Федеральное агентство связи

Сибирский государственный университет телекоммуникаций и информатики

Кафедра прикладной математики и кибернетики

КУРСОВАЯ РАБОТА

«Морской бой с применением ООП»

Выполнил: студент группы

Проверил:

Новосибирск 2014

**Содержание**

|  |  |
| --- | --- |
| 1. Постановка задачи------------------------------------------------------------------------- | стр 3 |
| 1. Основные идеи и характеристики применяемых методов------------------------ | стр 4 |
| 1. Текст программы-------------------------------------------------------------------------- | стр 5 |
| 1. Вывод---------------------------------------------------------------------------------------- | стр 14 |

1. **Постановка задачи.**

Программирование игры «Морской бой» для графического режима. Нужно использовать основные принципы объектно-ориентированного программирования.

1. **Основные идеи и характеристики применяемых методов.**

**Объектно-ориентированное программирование (ООП)** — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

*Инкапсуляция* — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя. Во всех классах программы реализовано разделение на protected и public блоки. Работать с полями из protected блока могут только методы из public и методы из public наследников. *Наследование* — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом. Создан класс, который является родительским и от которого наследуются разно палубные корабли. Родитель имеет инициализацию, к которой в будущем будут обращаться потомки.

*Полиморфизм* — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта. При использовании термина «полиморфизм» в сообществе ООП подразумевается полиморфизм подтипов; а использование параметрического полиморфизма называют обобщённым программированием.

*Класс* является описываемой на языке терминологии исходного кода моделью ещё не существующей сущности (объекта). Фактически он описывает устройство объекта, являясь своего рода чертежом. Говорят, что объект — это экземпляр класса. При этом в некоторых исполняющих системах класс также может представляться некоторым объектом при выполнении программы посредством динамической идентификации типа данных. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.

*Объект* - сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса или копирования прототипа (например, после запуска результатов компиляции и связывания исходного кода на выполнение).

1. **Текст программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#include <graphics.h>

#include <vector>

int Nastup = 0; // режим "Наступления"

int NX = 0, NY = 0; // координаты наступления

class Map{

private:

int matrix[11][11]; // матрица, содержащая в себе игровое поле

int xM, yM; // координаты поля

int k; // количество кораблей

public:

int GetK(){ return k; } // метод, возвращающий количество кораблей

int MinusK(){ this->k--; } // метод, уменьшающий количество кораблей на 1

void Draw(){

char\* massC[12] = {"1","2","3","4","5","6","7","8","9","10"," "," "}; // массив цифр

for(int i = 0; i <= 275; i+=25){ // рисуем игровое поле

outtextxy(xM+i+25+5,yM+5,massC[i/25]); // выводим цифры по X и Y

outtextxy(xM+5,yM+i+25+5,massC[i/25]);

moveto(xM,yM+i);

lineto(xM+275,yM+i);

moveto(xM+i,yM);

lineto(xM+i,yM+275);

}

for(int i = 0; i <= 10; i++){ // проходим по матрице, рисуя то, что в ней находиться

for(int j = 0; j <= 10; j++){

if(matrix[i][j]==1){ // 1 - cвой корабль, 2 - утопленник, 4 - неактивная ячейка, 3 - враг, 6 - мимо, 5 - попадание

bar(i\*25+xM+25,j\*25+yM+25,i\*25+xM+50,j\*25+yM+50); // белый закращенный прямоугольник

} else if(matrix[i][j]==5){

setfillstyle(1,GREEN); // устанавливаем зеленый цвет заливки

bar(i\*25+xM+25,j\*25+yM+25,i\*25+xM+50,j\*25+yM+50); // зеленый прямоугольник

setfillstyle(1,WHITE); // возвращаем цвет заливки в исходное состояние

} else if(matrix[i][j]==6){ // если удар был мимо, то нарисуем на этом месте несколько кругов различного радиуса

circle(i\*25+xM+25+25/2,j\*25+yM+25+25/2,25/2);

circle(i\*25+xM+25+25/2,j\*25+yM+25+25/2,25/3);

circle(i\*25+xM+25+25/2,j\*25+yM+25+25/2,25/4);

circle(i\*25+xM+25+25/2,j\*25+yM+25+25/2,25/5);

circle(i\*25+xM+25+25/2,j\*25+yM+25+25/2,25/6);

circle(i\*25+xM+25+25/2,j\*25+yM+25+25/2,25/7);

}

}

}

}

Map(int xT, int yT){ // конструктор для карты, устанавливающий ее начальные координаты и обнуляющий все ее значения, а так же количество кораблей на карте

this->xM = xT; this->yM = yT;

for(int i = 0; i < 10; i++){

for(int j = 0; j < 10; j++){

matrix[i][j] = 0;

}

}

this->k = 10;

Draw(); // отрисовываем игровое поле

}

int SetYa(int x, int y, int P, int status, int Z){ // метод для изменения данных в ячейках игрового поля

if(P == 1){ // P – показатель, какое количество палуб у корабля

if(matrix[x-1][y-1] != 1 && matrix[x-1][y-1] != 4){ // мы можем поставить корабль, если он не граничит с другим и не ставиться на другой корабль

for(int i = x-2; i <= x; i++){ // заполняем область однопалубного корабля

if(i == -1 || i == 10) continue; // не заполняем у матрицы индексы -1 и 10 по Y

for(int j = y-2; j <= y; j++){

if(j == 10 || j == -1) continue; // аналогично не заполняем по X

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z;

} else { return 1; }

} else if(P == 2){ // если двухпалубный корабль, то добавляется переменная, отвечающая за направление

switch(status){ // в зависимости от направления заполняем разную окружную территорию корабля, 0 – вверх, 1 – вправо, 2 – вниз, 3 - влево

case 0:{

if((y-1<1) ||matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-1][y-2] == 1 || matrix[x-1][y-2] == 4) return 1;

for(int i = x-2; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-3; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-1][y-2] = Z;

} break;

case 1:{

if((x-1 > 8) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x][y-1] == 1 || matrix[x][y-1] == 4) return 1;

for(int i = x-2; i <= x+1; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x][y-1] = Z;

} break;

case 2:{

if((y-1 > 8) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x][y-1] == 1 || matrix[x][y-1] == 4) return 1;

for(int i = x-2; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y+1; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-1][y] = Z;

} break;

case 3:{

if((x-1 < 1) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-2][y-1] == 1 || matrix[x-2][y-1] == 4) return 1;

for(int i = x-3; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-2][y-1] = Z;

} break;

}

} else if(P == 3){

switch(status){

case 0:{

if((y-1 < 2) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-1][y-2] == 1 || matrix[x-1][y-2] == 4 || matrix[x-1][y-3] == 1 || matrix[x-1][y-3] == 4) return 1;

for(int i = x-2; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-4; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-1][y-2] = Z; matrix[x-1][y-3] = Z; } break;

case 1:{

if((x-1 > 7) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x][y-1] == 1 || matrix[x][y-1] == 4 || matrix[x+1][y-1] == 1 || matrix[x+1][y-1] == 4) return 1;

for(int i = x-2; i <= x+2; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x][y-1] = Z; matrix[x+1][y-1] = Z; } break;

case 2:{

if((y - 1 > 7) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-1][y] == 1 || matrix[x-1][y] == 4 || matrix[x-1][y+1] == 1 || matrix[x-1][y+1] == 4) return 1;

for(int i = x-2; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y+2; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-1][y] = Z; matrix[x-1][y+1] = Z; } break;

case 3:{

if((x - 1 < 2) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-2][y-1] == 1 || matrix[x-2][y-1] == 4 || matrix[x-3][y-1] == 1 || matrix[x-3][y-1] == 4) return 1;

for(int i = x-4; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-2][y-1] = Z; matrix[x-3][y-1] = Z; } break;

}

} else if(P == 4){

switch(status){

case 0:{

if((y-1 < 3) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-1][y-2] == 1 || matrix[x-1][y-2] == 4 || matrix[x-1][y-3] == 1 || matrix[x-1][y-3] == 4 || matrix[x-1][y-4] == 1 || matrix[x-1][y-4] == 4) return 1;

for(int i = x-2; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-5; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-1][y-2] = Z; matrix[x-1][y-3] = Z; matrix[x-1][y-4] = Z; } break;

case 1:{

if((x - 1 > 6) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x][y-1] == 1 || matrix[x][y-1] == 4 || matrix[x+1][y-1] == 1 || matrix[x+1][y-1] == 4 || matrix[x+2][y-1] == 1 || matrix[x+2][y-1] == 4) return 1;

for(int i = x-2; i <= x+3; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x][y-1] = Z; matrix[x+1][y-1] = Z; matrix[x+2][y-1] = Z; } break;

case 2:{

if((y - 1 > 6) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-1][y] == 1 || matrix[x-1][y] == 4 || matrix[x-1][y+1] == 1 || matrix[x-1][y+1] == 4 || matrix[x-1][y+2] == 1 || matrix[x-1][y+2] == 4) return 1;

for(int i = x-2; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y+3; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-1][y] = Z; matrix[x-1][y+1] = Z; matrix[x-1][y+2] = Z; } break;

case 3:{

if((x - 1 < 3) || matrix[x-1][y-1] == 1 || matrix[x-1][y-1] == 4 || matrix[x-2][y-1] == 1 || matrix[x-2][y-1] == 4 || matrix[x-3][y-1] == 1 || matrix[x-3][y-1] == 4 || matrix[x-4][y-1] == 1 || matrix[x-4][y-1] == 4) return 1;

for(int i = x-5; i <= x; i++){

if(i == -1 || i == 10) continue;

for(int j = y-2; j <= y; j++){

if(j == 10 || j == -1) continue;

matrix[i][j] = 4;

}

}

matrix[x-1][y-1] = Z; matrix[x-2][y-1] = Z; matrix[x-3][y-1] = Z; matrix[x-4][y-1] = Z; } break;

}

} else { return 1; }

return 0;

}

int Izm(int x, int y){

if(matrix[x-1][y-1] == 5 || matrix[x-1][y-1] == 6){

return 0;

} else if(matrix[x-1][y-1] == 3 || matrix[x-1][y-1] == 1){

matrix[x-1][y-1] = 5;

int status = 1;

if((matrix[x-1][y-2] == 1 || matrix[x-2][y-1] == 1 || matrix[x][y-1] == 1 || matrix[x-1][y] == 1) || (matrix[x-1][y-2] == 3 || matrix[x-2][y-1] == 3 || matrix[x][y-1] == 3 || matrix[x-1][y] == 3)){

status = 0;

}

if(status == 1){

return 3;

}

return 1;

} else if(matrix[x-1][y-1] == 4 || matrix[x-1][y-1] == 0){

matrix[x-1][y-1] = 6;

return 2;

}

}

};

class Ship{

protected:

int pr; // принадлежность к определенному полю

int x,y; // координаты коробля

int status; // 0 - мертв, 1 - живой

public:

void Init(int xT, int yT, int prT){ // метод инициализации, задающий начальные значения

this->x = xT;

this->y = yT;

this->pr = prT;

this->status = 1;

}

void Kill(){ // если корабль был убит, то изменяем его статус

status = 0;

}

};

class Ship\_1:public Ship{ // однопалубные корабли

public:

void Init(int xT, int yT, int prT){ // функция инициализации, отправляющая данные в родительский класс

Ship::Init(xT,yT,prT);

}

};

int Ship1Init = 0; // количество однопалубных кораблей

std::vector<Ship\_1> vShip\_1(8); // вектор, к которому можно обращаться, как к списку, так и как к массиву

class Ship\_2:public Ship{

protected:

int napr; // направление корабля

public:

void Init(int xT, int yT, int n, int prT){

Ship::Init(xT,yT,prT);

this->napr = n;

}

};

int Ship2Init = 0;

std::vector<Ship\_2> vShip\_2(6);

class Ship\_3:public Ship{

protected:

int napr; // направление корабля

public:

void Init(int xT, int yT, int n, int prT){

Ship::Init(xT,yT,prT);

napr = n;

}

};

int Ship3Init = 0;

std::vector<Ship\_3> vShip\_3(4);

class Ship\_4:public Ship{

protected:

int napr; // направление корабля

public:

void Init(int xT, int yT, int n, int prT){

Ship::Init(xT,yT,prT);

napr = n;

}

};

int Ship4Init = 0;

std::vector<Ship\_4> vShip\_4(2);

void War(Map MyMap, Map WarMap){ // функция «война», принимающая два объекта (свою и вражескую карту)

printf("Да начнется битва!");

printf("Введите координаты: ");

while(1){ // бесконечный цикл, который выходит в том условии, если у врага или у игрока не осталось кораблей

if(WarMap.GetK() == 0 || MyMap.GetK() == 0) break; // условие выхода

int x = 0, y = 0;

metka:

scanf("%d %d",&x,&y); // считываем x и y, в которые будем стрелять

int P = WarMap.Izm(x,y); // изменяем значение в ячейке на вражеской карте

if(P == 0){ // если метод вернул 0, то в данную ячейку стрелять нельзя

printf("\nВы уже стреляли в эту ячейку! Повторите:\n");

goto metka; // возвращаемся к считыванию координат

} else if(P == 1){ // если вернулась 1, то это означает попадание

printf("Вы попали!\n");

} else if(P == 2){ // если 2, то игрок промахнулся

printf("Мимо!\n");

} else if(P == 3){ // 3 возвращается в том случае, когда корабль врага был полностью уничтожен

printf("Корабль врага уничтожен! :))\n");

WarMap.MinusK(); // в этом случае уменьшаем количество кораблей на вражеской части

printf("У врага осталось кораблей: %d\n",WarMap.GetK()); // выводим количество оставшихся кораблей

}

WarMap.Draw(); // перерисовываем вражескую карту

metka8:

if(Nastup == 0){ // если режим «наступления» не активен, то выбираем случайную ячейку

x = rand()%10+1;

y = rand()%10+1;

} else if(Nastup <= 4){ // если «наступление» активно и меньше 4

int r = rand()%4; // выбираем одно из четырех направлений

switch(r){ // в зависимости от выбранного направления выбираем ближайшую ячейка

case 0: if(NX < 10){ x = NX+1; y = NY; } break;

case 1: if(NX > 0){ x = NX-1; y = NY; } break;

case 2: if(NY > 0){ x = NX; y = NY-1; } break;

case 3: if(NY < 10){ x = NX; y = NY+1; } break;

}

}

P = MyMap.Izm(x,y); // изменяем статус ячейки

int Fl = 0; // сколько раз в невозможное место

if(P == 0){ // для компьютера такие же условия и возвращаемые значения, как и для игрока

Fl++;

if(Fl == 4){

Nastup = NX = NY = Fl = 0;

printf("Наступление окончено!");

}

goto metka8; // возврат и выбор другой случайной ячейки

} else if(P == 1){

printf("Увы, но в ваш корабль попали!\n");

Nastup++;

NX = x;

NY = y;

} else if(P == 2){

printf("Уху, враг промахнулся!\n");

} else if(P == 3){

printf("Ваш корабль потоплен :((\n");

Nastup = 0;

MyMap.MinusK();

printf("У вас осталось кораблей: %d\n",MyMap.GetK());

}

MyMap.Draw(); // перерисовываем карту игрока

}

if(MyMap.GetK() == 0){ // если у игрока не осталось кораблей, то выводим надпись о проигрыше

printf("Вы проиграли! :((");

}

if(WarMap.GetK() == 0){ // если у компьютера не осталось кораблей, то выводим надпись о победе

printf("Вы выиграли! :)");

}

system("PAUSE");

}

void Start(){ // функция, вызываемая при старте игры

Map map1(25,25); // первое игровое поле

Map map2(350,25); // игровое поле врага

for(int i = 0; i < 4; i++){

printf("Разместите 1-нопалубные корабли введя координаты[X Y]:\n");

metka:

int k = 0;

int b = 0;

scanf("%d %d",&b,&k); // считываем координаты для постановки

if(map1.SetYa(b,k,1,1,1) == 0){ // если удачно поставили корабль, то инициализируем его, прибавляем количество уже поставленных кораблей и перерисовываем карту игрока

vShip\_1[Ship1Init].Init(b,k,1);

Ship1Init++;

map1.Draw();

} else { // иначе говорим, что нельзя поставить

printf("Невозможно установить! Повторите: ");

goto metka; // возвращаемся к вводу координат

}

}

printf("\n0 - вверх, 1 - вправо, 2 - вниз, 3 - влево\n");

for(int i = 0; i < 3; i++){ // для другого количества палуб добавляется только направление

printf("Разместите 2-хпалубные корабли введя координаты [X Y направление]:\n");

metka1:

int k = 0;

int b = 0;

int n = 0;

scanf("%d %d %d",&b,&k,&n);

if(map1.SetYa(b,k,2,n,1) == 0){

vShip\_2[Ship2Init].Init(b,k,n,1);

Ship2Init++;

map1.Draw();

} else {

printf("Невозможно установить! Повторите: ");

goto metka1;

}

}

for(int i = 0; i < 2; i++){

printf("Разместите 3-хпалубные корабли введя координаты [X Y направление]:\n");

metka2:

int k = 0;

int b = 0;

int n = 0;

scanf("%d %d %d",&b,&k,&n);

if(map1.SetYa(b,k,3,n,1) == 0){

vShip\_3[Ship3Init].Init(b,k,n,1);

Ship3Init++;

map1.Draw();

} else {

printf("Невозможно установить! Повторите: ");

goto metka2;

}

}

for(int i = 0; i < 1; i++){

printf("Разместите 4-хпалубный корабль введя координаты [X Y направление]:\n");

metka3:

int k = 0;

int b = 0;

int n = 0;

scanf("%d %d %d",&b,&k,&n);

if(map1.SetYa(b,k,4,n,1) == 0){

vShip\_4[Ship4Init].Init(b,k,n,1);

Ship3Init++;

map1.Draw();

} else {

printf("Невозможно установить! Повторите:\n");

goto metka3;

}

}

srand(time(NULL));

for(int i = 0; i < 4; i++){ // подобным образом компьютер расставляет свои корабли, только координаты он выбирает случайно

metka4:

int k = rand()%10+1;

int b = rand()%10+1;

if(map2.SetYa(b,k,1,1,3) == 0){

vShip\_1[Ship1Init].Init(b,k,2);

Ship1Init++;

} else {

goto metka4;

}

}

for(int i = 0; i < 3; i++){

metka5:

int k = rand()%10+1;

int b = rand()%10+1;

int n = rand()%4;

if(map2.SetYa(b,k,2,n,3) == 0){

vShip\_2[Ship2Init].Init(b,k,n,2);

Ship2Init++;

} else {

goto metka5;

}

}

for(int i = 0; i < 2; i++){

metka6:

int k = rand()%10+1;

int b = rand()%10+1;

int n = rand()%4;

if(map2.SetYa(b,k,3,n,3) == 0){

vShip\_3[Ship3Init].Init(b,k,n,2);

Ship3Init++;

} else {

goto metka6;

}

}

for(int i = 0; i < 1; i++){

metka7:

int k = rand()%10+1;

int b = rand()%10+1;

int n = rand()%4;

if(map2.SetYa(b,k,4,n,3) == 0){

vShip\_4[Ship4Init].Init(b,k,n,2);

Ship4Init++;

} else {

goto metka7;

}

}

cleardevice(); // очищаем экран

map2.Draw(); // перерисовываем обе карты

map1.Draw();

War(map1,map2); // переходим к «войне»

}

int main(){

setlocale(LC\_ALL, "Russian"); // устанавливаем русский язык

initwindow(700,400);

Start();

closegraph();

return 0;

}

1. **Вывод**

В курсовой работе реализована игра «Морской бой» для графического режима. В программе используется большая часть изученного материала по дисциплине Объектно-ориентированное программирование.