МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №16

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Классы»

Работу выполнил

студент гр. 4941 Н.С. Горбунов

Санкт-Петербург

2020

**Цель лабораторной работы:** изучение структуры, свойств и видов объектов; изучение способов доступа к полям и правил вызова методов объектов; получение навыков объектно-ориентированного программирования на языке C/C++.

**Задание на программирование:** используя технологию объектно-ориентированного программирования разработать два варианта программы, реализующей движущийся графический объект в соответствии с индивидуальным заданием:

- с использованием статического объекта;

- с использованием динамического объекта.

**Порядок выполнения работы:**

1) Получить у преподавателя индивидуальное задание.

2) Разработать иерархию и структуру объектов, связанных на принципах наследования, в соответствии с индивидуальным заданием.

3) Описать типы объектов и методы обработки их полей.

4) Составить две программы на языке C/C++, реализующие движение графического объекта по заданной траектории: в виде динамического объекта и в виде статического объекта описанного типа.

5) Проверить и продемонстрировать преподавателю работу программ.

6) Оформить отчет о лабораторной работе в составе: постановка задачи, математическая модель, текст программы, контрольные примеры (скриншоты).

**Вариант 13**

Движение прямоугольника по синусоиде по центру экрана слева направо и обратно с изменением цвета при изменении направления движения

***Математическая модель***

Отрисовывается вертикальная линия в центре окна и двигается по вертикали.

Используется технология объектно-ориентированного программирования и разрабатываются два варианта программы, реализующей движущийся графический объект в соответствии с индивидуальным заданием:

- с использованием статического объекта;

- с использованием динамического объекта.

Работа выполнена с использованием средств графической библиотеки «windows.h»

***Специализация функций***

class graph – родительский класс класса world

class world – производный класс graph для создания окна

class rectangle– класс для прорисовки прямоугольника

Статические объекты

//Движение прямоугольника по синусоиде по центру экрана слева направо и обратно с изменением цвета при изменении направления движения

//Статические объекты

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include <conio.h>

#include <windows.h>

class graph

{

protected:

HWND hwnd;

public:

HDC hdc;

graph()

{

hwnd = GetForegroundWindow();

hdc = GetDC(hwnd);

}

};

//

class world :public graph

{

POINT\* arr;

int x0;

int y0;

public:

int n;

HPEN pBlack;

HPEN pYellow;

HPEN pRed;

HBRUSH bGreen;

HBRUSH bBlue;

HBRUSH bBlack;

public:

world(int x0, int y0)

{

this->x0 = x0;

this->y0 = y0;

pBlack = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

pYellow = CreatePen(PS\_SOLID, 1, RGB(255, 255, 0));

pRed = CreatePen(PS\_SOLID, 1, RGB(255, 0, 0));

bGreen = CreateSolidBrush(RGB(0, 255, 0));

bBlue = CreateSolidBrush(RGB(0, 0, 255));

bBlack = CreateSolidBrush(RGB(0, 0, 0));

int h = 80; //амплитуда синусоиды траектории движения

double da = 2.14; //величина шага смещения по оси x

double a = 0; //значение координаты x в точке старта

n = 300;

arr = new POINT[n];

for (int i = 0; i < n; i++)

{

arr[i].x = (int)a;

arr[i].y = y0 - (int)(h \* sin(a / 180 \* M\_PI));

a += da;

}

}

~world()

{

delete[] arr;

}

void clear()

{

SelectObject(hdc, bBlack);

SelectObject(hdc, pBlack);

Rectangle(hdc, 0, 0, x0 \* 2, y0 \* 2);

}

void path()

{

SelectObject(hdc, pYellow);

Polyline(hdc, arr, n);

}

POINT getPoint(int n)

{

POINT p = { arr[n].x, arr[n].y };

return p;

}

};

//

class rectangle

{

int step, ds;

int a, b;

HBRUSH firstGround;

HBRUSH secondGround;

bool flag; //флаг направления

public:

rectangle(int a, int b, HBRUSH Ground1, HBRUSH Ground2)

{

this->a = a; //размеры сторон прямоугольника

this->b = b;

step = 0;

ds = 4;

firstGround = Ground1; // фоны прямоугольника

secondGround = Ground2;

flag = true;

}

void show(world\* wrd)

{

POINT p = wrd->getPoint(step);

if (flag)

{

SelectObject(wrd->hdc, firstGround);

}

else

{

SelectObject(wrd->hdc, secondGround);

}

SelectObject(wrd->hdc, wrd->pRed);

Rectangle(wrd->hdc, p.x - a, p.y - b, p.x + a, p.y + b);

}

void go(world\* wrd)

{

step += ds;

if (step >= wrd->n || step <= 0)

{

ds = -ds;

flag = !flag;

}

}

HBRUSH changeColor(world\* wrd)

{

}

};

//

int main()

{

world\* wrd = new world(320, 150);

rectangle rc(50, 30, wrd->bGreen, wrd->bBlue); //создается статический объект

do

{

wrd->clear();

wrd->path();

rc.show(wrd);

Sleep(150);

rc.go(wrd);

} while (!\_kbhit());

delete wrd; //удаление объекта wrd из динамической памяти

return 0;

}

Динамические объекты

//Движение прямоугольника по синусоиде по центру экрана слева направо и обратно с изменением цвета при изменении направления движения

//Динамические объекты

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include <conio.h>

#include <windows.h>

class graph

{

protected:

HWND hwnd;

public:

HDC hdc;

graph()

{

hwnd = GetForegroundWindow();

hdc = GetDC(hwnd);

}

};

//

class world :public graph

{

POINT\* arr;

int x0;

int y0;

public:

int n;

HPEN pBlack;

HPEN pYellow;

HPEN pRed;

HBRUSH bGreen;

HBRUSH bBlue;

HBRUSH bBlack;

public:

world(int x0, int y0) //конструктор

{

this->x0 = x0;

this->y0 = y0;

pBlack = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

pYellow = CreatePen(PS\_SOLID, 1, RGB(255, 255, 0));

pRed = CreatePen(PS\_SOLID, 1, RGB(255, 0, 0));

bGreen = CreateSolidBrush(RGB(0, 255, 0));

bBlue = CreateSolidBrush(RGB(0, 0, 255));

bBlack = CreateSolidBrush(RGB(0, 0, 0));

int h = 80; //амплитуда синусоиды траектории движения

double da = 2.14;

double a = 0;

n = 300;

arr = new POINT[n];

for (int i = 0; i < n; i++)

{

arr[i].x = (int)a;

arr[i].y = y0 - (int)(h \* sin(a / 180 \* M\_PI));

a += da;

}

}

~world() //деструктор

{

delete[] arr;

}

void clear()

{

SelectObject(hdc, bBlack);

SelectObject(hdc, pBlack);

Rectangle(hdc, 0, 0, x0 \* 2, y0 \* 2);

}

void path()

{

SelectObject(hdc, pYellow);

Polyline(hdc, arr, n);

}

POINT getPoint(int n)

{

POINT p = { arr[n].x, arr[n].y };

return p;

}

};

//

class rectangle

{

int step, ds;

int a, b;

HBRUSH firstGround;

HBRUSH secondGround;

bool flag; //флаг направления

public:

rectangle(int a, int b, HBRUSH Ground1, HBRUSH Ground2) //конструктор

{

this->a = a;

this->b = b;

step = 0;

ds = 4;

firstGround = Ground1; // фоны прямоугольника

secondGround = Ground2;

flag = true;

}

void show(world\* wrd)

{

POINT p = wrd->getPoint(step);

if (flag)

{

SelectObject(wrd->hdc, firstGround);

}

else

{

SelectObject(wrd->hdc, secondGround);

}

SelectObject(wrd->hdc, wrd->pRed);

Rectangle(wrd->hdc, p.x - a, p.y - b, p.x + a, p.y + b);

}

void go(world\* wrd)

{

step += ds;

if (step >= wrd->n || step <= 0)

{

ds = -ds;

flag = !flag;

}

}

};

//

int main()

{

world\* wrd = new world(320, 150);

rectangle\* rc = new rectangle(50, 30, wrd->bGreen, wrd->bBlue); /\*создание объектов в динамической

памяти\*/

do

{

wrd->clear();

wrd->path();

rc->show(wrd);

Sleep(150);

rc->go(wrd);

} while (!\_kbhit()); //пока не нажата любая клавиша

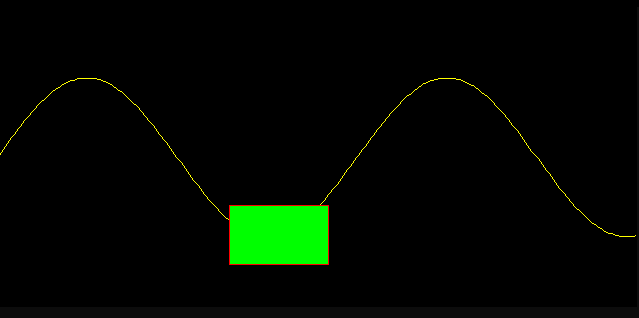
delete wrd; //удаление объекта wrd из динамической памяти

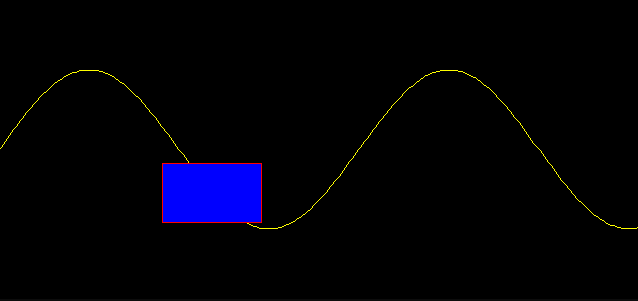
delete rc; //удаление объекта rc из динамической памяти

return 0;

}

**Скриншоты работы программы:**





**Вывод:** используя технологию объектно-ориентированного программирования разработал два варианта программы, реализующей движущийся графический объект в соответствии с индивидуальным заданием:

- с использованием статического объекта;

- с использованием динамического объекта.