#pragma once

class Rabbit

{

private:

//1. Опис полів

int position\_;

int health\_;

int maxPositionValue\_;

int maxStep\_;

void set\_position(int value);

void set\_health(int value);

void set\_maxPositionValue(int value);

void set\_maxStep(int value);

public:

//2. Описуємо методи доступу get, set

int get\_position();

int get\_health();

int get\_maxPositionValue();

int get\_maxStep();

//3. Конструктори

Rabbit(int position, int health, int maxPositionValue, int maxStep);

Rabbit(int maxPositionValue, int maxStep);

Rabbit(int maxPositionValue);

Rabbit();

//4.Інші методи

void move();

bool shoot(int shootPosition, int damageValue);

bool isAlive();

};

//====================================================================

#include "stdafx.h"

#include "Rabbit.h"

#include<cstdlib>

#include <math.h>

using namespace std;

int Rabbit::get\_position()

{

return position\_;

}

int Rabbit::get\_health()

{

return health\_;

}

int Rabbit::get\_maxPositionValue()

{

return maxPositionValue\_;

}

int Rabbit::get\_maxStep()

{

return maxStep\_;

}

Rabbit::Rabbit(int position, int health, int maxPositionValue, int maxStep)

{

set\_position(position);

set\_health(health);

set\_maxPositionValue(maxPositionValue);

set\_maxStep(maxStep);

}

Rabbit::Rabbit(int maxPositionValue, int maxStep):Rabbit(0,100,maxPositionValue,maxStep)

{

set\_position(rand()%maxPositionValue);

}

Rabbit::Rabbit(int maxPositionValue):Rabbit(maxPositionValue,1)

{

}

Rabbit::Rabbit() : Rabbit(10, 1)

{

}

void Rabbit::move()

{

int shift = -maxStep\_ + rand() % (maxStep\_ \* 2 + 1);

set\_position(position\_+ shift );

}

bool Rabbit::shoot(int shootPosition, int damageValue)

{

if (shootPosition==position\_)

{

set\_health(health\_ - damageValue);

return true;

}

else

{

return false;

}

}

bool Rabbit::isAlive()

{

return health\_>0;

}

void Rabbit::set\_maxStep(int value)

{

if (value>0 && value<maxPositionValue\_)

{

maxStep\_ = value;

}

else

{

throw "Max step can't be negative and greate than max";

}

}

void Rabbit::set\_maxPositionValue(int value)

{

if (value>0)

{

maxPositionValue\_ = value;

}

else

{

throw "Max position can't be negative";

}

}

void Rabbit::set\_position(int value)

{

if (value<0)

{

position\_= value + maxPositionValue\_;

}

else

{

if (value>maxPositionValue\_)

{

position\_ = value - maxPositionValue\_;

}

else

{

position\_ = value;

}

}

}

void Rabbit::set\_health(int value)

{

health\_ = value;

}

//============================================================

#pragma once

#include "Rabbit.h"

class Game

{

private:

int maxPosition\_;

int maxStep\_;

int bulletsCount\_;

int rabbitsCount\_;

int countShootedRabbits\_;

int damageShootValue\_;

Rabbit\* \* rabbits\_;

void set\_rabbitsCount(int value);

void set\_bulletsCount(int value);

void set\_maxPosition(int value);

void set\_maxStep(int value);

int shoot(int shootPosition);

void moveRabbits();

void print();

public:

int get\_bulletsCount();

int get\_rabbitsCount();

Game(int bulletsCount,int rabbitsCount, int maxPosition, int maxStep);

Game();

void play();

~Game();

};

//============================================================

#include "stdafx.h"

#include "Game.h"

#include<cstdlib>

#include<iostream>

using namespace std;

void Game::set\_rabbitsCount(int value)

{

if (value > 0)

{

rabbitsCount\_ = value;

rabbits\_ = new Rabbit\*[rabbitsCount\_];

for (int i = 0; i < rabbitsCount\_; i++)

{

rabbits\_[i] = new Rabbit(maxPosition\_, maxStep\_);

}

}

else

{

throw "Rabbits count must be >0";

}

}

void Game::set\_bulletsCount(int value)

{

if (value > 0)

{

bulletsCount\_ = value;

}

else

{

throw "Bullets count can't be <=0";

}

}

void Game::set\_maxPosition(int value)

{

if (value > 0)

{

maxPosition\_ = value;

}

else

{

throw "Max positin must be >0";

}

}

void Game::set\_maxStep(int value)

{

if (value > 0 && value < maxPosition\_)

{

maxStep\_ = value;

}

else

{

throw "Max positin must be >0 && < maxPosition";

}

}

int Game::shoot(int shootPosition)

{

int countStroked = 0;

for (int i = 0; i < rabbitsCount\_; i++)

{

if (rabbits\_[i]->isAlive())

{

if (rabbits\_[i]->shoot(shootPosition, damageShootValue\_))

{

countStroked++;

if (!rabbits\_[i]->isAlive())

{

countShootedRabbits\_++;

}

}

}

}

return countStroked;

}

void Game::moveRabbits()

{

for (int i = 0; i < rabbitsCount\_; i++)

{

if (rabbits\_[i]->isAlive())

{

rabbits\_[i]->move();

}

}

}

void Game::print()

{

system("cls");

printf("Are alive %d \n", rabbitsCount\_ - countShootedRabbits\_);

for (int i = 0; i < rabbitsCount\_; i++)

{

if (rabbits\_[i]->isAlive())

{

printf("%d\n", rabbits\_[i]->get\_position());

}

}

}

int Game::get\_bulletsCount()

{

return bulletsCount\_;

}

int Game::get\_rabbitsCount()

{

return rabbitsCount\_;

}

Game::Game(int bulletsCount, int rabbitsCount, int maxPosition, int maxStep)

{

set\_bulletsCount(bulletsCount);

set\_maxPosition(maxPosition);

set\_maxStep(maxStep);

set\_rabbitsCount(rabbitsCount);

countShootedRabbits\_ = 0;

damageShootValue\_ = 50;

}

Game::Game() :Game(10, 2, 10, 1)

{

}

void Game::play()

{

while (rabbitsCount\_ > countShootedRabbits\_ && bulletsCount\_ > 0)

{

print();

//-----------------

printf("Enter shoot position:");

int shootPosition;

cin >> shootPosition;

//------------------

int stroked = shoot(shootPosition);

printf("Strked : %d\n", stroked);

bulletsCount\_--;

//--------------------

moveRabbits();

system("pause");

}

if (rabbitsCount\_ > countShootedRabbits\_)

{

printf("You loose!!!\n");

}

else

{

printf("You won!!!\n");

}

}

Game::~Game()

{

for (int i = 0; i < rabbitsCount\_; i++)

{

delete rabbits\_[i];

}

delete[] rabbits\_;

}

//==========================================================

// ConsoleApplication37.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <string>

#include <iostream>

#include "Game.h"

using namespace std;

int main()

{

Game\* game = new Game(5,2,11,2);

game->play();

delete game;

system("pause");

return 0;

}