|  |  |
| --- | --- |
| *voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-02 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | О |  | Естественнонаучный |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | Информационные технологии и программирование | | |

|  |
| --- |
| Практическая работа №2 |
| Наследование |
| Вариант №25 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | |  | И924Б |
| Фокин В. К. | | | | | |
| Фамилия И.О. | | | | | |
| **Преподаватель:** | | | | | |
| Гладевич А. А. | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Оценка |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2023г. |

САНКТ-ПЕТЕРБУРГ

2023 г.

**Постановка задачи**

В рамках работы необходимо разработать иерархию классов на указанную в варианте тему. В иерархии должен быть корневой класс (базовый), который должен быть абстрактным и содержать общие для остальных поля и методы. У каждого из классов должен быть хотя бы один собственный метод. Также должны быть продемонстрированы виртуальные и переопределённые методы. Переопределенные методы должны также вызывать методы базового класса если это оправдано.

В каждом классе должны быть:

− метод, выводящий имя данного класса;

− метод, выводящий в виде строки всю информацию об объекте – тип, имена и значения полей;

− конструктор без параметров;

− конструктор с параметрами;

− деструктор.

Рекомендуемые поля и методы указаны в варианте, однако возможно придумать и свои. Также необходимо написать программу с меню, позволяющую протестировать разработанную иерархию. В ходе программы создаётся массив минимум из 4-х элементов, в котором хранятся указатели на объекты базового класса.

Обязательные пункты меню:

− добавление нового объекта выбранного пользователем класса в массив;

− выполнение указанного метода базового класса для всех объектов массива (у каждого должен выполниться перегруженный вариант);

− вывод всех объектов в массиве;

− выполнение уникального для указанного объекта метода.

**Вариант 25.**

Герои тёмных подземелий.

**Варианты полей:** имя, здоровье, уровень стресса, яркость факела.

**Варианты методов:** смело идти вперёд, храбро бежать назад, сражаться, привал

**Возможные классы:** герой (базовый), Антиквар, Арбалетчик, Воитель, Крестоносец.

**Текст программы:**

#include <iostream>

#include <string>

#define c 50

using namespace std;

class character {

protected:

string name;

int health;

int stress;

int brightness;

public:

character();

character(string, int, int, int);

~character();

void set\_name(string);

void set\_health(int);

void set\_stress(int);

void set\_brightness(int);

string get\_name();

int get\_health();

int get\_stress();

int get\_brightness();

virtual void getClass() = 0 {

cout << "Character" << endl;

}

void getInfo() {

cout << "Name: " << get\_name() << ", Health: " << get\_health() << ", Stress level: " << get\_stress() << ", Brightness level: " << get\_brightness() << endl;

}

void forward() {

cout << this->get\_name() << " very boldly went forward" << endl;

}

void back() {

cout << this->get\_name() << " brave enough to run back" << endl;

}

void fight() {

cout << this->get\_name() << " trying to fight" << endl;

}

friend ostream& operator << (ostream&, const character&);

};

class antiquarian : public character {

public:

antiquarian();

antiquarian(string, int, int, int);

~antiquarian();

void getClass() override {

cout << "Antiquarian" << endl;

}

void find() {

cout << "The antiquarian " << get\_name() << " found a rare item!" << endl;

}

};

class arbalester : public character {

public:

arbalester();

arbalester(string, int, int, int);

~arbalester();

void getClass() override {

cout << "Arbalester" << endl;

}

void shot() {

cout << "The arbalester " << get\_name() << " cleverly missed and upset :(" << endl;

}

};

class warrior : public character {

public:

warrior();

warrior(string, int, int, int);

~warrior();

void getClass() override {

cout << "Warrior" << endl;

}

void kill() {

cout << "The warrior " << get\_name() << " beheaded three, including himself too..." << endl;

}

};

class crusader : public character {

public:

crusader();

crusader(string, int, int, int);

~crusader();

void getClass() override {

cout << "Crusader" << endl;

}

void pray() {

cout << "The crusader " << get\_name() << " banishes freak who believes in spaghetti monster" << endl;

}

};

int main() {

character\* chrc[c];

string n;

int cnt, h = 0, s = 0, b = 0;

char menu, men, me, nemu;

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

{

chrc[i] = nullptr;

}

chrc[0] = new antiquarian("Ezio Auditore", 70, 20, 30);

chrc[1] = new antiquarian("Hendrik", 60, 15, 30);

chrc[2] = new arbalester("Legolas", 70, 10, 25);

chrc[3] = new arbalester("Connor Kenway", 75, 10, 25);

chrc[4] = new warrior("Kratos", 150, 0, 50);

chrc[5] = new warrior("Geralt z Rivii", 150, 0, 50);

chrc[6] = new crusader("Yennefer", 100, 10, 50);

chrc[7] = new crusader("Merlin", 100, 10, 50);

for (int i = 0; i < c && chrc[i] != NULL; i++) {

cnt = i + 1;

}

do {

cout << "----------------------" << endl;

cout << "1. Add a new character" << endl;

cout << "2. Show all characters" << endl;

cout << "3. Take action" << endl;

cout << "4. Ultimate" << endl;

cout << "0. Exit" << endl;

cout << "----------------------" << endl;

cin >> menu;

switch (menu) {

case '1':

system("cls");

do {

cout << "-----------------------------------" << endl;

cout << "What class to add the character to?" << endl;

cout << "1. Antiquarian" << endl;

cout << "2. Arbalester" << endl;

cout << "3. Warrior" << endl;

cout << "4. Crusader" << endl;

cout << "0. Back" << endl;

cout << "-----------------------------------" << endl;

cin >> men;

switch (men) {

case '1':

cout << "-----------------------------------" << endl;

cout << "Enter name: ";

cin >> n;

do {

cout << "\nEnter number of health points: ";

cin >> h;

if (h > 200 || h < 0) {

cout << "Enter a value from 0 to 200" << endl;

}

} while (h > 200 || h < 0);

do {

cout << "\nEnter stress level: ";

cin >> s;

if (s > 50 || s < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (s > 50 || s < 0);

do {

cout << "\nEnter torch brightness: ";

cin >> b;

if (b > 50 || b < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (b > 50 || b < 0);

cout << "-----------------------------------" << endl;

chrc[cnt] = new antiquarian(n, h, s, b);

cout << "Character was added" << endl;

cout << "-----------------------------------" << endl;

cnt += 1;

break;

case '2':

cout << "-----------------------------------" << endl;

cout << "Enter name: ";

cin >> n;

do {

cout << "\nEnter number of health points: ";

cin >> h;

if (h > 200 || h < 0) {

cout << "Enter a value from 0 to 200" << endl;

}

} while (h > 200 || h < 0);

do {

cout << "\nEnter stress level: ";

cin >> s;

if (s > 50 || s < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (s > 50 || s < 0);

do {

cout << "\nEnter torch brightness: ";

cin >> b;

if (b > 50 || b < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (b > 50 || b < 0);

cout << "-----------------------------------" << endl;

chrc[cnt] = new arbalester(n, h, s, b);

cout << "Character was added" << endl;

cout << "-----------------------------------" << endl;

cnt += 1;

break;

case '3':

cout << "-----------------------------------" << endl;

cout << "Enter name: ";

cin >> n;

do {

cout << "\nEnter number of health points: ";

cin >> h;

if (h > 200 || h < 0) {

cout << "Enter a value from 0 to 200" << endl;

}

} while (h > 200 || h < 0);

do {

cout << "\nEnter stress level: ";

cin >> s;

if (s > 50 || s < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (s > 50 || s < 0);

do {

cout << "\nEnter torch brightness: ";

cin >> b;

if (b > 50 || b < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (b > 50 || b < 0);

cout << "-----------------------------------" << endl;

chrc[cnt] = new warrior(n, h, s, b);

cout << "Character was added" << endl;

cout << "-----------------------------------" << endl;

cnt += 1;

break;

case '4':

cout << "-----------------------------------" << endl;

cout << "Enter name: ";

cin >> n;

do {

cout << "\nEnter number of health points: ";

cin >> h;

if (h > 200 || h < 0) {

cout << "Enter a value from 0 to 200" << endl;

}

} while (h > 200 || h < 0);

do {

cout << "\nEnter stress level: ";

cin >> s;

if (s > 50 || s < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (s > 50 || s < 0);

do {

cout << "\nEnter torch brightness: ";

cin >> b;

if (b > 50 || b < 0) {

cout << "Enter a value from 0 to 50" << endl;

}

} while (b > 50 || b < 0);

cout << "-----------------------------------" << endl;

chrc[cnt] = new crusader(n, h, s, b);

cout << "Character was added" << endl;

cout << "-----------------------------------" << endl;

cnt += 1;

break;

case '0':

system("cls");

break;

default:

cout << "-----------------------------------" << endl;

cout << "Invalid choice. Try again." << endl;

cout << "-----------------------------------" << endl;

}

} while (men != '0');

break;

case '2':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

cout << "----------------------------------------------------------------------" << endl;

chrc[i]->getClass();

chrc[i]->getInfo();

cout << endl;

cout << "----------------------------------------------------------------------" << endl;

}

break;

case '3':

system("cls");

do {

cout << "-----------------------------------" << endl;

cout << "What action to take?" << endl;

cout << "1. Fight" << endl;

cout << "2. Go foward" << endl;

cout << "3. Run back" << endl;

cout << "0. Exit" << endl;

cout << "-----------------------------------" << endl;

cin >> me;

switch (me) {

case'1':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

chrc[i]->fight();

}

break;

case'2':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

chrc[i]->forward();

}

break;

case '3':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

chrc[i]->back();

}

break;

case '0':

system("cls");

break;

default:

cout << "-----------------------------------" << endl;

cout << "Invalid choice. Try again." << endl;

cout << "-----------------------------------" << endl;

}

} while (me != '0');

break;

case '4':

system("cls");

do {

cout << "-----------------------------------" << endl;

cout << "Enter class" << endl;

cout << "1. Antiquarian" << endl;

cout << "2. Arbalester" << endl;

cout << "3. Warrior" << endl;

cout << "4. Crusader" << endl;

cout << "0. Back" << endl;

cout << "-----------------------------------" << endl;

cin >> nemu;

switch (nemu) {

case '1':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

antiquarian\* antiq = dynamic\_cast<antiquarian\*>(chrc[i]);

if (antiq) {

cout << "--------------------------------------------" << endl;

cout << antiq->get\_name() << endl;

antiq->find();

cout << "--------------------------------------------" << endl;

}

}

break;

case '2':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

arbalester\* arbal = dynamic\_cast<arbalester\*>(chrc[i]);

if (arbal) {

cout << "--------------------------------------------" << endl;

cout << arbal->get\_name() << endl;

arbal->shot();

cout << "--------------------------------------------" << endl;

}

}

break;

case '3':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

warrior\* warr = dynamic\_cast<warrior\*>(chrc[i]);

if (warr) {

cout << "--------------------------------------------" << endl;

cout << warr->get\_name() << endl;

warr->kill();

cout << "--------------------------------------------" << endl;

}

}

break;

case '4':

for (int i = 0; i < c && chrc[i] != NULL; i++) {

crusader\* crus = dynamic\_cast<crusader\*>(chrc[i]);

if (crus) {

cout << "--------------------------------------------" << endl;

cout << crus->get\_name() << endl;

crus->pray();

cout << "--------------------------------------------" << endl;

}

}

break;

case '0':

system("cls");

break;

default:

cout << "-----------------------------------" << endl;

cout << "Invalid choice. Try again." << endl;

cout << "-----------------------------------" << endl;

}

} while (nemu != '0');

break;

case '0':

for (int i = 0; i < c && chrc[i] != NULL; i++)

{

delete chrc[i];

}

return 0;

default:

cout << "-----------------------------------" << endl;

cout << "Invalid choice. Try again." << endl;

cout << "-----------------------------------" << endl;

}

} while (menu != '0');

}

//сеттеры

void character::set\_name(string n) {

name = n;

}

void character::set\_health(int h) {

health = h;

}

void character::set\_stress(int s) {

stress = s;

}

void character::set\_brightness(int b) {

brightness = b;

}

//геттеры

string character::get\_name() {

return name;

}

int character::get\_health() {

return health;

}

int character::get\_stress() {

return stress;

}

int character::get\_brightness() {

return brightness;

}

//конструкторы

//базовый

character::character() {

this->name = "No name";

this->health = 100;

this->stress = 0;

this->brightness = 50;

}

character::character(string n, int h, int s, int b) {

this->name = n;

this->health = h;

this->stress = s;

this->brightness = b;

}

character::~character() {

cout << "Character no longer exists" << endl;

}

//антиквар

antiquarian::antiquarian() {

this->name = "No name";

this->health = 100;

this->stress = 0;

this->brightness = 50;

}

antiquarian::antiquarian(string n, int h, int s, int b) {

this->name = n;

this->health = h;

this->stress = s;

this->brightness = b;

}

antiquarian::~antiquarian() {

cout << "Antiquarian no longer exists" << endl;

}

//арбалетчик

arbalester::arbalester() {

this->name = "No name";

this->health = 100;

this->stress = 0;

this->brightness = 50;

}

arbalester::arbalester(string n, int h, int s, int b) {

this->name = n;

this->health = h;

this->stress = s;

this->brightness = b;

}

arbalester::~arbalester() {

cout << "Arbalester no longer exists" << endl;

}

//воитель

warrior::warrior() {

this->name = "No name";

this->health = 100;

this->stress = 0;

this->brightness = 50;

}

warrior::warrior(string n, int h, int s, int b) {

this->name = n;

this->health = h;

this->stress = s;

this->brightness = b;

}

warrior::~warrior() {

cout << "Warrior no longer exists" << endl;

}

//крестоносец

crusader::crusader() {

this->name = "No name";

this->health = 100;

this->stress = 0;

this->brightness = 50;

}

crusader::crusader(string n, int h, int s, int b) {

this->name = n;

this->health = h;

this->stress = s;

this->brightness = b;

}

crusader::~crusader() {

cout << "Crusader no longer exists" << endl;

};

ostream& operator << (ostream& out, const character& o) {

cout << "Name: " << o.name << "\nHealth: " << o.health << "\nStress level: " << o.stress << "\nBrightness level: " << o.brightness << endl;

return out;

}

**Результат работы программы:**

При включении программы запускается меню, что предоставлено на рисунке 1.

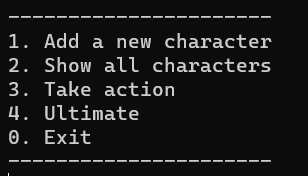


Рисунок 1 − Запуск меню

При нажатии на «1» можно добавить нового персонажа. Сначала появляется меню выбора класса, что представлено на рисунке 2.

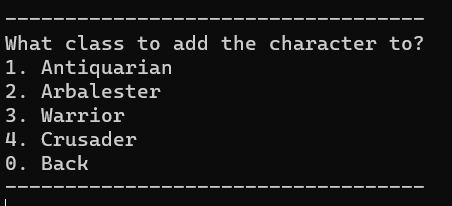


Рисунок 2 − Меню выбора класса для добавления персонажа

При нажатии на «1», «2», «3» и «4» создается новый персонаж в выбранный класс. С клавиатуры вводится имя, здоровье, уровень стресса и уровень света факела, что представлено на рисунке 3.

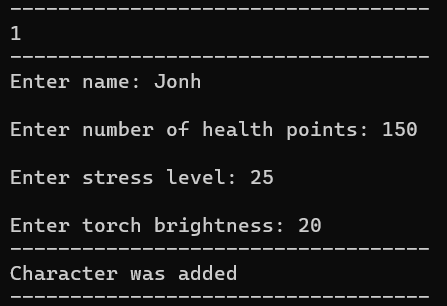


Рисунок 3 − Создание нового персонажа

У здоровья, уровня стресса и уровня освещения есть границы. Если значение слишком мало или слишком велико, выводится сообщение об ошибке, что представлено на рисунке 4.

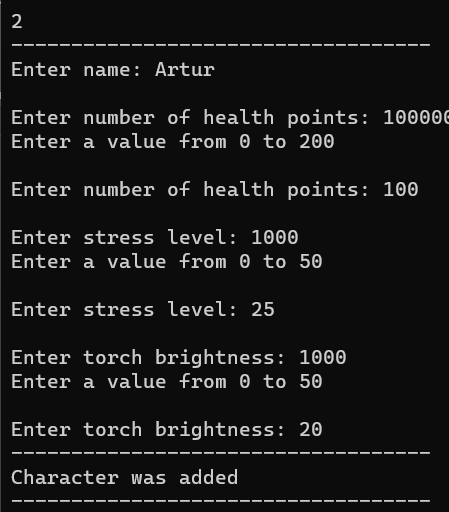


Рисунок 4 − Результат работы программы при недопустимых значениях

При нажатии на «0» произведется выход в главное меню.

При нажатии на «2» можно увидеть список всех имеющихся персонажей, что показано на рисунке 5.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 − Вывод имеющихся персонажей

При нажатии на «3» можно выбрать, какой метод совершить. Появляется меню, что представлено на рисунке 6.

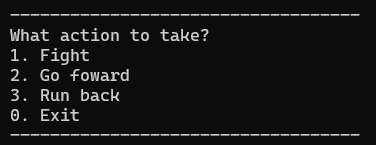


Рисунок 6 − Меню выбора метода

При нажатии на «1» выполняется метод для всех классов, что представлено на рисунке 7.

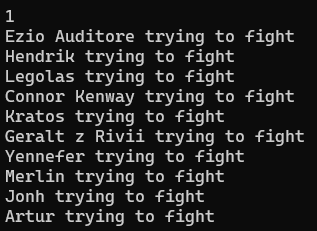


Рисунок 7 − Выполнение метода

При нажатии на «2» выполняется метод для всех классов, что представлено на рисунке 8.

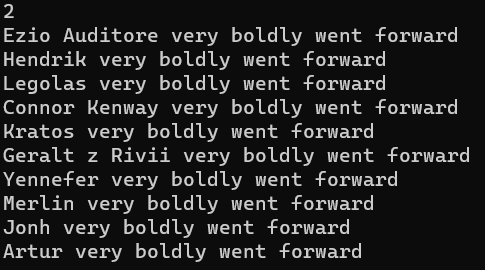


Рисунок 8 − Выполнение метода

При нажатии на «3» выполняется метод для всех классов, что представлено на рисунке 9.

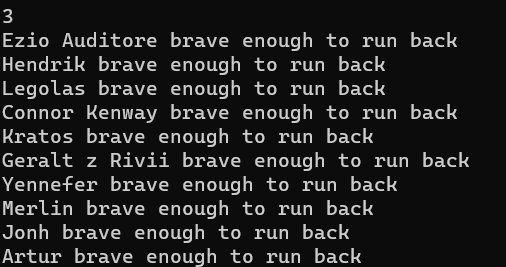


Рисунок 9 − Выполнение метода

При нажатии на «0» произведется выход в главное меню.

При нажатии на «4» можно выполнить уникальный метод. Появится меню выбора класса, что представлено на рисунке 10.

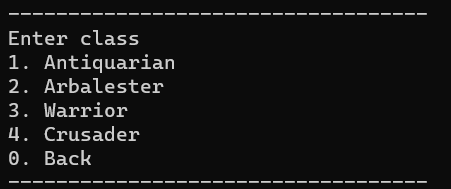


Рисунок 10 − Выбор класса для выполнения уникального метода

При нажатии на «1» совершается уникальный метод для выбранного класса, что представлено на рисунке 11.

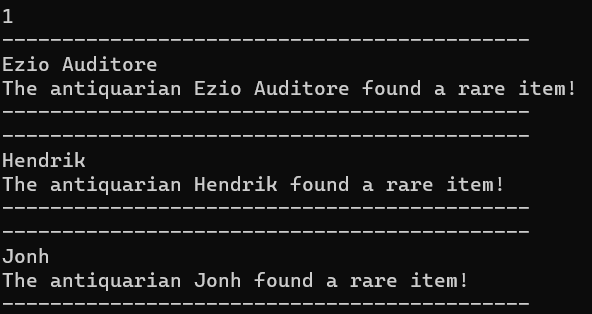


Рисунок 11 − Выполнение уникального метода

При нажатии на «2» совершается уникальный метод для выбранного класса, что представлено на рисунке 12.

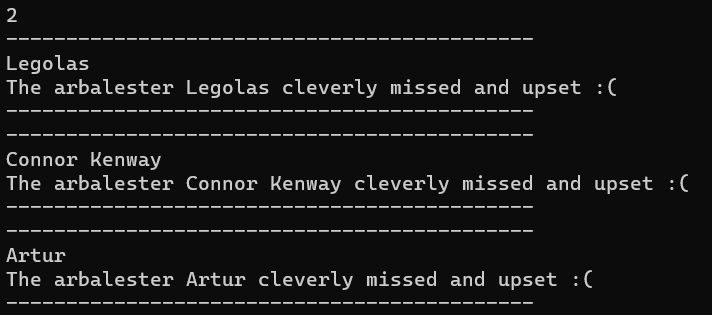


Рисунок 12 − Выполнение уникального метода

При нажатии на «3» совершается уникальный метод для выбранного класса, что представлено на рисунке 13.

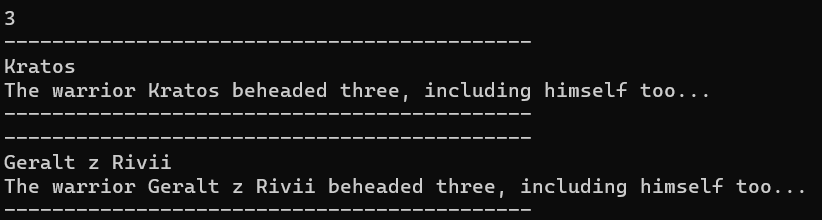


Рисунок 13 − Выполнение уникального метода

При нажатии на «4» совершается уникальный метод для выбранного класса, что представлено на рисунке 14.

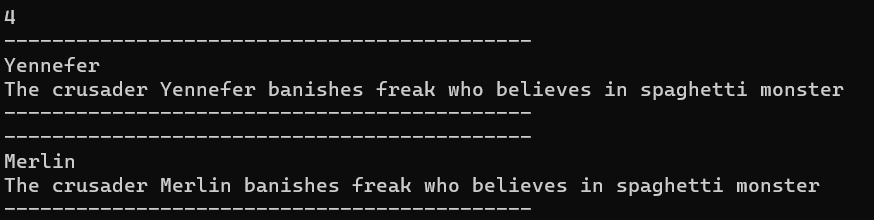


Рисунок 14 − Выполнение уникального метода

При нажатии на «0» произведется выход в главное меню.

При нажатии на «0» в главном меню сработают деструкторы, и программа завершится, что представлено на рисунке 15.

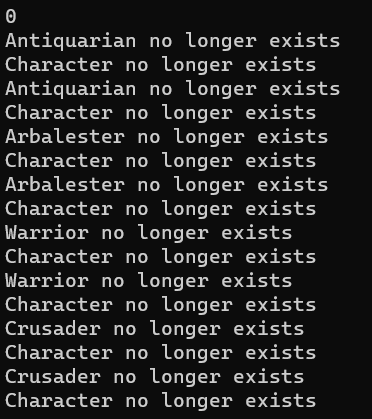


Рисунок 15 − Завершение программы

Если в любом из меню нажать на несуществующий пункт меню, то появится ошибка, что представлено на рисунке 16.

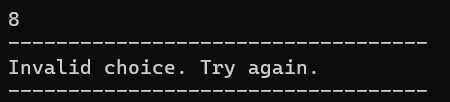


Рисунок 16 − Выбор несуществующего пункта меню