

**Федеральное государственное бюджетное образовательное  
учреждение «Сибирский государственный университет  
телекоммуникаций и информатики»**

**кафедра ПМ и К**

**КУРСОВАЯ РАБОТА**

**Тема: «Круговое движение с вращением произвольного составного  
графического объекта»**

Выполнил:

студент 2 курса факультета

ИВТ группы ИП-712

Алексеев С.В.

Проверил:

ассистент кафедры ПМ и К

Суходоева Н.Н.

Новосибирск – 2018

Федеральное государственное бюджетное образовательное  
учреждение «Сибирский государственный университет  
телекоммуникаций и информатики»

кафедра ПМ и К

**КУРСОВАЯ РАБОТА**

Тема: «Круговое движение с вращением произвольного составного  
графического объекта»

Выполнил:

студент 2 курса факультета

ИВТ группы ИП-712

Алексеев С.В.

Проверил:

ассистент кафедры ПМ и К

Суходоева Н.Н.

25.12.18 *хорошо*  
*Ан*

Новосибирск – 2018

## **Оглавление**

Постановка задачи.....	4
Технологии ООП.....	4
Структура классов.....	5
Алгоритм и результат работы.....	5
Заключение.....	6
Приложение. Листинг.....	6

## Постановка задачи

Необходимо написать программу, используя объектно-ориентированный подход, которая будет реализовывать круговое движение с вращением произвольного составного графического объекта.

Программа написана при помощи графической библиотеки *Graphics.h*, функциями библиотеки реализованы все графические объекты и функции движения.

За основу составного графического объекта были взяты простые фигуры:

1. Окружность
2. Прямоугольник
3. Треугольник

## Технологии ООП

В курсовой работе были применены такие принципы объектно-ориентированного программирования, как:

1. Инкапсуляция
2. Полиморфизм
3. Наследование

Из технологий ООП были применены:

1. Чистые виртуальные функции
2. Абстрактный класс
3. Конструктор, перегрузка конструкторов
4. Списки инициализации
5. Массив указателей на объекты

## Структура классов

В программе реализован абстрактный класс (*shape*), который содержит в себе виртуальные и чистые виртуальные функции, для возвращения и записи координат, рисования фигур. Так же в нем есть защищенные поля, хранящие координаты фигур.

Shape является родительским для классов circleCls, triangleCls, triangLCls, triangRCls, rectangleCls, которые реализуют инициализацию полей с помощью конструкторов. Файл methods.cpp содержит в себе реализацию методов классов. В нём объявлены и реализованы методы установки и возвращения координат, методы рисования и движения фигур. В файле constants.h описаны константы для рисования фигур.

## Алгоритм и результат работы

Инициализируется окно. Создаются пять объектов фигур с заданными координатами и цветом. Создаётся массив указателей на объекты типа shape. Для каждого объекта поочерёдно вызываются функции рисования и движения(изменения его координат). В основе функций рисования стандартные функции из библиотеки graphics.h.

## Заключение

Были изучены и реализованы принципы ООП: полиморфизм, наследование, инкапсуляция. Полиморфизм заключается в возможности использования разных вариаций класса `shape`, его поля и методы используются по-разному при создании объектов. Наследование реализовано при использовании классами фигур класса `shape`. Одни и те же методы необходимо было переопределить и не было необходимости прописывать поля координат в каждом производном классе. Инкапсуляция заключалась в сокрытии полей координат от внешнего воздействия при помощи модификатора доступа `protected`. При этом доступ к ним был реализован с помощью функций, наследующих от данного класса с соответствующим модификатором `protected`.

## Приложение. Листинг

```
Main.cpp
#include "constants.h"
#include "classes.h"
#include <stdio.h>
#include <stdlib.h>
#include <ctime>
#include <iostream>
#include <graphics.h>

main(){
```

```

srand(time(NULL));
initwindow(750, 750);

circleCls circleObj( 220 ,150 ,15 );
circleCls *Circle = &circleObj;

rectangleCls rectanObj( 200, 110,11);
rectangleCls *Rectangle = &rectanObj;

triangleCls trianObj( 220, 49, 4);
triangleCls *Triangle = &trianObj;

triangLCls trianLObj( 241, 160, 1);
triangLCls *LTriangle = &trianLObj;

trianRCls trianRObj(199, 160, 1);
trianRCls *RTriangle = &trianRObj;

shape **array = new shape*[ARR_SIZE];

        array[0] = Rectangle;
        array[1] = Circle;
        array[2] = Triangle;
        array[3] = LTriangle;
        array[4] = RTriangle;
while(true){
    cleardevice();
    for(int i = 0; i < ARR_SIZE; i++){
        array[i] -> draw();
    }
}

```

```

        array[i] -> move();
    }
    delay(10);
}
closegraph();
}

```

## Methods.cpp

```

#include "classes.h"
#include <math.h>
#include "constants.h"
#define N 0.09

float t = 0;
float r = 0;
float s = 0;
float g = 0;
float k = 0;

float shape::getXpos() const{
    return x;
}

void shape::setXpos(const float x){
    shape::x = x;
}

```



```
float shape::getYpos() const{  
    return y;  
}
```

```
void shape::setYpos(const float y){  
    shape::y = y;  
}
```

```
int shape::getClr() const{  
    return color;  
}
```

```
void shape::setClr(const int color){  
    shape::color = color;  
}
```

```
void circleCls::move(){  
  
    x = x + 15 * cos(t);  
    y = y + 15 * sin(t);  
    t+= N;  
  
}
```

```
void circleCls::draw(){  
    setcolor(getClr());
```

```

        circle(getXpos(),
               getYpos(), CIRCLE_RADIUS);
    }

```

```

void rectangleCls::move(){

    x = x + 15 * cos(r);
    y = y + 15 * sin(r);
    r+= N;
}

```

```

void rectangleCls::draw(){
    setcolor(getClr());
    rectangle(getXpos(), getYpos(),
              getXpos() + RECTAN_SIZE/ 2,
              getYpos() + RECTAN_SIZE );
}

```

```

void triangleCls::move(){

    x = x + 15 * cos(s);
    y = y + 15 * sin(s);
    s+= N;
}

```

```

void triangleCls::draw(){
    setcolor(color);
    line(x, y, x + 17, y + 60);
    line(x, y, x - 17, y + 60);
}

```

```

        line(x - 19, y + 60,
              x + 19, y + 60);
    }

```

```

void triangLCls::move() {

```

```

    x = x + 15 * cos(k);
    y = y + 15 * sin(k);
    k += N;
}

```

```

void triangLCls::draw() {

```

```

    setcolor(color);
    line(x, y, x + 17, y + 60);
    line(x, y, x, y + 60);
    line(x - 1, y + 60,
          x + 18, y + 60);
}

```

```

void trianRCls::move() {

```

```

    x = x + 15 * cos(g);
    y = y + 15 * sin(g);
    g += N;
}

```

```

void trianRCls::draw() {

```

```

    setcolor(color);
    line(x, y, x-17, y + 60);
    line(x, y, x, y + 60);

```

```
        line(x - 18, y + 60,  
            x + 1, y + 60);  
    }
```

Classes.h

```
#ifndef CLASS_H  
#define CLASS_H
```

```
#include <stdio.h>  
#include <ctime>  
#include <stdlib.h>  
#include <iostream>  
#include <graphics.h>  
#include "constants.h"
```

```
class shape  
{  
public:  
    float getXpos() const;  
    void setXpos(const float x);  
  
    float getYpos() const;  
    void setYpos(const float y);  
  
    int getClr() const;  
    void setClr(const int color);
```

```
virtual void move() = 0;  
    virtual void draw() = 0;
```

```
shape() {}
```

```
protected:
```

```
    float x;  
    float y;  
    int color;  
};
```

```
class circleCls : public shape
```

```
{
```

```
public:
```

```
    void move();  
    void draw();
```

```
    circleCls(float x, float y, int color):shape(){
```

```
        setXpos(x);  
        setYpos(y);  
        setClr(color);
```

```
    }
```

```
};
```

```
class triangleCls : public shape
```

```
{
```

```
public:
```

```

        void move();
void draw();

        triangleCls(float x, float y, int color):shape(){
            setXpos(x);
            setYpos(y);
            setClr(color);
        }
};

```

```

class triangLCls : public shape
{
public:
    void move();
    void draw();

```

```

        triangLCls(float x, float y, int color):shape(){
            setXpos(x);
            setYpos(y);
            setClr(color);
        }
};

```

```

class trianRCls : public shape
{
public:
    void move();
    void draw();

```

```

        trianRCls(float x, float y, int color):shape(){

```

```

        setXpos(x);
        setYpos(y);
        setClr(color);
    }
};

class rectangleCls : public shape
{
public:
    void move();
    void draw();
    int pl;

    rectangleCls(float x, float y, int color):shape(){
        setXpos(x);
        setYpos(y);
        setClr(color);
    }
    rectangleCls(void) : pl(300){
    }
};

#endif

```

Constants.h

```

#ifndef CONST_H
#define CONST_H

```

```
#define CIRCLE_RADIUS 12
```

```
#define RECTAN_SIZE 75
```

```
#define TRIAN_SIZE 40
```

```
#define ARR_SIZE 5
```

```
#endif
```