Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: Разработка системы классов для игры “Путь в колледж”**

**Специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 31ИС-23** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **К.В. Терсков** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Л.Б. Гусятинер** |

**Москва 2024**

Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С. Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Студент группы 302ИС-22 Терсков Кирилл**

**ТЕМА: «: Разработка системы классов для игры “Путь в колледж”»**

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи проекта «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Москва 2024**

**Оглавление**

[Введение 4](#_Toc184267885)

[Глава 1. Теоретическая область 5](#_Toc184267886)

[1.1. Описание предметной области 5](#_Toc184267887)

[1.2. Описание существующих разработок 8](#_Toc184267888)

[Глава 2. Проектная часть 10](#_Toc184267889)

[2.1. Выбор инструментов 10](#_Toc184267890)

[2.2. Проектирование сценария 11](#_Toc184267891)

[2.3. Диаграмма классов 13](#_Toc184267892)

[2.4. Описание главного модуля 14](#_Toc184267893)

[2.5. Описание спецификаций к модулям 19](#_Toc184267894)

[2.6. Описание модулей 26](#_Toc184267895)

[2.7. Описание тестовых наборов модулей 27](#_Toc184267896)

[Глава 3. Эксплуатационная часть 32](#_Toc184267897)

[3.1. Руководство оператора 32](#_Toc184267898)

[Заключение 36](#_Toc184267899)

[Список литературы и интернет-источников 37](#_Toc184267900)

[Приложение 1. Код главного модуля main.c 38](#_Toc184267901)

[Приложение 2. Код модуля character.h 41](#_Toc184267902)

[Приложение 3. Код модуля location.h 42](#_Toc184267903)

[Приложение 4. Код модуля item.h 43](#_Toc184267904)

[Приложение 5. Код модуля npc.h 44](#_Toc184267905)

[Приложение 6. Код модуля func.c 45](#_Toc184267906)

[Приложение 7. Код модуля func.h 49](#_Toc184267907)

# Введение

Целью данного курсового проекта является написание системы классов для игры “Путь в колледж”, которая пригодиться в дальнейшей в разработке игры. В каждой игре есть классы, без них невозможно функционирование самой игры. Ведь классы состоят из данных и переменных, которые используются в дальнейшем в разработке игр для взаимодействия с игроком или составляющее гейм-дизайна уровней.

В первой части будет рассмотрена предметная область моей темы, а также будут приведены несколько примеров. В примерах будут указаны игры, которые можно скачать легально в виде триала, демо или бесплатной версии.

Во второй части будут рассмотрены разработанные модули

В третей части будет рассмотрено руководство оператора

# Глава 1. Теоретическая область

## **Описание предметной области**

Игровая индустрия – на данный момент является одним из самых быстро развивающимся сектором развлечения в мире, а также является прибыльным бизнесом.

Игровая индустрия включает в себя несколько направлений разработки такие как:

* Мобильный игры. Мобильные игры составляют половину рынка и это всё из-за доступности смартфонов и планшетов.
* Консоли. Консоли уже давно существуют в данной индустрии и занимают хорошую часть рынка в мире. Можно выделить 2 компании, которые занимаются созданием консолей это: PlayStation и Xbox. Консоли удобны тем, что за средний бюджет пользователь получает мощное устройство, которое позволит ему получить стабильную работы игры. Также есть портативные консоли, которые позволяют получить доступ к играм в любом место земли, где есть интернет. Тут также можно выделить 2 компании, которые занимаются созданием данных консолей, это: Nintendo и Steam. Консоли являются более закрытой системой в отличие от PC, игры на консоли не поддерживают модификации и кастомизации, также производители консолей осуществляют строгий контроль над аппаратным и программным обеспечением.
* PC. Также существует давно в данной индустрии и занимает значительную часть рынка. PC удобен для пользователя тем, что PC всегда можно улучшить в техническом плане, но также обновления PC под определённые требования игр пользователю может обойтись в большую сумму. Но также игры на PC удобны тем, что данные игры могут легко поддерживать модификации и кастомизации для игры, для более разнообразного контента.
* VR и AR. VR и AR игры являются самым молодым направлением в игровой индустрии, но также и самым перспективным и занимает малую часть рынка. Они позволяют пользователю получить особенный игровой опыт, который не сравнится ни с какой платформой. Играя в данные игры, игрок может оказаться в любой точке мира и также получить ощущение настоявшей реальности, но в итоге это оказывается игрой.

Также игры разделаются на классификации и можно выделить несколько классификаций:

* По жанрам. В игривой индустрии игровых жанров очень много, но самыми популярными являются: шутеры, приключения, RPG, стратегии и симуляторы. Такое большое разнообразии жанров позволяет игроку находить новые для себя игры по жанрам.
* По целевой аудитории. Тут разделение идёт по интересам и возрастным группам. Тут можно выделать 3 группы: Детские игры, подростковые игры и взрослые игры. Детские игры разработаны для детей и часто несут образовательный характер. Подростковые игры рассчитаны под молодёжную аудиторию и содержит более серьёзные темы. Взрослые игры содержат более сложный сюжет и тяжёлые темы, а также может содержать элементы насилия.
* По механикам. Игры могут быть однопользовательскими – игры рассчитаны под одного игрока. Многопользовательскими – игры рассчитаны под игру несколько игроков одновременно. Кроссплатформенными – игры, поддерживающие игру на разных устройствах.
* По контексту использования. Игры могут использоваться для обучения, в медицинских или социальных целях.

В наше время игры являются уже не просто развлечением, а искусством. Через игры автор может показывать, как он видит свой мир, делиться своими переживаниями и мыслями. В новых играх сюжет становиться глубоким и сложным, который поднимает очень важные проблемы в наше время. Заставляет игрока задуматься над данным сюжетом, также пережить те же самые ощущение, как и его любимый герой в игре. Также игры могут помочь игроку справиться с разными сложностями в жизни, ведь в игре он может найти героя, у которого похожая ситуация в жизни и во время игры герой справляется с ней. И это может помочь игроку также справиться со сложностями в жизни.

Игровая индустрия развивается большими шагами, особенно в развитии графики, механик и т. д. Как пример можно рассмотреть игровой движок Unreal Engine 5.5. На данный момент данный движок позволяет создавать такие игры, что в моментах нельзя отличить от реальной жизни. И с каждой новой версией движка представляются всё больше функций, который могут облегчить и ускорить разработку игр.

## **Описание существующих разработок**

На игровом рынке существует большое количество игр, и каждая отличается друг от друга своими механиками, сюжетом и т. д.

Первое рассмотрение игры – это “Atomic Heart” от российской компании “Mindfish” и в неё можно поиграть, но только в бесплатной демоверсии. Данная игра является ААА однопользовательской и сюжет повествует о будущем, где остался СССР и как же будущее без роботов. Но в один момент случился сбой, и роботы стали нападать на людей. И главному герою требуется это исправить. В данной игре реализация классов и работы с ними на высшем уровне. Данный вывод можно сделать из того, что в игре большая детализация локации, предметов, а это означает, что в игре большое количество классов. Из-за того, что это ААА проект, то используется модульный метод, который позволяет им быстро добавлять или изменять классы и не переписывать большую часть кода. Также большое количество классов сильно сказывается на оптимизации работы игры, а чтобы это избежать требуется оптимизировать систему классов.

Плюсами данной игры:

1. Модульное программирование
2. Оптимизация классов

К минусам можно отнести:

1. Высокое требование к аппаратным средствам

Второе рассмотрение игры – это “Genshin Impact” от китайской компании “MiHoYo”. Игра была выпушена на телефоны/планшеты. В данной игре сюжет заключается в том, что путешественник ищет свою/его сестру/брата (это зависит от выбора персонажа). Игрок путешествует по большому миру, изучает его историю и знакомится с новыми персонажами и вместе с ними пытается найти свою/его сестру/брата. Из-за того, что игра вышла сначала под мобильные устройства, то игре требовалась очень хорошая оптимизация. Игра содержит большое количество классов, ведь в игре открытый мир, а это означает ещё большее количество предметов и т. д. Данная игра является ААА проектом, а значит в разработке использовалась модульное программирование и паттерны. И из-за большого количества классов в игре, нужно было сделать хорошую оптимизацию классов, с чем разработчики хорошо справились.

Плюсами данной игры:

1. Модульное программирование
2. Оптимизация классов
3. Минимальное требование к аппаратным средствам

К минусам можно отнести:

1. Требование к большому кол-во свободного памяти на устройстве

# Глава 2. Проектная часть

## **Выбор инструментов**

При выборе инструмента и языка программирование я отталкивался от своего личного опыта работы с инструментами и также при учёте своих целей.

Мой курсовой проект написан на С. Мой выбор упал на данный язык из-за нескольких факторов, вот такие:

1. В играх часто важна работа с оперативной памятью и быстрая работа кода, что может дать нам С.
2. Использование С позволит в данном курсовом проекте использовать мои знания на практике. Также писать программы на С мне удобнее, чем на том же Python

Курсовой проект пишется в Visual Studio Code. Данный выбор я сделал из-за того, что до этого имел опыт работы в данном IDE. Поставить поддержку С было не сложно, всё можно сделать с помощью MinGW.

Visual Studio Code – инструментальная среда разработки, которая включает в себя интеграцию среды разработки, встроенный отладчик и редактор исходного кода. Также поддерживается установка плагинов, которые могут позволить пользователю добавить другие инструменты, также возможны разные темы кастомизации.

## **Проектирование сценария**

В этом разделе приведён пример использование программы пользователем.

После того как пользователь запустит программу ему необходимо написать название файла, который в дальнейшем будет создан для сохранения информации, которую введёт пользователь. Узнать о функциях можно будет узнать в “Руководство оператора”.

Изображение выглядит как текст, диаграмма, документ, Шрифт

Автоматически созданное описание

Рисунок 1. Сценарий использования

## **Диаграмма классов**

В диаграмме классов будет представлено 4 класса Character (для персонажа), Location (для локации), Item (для предметов) и Npc (для npc).

В классах Character, Location, Item и Npc используется public по умолчанию, ведь в C нет механизма для разделения доступа к полям структуры.

Блок, находящийся после названия класса, показывает объявленные переменные, а знак “+” указывает на public, в моём случае будет только public.

После блока с переменными находится блок с операциями.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 2. Диаграмма классов

## **Описание главного модуля**

Главный модуль представляется в виде файла main.c.

К данному модулю подключаются и остальные модули, содержащие в себе функции и классы.

Листинг 1. Код главного модуля

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "character.h"

#include "location.h"

#include "item.h"

#include "npc.h"

#include "func.h"

#define MAX\_NAME\_LENGTH 100

#define MAX\_CHARACTERS 10

#define MAX\_ITEMS 10

#define MAX\_NPCS 10

int main() {

    Character characters[MAX\_CHARACTERS];

    Location locations[MAX\_CHARACTERS];

    Item items[MAX\_ITEMS];

    NPC npcs[MAX\_NPCS];

    int characterCount = 0, locationCount = 0, itemCount = 0, npcCount = 0;

    char filename[MAX\_NAME\_LENGTH];

    printf("Enter filename for saving: ");

    fgets(filename, MAX\_NAME\_LENGTH, stdin);

    filename[strcspn(filename, "\n")] = 0;

    char choice;

    do {

        if (characterCount < MAX\_CHARACTERS) {

            inputCharacter(&characters[characterCount]);

            characterCount++;

        } else {

            printf("Maximum number of characters reached.\n");

            break;

        }

        printf("Do you want to enter another character? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    do {

        if (locationCount < MAX\_CHARACTERS) {

            inputLocation(&locations[locationCount]);

            locationCount++;

        } else {

            printf("Maximum number of locations reached.\n");

            break;

        }

        printf("Do you want to enter another location? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    do {

        if (itemCount < MAX\_ITEMS) {

            inputItem(&items[itemCount]);

            itemCount++;

        } else {

            printf("Maximum number of items reached.\n");

            break;

        }

        printf("Do you want to enter another item? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    do {

        if (npcCount < MAX\_NPCS) {

            inputNPC(&npcs[npcCount]);

            npcCount++;

        } else {

            printf("Maximum number of NPCs reached.\n");

            break;

        }

        printf("Do you want to enter another NPC? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    saveToFile(filename, characters, characterCount, locations, locationCount, items, itemCount, npcs, npcCount);

    printf("Data saved to file %s.\n", filename);

    return 0;

}

Ниже будет представлена схема главного модуля программы

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок 3. Блок-схема главного модуля

## **Описание спецификаций к модулям**

Спецификация к модулю Character (Листинг 2).

Листинг 2. Код класса Character

#ifndef CHARACTER\_H\_INCLUDED

#define CHARACTER\_H\_INCLUDED

#define MAX\_NAME\_LENGTH 100

#define MAX\_HISTORY\_LENGTH 1000

typedef struct {

    char firstName[MAX\_NAME\_LENGTH];

    char lastName[MAX\_NAME\_LENGTH];

    float height;

    float weight;

    char history[MAX\_HISTORY\_LENGTH];

} Character;

#endif // CHARACTER\_H\_INCLUDED

Спецификация к модулю Location (Листинг 3).

Листинг 3. Код класса Location

#ifndef LOCATION\_H\_INCLUDED

#define LOCATION\_H\_INCLUDED

#define MAX\_LOCATION\_TYPE\_LENGTH 50

#define MAX\_LOCATION\_NAME\_LENGTH 50

#define MAX\_DESCRIPTION\_LENGTH 500

typedef struct {

    char type[MAX\_LOCATION\_TYPE\_LENGTH];

    char name\_loc[MAX\_LOCATION\_NAME\_LENGTH];

    float size;

    char description[MAX\_DESCRIPTION\_LENGTH];

} Location;

#endif // LOCATION\_H\_INCLUDED

Спецификация к модулю Item (Листинг 4).

Листинг 4. Код класса Item

#ifndef ITEM\_H\_INCLUDED

#define ITEM\_H\_INCLUDED

#define MAX\_ITEM\_TYPE\_LENGTH 50

#define MAX\_ITEM\_NAME\_LENGTH 100

typedef struct {

    char type[MAX\_ITEM\_TYPE\_LENGTH];

    char name\_item[MAX\_ITEM\_NAME\_LENGTH];

    int quantity;

} Item;

#endif //ITEM\_H\_INCLUDED

Спецификация к модулю Npc (Листинг 5)

Листинг 5. Код для класса Npc

#ifndef NPC\_H\_INCLUDED

#define NPC\_H\_INCLUDED

#include "item.h"

#include "location.h"

#define MAX\_ITEMS 10

#define MAX\_NPC\_NAME\_LENGTH 100

typedef struct {

    char name\_npc[MAX\_NPC\_NAME\_LENGTH];

    Location location;

    Item items[MAX\_ITEMS];

    int itemCount;

} NPC;

#endif //NPC\_H\_INCLUDED

Спецификация к модулю func (Листинг 6)

Листинг 6. Код модуля func

#include <stdio.h>

#include "item.h"

#include "character.h"

#include "location.h"

#include "npc.h"

#include "func.h"

#define MAX\_HISTORY\_LENGTH 1000

#define MAX\_DESCRIPTION\_LENGTH 500

#define MAX\_ITEM\_TYPE\_LENGTH 50

#define MAX\_LOCATION\_NAME\_LENGTH 50

#define MAX\_ITEM\_NAME\_LENGTH 100

#define MAX\_LOCATION\_TYPE\_LENGTH 50

#define MAX\_NPC\_NAME\_LENGTH 100

#define MAX\_CHARACTERS 100

#define MAX\_LOCATIONS 100

#ifndef MAX\_ITEMS

#define MAX\_ITEMS 10

#endif

#define MAX\_NAME\_LENGTH 100

#ifndef MAX\_DESCRIPTION\_LENGTH

#define MAX\_DESCRIPTION\_LENGTH 1000

#endif

Location locations[MAX\_LOCATIONS];

int locationCount = 0;

Item items[MAX\_ITEMS];

int itemCount = 0;

void inputCharacter(Character \*character\_p) { //Данная функция отвечает за персонажа(ей)

    printf("Enter character's first name: ");

    scanf("%s", character\_p->firstName);

    printf("Enter character's last name: ");

    scanf("%s", character\_p->lastName);

    printf("Enter height (in meters): ");

    scanf("%f", &character\_p->height);

    printf("Enter weight (in kilograms): ");

    scanf("%f", &character\_p->weight);

    printf("Enter character's history: ");

    getchar();

    fgets(character\_p->history, MAX\_HISTORY\_LENGTH, stdin);

}

void inputLocation(Location \*location\_p) {

    printf("Enter location type: ");

    scanf("%s", location\_p->type);

    printf("Enter location name: ");

    scanf("%s", location\_p->name\_loc);

    printf("Enter location size: ");

    scanf("%f", &location\_p->size);

    printf("Enter location description: ");

    getchar();

    fgets(location\_p->description, MAX\_DESCRIPTION\_LENGTH, stdin);

    locations[locationCount++] = \*location\_p;

}

void inputItem(Item \*item\_p) {

    printf("Enter item type: ");

    scanf("%s", item\_p->type);

    printf("Enter item name: ");

    scanf("%s", item\_p->name\_item);

    printf("Enter item quantity: ");

    scanf("%d", &item\_p->quantity);

    items[itemCount++] = \*item\_p;

}

void inputNPC(NPC \*npc\_p) {

    printf("Enter NPC name: ");

    scanf("%s", npc\_p->name\_npc);

    if (locationCount > 0) {

        printf("Select a location (0 to %d):\n", locationCount - 1);

        for (int i = 0; i < locationCount; i++) {

            printf("%d: %s\n", i, locations[i].name\_loc);

        }

        int locIndex;

        scanf("%d", &locIndex);

        npc\_p->location = locations[locIndex];

    } else {

        inputLocation(&npc\_p->location);

    }

    printf("Enter number of items with the NPC: ");

    scanf("%d", &npc\_p->itemCount);

    for (int i = 0; i < npc\_p->itemCount; i++) {

        if (itemCount > 0) {

            printf("Select an item (0 to %d):\n", itemCount - 1);

            for (int j = 0; j < itemCount; j++) {

                printf("%d: %s\n", j, items[j].name\_item);

            }

            int itemIndex;

            scanf("%d", &itemIndex);

            npc\_p->items[i] = items[itemIndex];

        } else {

            inputItem(&npc\_p->items[i]);

        }

    }

}

void saveToFile(const char \*filename, Character \*characters, int characterCount, Location \*locations, int locationCount, Item \*items, int itemCount, NPC \*npcs, int npcCount) {

    FILE \*file = fopen(filename, "a+");

    if (file == NULL) {

        printf("Error opening file!\n");

        return;

    }

    for (int i = 0; i < characterCount; i++) {

        fprintf(file, "Character: %s %s, Height: %.2f, Weight: %.1f, History: %s\n",

                characters[i].firstName, characters[i].lastName,

                characters[i].height, characters[i].weight, characters[i].history);

    }

    for (int i = 0; i < locationCount; i++) {

        fprintf(file, "Location: %s, Location name: %s, Size: %.2f, Description: %s\n",

                locations[i].type, locations[i].name\_loc, locations[i].size, locations[i].description);

    }

    for (int i = 0; i < itemCount; i++) {

        fprintf(file, "Item: %s, Name: %s, Quantity: %d\n",

                items[i].type, items[i].name\_item, items[i].quantity);

    }

    for (int i = 0; i < npcCount; i++) {

        fprintf(file, "NPC: %s, Location: %s, ItemCount: %d\n",

                npcs[i].name\_npc, npcs[i].location.name\_loc, npcs[i].itemCount);

        for (int j = 0; j < npcs[i].itemCount; j++) {

            fprintf(file, "  Item %d: %s\n", j + 1, npcs[i].items[j].name\_item);

        }

    }

    fclose(file);

}

## **Описание модулей**

В данной главе будут описаны используемые модули.

1. Главный модуль

Главный модуль представляется в виде файла main.c. К данному модулю

подключаются и остальные модули, содержащие в себе функции и классы.

1. Модуль функций

В модуле функций находятся функции, которые используются в проекте:

* Printf() – вывод информации на экран
* Scanf() – используется для считывания вводимой информации.
* Fgets() – позволяет считывать строку текста из указанного потока и сохраняет её в массив символов.
* Getchar – используется для считывания следующего символа из стандартного потока и возвращает его как unsigned char, приведённый к типу int. В моём случае используется для очистки буфера перед использованием Fgets()
* Fopen() – используется для открытия файла и возвращения указателя на него
* Fprintf() – используется для записи данный в указанный файл.
* Fclose – закрывает файл

1. Модуль содержащий класс Character(character.h)

Класс Character предназначен для хранения данных о персонаже в программе (Имя, фамилия, вес, рост и история персонажа)

1. Модуль содержащий класс Location(location.h)

Класс Location предназначен для хранения данных о локации, где будут происходить действия (тип локации, размер и описание локации)

1. Модуль содержащий класс Item(item.h)

Класс Item предназначен для хранения данных о предмете (тип предмета, название и количество)

1. Модуль содержащий класс Npc(npc.h)

Класс Npc предназначен для хранения данных о Npc (Имя, локация, где будет находиться Npc, какой предмет у него будет и количество данного предмета)

1. Модуль, содержащий функции(func.c)

Данный модуль с функциями предназначен для ввода и сохранения введённой информации(character, location, item, npc)

## **Описание тестовых наборов модулей**

В данном разделе будут показаны результаты тестирования “Чёрного ящика”.

Тест 1. Ввод информации только с 1 персонажем, локацией, предметом и npc

Действие: запустить программу, ввести название файла, заполнить информацию о персонаже и когда появиться вопрос “Do you want to enter another character? (y/n):” написать “n” и нажать Enter, после чего повторить такие же действия с остальными не заполненными полями.

Ожидаемый результат: программа создаст файл с нужным названием и сохранит его в папку с проектом. В самом файле будет написана информация, которую мы вводили.

Результат теста:

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание

Рисунок 4. Ввод всех данных

Изображение выглядит как текст, чек, алгебра, Шрифт

Автоматически созданное описание

Рисунок 5. Файл с сохранёнными данными

Тест 2. Ввод информации уже с 2 персонажами, локациями, предметами и npc

Действие: запустить программу, ввести название файла, заполнить информацию о персонаже и когда появиться вопрос “Do you want to enter another character? (y/n):” написать “n” и нажать Enter, после чего повторить такие же действия с остальными не заполненными полями.

Ожидаемый результат: программа создаст файл с нужным названием и сохранит его в папку с проектом. В самом файле будет написана информация, которую мы вводили.

Результат теста:

Изображение выглядит как текст, снимок экрана, меню, черно-белый

Автоматически созданное описание

Рисунок 6. Ввод всех данных (по 2 раза)

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 7. Файл с сохранёнными данными

# Глава 3. Эксплуатационная часть

## **Руководство оператора**

1. **Назначение программы**

**Функциональное назначение программы**

В проекте “Proga” основной функцией является автоматизация сохранения данных, которые пригодятся в создание игры.

**Эксплуатационное назначение программы**

Программное обеспечение “Proga” может использоваться любой компанией, а также частным лицом.

**Состав функций**

Функции оператора:

1. Функция сохранения персонажа – отвечает за информацию о персонаже, которая, будет сохранена в файле
2. Функция сохранения локации – отвечает за информацию о локации, которая, будет сохранена в файле
3. Функция сохранения предмета – отвечает за информацию о предмете, которая, будет сохранена в файле
4. Функция сохранения npc – отвечает за информацию о npc, которая, будет сохранена в файле
5. Функция загрузки сохранённых данных – отвечает за загрузку заранее сохранённой информации
6. Функция выбора файла для сохранения – отвечает за выбор файла, куда будет сохранена информация
7. **Условия выполнения программы**

**Минимальный состав аппаратных средств**

Минимальный состав используемых аппаратных (технических) средств:

* Процессор с тактовой частотой не ниже 1,8 ГГц.
* ОЗУ – 20 МБ;
* Windows 7, 8.1, 9, 10, 11
* Видеоадаптер с минимальным разрешением 144p.
* Место на жёстком диске: 100+ КБ

**Минимальный состав программных средств**

Дополнительные программы для использование данной программы не требуются

**Требования к персоналу(оператору)**

Базовые умения работы с компьютером, и базовые знания английского.

1. Выполнение программы

**Загрузка и запуск программы**

До запуска программы программа состоит из 1 файла “Proga.exe”. Но после использования программы количество файлов может увеличиться на неопределённое количество, всё зависит от того сколько файлов будут созданы для хранения информации.

При запуске программы должно появиться данное окно:

Изображение выглядит как текст, снимок экрана, компьютер, мультимедиа

Автоматически созданное описание

Рисунок 8. Главное окно программы

Для дальнейшего выполнения программы необходимо написать название файла и после этого появиться поле для создания персонажа

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9. Начало ввода информации о персонаже

После того, как будет, верно, введена вся информация о персонаже, появиться выбор о создание ещё одного персонажа.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10. Конец ввода информации о персонаже

Если вести “y”, то у вас начнётся ввод данных о новом персонаже, а если вести “n”, то перейдёт на ввод информации о локации

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 11. Начало ввода информации о локации

После того, как будет верно, введена информация о локации, то появится выбор и создание ещё одной локации. От выбора “y” и ”n” будет зависит переход к следующему вводу информация. Такие же действия необходимо проделать с заполнением информации о предмете и Npc. После того как вы ведёте всё верно у вас появится последний выбор о создание ещё одного npc. При вводе “y” можно будет создать ещё одного Npc, при вводе “n” программа завершится и будет создан файл с введённой раньше информацией

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 12. Последний выбор перед завершением программы

Вот так будет выглядит сохранённый файл



Рисунок 13. Сохранённый файл

# Заключение

В результате выполнения курсового проекта была написана программа “Proga”, которая позволяет пользователю, сохранят информацию для игры.

В ходе написания курсовой работы были проанализированы существующие разработки, предметная область и получены практические навыки с работой в Visual Studio Code.

Так же я планирую продолжить работу над данным проектом с целью расширения возможностей, а также использование при создании игры на дипломную работу. Планы на данный проект будут представлены ниже:

1. Переписать код на с++
2. Исправить баги
3. Усовершенствовать код
4. Использовать данные классы уже в самой игре

# Список литературы и интернет-источников

1. Сайт Visual Studio Code

<https://code.visualstudio.com>

1. Сайт компании “Mindfish”

<https://mundfish.com>

1. Теоретические материалы:

<https://cmp.phys.msu.ru/sites/default/files/MetC_complete.pdf>

Книга “Сделай видеоигру один и не свихнись - Грис Слава”

https://hsbi.hse.ru/articles/igrovaya-industriya-geymdev/

# Приложение 1. Код главного модуля main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "character.h"

#include "location.h"

#include "item.h"

#include "npc.h"

#include "func.h"

#define MAX\_NAME\_LENGTH 100

#define MAX\_CHARACTERS 10

#define MAX\_ITEMS 10

#define MAX\_NPCS 10

int main() {

    Character characters[MAX\_CHARACTERS];

    Location locations[MAX\_CHARACTERS];

    Item items[MAX\_ITEMS];

    NPC npcs[MAX\_NPCS];

    int characterCount = 0, locationCount = 0, itemCount = 0, npcCount = 0;

    char filename[MAX\_NAME\_LENGTH];

    printf("Enter filename for saving: ");

    fgets(filename, MAX\_NAME\_LENGTH, stdin);

    filename[strcspn(filename, "\n")] = 0;

    char choice;

    do {

        if (characterCount < MAX\_CHARACTERS) {

            inputCharacter(&characters[characterCount]);

            characterCount++;

        } else {

            printf("Maximum number of characters reached.\n");

            break;

        }

        printf("Do you want to enter another character? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    do {

        if (locationCount < MAX\_CHARACTERS) {

            inputLocation(&locations[locationCount]);

            locationCount++;

        } else {

            printf("Maximum number of locations reached.\n");

            break;

        }

        printf("Do you want to enter another location? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    do {

        if (itemCount < MAX\_ITEMS) {

            inputItem(&items[itemCount]);

            itemCount++;

        } else {

            printf("Maximum number of items reached.\n");

            break;

        }

        printf("Do you want to enter another item? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    do {

        if (npcCount < MAX\_NPCS) {

            inputNPC(&npcs[npcCount]);

            npcCount++;

        } else {

            printf("Maximum number of NPCs reached.\n");

            break;

        }

        printf("Do you want to enter another NPC? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y');

    saveToFile(filename, characters, characterCount, locations, locationCount, items, itemCount, npcs, npcCount);

    printf("Data saved to file %s.\n", filename);

    return 0;

}

# Приложение 2. Код модуля character.h

#ifndef CHARACTER\_H\_INCLUDED

#define CHARACTER\_H\_INCLUDED

#define MAX\_NAME\_LENGTH 100

#define MAX\_HISTORY\_LENGTH 1000

typedef struct {

    char firstName[MAX\_NAME\_LENGTH];

    char lastName[MAX\_NAME\_LENGTH];

    float height;

    float weight;

    char history[MAX\_HISTORY\_LENGTH];

} Character;

#endif // CHARACTER\_H\_INCLUDED

# Приложение 3. Код модуля location.h

#ifndef LOCATION\_H\_INCLUDED

#define LOCATION\_H\_INCLUDED

#define MAX\_LOCATION\_TYPE\_LENGTH 50

#define MAX\_LOCATION\_NAME\_LENGTH 50

#define MAX\_DESCRIPTION\_LENGTH 500

typedef struct {

    char type[MAX\_LOCATION\_TYPE\_LENGTH];

    char name\_loc[MAX\_LOCATION\_NAME\_LENGTH];

    float size;

    char description[MAX\_DESCRIPTION\_LENGTH];

} Location;

#endif // LOCATION\_H\_INCLUDED

# Приложение 4. Код модуля item.h

#ifndef ITEM\_H\_INCLUDED

#define ITEM\_H\_INCLUDED

#define MAX\_ITEM\_TYPE\_LENGTH 50

#define MAX\_ITEM\_NAME\_LENGTH 100

typedef struct {

    char type[MAX\_ITEM\_TYPE\_LENGTH];

    char name\_item[MAX\_ITEM\_NAME\_LENGTH];

    int quantity;

} Item;

#endif //ITEM\_H\_INCLUDED

# Приложение 5. Код модуля npc.h

#ifndef NPC\_H\_INCLUDED

#define NPC\_H\_INCLUDED

#include "item.h"

#include "location.h"

#define MAX\_ITEMS 10

#define MAX\_NPC\_NAME\_LENGTH 100

typedef struct {

    char name\_npc[MAX\_NPC\_NAME\_LENGTH];

    Location location;

    Item items[MAX\_ITEMS];

    int itemCount;

} NPC;

#endif //NPC\_H\_INCLUDED

# Приложение 6. Код модуля func.c

#include <stdio.h>

#include "item.h"

#include "character.h"

#include "location.h"

#include "npc.h"

#include "func.h"

#define MAX\_HISTORY\_LENGTH 1000

#define MAX\_DESCRIPTION\_LENGTH 500

#define MAX\_ITEM\_TYPE\_LENGTH 50

#define MAX\_LOCATION\_NAME\_LENGTH 50

#define MAX\_ITEM\_NAME\_LENGTH 100

#define MAX\_LOCATION\_TYPE\_LENGTH 50

#define MAX\_NPC\_NAME\_LENGTH 100

#define MAX\_CHARACTERS 100

#define MAX\_LOCATIONS 100

#ifndef MAX\_ITEMS

#define MAX\_ITEMS 10

#endif

#define MAX\_NAME\_LENGTH 100

#ifndef MAX\_DESCRIPTION\_LENGTH

#define MAX\_DESCRIPTION\_LENGTH 1000

#endif

Location locations[MAX\_LOCATIONS];

int locationCount = 0;

Item items[MAX\_ITEMS];

int itemCount = 0;

void inputCharacter(Character \*character\_p) { //Данная функция отвечает за персонажа(ей)

    printf("Enter character's first name: ");

    scanf("%s", character\_p->firstName);

    printf("Enter character's last name: ");

    scanf("%s", character\_p->lastName);

    printf("Enter height (in meters): ");

    scanf("%f", &character\_p->height);

    printf("Enter weight (in kilograms): ");

    scanf("%f", &character\_p->weight);

    printf("Enter character's history: ");

    getchar();

    fgets(character\_p->history, MAX\_HISTORY\_LENGTH, stdin);

}

void inputLocation(Location \*location\_p) {

    printf("Enter location type: ");

    scanf("%s", location\_p->type);

    printf("Enter location name: ");

    scanf("%s", location\_p->name\_loc);

    printf("Enter location size: ");

    scanf("%f", &location\_p->size);

    printf("Enter location description: ");

    getchar();

    fgets(location\_p->description, MAX\_DESCRIPTION\_LENGTH, stdin);

    locations[locationCount++] = \*location\_p;

}

void inputItem(Item \*item\_p) {

    printf("Enter item type: ");

    scanf("%s", item\_p->type);

    printf("Enter item name: ");

    scanf("%s", item\_p->name\_item);

    printf("Enter item quantity: ");

    scanf("%d", &item\_p->quantity);

    items[itemCount++] = \*item\_p;

}

void inputNPC(NPC \*npc\_p) {

    printf("Enter NPC name: ");

    scanf("%s", npc\_p->name\_npc);

    if (locationCount > 0) {

        printf("Select a location (0 to %d):\n", locationCount - 1);

        for (int i = 0; i < locationCount; i++) {

            printf("%d: %s\n", i, locations[i].name\_loc);

        }

        int locIndex;

        scanf("%d", &locIndex);

        npc\_p->location = locations[locIndex];

    } else {

        inputLocation(&npc\_p->location);

    }

    printf("Enter number of items with the NPC: ");

    scanf("%d", &npc\_p->itemCount);

    for (int i = 0; i < npc\_p->itemCount; i++) {

        if (itemCount > 0) {

            printf("Select an item (0 to %d):\n", itemCount - 1);

            for (int j = 0; j < itemCount; j++) {

                printf("%d: %s\n", j, items[j].name\_item);

            }

            int itemIndex;

            scanf("%d", &itemIndex);

            npc\_p->items[i] = items[itemIndex];

        } else {

            inputItem(&npc\_p->items[i]);

        }

    }

}

void saveToFile(const char \*filename, Character \*characters, int characterCount, Location \*locations, int locationCount, Item \*items, int itemCount, NPC \*npcs, int npcCount) {

    FILE \*file = fopen(filename, "a+");

    if (file == NULL) {

        printf("Error opening file!\n");

        return;

    }

    for (int i = 0; i < characterCount; i++) {

        fprintf(file, "Character: %s %s, Height: %.2f, Weight: %.1f, History: %s\n",

                characters[i].firstName, characters[i].lastName,

                characters[i].height, characters[i].weight, characters[i].history);

    }

    for (int i = 0; i < locationCount; i++) {

        fprintf(file, "Location: %s, Location name: %s, Size: %.2f, Description: %s\n",

                locations[i].type, locations[i].name\_loc, locations[i].size, locations[i].description);

    }

    for (int i = 0; i < itemCount; i++) {

        fprintf(file, "Item: %s, Name: %s, Quantity: %d\n",

                items[i].type, items[i].name\_item, items[i].quantity);

    }

    for (int i = 0; i < npcCount; i++) {

        fprintf(file, "NPC: %s, Location: %s, ItemCount: %d\n",

                npcs[i].name\_npc, npcs[i].location.name\_loc, npcs[i].itemCount);

        for (int j = 0; j < npcs[i].itemCount; j++) {

            fprintf(file, "  Item %d: %s\n", j + 1, npcs[i].items[j].name\_item);

        }

    }

    fclose(file);

}

# Приложение 7. Код модуля func.h

#ifndef FUNC\_H\_INCLUDED

#define FUNC\_H\_INCLUDED

#include "item.h"

#include "character.h"

#include "location.h"

#include "npc.h"

void inputCharacter(Character \*character\_p);

void inputLocation(Location \*location\_p);

void inputItem(Item \*item\_p);

void inputNPC(NPC \*npc\_p);

void saveToFile(const char \*filename, Character \*characters, int characterCount, Location \*locations, int locationCount, Item \*items, int itemCount, NPC \*npcs, int npcCount);

#endif //FUNC\_H\_INCLUDED