fopen("excursions.txt", "r"). Каждая альтернативная функция построена по одному плану и имеет два параметра – указатель на структуру и файл:

1. Создаются массивы типа char для всех полей структуры, размеры которых определены при её объявлении (для целочисленных переменных picture\_year и price char-массивы состоят из 6 символов, с учётом длины в 4 цифры и символа конца строки вместе с «\n»).
2. Чтобы в дальнейшем получилось вывести список в консоль, вводится указатель на структуру \*firstelem для сохранения адреса начала списка. Он же записан в операторе return.
3. Из файла считываются первые строки определённой длины, равной размеру символьных массивов и помещаются в соответствующие переменные типа char (см. пункт 1).
4. Структура инициализируется посредством вызова функции «init\_...», а firstelem теперь указывает на первый заполненный объект списка.
5. В цикле while из файла по очереди считываются оставшиеся строки также с помощью fgets (см. пункт 3) и вызывается функция «add\_elem\_...», записывая их в должные поля списка. Условие цикла – пока не будет просмотрена вся информация в файле.
6. Возвращается результат функции – переменная firstelem и fclose закрывает поток данных.

Программный код функции заполнения структуры-экскурсии из файла «excursions» представлен на рисунке 20.

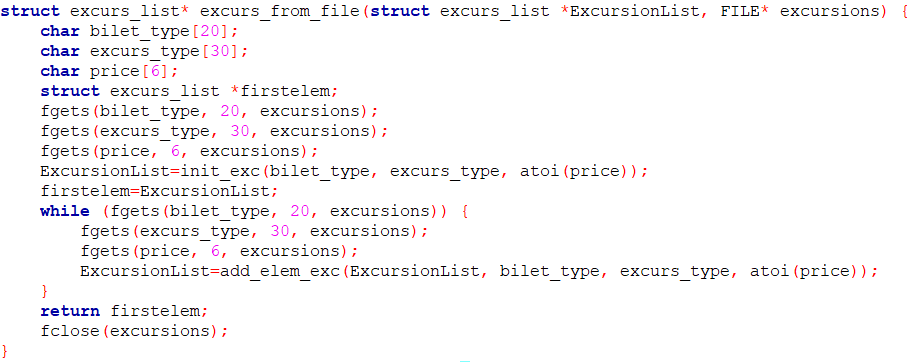


Рисунок 20 – Функция excurs\_from\_file

Вывод списков экспонатов и экскурсий реализован функциями print\_picture и print\_excursion соответственно. Они имеют тип void (ничего не возвращают), а в виде параметра – указатель lst на структуру, которую надо показать на экране. С целью сохранения адреса первого объекта в обеих функциях создаётся новый указатель на требуемый список – p, аналог lst (рис. 21-22).

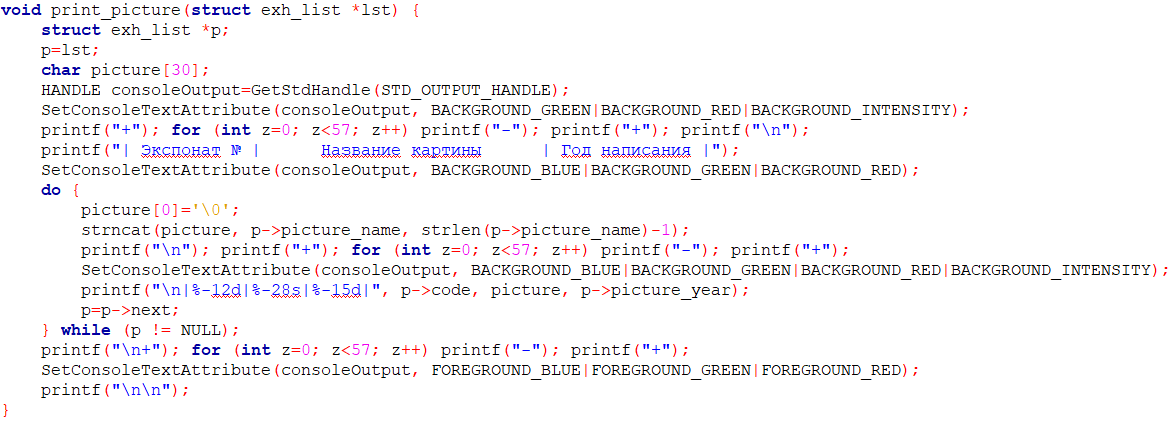


Рисунок 21 – Функция print\_picture

Т.к. в используемых текстовых файлах записи идут с новой строчки, то для полей типа char созданы дополнительные массивы, первым элементам которых присваивается символ '\0', превращая длину строки в нулевую. Делается это для реализации красивого табличного вывода благодаря функции strncat. Конкретнее: опустошённый массив char объединяется с искомой строкой из структуры данных, но с ограничением длины последней: strlen-1, т.е. без символа переноса строки.

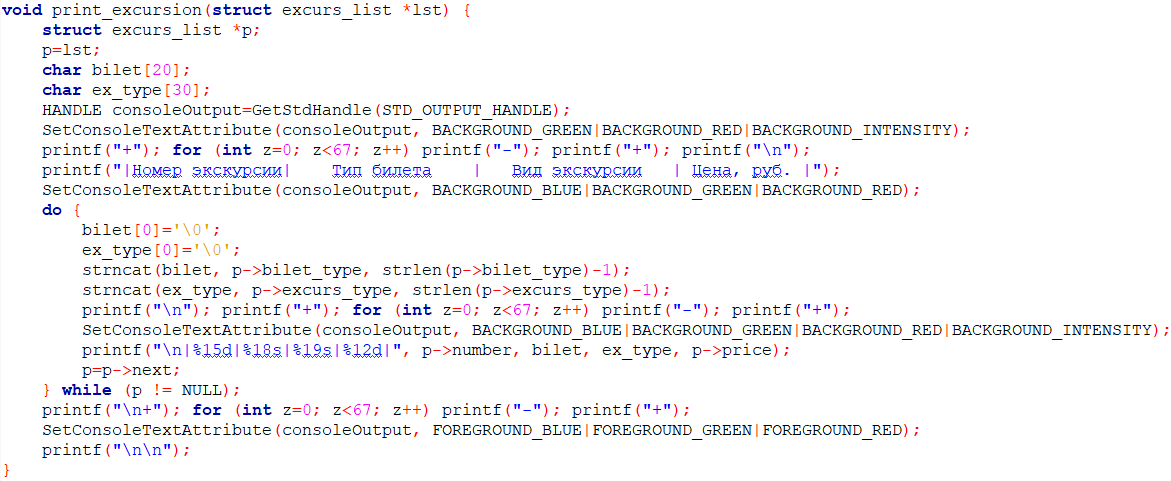


Рисунок 22 – Функция print\_excursion

Функция поиска картины по названию (рис. 23) основана на цикле с постусловием do while. Её тип, аргумент lst и объявляемый в теле указатель p идентичны описанным для функции вывода. Вторым же параметром выступает строка exponat\_name, вводимая пользователем с клавиатуры.

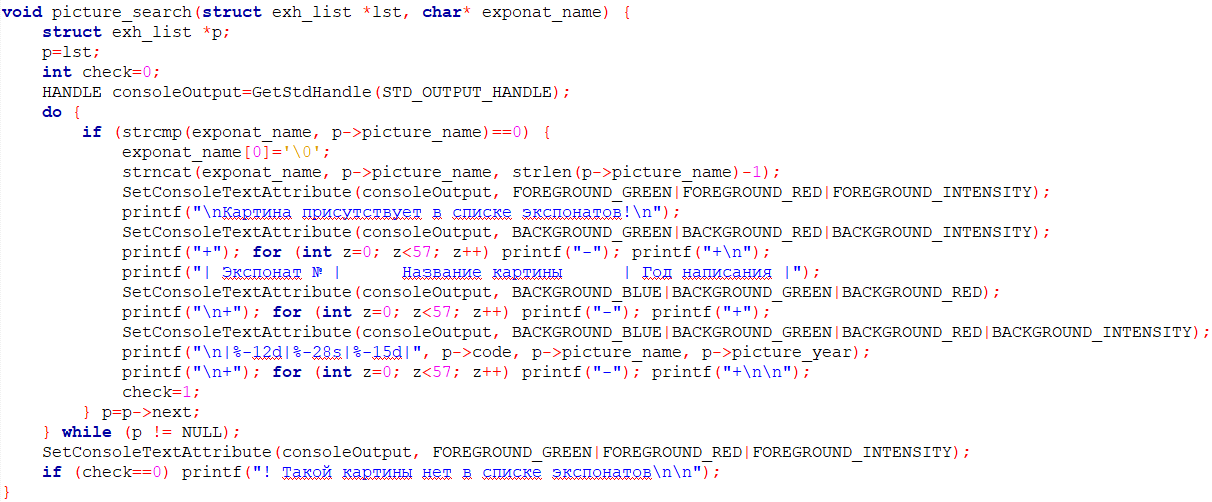


Рисунок 23 – Функция picture\_search

В блоке do проверяется условие её тождественности со строкой имени экспоната из списка: strcmp(exponat\_name, p->picture\_name)==0. В самом if с exponat\_name проводятся аналогичные действия для правильного вывода в виде таблицы (удаление «\n»), а также происходит инициализация переменной check=1 при нахождении картины. Цикл выполняется пока не будет достигнут последний элемент структуры.

Функции сортировки экспонатов и экскурсий используют в работе составленные при них вспомогательные, которые различаются лишь полем, выбранным для сравнения (буква или число), но функционируют одинаково. Изучим их действие на примере упорядочения картин по алфавиту (рис. 24).

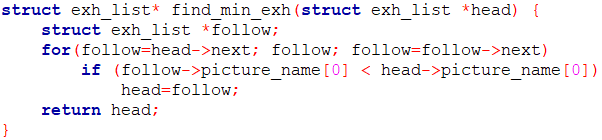


Рисунок 24 – Вспомогательная функция для сортировки экспонатов

Функция find\_min\_exh возвращает ссылку на элемент списка, где хранится название картины, которое начинается с ближайшего к началу алфавита символа. А определяется такой объект в цикле for, идущем по всей структуре с экспонатами до самого конца. Чтобы сравнивать значение, хранимое в начале списка, head, с тем, что находится в его следующем потомке head->next, создан указатель на этот перечень картин \*follow. Если первая буква имени его экспоната, располагаемого в follow->next, меньше чем данная буква в названии самого первого из списка, то они меняются местами: head=follow.

Сама функция сортировки в алфавитном порядке (рис. 25) принимает на вход один параметр – указатель на первый элемент структуры с экспонатами. Для начала создаётся массив char для имени картины picture\_name\_tmp. Внутри цикла прохода по всему списку for объявляется указатель minPtr на struct exh\_list. Туда записывается адрес объекта структуры с начальной алфавитной буквой в поле названия экспоната, возвращаемый функцией find\_min\_exh (здесь и происходит сортировка элементов). Далее идёт условие minPtr != list, чтобы в случае равенства не переписывать одну и ту же строку. Пройдя защиту от лишних действий if, при помощи strcpy и переменной picture\_name\_tmp производится обмен полей picture\_name между списками: только что найденным minPtr и непосредственно list. Аналогично меняются поля с годом picture\_year, чтобы при будущем оперировании данными о картинах ничего не перепуталось.

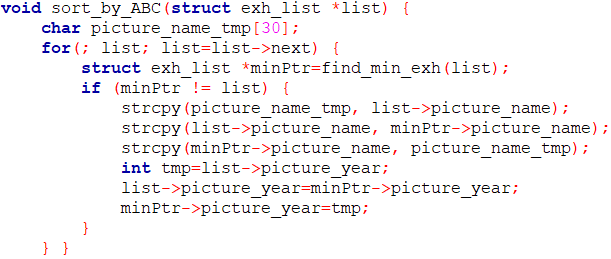


Рисунок 25 – Функция sort\_by\_ABC

Сортировка экскурсий по цене происходит похожим образом, просто изменяемые поля становятся другими, а функция find\_min\_excurs будет возвращать ссылку на экскурсию, билет на которую обойдётся дешевле. Программный код данной функции изображён на рисунке 26.

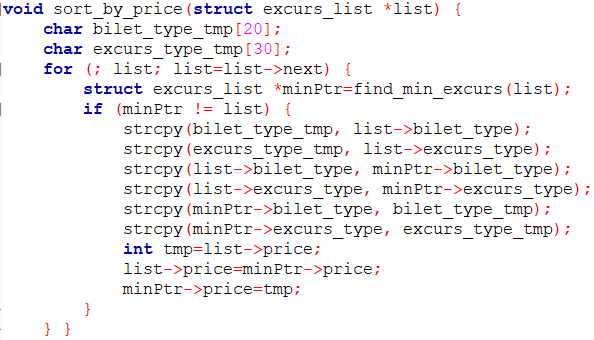


Рисунок 26 – Функция sort\_by\_price

Следующей значимой опцией программы является удаление записи. Тут предусмотрены несколько функций: deletehead для удаления головного элемента и delete\_exh для остальных. Первая функция (рис. 27) возвращает указатель temp, который в ней создаётся, на новый корень списка. Он ссылается на поле next текущего элемента, передаваемого в параметр – root, чья память и освобождается далее.

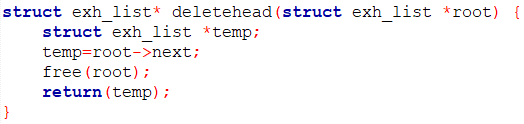


Рисунок 27 – Функция удаления корня списка

Вторая функция удаления delete\_exh(int N, struct exh\_list \*lst), принимающая на вход номер удаляемого элемента и ссылку lst на начало структуры, использует дополнительную – deletelem для исключения узла списка. Её код:

struct exh\_list\* deletelem(struct exh\_list \*lst, struct exh\_list \*root) {

struct exh\_list \*temp;

temp=root;

while (temp->next != lst) //просматриваем список начиная с корня, пока не найдем узел, предшествующий lst

temp=temp->next;

temp->next=lst->next; //переставляем указатель

free(lst); //освобождаем память удаляемого узла

return(temp);

В самой функции delete\_exh (рис. 28) в привычном цикле, пока не достигнут конец списка, на который указывает \*p и благодаря проверке условия N= =p->code указателю current присваивается p, тем самым сохраняется ссылка на удаляемый элемент. После этого вызывается deletelem с параметрами current и lst (фактический – ExhibitionList). Элемент успешно удалён.

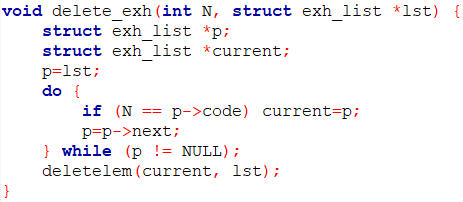


Рисунок 28 – Функция удаления записи delete\_exh

Обновление списка экспонатов с учётом удалённых элементов делает функция push\_in\_picture(struct exh\_list \*lst) типа void (рис. 29). Файл с картинами открывается для перезаписи и через указатель p на структуру exh\_list про- исходит доступ к нужным полям. Содержащаяся в них информация выводится в файл pictures.txt (fprintf), и он закрывается.

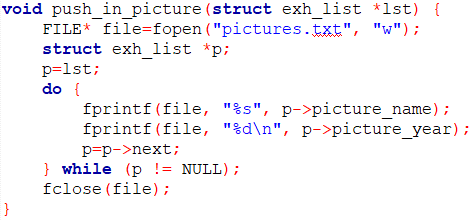


Рисунок 29 – Функция занесения в файл нового списка экспонатов после

удаления записей

Не менее важными являются и такие функции, как занесение в корзину билета и резервирование чека о его приобретении. Аргументы первой из них – номер экскурсии и сам список. Функция, представленная на рисунке 30, открывает уже существующий файл tickets.txt для записи в конец, чтобы можно было сохранить данные о нескольких позициях. Само же фиксирование экскурсий, выбранных человеком, осуществляется в блоке do, где сперва проверяется условие на совпадение введённого пользователем номера со значением поля number структуры excurs\_list, на которую указывает p: if (N==p->number). При положительном исходе в файл поступает необходимая информация благодаря fprintf. Затем происходит переход к следующему элементу. Цикл while выполняется пока переменная p не станет равной NULL, по окончании файл закрывается.

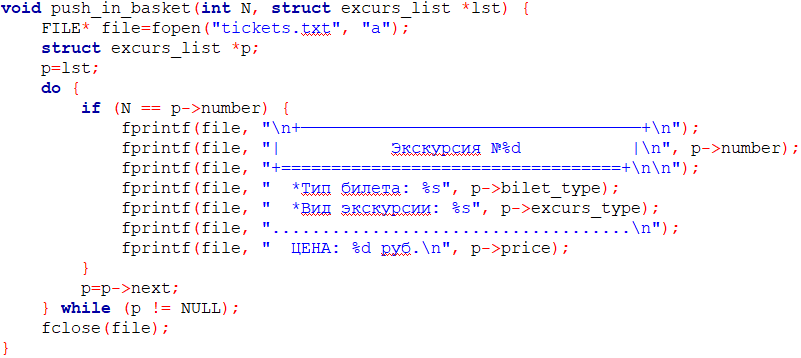


Рисунок 30 – Функция занесения экскурсии в корзину push\_in\_basket

Очистка корзины реализована в главной функции main открытием файла tickets.txt в режиме «w». Поскольку файл с таким именем уже существует – он стирается, что и требовалось сделать.

Ещё одна функция типа void (рис. 31) служит для сохранения чека на билет в специальном файле check.txt, который открывается для записи: FILE\* file\_check=fopen("check.txt", "w"). В цикле while считываются данные из «корзины», файла tickets.txt, открытого ранее для чтения: fscanf(file\_tickets, "%c", &ch) и переносятся в чек: fprintf(file\_check, "%c", ch). Потом оба потока закрываются.

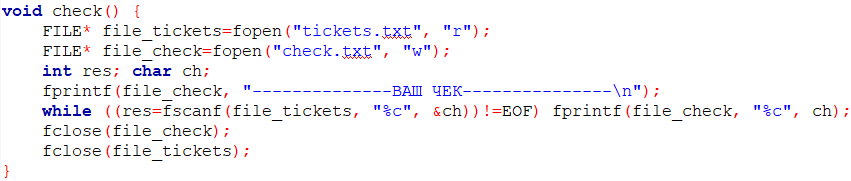


Рисунок 31 – Функция резервирования билетного чека check() в файл

Таким образом, полученный программный продукт был разработан на базе вышеописанных функций, а динамическая память, выделяемая под объекты двух структур, была освобождена перед выходом, дабы миновать противоестественное поведение приложения.

2.2 Тестирование программного решения

Проверка в действии готового программного обеспечения для выставки производилась на ноутбуке с шестиядерным процессором девятого поколения Intel Core i7-9750H, тактовая частота которого равна 2.60 GHz. Установленная операционная система – Windows 10, 64-разрядная, а объём оперативной памяти ноутбука составляет 8 ГБ.

Тесты на корректность отображения некоторых пунктов меню приложения, не исследованных в главах выше, показаны на рисунках 32-34.

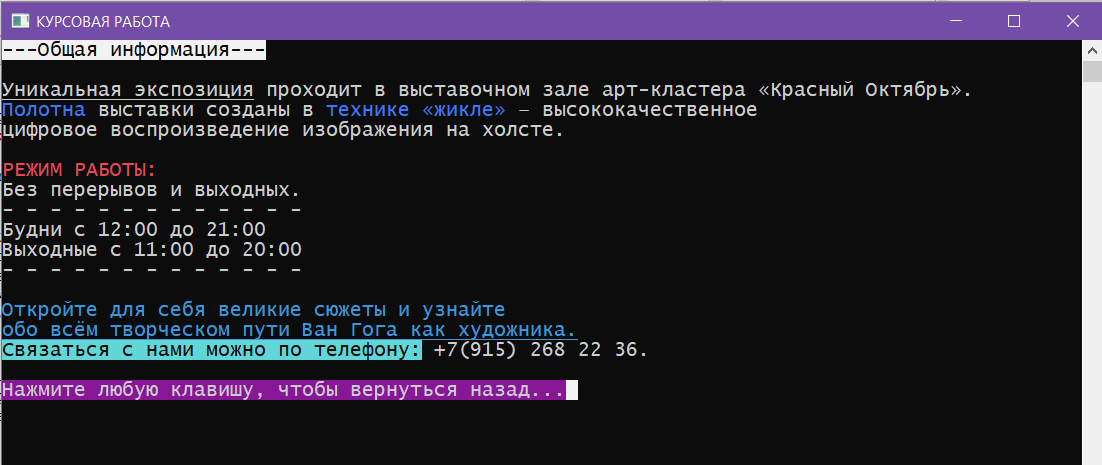


Рисунок 32 – Текст пункта «Общая информация»

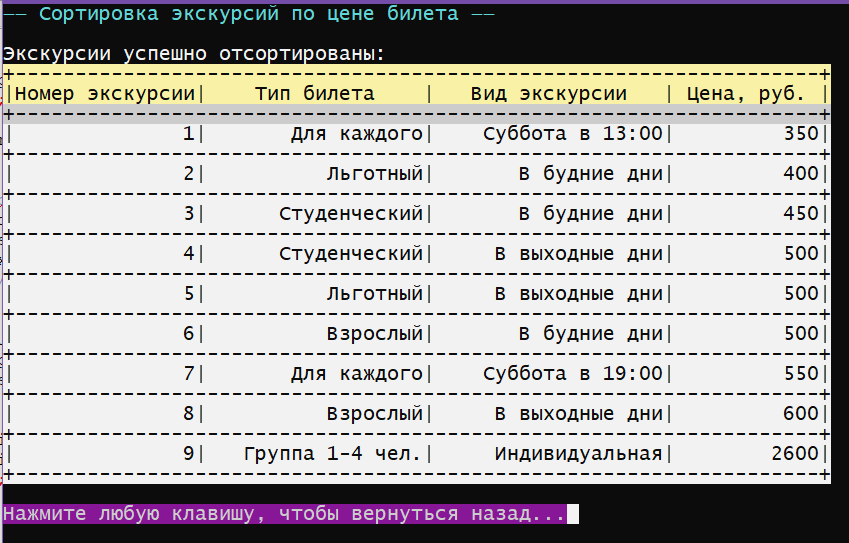


Рисунок 33 – Вывод отсортированного по цене списка экскурсий

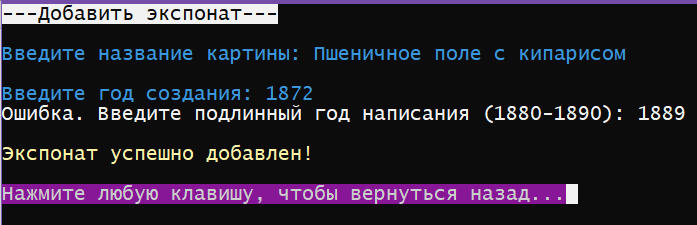


Рисунок 34 – Добавление картины

Исходя из проведённого тестирования можно заключить, что программное средство работает должным образом. Подробно разобраны принципы работы каждой из выявленных в ходе исследования функций CRUD, поиска, сортировки и работы с файлами. Приложение исправно действует и отвечает всем требованиям.

ЗАКЛЮЧЕНИЕ

Прикладное решение позволяет автоматизировать основные задачи, возникающие при организации выставочной деятельности и вести учёт по произведениям искусства и купленным экскурсионным билетам.

Итогом выполнения курсовой работы стало разработанное на языке С программное средство для предметной области «Выставка», которое позволяет пользователю комфортно пользоваться системой. Например, благодаря созданному продукту человек может без труда ознакомиться с экспонатами выставки, быстро проверить по названию наличие какой-либо картины на ней, отсортировать список экспонатов по алфавиту, а имеющиеся в наличии билеты по цене и др.

Выполнены следующие задачи, установленные при проведении исследования и подробном анализе его области:

* Разработан алгоритм с логичной архитектурой и понятным дизайном приложения за счёт оформления интерфейса.
* Элементы имеют иерархию по важности и согласно ей располагаются.
* Реализовано подключение к файлам и сохранение в них информации.
* Возможны поиск и сортировка данных.
* Подобраны осмысленные имена для функций и переменных.

Знания, полученные по дисциплине «Языки программирования», применены в ходе разработки данного программного решения, а функциональность готовой программы протестирована на всех возможных вариантах входных значений, следовательно, цель курсовой работы выполнена.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Зачем мы ходим на выставки и в галереи? [Электронный ресурс] // Виртуальная газета «Творчество в Петербурге». URL: http://gz.raz-vmeste.ru/2017/ 11/09/zachem-my-hodim-na-vystavki-i-v-galerei/ (Дата обращения: 05.06.2022).

2. Выставки. Кому и зачем они нужны? [Электронный ресурс] // Познавательный журнал ШколаЖизни. URL: https://www.shkolazhizni.ru/culture/ articles/31887/ (Дата обращения: 05.06.2022).

3. Базы данных «Выставочный комплекс» Visual FoxPro [Электронный ресурс] // Образовательный портал MyUniverCity. URL: https://www.myuniver-city.ru/Информатика/Базы\_данных\_Выставочный\_комплекс\_Visual/295194\_2788 990\_ страница1.html (Дата обращения: 06.06.2022).

4. Подбельский В. В. Курс программирования на языке Си : учебник / В. В. Подбельский, С. С. Фомин. – Москва : ДМК Пресс, 2012. // Лань: электронно-библиотечная система. URL: https://e.lanbook.com/book/4148 (Дата обращения: 6.06.2022). – Режим доступа: для авториз. пользователей.

5. Структуры в Си: конструктор, массивы и указатель на структуру [Электронный ресурс] // Портал для помощи программистам CoderNet. URL: https://codernet.ru/articles/drugoe/strukturyi\_v\_si\_konstruktor\_massivyi\_i\_ukazatel\_na\_strukturu/ (Дата обращения: 6.06.2022).

6. Двусвязный список [Электронный ресурс] // Лекции по С - learnc.info. URL: https://learnc.info/adt/double\_linked\_list.html (Дата обращения: 06.06.2022).

7. Приложение 1. Стандартные библиотеки языка Си [Электронный ресурс] // Информационный ресурс для обучения Лекции.Орг. URL: https://lektsii.org/7-60721.html (Дата обращения: 06.06.2022).

8. Дескрипторы консоли [Электронный ресурс] // Проект MSDN - Windows API по-русски веб-сайта Narovol.ru. URL: http://narovol.narod.ru/\_ tbkp/New\_MSDN\_API/Console/hend\_con.htm (Дата обращения: 07.06.2022).

ПРИЛОЖЕНИЕ А

Листинг полного программного кода

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <string.h>

struct exh\_list {

int code, picture\_year;

char picture\_name[30];

struct exh\_list\* next;

struct exh\_list\* prev;

};

struct excurs\_list {

int number, price;

char bilet\_type[20];

char excurs\_type[30];

struct excurs\_list\* next;

struct excurs\_list\* prev;

};

//ЭКСПОНАТЫ:

struct exh\_list\* init\_exh(int year, char\* name) {

struct exh\_list \*lst=(struct exh\_list\*)malloc(sizeof(struct exh\_list));

lst->code=1;

lst->picture\_year=year;

strcpy(lst->picture\_name, name);

lst->next=NULL;

lst->prev=NULL;

return(lst);

}

struct exh\_list\* add\_elem\_exh(struct exh\_list \*lst, int year, char\* name) {

struct exh\_list \*temp, \*p;

temp=(struct exh\_list\*)malloc(sizeof(struct exh\_list));

temp->code=lst->code+1;

p=lst->next;

lst->next=temp;

temp->picture\_year=year;

strcpy(temp->picture\_name, name);

temp->next=p;

temp->prev=lst;

if (p != NULL)

p->prev=temp;

return(temp);

}

struct exh\_list\* picture\_from\_file(struct exh\_list \*ExhibitionList, FILE\* pictures) {

char picture\_name[30];

char picture\_year[6];

struct exh\_list \*firstelem;

fgets(picture\_name, 30, pictures);

fgets(picture\_year, 6, pictures);

ExhibitionList=init\_exh(atoi(picture\_year), picture\_name);

firstelem=ExhibitionList;

while (fgets(picture\_name, 30, pictures)) {

fgets(picture\_year, 6, pictures);

ExhibitionList=add\_elem\_exh(ExhibitionList, atoi(picture\_year), picture\_name);

}

return firstelem;

fclose(pictures);

}

void print\_picture(struct exh\_list \*lst) {

struct exh\_list \*p;

p=lst;

char picture[30];

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

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("+"); for (int z=0; z<57; z++) printf("-"); printf("+"); printf("\n");

printf("| Экспонат № | Название картины | Год написания |");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED);

do {

picture[0]='\0';

strncat(picture, p->picture\_name, strlen(p->picture\_name)-1);

printf("\n"); printf("+"); for (int z=0; z<57; z++) printf("-"); printf("+");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("\n|%-12d|%-28s|%-15d|", p->code, picture, p->picture\_year);

p=p->next;

} while (p != NULL);

printf("\n+"); for (int z=0; z<57; z++) printf("-"); printf("+");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

printf("\n\n");

}

void picture\_search(struct exh\_list \*lst, char\* exponat\_name) {

struct exh\_list \*p;

p=lst;

int check=0;

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

do {

if (strcmp(exponat\_name, p->picture\_name)==0) {

exponat\_name[0]='\0';

strncat(exponat\_name, p->picture\_name, strlen(p->picture\_name)-1);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\nКартина присутствует в списке экспонатов!\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("+"); for (int z=0; z<57; z++) printf("-"); printf("+\n");

printf("| Экспонат № | Название картины | Год написания |");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED);

printf("\n+"); for (int z=0; z<57; z++) printf("-"); printf("+");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("\n|%-12d|%-28s|%-15d|", p->code, exponat\_name, p->picture\_year);

printf("\n+"); for (int z=0; z<57; z++) printf("-"); printf("+\n\n");

check=1;

} p=p->next;

} while (p != NULL);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

if (check==0) printf("! Такой картины нет в списке экспонатов\n\n");

}

struct exh\_list\* find\_min\_exh(struct exh\_list \*head) { //вспомогательная

struct exh\_list \*follow;

for(follow=head->next; follow; follow=follow->next)

if (follow->picture\_name[0] < head->picture\_name[0])

head=follow;

return head;

}

void sort\_by\_ABC(struct exh\_list \*list) {

char picture\_name\_tmp[30];

for(; list; list=list->next) {

struct exh\_list \*minPtr=find\_min\_exh(list);

if (minPtr != list) {

strcpy(picture\_name\_tmp, list->picture\_name);

strcpy(list->picture\_name, minPtr->picture\_name);

strcpy(minPtr->picture\_name, picture\_name\_tmp);

int tmp=list->picture\_year;

list->picture\_year=minPtr->picture\_year;

minPtr->picture\_year=tmp;

}

} }

struct exh\_list\* deletehead(struct exh\_list \*root) {

struct exh\_list \*temp;

temp=root->next;

free(root);

return(temp);

}

struct exh\_list\* deletelem(struct exh\_list \*lst, struct exh\_list \*root) {

struct exh\_list \*temp;

temp=root;

while (temp->next != lst)

temp=temp->next;

temp->next=lst->next;

free(lst);

return(temp);

}

void delete\_exh(int N, struct exh\_list \*lst) {

struct exh\_list \*p;

struct exh\_list \*current;

p=lst;

do {

if (N == p->code) current=p;

p=p->next;

} while (p != NULL);

deletelem(current, lst);

}

void push\_in\_picture(struct exh\_list \*lst) { //с учётом удалённых

FILE\* file=fopen("pictures.txt", "w");

struct exh\_list \*p;

p=lst;

do {

fprintf(file, "%s", p->picture\_name);

fprintf(file, "%d\n", p->picture\_year);

p=p->next;

} while (p != NULL);

fclose(file);

}

//ЭКСКУРСИИ:

struct excurs\_list\* init\_exc(char\* bilet, char\* excurs, int cost) {

struct excurs\_list \*lst;

lst=(struct excurs\_list\*)malloc(sizeof(struct excurs\_list));

lst->number=1;

strcpy(lst->bilet\_type, bilet);

strcpy(lst->excurs\_type, excurs);

lst->price=cost;

lst->next=NULL;

lst->prev=NULL;

return(lst);

}

struct excurs\_list\* add\_elem\_exc(struct excurs\_list \*lst, char\* bilet, char\* excurs, int cost) {

struct excurs\_list \*temp, \*p;

temp=(struct excurs\_list\*)malloc(sizeof(struct excurs\_list));

temp->number=lst->number+1;

p=lst->next;

lst->next=temp;

strcpy(temp->bilet\_type, bilet);

strcpy(temp->excurs\_type, excurs);

temp->price=cost;

temp->next=p;

temp->prev=lst;

if (p != NULL)

p->prev=temp;

return(temp);

}

struct excurs\_list\* excurs\_from\_file(struct excurs\_list \*ExcursionList, FILE\* excursions) {

char bilet\_type[20];

char excurs\_type[30];

char price[6];

struct excurs\_list \*firstelem;

fgets(bilet\_type, 20, excursions);

fgets(excurs\_type, 30, excursions);

fgets(price, 6, excursions);

ExcursionList=init\_exc(bilet\_type, excurs\_type, atoi(price));

firstelem=ExcursionList;

while (fgets(bilet\_type, 20, excursions)) {

fgets(excurs\_type, 30, excursions);

fgets(price, 6, excursions);

ExcursionList=add\_elem\_exc(ExcursionList, bilet\_type, excurs\_type, atoi(price));

}

return firstelem;

fclose(excursions);

}

void print\_excursion(struct excurs\_list \*lst) {

struct excurs\_list \*p;

p=lst;

char bilet[20];

char ex\_type[30];

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

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("+"); for (int z=0; z<67; z++) printf("-"); printf("+"); printf("\n");

printf("|Номер экскурсии| Тип билета | Вид экскурсии | Цена, руб. |");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED);

do {

bilet[0]='\0';

ex\_type[0]='\0';

strncat(bilet, p->bilet\_type, strlen(p->bilet\_type)-1);

strncat(ex\_type, p->excurs\_type, strlen(p->excurs\_type)-1);

printf("\n"); printf("+"); for (int z=0; z<67; z++) printf("-"); printf("+");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("\n|%15d|%18s|%19s|%12d|", p->number, bilet, ex\_type, p->price);

p=p->next;

} while (p != NULL);

printf("\n+"); for (int z=0; z<67; z++) printf("-"); printf("+");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

printf("\n\n");

}

struct excurs\_list\* find\_min\_excurs(struct excurs\_list \*head) { //вспомогательная

struct excurs\_list \*follow;

for (follow=head->next; follow; follow=follow->next)

if (follow->price < head->price)

head=follow;

return head;

}

void sort\_by\_price(struct excurs\_list \*list) {

char bilet\_type\_tmp[20];

char excurs\_type\_tmp[30];

for (; list; list=list->next) {

struct excurs\_list \*minPtr=find\_min\_excurs(list);

if (minPtr != list) {

strcpy(bilet\_type\_tmp, list->bilet\_type);

strcpy(excurs\_type\_tmp, list->excurs\_type);

strcpy(list->bilet\_type, minPtr->bilet\_type);

strcpy(list->excurs\_type, minPtr->excurs\_type);

strcpy(minPtr->bilet\_type, bilet\_type\_tmp);

strcpy(minPtr->excurs\_type, excurs\_type\_tmp);

int tmp=list->price;

list->price=minPtr->price;

minPtr->price=tmp;

}

} }

void push\_in\_basket(int N, struct excurs\_list \*lst) {

FILE\* file=fopen("tickets.txt", "a");

struct excurs\_list \*p;

p=lst;

do {

if (N == p->number) {

fprintf(file, "\n+——————————————————————————————————+\n");

fprintf(file, "| Экскурсия №%d |\n", p->number);

fprintf(file, "+==================================+\n\n");

fprintf(file, " \*Тип билета: %s", p->bilet\_type);

fprintf(file, " \*Вид экскурсии: %s", p->excurs\_type);

fprintf(file, "....................................\n");

fprintf(file, " ЦЕНА: %d руб.\n", p->price);

}

p=p->next;

} while (p != NULL);

fclose(file);

}

void check() {

FILE\* file\_tickets=fopen("tickets.txt", "r");

FILE\* file\_check=fopen("check.txt", "w");

int res; char ch;

fprintf(file\_check, "--------------ВАШ ЧЕК---------------\n");

while ((res=fscanf(file\_tickets, "%c", &ch))!=EOF) fprintf(file\_check, "%c", ch);

fclose(file\_check);

fclose(file\_tickets);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("title КУРСОВАЯ РАБОТА");

int choice=0, z;

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

struct exh\_list\* ExhibitionList=NULL;

FILE\* pictures=fopen("pictures.txt", "r");

if (!pictures) {

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("\n! ОШИБКА ОТКРЫТИЯ ФАЙЛА pictures.txt\n\n"); system("pause>nul|set/p=Проверьте файл и перезапустите программу."); exit(1); }

ExhibitionList=picture\_from\_file(ExhibitionList, pictures);

struct excurs\_list\* ExcursionList=NULL;

FILE\* excursions=fopen("excursions.txt", "r");

if (!excursions) {

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("\n! ОШИБКА ОТКРЫТИЯ ФАЙЛА excursions.txt\n\n"); system("pause>nul|set/p=Проверьте файл и перезапустите программу."); exit(1); }

ExcursionList=excurs\_from\_file(ExcursionList, excursions);

while (choice != 3) {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY|BACKGROUND\_BLUE);

printf("\n\t\t\t < ВЫСТАВКА РЕПРОДУКЦИЙ КАРТИН > \n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_INTENSITY);

for (z=0; z<90; z++) printf("."); printf("\n");

for (z=0; z<90; z++) printf("\*"); printf("\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("\n\nДобро пожаловать на выставку ");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("«ВАН ГОГ. ТАЙНЫ ГЕНИЯ И БЕЗУМЦА»\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n1| Вход для Пользователя\n2| Доступ в панель Администратора\n3| Выход из приложения\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("Для работы введите соответствующую цифру: ");

scanf("%d", &choice);

rewind(stdin);

switch(choice) {

case 1: {

int point\_1=0;

while (point\_1 != 3) {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

for (z=0; z<90; z++) printf("\_"); printf("\n");

printf("\n\t\t\t 1 | Главное меню Пользователя\n");

for (z=0; z<90; z++) printf("\_"); printf("\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n-> 1. Что Вас ждёт на выставке\n-> 2. Мои билеты\n-> 3. Выход в главное меню\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Выберите пункт: ");

scanf("%d", &point\_1);

rewind(stdin);

switch(point\_1) {

case 1: {

int point\_1\_0=0;

while (point\_1\_0 != 4) {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n\t\t\t\t Что Вас ждёт на выставке\n");

for (z=0; z<90; z++) printf("\_"); printf("\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n1) Общая информация\n2) Экспонаты\n3) Экскурсии\n4) Назад\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Интересующий раздел: ");

scanf("%d", &point\_1\_0);

rewind(stdin);

switch(point\_1\_0) {

case 1: {

system("cls");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Общая информация---\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|COMMON\_LVB\_UNDERSCORE);

printf("Уникальная экспозиция");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

FILE\* info\_1 = fopen("info\_1.txt", "r");

int res; char ch;

while ((res=fscanf(info\_1, "%c", &ch)) != EOF) printf("%c", ch);

fclose(info\_1);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_INTENSITY);

printf("\nПолотна");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

FILE\* info\_2 = fopen("info\_2.txt", "r");

while ((res=fscanf(info\_2, "%c", &ch)) != EOF) printf("%c", ch);

fclose(info\_2);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_INTENSITY);

printf("технике «жикле» ");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

FILE\* info\_3 = fopen("info\_3.txt", "r");

while ((res=fscanf(info\_3, "%c", &ch)) != EOF) printf("%c", ch);

fclose(info\_3);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n\nРЕЖИМ РАБОТЫ:\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

FILE\* info\_4 = fopen("info\_4.txt", "r");

while ((res=fscanf(info\_4, "%c", &ch)) != EOF) printf("%c", ch);

fclose(info\_4);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN);

printf("\n\nОткройте для себя великие сюжеты и узнайте\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|COMMON\_LVB\_UNDERSCORE);

FILE\* info\_5 = fopen("info\_5.txt", "r");

while ((res=fscanf(info\_5, "%c", &ch)) != EOF) printf("%c", ch);

fclose(info\_5);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_INTENSITY);

printf("\nСвязаться с нами можно по телефону:");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

printf(" +7(915) 268 22 36.\n\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад..."); //изменение текста от pause

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 2: {

int point\_1\_1=0;

while (point\_1\_1 != 4) {

system("cls");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Экспонаты---\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("1) Вывод всех экспонатов\n2) Поиск по названию\n3) Сортировка по алфавиту\n4) Назад\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Интересующий раздел: ");

scanf("%d", &point\_1\_1);

rewind(stdin);

switch(point\_1\_1) {

case 1: {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("—— На выставке представлены 12 масштабных картин Ван Гога ——\n\n");

print\_picture(ExhibitionList);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 2: {

system("cls");

char string[30];

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("—— Поиск картины ——\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Пожалуйста, введите точное название: ");

fflush(stdin);

gets(string);

strcat(string, "\n");

picture\_search(ExhibitionList, string);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад...");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 3: {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("—— Сортировка экспонатов по алфавиту ——\n\n");

sort\_by\_ABC(ExhibitionList);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Экспонаты успешно отсортированы:\n");

print\_picture(ExhibitionList);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад...");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 4: break;

} }

break;

}

case 3: {

int point\_1\_2=0;

while (point\_1\_2 != 4) {

system("cls");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Экскурсии---\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("1) Вывод всех экскурсий\n2) Сортировка по цене\n3) Добавление в «Мои билеты»\n4) Назад\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Интересующий раздел: ");

scanf("%d", &point\_1\_2);

rewind(stdin);

switch(point\_1\_2) {

case 1: {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("—— Варианты экскурсий ——\n\n");

print\_excursion(ExcursionList);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 2: {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("—— Сортировка экскурсий по цене билета ——\n\n");

sort\_by\_price(ExcursionList);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Экскурсии успешно отсортированы:\n");

print\_excursion(ExcursionList);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад...");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 3: {

system("cls");

int vibor;

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("—— Добавление в «Мои билеты» ——\n\n");

print\_excursion(ExcursionList);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Введите номер экскурсии для добавления в корзину: ");

scanf("%d", &vibor);

rewind(stdin);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

if ((vibor>0) && (vibor<10)) {

push\_in\_basket(vibor, ExcursionList);

printf("Экскурсия успешно добавлена!\n\n");

} else printf("Некорректное значение. Введите цифру от 1 до 9.\n\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 4: break;

} }

break;

}

case 4: break;

} }

break;

}

case 2: {

int point\_1\_3=0;

while (point\_1\_3 != 4) {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n\t\t\t\t\tМои билеты\n");

for (z=0; z<90; z++) printf("\_"); printf("\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n1) Корзина\n2) Очистить корзину\n3) Добавить билет\n4) Назад\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Интересующий раздел: ");

scanf("%d", &point\_1\_3);

rewind(stdin);

switch(point\_1\_3) {

case 1: {

system("cls");

int res; char ch;

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Корзина---\n\n");

FILE\* tickets=fopen("tickets.txt", "r");

fseek(tickets, 0, SEEK\_END); //коретка в конец файла

int pos=ftell(tickets); //текущая позиция в tickets

if (pos <= 0) {

fclose(tickets);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Ваша корзина пока что пустая!\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

printf("Перейдите в раздел: 3) Добавить билет\n\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

else {

tickets=fopen("tickets.txt", "r");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("Билеты в корзине:");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_INTENSITY);

while ((res=fscanf(tickets, "%c", &ch))!=EOF) printf("%c", ch);

fclose(tickets);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_INTENSITY);

printf("\n\nЖелаете приобрести? (1 - Да, 0 - Нет) ");

int buy;

do {

scanf("%d", &buy);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

switch(buy) {

case 0: {

system("cls");

point\_1\_3=0;

break;

}

case 1: {

check();

printf("\nСпасибо! Билеты приобретены.\nВаш ЧЕК находится в файле check.txt\n\n");

FILE\* file=fopen("tickets.txt", "w");

fclose(file);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

default: {

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

printf("\nВы ввели некоректное значение. Пожалуйста, попробуйте ещё раз ");

rewind(stdin);

break;

} }

} while ((buy!=0) && (buy!=1));

};

break;

}

case 2: {

system("cls");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Очистить корзину---\n\n");

FILE\* file=fopen("tickets.txt", "w");

fclose(file);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("~ КОРЗИНА УСПЕШНО ОЧИЩЕНА ~\n\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад...");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 3: {

system("cls");

int vibor;

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Добавить билет---\n\n");

print\_excursion(ExcursionList);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Введите номер экскурсии для добавления в корзину: ");

scanf("%d", &vibor);

rewind(stdin);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

if ((vibor>0) && (vibor<10)) {

push\_in\_basket(vibor, ExcursionList);

printf("Экскурсия успешно добавлена!\n\n");

} else printf("Некорректное значение. Введите цифру от 1 до 9.\n\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break;

}

case 4: break;

} }

break;

}

case 3: break;

} } break;

}

case 2: {

int point\_2 = 0;

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

for (z=0; z<90; z++) printf("\_"); printf("\n");

printf("\n\t\t\t 2 | Меню управления - АДМИНИСТРАТОР\n");

for (z=0; z<90; z++) printf("\_"); printf("\n");

const char str[10]="password2";

char s[10];

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\nВведите пароль для входа в систему: ");

scanf("%s", s);

if (strcmp(str, s)==0) {

while (point\_2 != 4) {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n\t\t\t\t Редактирование записей\n");

for (z=0; z<90; z++) printf("\_"); printf("\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n1. Поиск картины\n2. Добавить экспонат\n3. Удалить запись\n4. Выход в главное меню\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\nВыберите действие: ");

scanf("%d", &point\_2);

rewind(stdin);

switch(point\_2) {

case 1: {

system("cls");

char string[30];

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Поиск картины---\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN);

printf("Пожалуйста, введите точное название: ");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_RED|FOREGROUND\_GREEN);

fflush(stdin);

gets(string);

strcat(string, "\n");

picture\_search(ExhibitionList, string);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад...");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break; }

case 2: {

system("cls");

char picture\_name[30];

int year;

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Добавить экспонат---\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN);

printf("\nВведите название картины: ");

fflush(stdin);

gets(picture\_name);

strcat(picture\_name, "\n");

printf("\nВведите год создания: ");

scanf("%d", &year);

while ((year>1890) || (year<1880)) {

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Ошибка. Введите подлинный год написания (1880-1890): ");

fflush(stdin);

scanf("%d", &year); }

struct exh\_list \*ptr, \*current;

ptr=ExhibitionList;

do ptr=ptr->next; while (ptr->next != NULL);

current=add\_elem\_exh(ptr, year, picture\_name);

FILE\* file=fopen("pictures.txt", "a");

fprintf(file, "%s\n", current->picture\_name);

fprintf(file, "%d\n", current->picture\_year);

fclose(file);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\nЭкспонат успешно добавлен!\n\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад...");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break; }

case 3: {

system("cls");

int count=0, delete\_num;

FILE\* file=fopen("pictures.txt", "r");

while (!feof(file)) {

if (fgetc(file)=='\n')

count++; //кол-во строк в файле

} fclose(file);

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_GREEN|BACKGROUND\_RED|BACKGROUND\_INTENSITY);

printf("---Удалить запись---\n\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_INTENSITY);

printf("Список экспонатов:\n");

print\_picture(ExhibitionList);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("Введите номер экспоната для удаления: ");

scanf("%d", &delete\_num);

rewind(stdin);

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

if ((delete\_num>0) && (delete\_num<=(count/2))) {

delete\_exh(delete\_num, ExhibitionList);

printf("Запись успешно удалена!\n\n");

push\_in\_picture(ExhibitionList); }

else if (delete\_num==1) {

ExhibitionList=deletehead(ExhibitionList);

printf("Запись успешно удалена!\n\n");

push\_in\_picture(ExhibitionList);

} else printf("\nВы ввели некоректное значение.\n");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|BACKGROUND\_RED|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Нажмите любую клавишу, чтобы вернуться назад...");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

break; }

case 4: break;

} } }

else {

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_RED|BACKGROUND\_GREEN); printf("\nПароль неверный.\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

system("pause>nul|set/p=Вы будете перенаправлены в главное меню, нажав любую клавишу...");

} break;

}

case 3: {

system("cls");

SetConsoleTextAttribute(consoleOutput, BACKGROUND\_BLUE|FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\nВы действительно хотите выйти из программы?\n");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

printf("\n1 - да / 0 - нет: ");

int figure;

do {

scanf("%d", &figure);

rewind(stdin);

switch(figure) {

case 0: {

system("cls");

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED|FOREGROUND\_INTENSITY);

choice=0;

break;

}

case 1: {

system("cls");

break;

}

default: {

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

printf("\nВы ввели некоректное значение. Пожалуйста, попробуйте ещё раз ");

rewind(stdin);

break;

} }

} while ((figure!=0) && (figure!=1));

break;

} default: {

SetConsoleTextAttribute(consoleOutput, FOREGROUND\_BLUE|FOREGROUND\_GREEN|FOREGROUND\_RED);

printf("\nНеверно выбран режим работы. Допустимые варианты: 1, 2, 3.\nНажмите Enter для повторного ввода");

getchar();

break;

} } }

free(ExhibitionList);

free(ExcursionList);

return 0;

}