Федеральное агентство связи (Россвязь)

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Курсовая работа

«Объектно-ориентированное программирование»

Вариант 25

«Битва противников»

Выполнили:

студент группы

Проверил:

Суходоева Н.Н.

Новосибирск 2018

Оглавление

[Задание на курсовую работу. 3](file:///C:\Users\TheEsus\Desktop\Federalnoe_agentstvo_svyazi.docx#_Toc530662838)

[Описание иерархии объектов и методов объектов. 4](file:///C:\Users\TheEsus\Desktop\Federalnoe_agentstvo_svyazi.docx#_Toc530662839)

[Описание алгоритма основной программы. 7](file:///C:\Users\TheEsus\Desktop\Federalnoe_agentstvo_svyazi.docx#_Toc530662840)

**Задание на курсовую**

* \*\*\*Смоделировать битву двух противников
* Программа должна быть написана с использованием **объектно-ориентированных** технологий.
* Описания объектов и методов необходимо оформить в отдельном модуле.

**Содержание работы:**

* Виртуальные методы
* Инкапсуляция (все поля данных не доступны из внешних функций)
* Наследование (минимум 3 класса, один из которых - абстрактный)
* Полиморфизм
* Конструкторы, Перегрузка конструкторов, Списки инициализации
* Статические элементы

**Описание иерархии объектов и методов объектов.**  
  
В данной курсовой работе присутствуют 3 класса, один из которых абстрактный.  
Абстрактные классы - это классы, которые содержат или наследуют без переопределения хотя бы одну чистую виртуальную функцию.  
Class Unit является абстрактным, в нем присутствует 21 виртуальный метод (каждый виртуальный метод переопределён в дочерних классах) :  
  
1)virtual void Show(){};  
Данная функция отвечает за вывод в графическом окне информации о состоянии героя, которого выбрал игрок (количество жизней, положение на карте, минимальный и максимальный урон, количество возможных супер ударов и дальность атаки (в клетках))  
  
2)virtual void Buff(){};  
Данная функция отвечает за повышение некоторых характеристик героя в соответствии с выбором цвета для героя.  
  
3)virtual void Redraw(){};  
Необходима для движения персонажа на карте. Убирает точку персонажа с карты.

4)virtual void Respawn(){};  
Переводит координаты персонажа, прописанные в полях объекта, в его координаты в графическом окне. Вызывает void Redraw().  
  
5)virtual void Draw(){};  
Рисует точку персонажа на карте по заданным параметрам (Форма - окружность + цвет + тип зарисовки круга).  
  
6)virtual void Spawn(){};  
Переводит координаты персонажа, прописанные в полях объекта, в его координаты в графическом окне. Вызывает void Draw().  
  
7)virtual int PX(){};  
Возвращает заданное значение положение игрока по  Х координате.

8) virtual int PY(){}  
Возвращает заданное значение положения игрока по Y координате.  
  
9) virtual int CheckX(){};

Проверяет, совпадают ли координата «х», передаваемая в функцию, и координата «х» объекта, который тоже подаётся в функцию. Возвращает 0 если они совпали и 1, если не совпали.

10)virtual int CheckY(){};

Проверяет, совпадают ли координата «у», передаваемая в функцию, и координата «у» объекта, который тоже подаётся в функцию. Возвращает 0, если они совпали, и 1, если не совпали.

11)virtual void Chekk(){};

Функция выполняет проверку, не совпали ли координаты двух объектов при первоначальном их создании. Если совпали, то изменяет координаты второго объекта.

12)virtual void Move(){};

Отвечает за перемещение персонажа по карте. В случае выхода за границы или хода на объект соперника выводит сообщение о невозможности такого хода.

13)virtual int MovN(){};

Возвращает значение скорости движения персонажа.

14)virtual int Color(){};

Возвращает цвет героя, который выбрал игрок.

15) virtual void attack\_norm(){};  
Генерирует случайный урон, который нанесёт герой со своей обычной атаки.

16)virtual void attack\_ult(){};

Генерирует случайный урон, который может нанести герой, исходя из параметров урона его объекта, при помощи супер-удара. Уменьшает общее количество супер-ударов, которые может нанести герой за бой.

17)virtual void DMG(){};

Уменьшает кол-во жизней героя в зависимости от полученного урона.

18)virtual int HHH(){};  
Проверяет количество жизней у героя. В том случае, когда кол-во жизней героя упало ниже нуля, функция вернёт «0». Если жизни ещё есть, то вернёт «1».

19)virtual int ULT(){};  
Управление количеством супер ударов, которые может нанести герой за весь бой. В случае если они есть, то возвращается 1, если их нет, то 0.

20)virtual void DDD(){};

Вывод нанесенного урона в консоль

21)virtual int RangeU(){};

Возвращает индивидуальное значение дальности атаки героя.

**Наследование:**

Класс Characters - наследник класса Unit, задача класса – хранить в себе

общие основные переменные, которые участвуют в программе и являются ведущими. В нём реализован список инициализации.

Класс Specialist – наследник класса Characters, задачей класса является переопределение переменных (цвет героя, скорость движения, количество супер ударов, нижний и верхний пределы атаки и дальность атаки), а так же структуры всех методов, которые были описаны выше.

**Доступность компонентов класса:**

Абстрактный класс Unit имеет только спецификатор доступа public для своих методов, полей не имеет.  
Класс Characters имеет спецификатор доступа protected для полей класса и public для конструктора со списком инициализации.

Класс Specialist имеет так же спецификатор доступа private для полей класса и public для определения этих переменных, а так же методов.

**Конструкторы и их перегрузка:**  
Класс Characters имеет конструктор со списком инициализации.

Класс Specialist имеет конструктор с параметрами. Если не подавать параметры в конструктор, то он заполнит объект заданными параметрами, если же подать параметры, то конструктор использует их для заполнения полей объекта.

**Статические переменные:**

В программе используется статическая переменная static int s\_def, которую мы приравниваем к 50 и используем в конструкторе класса Specialist для заполнения поля HP.

В данной работе также была использована технология ООП «передача объектов в качестве параметров функций и методов».

**Описание алгоритма основной программы**

Работа программы начинается с создания каждым из двух игроков своего персонажа. За это отвечают функции Stats(),players\_color() и players().При запуске программы в консоли появляется меню выбора героя для игрока. Выбрав героя, игрок может выбрать цвет героя. Затем ту же процедуру проводит и второй игрок.   
Затем вызываются функции Show\_window() – отрисовка окна и его разделение на поля, Show() – выводит характеристики в спец окнах по обе стороны от карты, map() – вывод карты. Все выводится в графическом окне.

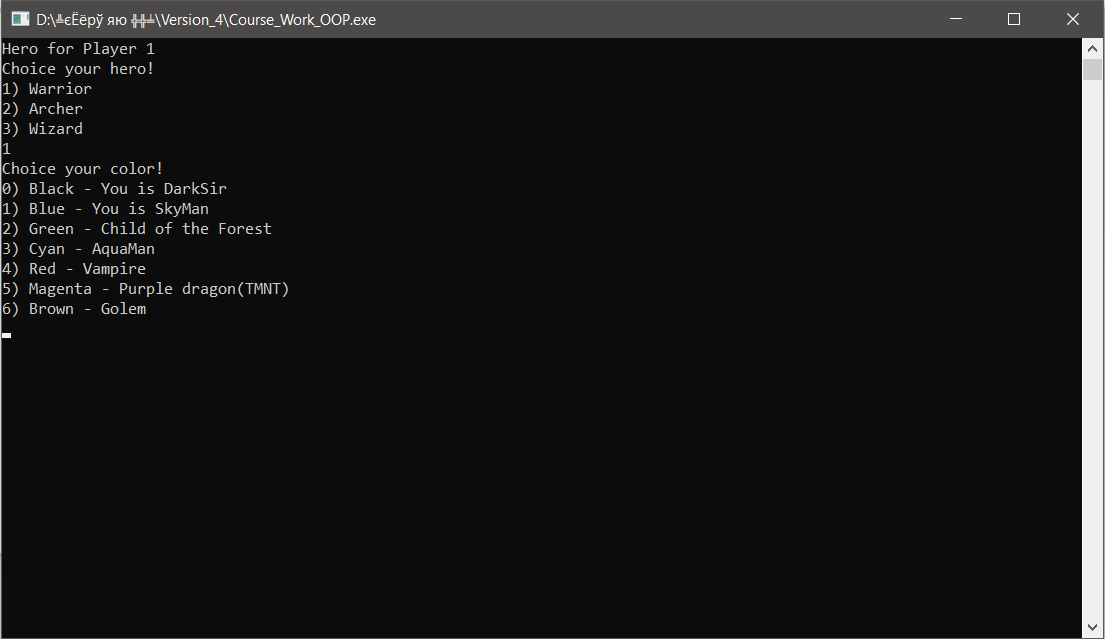
Далее начинается бой между игроками.

Бой координируется функциями Fight() и Turn().

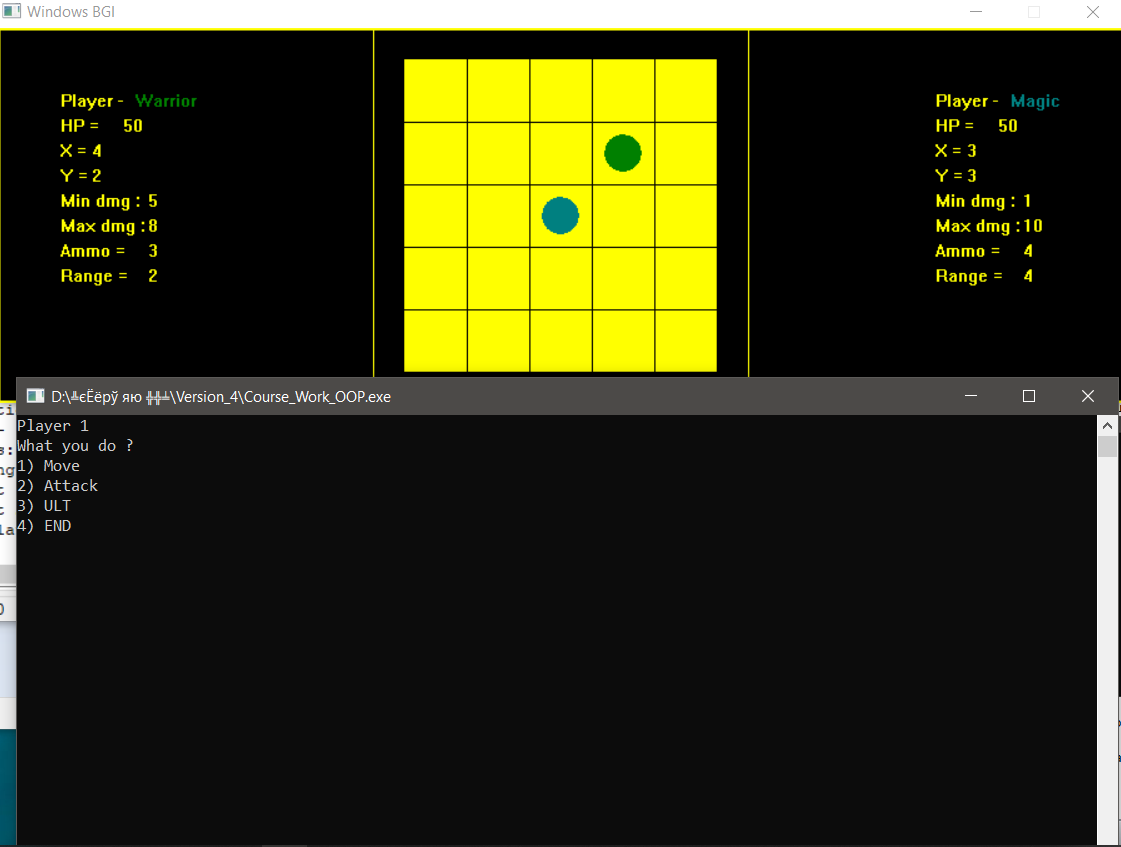
Каждому Игроку предоставляется возможность сделать выбор из 4-х возможностей:  
1)Куда-то походить  
2) Атаковать обычной атакой (если позволяет дальность атаки).  
3)Совершить супер удар (если позволяет дальность).

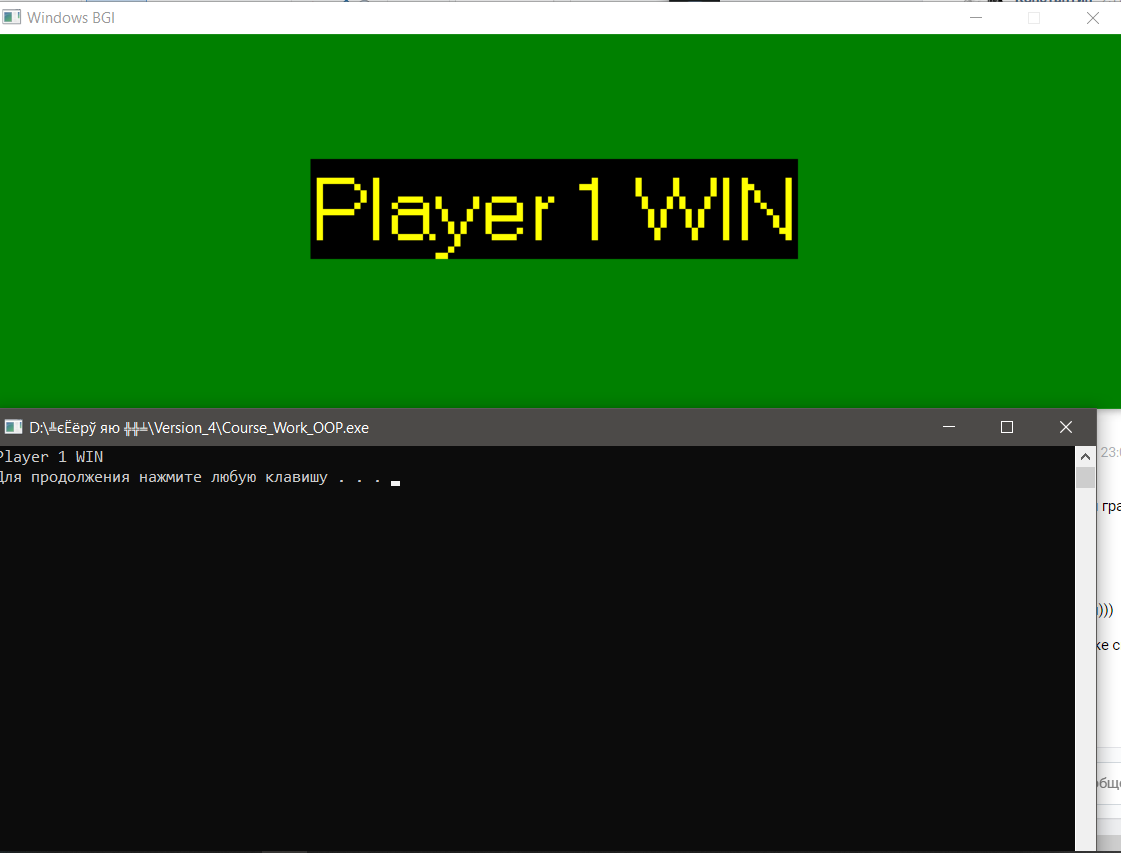
4)Закончить ход

Данные манипуляции повторяются до тех пор, пока один из игроков не одержит победу (пока его герой не убьет героя второго игрока, т.е. пока HP одного из героев не достигнут нуля).

**Скриншоты**Стартовый выбор героя (выбран 1) и цвета героя для первого игрока.  


Аналогично проходит и выбор для второго игрока.

Выводится графическое окно с расставленными персонажами игроков и характеристиками героев и меню хода в консоли.  


Конец игры  


# Вывод

Данная курсовая работы была разработана с использованием основных принципов ООП:

Инкапсуляция, наследование, полиморфизм, также использовались конструкторы и их перегрузка, инициализация объектов, статические элементы, виртуальные функции и передача объектов в качестве параметров функций и методов.

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. Благодаря ООП, мы можем быстро написать программу с таким кодом, который будет понятен остальным.

ООП даёт возможность делить программу на кучу «модулей»-классов, каждый из которых делает свою часть работы. Все объекты, взаимодействуя между собой, образуют работу нашей программы.

Кроме того, написанный нами код можно повторно использовать в другом месте программы, что также экономит большое количество времени.

Мы научились работать с основными принципами ООП: наследование, полиморфизм и инкапсуляция.

Состояние будущих объектов описывается в классе с помощью полей, а их поведение – с помощью методов. Возможность же изменения состояния и поведения осуществляется с помощью модификаторов доступа к полям и методам – private, protected, public.

Для работы с нашим объектом мы оставляем открытыми методы с помощью модификатора public.

Предоставление открытых методов для работы с объектом также является частью механизма инкапсуляции, тат как если полностью закрыть доступ к объекту – он станет бесполезным.

Наследование заключается в использовании уже существующих классов для описания новых.

Принцип в ООП, когда программа может использовать объекты с одинаковым интерфейсом без информации о внутреннем устройстве объекта, называется полиморфизмом.

**Код программы**  
Файл Libra.h

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

#include <graphics.h>

#include <ctime>

#include <conio.h>

const int N = 5;

using namespace std;

class Unit{

public:

static int s\_def;

public:

virtual void Show(){};

virtual void Buff(){};

virtual void Redraw(){};

virtual void Respawn(){};

virtual void Draw(){};

virtual void Spawn(){};

virtual int PX(){};

virtual int PY(){};

virtual int CheckX(){};

virtual int CheckY(){};

virtual void Chekk(){};

virtual void Move(){};

virtual int MovN(){};

virtual int Color(){};

virtual void attack\_norm(){};

virtual void attack\_ult(){};

virtual void DMG(){};

virtual int HHH(){};

virtual int ULT(){};

virtual void DDD(){};

virtual int RangeU(){};

};

class Characters: public Unit {

protected:

int pos\_x;

int pos\_y;

int hp;

int damage;

public:

Characters(): damage(0), pos\_x(1 + rand() % 5), pos\_y(1 + rand() % 5) {}

};

class Specialist: public Characters {

private:

int color;

int mov;

int super\_banck;

int a1;

int a2;

int range\_W;

public:

Specialist(int MOV = 1, int COLOR = 0, int A1 = 0, int A2 = 0, int Super\_Banck = 0, int RANGE\_W = 0){

mov = MOV;

hp = s\_def;

color = COLOR;

a1 = A1;

a2 = A2;

super\_banck = Super\_Banck;

range\_W = RANGE\_W;

}

void Show(int x0, int y0){

string s;

int x; int y;

x = x0; y = y0;

char healse[10], positionX[10],positionY[10];

char dmg1[10], dmg2[10], ammo[10], Range[10];

itoa(hp,healse,10);

itoa(pos\_x,positionX,10);

itoa(pos\_y,positionY,10);

itoa(a1,dmg1,10);

itoa(a2,dmg2,10);

itoa(super\_banck,ammo,10);

itoa(range\_W,Range,10);

setcolor(YELLOW);

outtextxy(x,y,"Player - ");

if(color == 0) setcolor(15);

else setcolor(color);

if(a2==10){

outtextxy(x+60,y,"Magic");

}

if(a2==9){

outtextxy(x+60,y,"Archer");

}

if(a2==8){

outtextxy(x+60,y,"Warrior");

}

setfillstyle(0,0);

if(hp<10) bar(x,y+20,x+100,y+40);

setcolor(YELLOW);

outtextxy(x,y+=20,"HP = ");

outtextxy(x+50,y,healse);

outtextxy(x,y+=20,"X = ");

outtextxy(x+25,y,positionX);

outtextxy(x,y+=20,"Y = ");

outtextxy(x+25,y,positionY);

outtextxy(x,y+=20,"Min dmg : ");

outtextxy(x+70,y,dmg1);

outtextxy(x,y+=20,"Max dmg : ");

outtextxy(x+70,y,dmg2);

outtextxy(x,y+=20,"Ammo = ");

outtextxy(x+70,y,ammo);

outtextxy(x,y+=20,"Range = ");

outtextxy(x+70,y,Range);

}

void Buff(){

if(color == 0) a1=a1+1;

if(color == 1) super\_banck++;

if(color == 2) super\_banck++;

if(color == 3) super\_banck++;

if(color == 4) a1=a1+1;

if(color == 5) hp=hp+7;

if(color == 6) hp=hp+7;

}

void Redraw(int position\_x, int position\_y){

setcolor(YELLOW);

setfillstyle(1,YELLOW);

circle(position\_x, position\_y, 15);

floodfill(position\_x, position\_y, YELLOW);

}

void Respawn(int cor\_x[], int cor\_y[]){

srand(time(NULL));

int position\_x = cor\_x[pos\_x];

int position\_y = cor\_y[pos\_y];

Redraw(position\_x, position\_y);

}

void Draw(int position\_x, int position\_y){

setcolor(color);

setfillstyle(1,color);

circle(position\_x, position\_y, 15);

floodfill(position\_x, position\_y, color);

}

void Spawn(int cor\_x[], int cor\_y[]){

srand(time(NULL));

int position\_x = cor\_x[pos\_x];

int position\_y = cor\_y[pos\_y];

Draw(position\_x, position\_y);

}

int PX(){

return pos\_x;

}

int PY(){

return pos\_y;

}

int CheckX(Specialist P, int xx){

int x0;

x0 = P.PX();

if(xx == x0) return 0;

else return 1;

}

int CheckY(Specialist P, int yy){

int y0;

y0 = P.PY();

if(yy == y0) return 0;

else return 1;

}

void Chekk(Specialist P){

int fx=0; int fy=0;

fx = CheckX(P, pos\_x);

fy = CheckY(P, pos\_y);

if(fx == 0 && fy == 0){

if(pos\_x <5 && pos\_y <5){

pos\_x--;

pos\_y--;

}

else {

pos\_x--;

pos\_y--;

}

}

}

void Move(int cor\_x[], int cor\_y[], Specialist P){

cout<<endl;

int mmm;

int fm=0;

int fx=0; int fy=0;

cout<<"Make move:"<<endl;

cout<<"1) Right"<<endl;

cout<<"2) Left"<<endl;

cout<<"3) Up"<<endl;

cout<<"4) Down"<<endl;

cout<<"Use 0, if you want stay on this point"<<endl;

while(fm==0){

fx=0; fy=0;

cin>>mmm;

if(mmm == 1)//право

{

fx = CheckX(P, pos\_x+1);

fy = CheckY(P, pos\_y);

if(fx == 1 or fy == 1){

if(pos\_x==5){

cout<<"not possible, change your choice"<<endl;

}

else{

Respawn(cor\_x,cor\_y);

pos\_x++;

fm=1;

}

}

else

cout<<"not possible, change your choice"<<endl;

}

if(mmm == 2)//лево

{

fx = CheckX(P, pos\_x-1);

fy = CheckY(P, pos\_y);

if(fx == 1 or fy == 1){

if(pos\_x==1){

cout<<"not possible, change your choice"<<endl;

}

else{

Respawn(cor\_x,cor\_y);

pos\_x--;

fm=1;

}

}

else

cout<<"not possible, change your choice"<<endl;

}

if(mmm == 3)//верх

{

fx = CheckX(P, pos\_x);

fy = CheckY(P, pos\_y-1);

if(fx == 1 or fy == 1){

if(pos\_y==1){

cout<<"not possible, change your choice"<<endl;

}

else{

Respawn(cor\_x,cor\_y);

pos\_y--;

fm=1;

}

}

else

cout<<"not possible, change your choice"<<endl;

}

if(mmm == 4)//низ

{

fx = CheckX(P, pos\_x);

fy = CheckY(P, pos\_y+1);

if(fx == 1 or fy == 1){

if(pos\_y==5){

cout<<"not possible, change your choice"<<endl;

}

else{

Respawn(cor\_x,cor\_y);

pos\_y++;

fm=1;

}

}

else

cout<<"not possible, change your choice"<<endl;

}

if(mmm == 0)//ничего

{

fm=1;

}

}

}

int MovN(){

return mov;

}

int Color(){

return color;

}

void attack\_ult(){

damage = a1 + rand()%(a2-a1);

super\_banck--;

}

void attack\_norm(){

damage = 1 + rand() % (10-1);

}

void DMG(Specialist P){

hp = hp - P.damage;

}

int HHH(){

if(hp <= 0) return 0;

else return 1;

}

int ULT(){

if(super\_banck > 0) return 1;

else return 0;

}

void DDD(){

cout<<damage;

}

int RangeU(){

return range\_W;

}

};

int coordinate(int cor\_x[],int cor\_y[]){

for(int i = 1; i <= 5; i++){

cor\_x[i] = 300 + 50 \* i;

cor\_y[i] = 50 \* i;

}

}

void Show\_window(){

setcolor(YELLOW);

line(0,0,900,0);

line(0,1,900,1);

line(0,299,900,299);

line(0,298,900,298);

line(0,0,0,300);

line(1,0,1,300);

line(899,0,899,300);

line(900,0,900,300);

line(300,0,300,300);

line(600,0,600,300);

}

void map(){

setfillstyle(1,YELLOW);

setcolor(0);

bar(325,25,575,275);

int x = 325, y = 25;

for(x = 375; x < 600; x+=50){

line(x,y, x, y+250);

}

x=325;

for( y = 75; y < 275; y+=50){

line(x,y, x+250, y);

}

}

int players(){

int flag = 0, ch = 0;

while(flag == 0){

cout<<"Choice your hero!\n";

cout<<"1) Warrior \n";

cout<<"2) Archer \n";

cout<<"3) Wizard \n";

cin>>ch;

if(ch == 1 or ch == 2 or ch == 3 ){

flag = 1;

break;

}

system("cls");

}

return ch;

}

int players\_color(){

int flag = 0;

int color;

while(flag == 0){

cout<<"Choice your color!\n";

cout<<"0) Black - You is DarkSir\n";

cout<<"1) Blue - You is SkyMan\n";

cout<<"2) Green - Child of the Forest \n";

cout<<"3) Cyan - AquaMan\n";

cout<<"4) Red - Vampire\n";

cout<<"5) Magenta - Purple dragon(TMNT)\n";

cout<<"6) Brown - Golem\n";

cin>>color;

if(color == 0 or color == 1 or color == 2 or color == 3 or color == 4 or color == 5 or color == 6){

flag = 1;

break;

}

system("cls");

}

return color;

}

void Stats(int chh, int &MOV, int &A1, int &A2, int &Super\_Banck, int &RANGE\_W){

if(chh == 1) //Warrior

{

MOV = 2;

A1 = 5;

A2 = 8;

Super\_Banck = 2;

RANGE\_W = 2;

}

if(chh == 2) //Archer

{

MOV = 1;

A1 = 3;

A2 = 9;

Super\_Banck = 3;

RANGE\_W = 3;

}

if(chh == 3) //Magic

{

MOV = 1;

A1 = 1;

A2 = 10;

Super\_Banck = 3;

RANGE\_W = 4;

}

}

void Turn(Specialist &P1, Specialist &P2, int cor\_x[], int cor\_y[]){

int fa;

int fm;

int ch = 0;

fa = fm = 0;

while((fa==0)||(fm==0))

{

cout<<"What you do ?\n";

cout<<"1) Move \n";

cout<<"2) Attack\n";

cout<<"3) ULT \n";

cout<<"4) END \n";

cin>>ch;

system("cls");

switch(ch){

case 1:

if(fm==0){

int MN;

MN = P1.MovN();

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

P1.Move(cor\_x,cor\_y, P2);

P1.Spawn(cor\_x,cor\_y);

}

fm = 1;

}

else

cout<<"You do it before"<<endl;

break;

case 2:

if(fa==0){

int x1,y1,x2,y2;

x1=P1.PX(); y1=P1.PY();

x2=P2.PX(); y2=P2.PY();

if(x1==x2 or x1==(x2-1) or x1==(x2+1)){

if(y1==y2 or y1==(y2-1) or y1==(y2+1)){

P1.attack\_norm();

P2.DMG(P1);

cout<<"P1 strike ";

P1.DDD();

cout<<" HP P2"<<endl;

fa = 1;

}

else

cout<<"Yoy cann't make this attack"<<endl;

}

else

cout<<"Yoy cann't make this attack"<<endl;

}

else

cout<<"You do it before"<<endl;

break;

case 3:

if(fa==0){

if(P1.ULT() == 1){

int x1,y1,x2,y2;

x1=P1.PX(); y1=P1.PY();

x2=P2.PX(); y2=P2.PY();

int r; r=P1.RangeU();

int i; int fr=0;

int j;

if(x1==x2 or y1==y2){

if(x1==x2){

cout<<"x1"<<endl;

for (i=0;i<=r;i++){

j=y1+i;

cout<<j<<endl;

if(j==y2){

fr=1; break;

}

}

cout<<"x2"<<endl;

for (i=0;i<=r;i++){

j=y1-i;

cout<<j<<endl;

if(j==y2){

fr=1; break;

}

}

}

if(y1==y2){

cout<<"y1"<<endl;

for (i=0;i<=r;i++){

j=x1+i;

cout<<j<<endl;

if(j==x2){

fr=1; break;

}

}

cout<<"y2"<<endl;

for (i=0;i<=r;i++){

j=x1-i;

cout<<j<<endl;

if(j==x2){

fr=1; break;

}

}

}

}

if(fr==1){

P1.attack\_ult();

P2.DMG(P1);

cout<<"You strike ";

P1.DDD();

cout<<" HP your enemy"<<endl;

fa = 1;

}

else

cout<<"Yoy cann't make this attack"<<endl;

}

else

cout<<"You haven't ammo"<<endl;

}

else

cout<<"You do it before"<<endl;

break;

case 4:

fa=1;

fm=1;

break;

}

}

}

void Fight(Specialist &P1, Specialist &P2, int cor\_x[], int cor\_y[]){

int life1 = 1;

int life2 = 1;

int ff = 1;

while((life1)&&(life2)){

system("cls");

if(ff == 1)

{

cout<<"Player 1\n";

Turn(P1,P2,cor\_x,cor\_y);

}

if(ff == -1)

{

cout<<"Player 2\n";

Turn(P2,P1,cor\_x,cor\_y);

}

P1.Show(50,50);

P2.Show(750,50);

life1 = P1.HHH();

life2 = P2.HHH();

ff = ff\*(-1);

system("pause");

}

system("cls");

settextstyle(3,0,8);

int CLR;

if(life1==0){

CLR = P2.Color();

setfillstyle ( 1, CLR);

bar(0,0,900,300);

outtextxy(250,100,"Player 2 WIN");

}

if(life2==0) {

CLR = P1.Color();

setfillstyle ( 1, CLR);

bar(0,0,900,300);

outtextxy(250,100,"Player 1 WIN");

cout<<"Player 1 WIN\n";

}

}

Файл main.cpp

#include "Libra.h"

int Unit::s\_def = 50;

main(){

srand(time(NULL));

int cor\_x[N], cor\_y[N];

coordinate(cor\_x, cor\_y);

int MOV;

int A1;

int A2;

int Super\_Banck;

int RANGE\_W;

int COLOR;

int chh;

cout<< "Hero for Player 1" <<endl;

chh = players();

Stats(chh,MOV,A1,A2,Super\_Banck,RANGE\_W);

COLOR = players\_color();

Specialist P1(MOV, COLOR, A1, A2, Super\_Banck, RANGE\_W);

P1.Buff();

system("cls");

cout<< "Hero for Player 2" <<endl;

chh = players();

Stats(chh,MOV,A1,A2,Super\_Banck,RANGE\_W);

COLOR = players\_color();

Specialist P2(MOV, COLOR, A1, A2, Super\_Banck, RANGE\_W);

P2.Buff();

system("cls");

initwindow(900,300);

map();

Show\_window();

P2.Chekk(P1);

P1.Spawn(cor\_x,cor\_y);

P2.Spawn(cor\_x,cor\_y);

cout<<endl;

P1.Show(50,50);

P2.Show(750,50);

system("pause");

Fight(P1,P2,cor\_x, cor\_y);

system("pause");

closegraph();

return 0;

}