МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

Отчет по лабораторной работе №7.

По ОБЪЕКТО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ

«Обработка собственных событий»

Выполнила:

студентка гр. ПМ-21-2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бутусова В.М.

Проверил:

доц., к.п.н. кафедры АСУ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кургасов В. В.

Липецк 2022

**Цель работы:** научиться создавать и использовать собственные события для возможности отслеживания другими классами наступления определенных условий в текущем классе.

**Задание кафедры:** Реализовать на языке C++ программу, требования для которой представлены в таблице, с использованием механизма событий.

|  |  |  |
| --- | --- | --- |
| Вариант | Описание задачи | События |
| **3** | Программа – эмулятор игры. Имеются классы: комната, игрок. Игроки находятся по два в комнате, обладают некоторым запасом здоровья и по очереди наносят друг другу случайной степени урон. Победа у того, кто первым исчерпал запас здоровья противника. Необходим класс, собирающий статистику по победам из всех комнат, и класс для сохранения сообщений в файл о сработавших событиях. | 1. Победа в игре. 2. Нанесение урона. |

Код программы:

#include<iostream>

#include<string>

#include<fstream>

#include <Windows.h>

using namespace std;

fstream file;

string location = "C:\\Users\\Валерия\\Desktop\\ЛАБЫ ПРОГА\\ООП\\7 лаба\\Project2\\Project2\\x64\\Debug\\doc.txt";

class File {

public:

static void save(string s) // сохранение

{

file.open(location, fstream::in | fstream::app);

file << s;

file.close();

}

static void load() // загрузка

{

file.open(location);

string stroka;

for (getline(file,stroka); !file.eof(); getline(file, stroka))

cout << stroka << endl;

file.close();

}

static void clear() // очистка

{

fstream clear\_file(location, ios::out);

clear\_file.close();

}

};

class Event : public File // класс - событие

{

public:

static void win(string n) // событие - победа

{

string e = "";

e = "Победил игрок " + n + "\n";

cout << e;

save(e); // вызов фук=нкции сохранения собития

}

static void nobody() // событие - ничья

{

string e = "";

e = "Победила дружба\n";

cout << e;

save(e);// вызов фук=нкции сохранения собития

}

static void damage(string n1, string n2, int d) // событие - нанесение урона

{

string e = "";

e = "Игрок " + n1 + " нанёс игроку " + n2 + ": " + to\_string(d) + " урона.\n";

cout << e;

save(e);// вызов фук=нкции сохранения собития

}

};

class Room // класс комнта

{

public:

// шаблонная функция ввода имён игроков

template <typename T>

Room(T& one, T& two)

{

cout << "Введите имя 1 игрока: ";

cin >> one.n;

cout << endl << "Введите имя 1 игрока: ";

cin >> two.n;

}

};

class Player : public Event // класс - игрок

{

public:

string n = "";

int health = 30;

};

// шаблонная функция для нанесения урона

template <typename T, typename P>

bool attac(T & p, P & name)

{

int damage = rand() % 20 + 1; // нанесение случайного урона до 20

p.health -= damage;

p.damage(name, p.n, damage); // событие - нанесение урона

if (p.health <= 0) // если здоровье <= 0

{

p.win(name); // событие - победа

return true;

}

}

int main()

{

setlocale(LC\_ALL, "ru");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char k = 0;

while (k != '4') // главное меню программы

{

cout << "\n1. Посмотреть прошлые соревнования\n";

cout << "2. Начать новую игру\n";

cout << "3. Очистить историю побед\n";

cout << "4. Выход" << endl;

cin >> k;

bool mark = false;

if (k == '1')

{

File::load(); //вызов функции загрузки данных в классе File

}

if (k == '2')

{

system("cls");

Player one; //создание первого игрока

Player two; //создание второго игрока

Room room(one, two); // создание комнаты с игроками

File::save("\nИгроки " + one.n + " и " + two.n +":\n"); // сохранение в файл имён игроков

for (int i = 0; i < 30; i++) // поочерёдная отака игроков

{

if (attac(two, one.n) == true) break;

if (attac(one, two.n) == true) break;

if (i == 30) // дойдя до 30 раунда

{

Event::nobody(); // событие - ничья

}

}

}

if (k == '3')

{

File::clear(); //вызов функции сохранения данных в классе File

cout << "История очищена\n";

}

}

}

**Пример работы программы.**

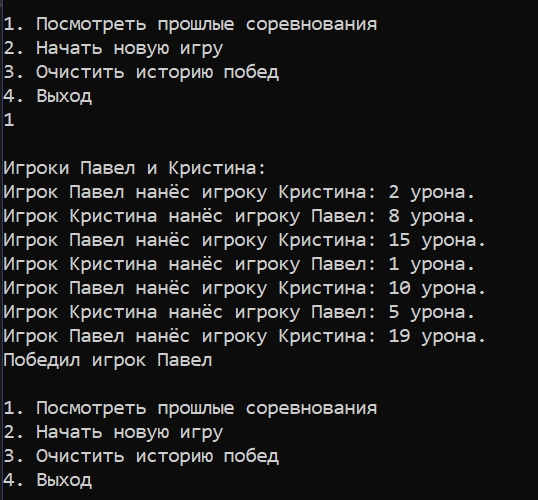


Рисунок 1 – демонстрация загрузки данных

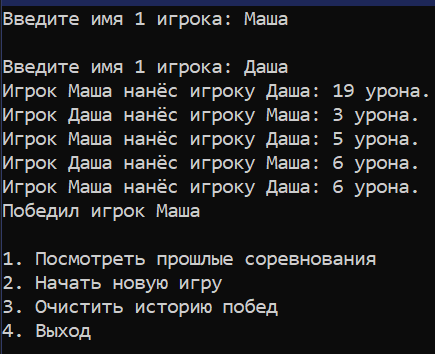


Рисунок 2 – демонстрация запуска игры

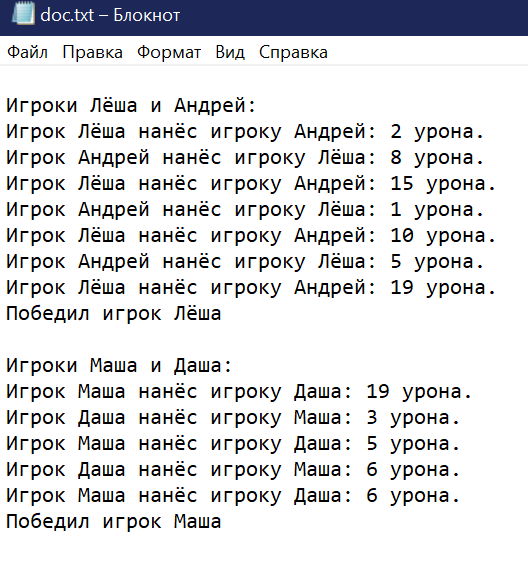


Рисунок 3 – хранение событий в txt файле.

**Вывод:** в данной лабораторной работе была осуществлена возможность отслеживания событий в одном классе другим классом.

**Контрольные вопросы.**

**1. Что такое делегат?**

Делегат - это указатель на статическую функцию или нестатический метод какого-либо класса. Сигнатура, указанная в описании делегата, должна совпадать с сигнатурой метода, на который указывает делегат.

**2. Что такое событие? Как его можно создать?**

Событие - некоторое состояние объекта класса, которому присвоено имя и которое может быть обработано некоторым кодом, который исполняется только при возникновении данного события. В контексте языка С++ событие - это объект, отдельные поля которого характеризуют некоторые свойства передаваемой информации. Создать событие можно с использованием следующего синтаксиса:

event SOME\_DELEGATE ^ EVENT\_NAME;

Где SOME\_DELEGATE - делегат, с которым связывается событие.

**3. Как передать в событие один или несколько параметров?**

Чтоб передать в событие один или несколько параметров, необходимо просто записать их в круглые скобки, которые ставятся после имени делегата, при объявлении делегата.

**4. Что такое обработчик события?**

Обработчики событий – это операторы или функции, которые выполняются в случае наступления определенного события (нажатия на кнопку, изменения содержимого текстового поля, щелчка мышью на гиперссылке и т.д.).

**5. Как отслеживать наступление события?**

Чтобы обрабатывать событие, нужно реализовать метод, соединяющий сигнал со слотом. Когда происходит, например, нажатие кнопки, генерируется сигнал щелчка кнопки мышки, в связи с этим срабатывает указатель на объект программы. Далее программа работает в обычном режиме – согласно логике программы. Таким образом, событие обработано.