Федеральное агентство связи

Сибирский государственный университет телекоммуникаций и информатики

Кафедра прикладной математики и кибернетики

КУРСОВАЯ РАБОТА

по предмету объектно-ориентированное программирование

«PAC-MAN»

Выполнил: студент группы ИП-312

Проверил:

Климова И.В.

Новосибирск 2014

**Содержание**

|  |  |
| --- | --- |
| 1. Постановка задачи------------------------------------------------------------------------- | стр 3 |
| 1. Основные идеи применяемых методов----------------------------------------------- | стр 4 |
| 1. Классы в программе----------------------------------------------------------------------- | стр 5 |
| 1. Вывод---------------------------------------------------------------------------------------- | стр 9 |

1. **Постановка задачи.**

С помощью ООП разработать игру PAC-MAN в графическом режиме.

**Необходимый минимум содержания работы:**

* Инкапсуляция (все поля данных не доступны из внешних функций)
* Наследование (минимум 3 класса, один из которых - абстрактный)
* Полиморфизм
* Конструкторы, Перегрузка конструкторов
* Желательно: минимум + 2 технологии ООП (статические элементы, дружественные функции, классы, виртуальные функции, шаблоны, множественное наследование, массивы указателей на объекты, конструкторы копирования, параметры по умолчанию, использование объектов в качестве аргументов или возвращаемых значений).

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

**Объектно-ориентированное программирование (ООП)** — парадигма программирования, в которой основными концепциями являются объекты и классы.

Инкапсуляция — это свойство, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя.

Наследование — это свойство, 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.

Полиморфизм — использование объектов 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.

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).

Абстрактный класс — базовый класс, который не предполагает создания экземпляров. Абстрактный класс может содержать (и не содержать) абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором объявлен, однако должен быть реализован для его неабстрактных потомков. Абстрактные классы представляют собой наиболее общие абстракции, то есть имеющие наибольший объём и наименьшее содержание.

Виртуальный метод — метод класса, который может быть переопределён в классах-наследниках так, что конкретная реализация метода для вызова будет определяться во время исполнения. Таким образом, программисту необязательно знать точный тип объекта для работы с ним через виртуальные методы: достаточно лишь знать, что объект принадлежит классу или наследнику класса, в котором метод объявлен.

1. **Классы программы**

class Map{

private:

int matrix[20][20]; // матрица для рисования карты

public:

int getM(int x, int y){ return matrix[(int)x/20][(int)y/20]; } // метод, для возвращения значений в ячейке

void setM(int x, int y){ matrix[(int)x/20][(int)y/20] = 0; } // метод для обнуления ячейки

int getStat(){ for(int i = 1; i < 19; i++){ // метод, проверяющий, остались ли ещё «бонусы»

for(int j = 1; j < 19; j++){

if(matrix[i][j] == 2) return 1;

}

}

return 0; }

Map(){ // конструктор

int matrix[20][20] = {{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}, // матрица, для рисования карты (1 – граница, 0 – «бонусы»)

{1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1},

{1,0,1,1,1,1,0,0,0,0,1,0,0,0,1,1,1,1,0,1},

{1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1},

{1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1},

{1,0,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,0,1},

{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1},

{1,0,0,0,1,0,0,0,1,1,0,1,1,0,0,1,0,0,0,1},

{1,1,1,0,1,0,0,0,1,0,0,0,1,0,0,1,0,1,1,1},

{1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,1,0,0,0,1},

{1,0,0,0,1,0,0,0,1,1,1,1,1,0,0,1,0,0,0,1},

{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{1,0,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,0,1},

{1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1},

{1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1},

{1,0,1,1,1,1,0,0,0,0,1,0,0,0,1,1,1,1,0,1},

{1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}};

for(int i = 0; i < 20; i++){ // переписываем матрицу в поле объекта

for(int j = 0; j < 20; j++){

if(matrix[i][j] == 0){

this->matrix[j][i] = 2;

} else { this->matrix[j][i] = matrix[i][j]; }

}

}

Draw(); // рисуем карту

}

void Draw(){ // метод для рисования карты

for(int i = 0; i < 20; i++){

for(int j = 0; j < 20; j++){

if(matrix[i][j] == 1){ // если в матрице встречается 1, то рисуем границу

setfillstyle(1,BLUE);

bar(i\*20,j\*20,i\*20+20,j\*20+20);

} else if(matrix[i][j] == 2){ // если 2, то рисуем «бонус»

setfillstyle(1,14);

fillellipse(i\*20+10,j\*20+10,2,2);

}

}

}

}

}; - класс для хранения карты, на которой происходят все перемещения

class Person{

protected:

int x, y; // координаты объекта

public:

virtual void Draw() = 0; // чистые виртуальные методы

virtual void Dvig() = 0;

}; - абстрактный класс, от которого нельзя создать объект, используется, как основа для персонажа и врагов

class Player:public Person{

private:

int status; // статус анимации (-1 – рот открыт, 1 – рот закрыт)

int score; // набранные очки

public:

Player(){ // конструктор, заполняющий поля начальными значения

this->status = -1;

this->x = 210; // начальные координаты (x и y)

this->y = 210;

this->score = 0; // начальное количество очков - 0

}

void Draw(){ // рисование персонажа

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

setfillstyle(1,13);

if(status == -1){ // если «рот открыт»

pieslice(x,y,25,335,10); // рисуем сектор круга от 25 до 335 градусов

} else if(status == 1){ // если «рот закрыт»

pieslice(x,y,5,355,10); // рисуем сектор круга от 5 до 355 градусов

}

setfillstyle(1,5);

fillellipse(x+2,y-4,3,3); // рисуем глаз

status \*= -1; // меняем статус на противоположный

char buf[255];

outtextxy(5,5,itoa(this->score,buf,10)); // выводим в левый верхний угол

}

void Dvig(){} // пустой метод

void Dvig(int key, Map &M){ // перегруженный метод

switch(key){ // в зависимости от пришедшей в метод клавиши

case 72: if(getpixel(x,y-20) == 14){ this->score++; M.setM(x,y); } // проверка на «бонус», если он был в той стороне, куда собирается персонаж, то прибавляем очки

if(getpixel(x,y-20) != 1){ // если не препятствие

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10); // закрашиваем предыдущие координаты

this->y -= 20; } // перемещаем кооржинаты

break;

case 77: if(getpixel(x+20,y) == 14){ this->score++; M.setM(x,y); }

if(getpixel(x+20,y) != 1){

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

this->x += 20; } break;

case 80: if(getpixel(x,y+20) == 14){ this->score++; M.setM(x,y); }

if(getpixel(x,y+20) != 1){

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

this->y += 20; } break;

case 75: if(getpixel(x-20,y) == 14){ this->score++; M.setM(x,y); }

if(getpixel(x-20,y) != 1){

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

this->x -= 20; } break;

default: break;

}

}

}; - класс для персонажа игры

class War:public Person{

protected:

int color; // цвет врага

public:

War(int x, int y, int color){ // конструктор, заполняющий поля начальными значениями

this->x = x;

this->y = y;

this->color = color;

Draw();

}

void Draw(){ // рисование

setfillstyle(1,color);

bar(x-9,y-9,x+9,y+9); // враг – квадрат

setfillstyle(1,15);

fillellipse(x-4,y,2,2); // с двумя глазами

fillellipse(x+4,y,3,3);

}

void Dvig(){} // пустой метод

void Dvig(int n, Map &M){ // перегруженный метод

switch(n){

case 0:{ if(getpixel(x-2,y-20) == 13){ // если в направлении движения был игрок, то игра окончена

closegraph();

printf("Game Over\n");

system("PAUSE");

exit(0);

}

if(getpixel(x,y-20) != 1){ // если не препятствие, то передвигаемся

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

if(M.getM(x,y) == 2){ // если в направлении движения был бонус, то запоминаем это

setfillstyle(1,14);

fillellipse(x,y,2,2);

}

this->y -= 20; // перемещаемся

} } break;

case 1:{ if(getpixel(x-2+20,y) == 13){

closegraph();

printf("Game Over\n");

system("PAUSE");

exit(0);

}

if(getpixel(x+20,y) != 1){

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

if(M.getM(x,y) == 2){

setfillstyle(1,14);

fillellipse(x,y,2,2);

}

this->x += 20;

} } break;

case 2:{ if(getpixel(x-2,y+20) == 13){

closegraph();

printf("Game Over\n");

system("PAUSE");

exit(0);

}

if(getpixel(x,y+20) != 1){

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

if(M.getM(x,y) == 2){

setfillstyle(1,14);

fillellipse(x,y,2,2);

}

this->y += 20;

} } break;

case 3:{ if(getpixel(x-2-20,y) == 13){

closegraph();

printf("Game Over\n");

system("PAUSE");

exit(0);

}

if(getpixel(x-20,y) != 1){

setfillstyle(1,0);

bar(x-10,y-10,x+10,y+10);

if(M.getM(x,y) == 2){

setfillstyle(1,14);

fillellipse(x,y,2,2);

}

this->x -= 20;

} } break;

default: break;

}

Draw();

}

}; - класс для вражеских персонажей

1. **Вывод**

В курсовой работе реализована игра PACMAN с использованием графической библиотеки <graphics.h>. Использовались такие методы ООП, как наследование, полиморфизм, абстрактные классы с чистыми виртуальными методами, передача объекта в функцию, объект класса является полем другого объекта.