Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

О Т Ч Е Т

по домашней работе № 3

Название: Программирование на C++ с использованием классов

Дисциплина: Объектно-ориентированное программирование

Студент ИУ6-25 Б — (Группа) — (Подпись, дата) Дуленина И.Н. — (И.О. Фамилия) — 20.03.2023

Преподаватель — (Подпись, дата) Веселовская О.А. — (И.О. Фамилия) — 20.03.2023

Москва, 2023

Вариант 8

Часть 3.1. Композиция

Задание: Разработать и реализовать диаграмму классов для описанных объектов предметной области, используя механизм композиции. Протестировать все методы каждого класса. Все поля классов должны быть скрытыми (private) или защищенными (protected). Методы не должны содержать операций ввода/вывода, за исключением процедуры, единственной задачей которой является вывод информации об объекте на экран.

Объект – крепостная башня. Поля: название, высота, признак "проездная". Методы: процедура инициализации, процедура вывода информации о башне на экран и функции, возвращающие значения полей по запросу.

Объект – крепость. Параметры: название, количество башен и сами башни. Методы объекта должны позволять: инициализировать объект, выводить на экран информацию об объекте, определять количество проездных башен в крепости, получать название самой высокой башни.

В отчете привести диаграмму разработанных классов и объектную декомпозицию.

Код программы:

```
#include <locale.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
#include<string>
#include <conio.h>
using namespace std;
const int kol = 15;
class Bash
{
protected:
    char str[10];
    int height;
    bool pr;
public:
    Bash() {}
    Bash(char s[10], int h, bool p) { strcpy_s(str, s); height = h; pr = p;}
    ~Bash() {}
    void print(void)
    { cout << "Название: " << str << "\nВысота: "<<height<<"\nПроездная: "<<pr<<"\n"; }
    void Init(char s[10], int h, bool p) { strcpy_s(str, s); height = h; pr = p; }
    char* Name() {return str;}
    int High() { return height; }
    bool Proezd() { return pr; }
};
class Krep
{
private:
    char name[22]="КРЕПОСТЬ";
    int  size;
    Bash mas[kol];
public:
    Krep() {}
    Krep(int af, Bash m1[kol])
    {
        setmas(af, m1);
    }
    ~Krep() {}
    void  printm();
    void setmas(int af, Bash m1[]);
    int kol();
```

```
    string thehighest();
};
void Krep::setmas(int af, Bash m1[])
{
    int i;
    size = af;
    for (i = 0; i < size; i++)   mas[i].Init(m1[i].Name(), m1[i].High(), m1[i].Proezd());
}
void Krep::printm()
{
    int  i; char h[10];
    cout<<"----------\nСодержимое объекта:"<<name<<"\n";
    for (i = 0; i < size; i++) {
        cout << i+1 << ":";  mas[i].print();
    }
    cout << "Количество проездных башен: " << kol();
    cout << "\nНазвание самой высокой башни: " << thehighest();
}
int Krep::kol() {
    int k=0;
    for (int i = 0; i < size; i++) {
        if (mas[i].Proezd() == 1) k += 1;
    }
    return k;
}
string Krep::thehighest() {
    int max = 0; char name[10];
    for (int i = 0; i < size; i++) {
        if (mas[i].High() > max) {
            max = mas[i].High();
            strcpy_s(name, mas[i].Name());
        }
    }
    return name;
}
int main()
{
    setlocale(0, "russian");
    int numb; char name[10]; int h; bool p;
    cout << "Введите число башен: \n";
    cin >> numb;
    Bash mn[kol]; Bash A;
    for (int i = 0; i < numb; i++) {
        cout << "the " << i+1 << " tower:\nname: ";
        cin.ignore();
        //cin >> name;
        gets_s(name, 10);
        //cin.ignore();
        cout << "the height:"; cin >> h;
        cout << "proezd (0/1) ? "; cin >> p;
        A.Init(name, h, p);
        mn[i] = A;
    }
    Krep a(numb, mn);
    a.printm();
    return 0;
}
```

```
3
the 1 tower:
name: er
the height:4
proezd (0/1) ? 1
the 2 tower:
name: etr
the height:0
proezd (0/1) ? 0
the 3 tower:
name: erty
the height:10
proezd (0/1) ? 1
----------
Содержимое объекта
1:Название: er
Высота: 4
Проездная: 1
2:Название: etr
Высота: 0
Проездная: 0
3:Название: erty
Высота: 10
Проездная: 1
Количество проездных башен: 2
Название самой высокой башни: erty
```

Рисунок 1 – работающая версия программы

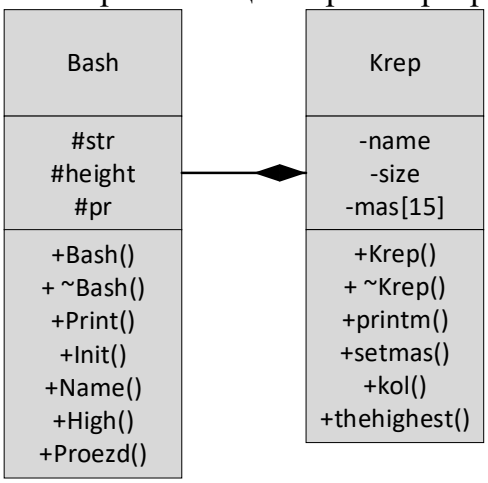

Рисунок 2 – диаграмма классов

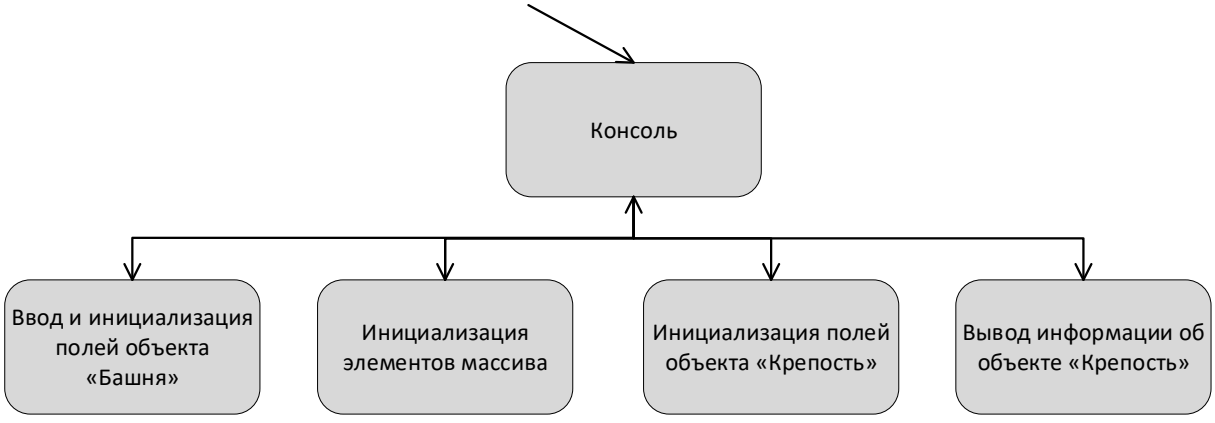

Рисунок 3 – объектная декомпозиция

Вывод: мы познакомились с таким видом отношений между классами как композиция на языке C++.

Часть 3.2. Qt. Полиморфное наследование

Задание: Разработать программу, содержащую описание трех графических объектов:

квадрат, ромб, два одинаковых взаимно ортогональных ромба с общим центром.

Реализуя механизм полиморфизма, привести объекты в горизонтальное движение по экрану с различными скоростями с отражением от границ экрана.

В отчете привести диаграмму используемых классов Qt и разработанных классов, граф состояний пользовательского интерфейса и объектную декомпозицию.

Код модуля main.cpp:

```
#include "widget.h"

#include <QApplication>

#include <QLocale>
#include <QTranslator>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "part2_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    Window w;
    w.show();
    return a.exec();
}
```

Код модуля area.h:

```
#ifndef AREA_H
#define AREA_H
#include <QWidget>
#include "figura.h"
class Area : public QWidget
{
    int myTimer; // идентификатор таймера
    int width;
    float alpha; // "таймер"
```

```
public:
   Area(QWidget *parent = 0);
   ~Area();
   MyRomb *myromb;
   MySquare *mysquare;
   MyRRomb *myrromb;
protected:
   //      обработчики событий
   void paintEvent(QPaintEvent *event);
   void timerEvent(QTimerEvent *event);
   void showEvent(QShowEvent *event);
   void hideEvent(QHideEvent *event);
};

#endif // AREA_H
```

Код модуля area.cpp:

```
#include "area.h"
Area::Area(QWidget *parent):QWidget(parent)
{
   width=300;
   setFixedSize(QSize(width,500));
   mysquare=new MySquare(100,60,50);
   myromb = new MyRomb(150,197,40);
   myrromb=new MyRRomb(200, 350, 100);
   alpha=0;
}
void Area::showEvent(QShowEvent *)
{
   myTimer=startTimer(50);          // создать таймер
}
void Area::paintEvent(QPaintEvent *)
{
   QPainter painter(this);
   painter.setPen(Qt::red);
   double v1=15; double v2=9; double v3=7.5;//скорости
   mysquare->move(alpha*v1,&painter, v1, width);
   myromb->move(alpha*v2,&painter, v2, width);
   myrromb->move(alpha*v3, &painter, v3, width);
}
void Area::timerEvent(QTimerEvent *event)
{
   if (event->timerId() == myTimer) // если наш таймер
   {
      alpha=alpha+0.5;
      update();                  // обновить внешний вид
   }
```

```
    else
        QWidget::timerEvent(event);   // иначе передать для стандартной
            // обработки
}
void Area::hideEvent(QHideEvent *)
{
    killTimer(myTimer);             // уничтожить таймер
}
Area::~Area()
{
    delete mysquare;
    delete myromb;
    delete myrromb;
}
```

Код модуля figure.h:

```
#ifndef FIGURA_H
#define FIGURA_H

#include <QtGui>
class Figura
{
protected:
    int x,y,halflen,dx,f, k;float t2, tt;
    virtual void draw(QPainter *Painter)=0;
public:
    Figura(int X,int Y,int Halflen):
        x(X),y(Y),halflen(Halflen){t2=0; tt=0; f=1; k=1;}
    void move(float Alpha,QPainter *Painter, double n, int width);
    virtual ~Figura(){};
};
class MyRomb:public Figura
{
protected:
    void draw(QPainter *Painter) override;
public:
    MyRomb(int x,int y,int halflen):Figura(x,y,halflen){}
};
class MySquare:public Figura
{
protected:
    void draw(QPainter *Painter) override;
public:
    MySquare(int x,int y,int halflen):Figura(x,y,halflen){}
};
class MyRRomb:public Figura
```

```
{
protected:
   void draw(QPainter *Painter) override;
public:
   MyRRomb(int x,int y,int halflen):Figura(x,y,halflen){}
};
#endif // FIGURA_H
```

Код модуля figure.cpp:

```
#include "figura.h"
#include <math.h>
void Figura::move(float Alpha,QPainter *Painter, double n, int width)//int t, int n
{
   if (((x+dx+halflen>width-0.5*n) && (f==1)) ||((x+dx-halflen<0.5*n) && (f==-
1))){ //0.5*10*3(n)
      if (k==1){t2=Alpha-t2; k=2;}
      if (f==1) tt=t2;
      f=-f;}
   if (f==1)tt=tt+0.5*n;
   else tt=tt-0.5*n;
   dx=round(tt);
   draw(Painter);
}
void MyRomb::draw(QPainter *Painter)
{
   Painter->drawLine(x+dx+halflen, y,x+dx,y+2*halflen);
   Painter->drawLine(x+dx,y+2*halflen, x+dx-halflen,y);
   Painter->drawLine(x+dx-halflen,y, x+dx,y-2*halflen);
   Painter->drawLine(x+dx,y-2*halflen, x+dx+halflen,y);
}
void MySquare::draw(QPainter *Painter)
{
   Painter->drawLine(x+dx+halflen, y+halflen,x+dx-halflen,y+halflen);
   Painter->drawLine(x+dx-halflen,y+halflen,x+dx-halflen,y-halflen);
   Painter->drawLine(x+dx-halflen,y-halflen,x+dx+halflen,y-halflen);
   Painter->drawLine(x+dx+halflen,y-halflen,x+dx+halflen,y+halflen);
}
void MyRRomb::draw(QPainter*Painter){
   int a, b;
   a=halflen/3;
   Painter->drawLine(x+dx+a, y,x+dx,y+2*a);
   Painter->drawLine(x+dx,y+2*a,x+dx-a,y);
   Painter->drawLine(x+dx-a,y,x+dx,y-2*a);
   Painter->drawLine(x+dx,y-2*a,x+dx+a,y);
   b=halflen/2;
   Painter->drawLine(x+dx+halflen, y, x+dx,y+b);
```

```
    Painter->drawLine(x+dx,y+b, x+dx-halflen,y);
    Painter->drawLine(x+dx-halflen,y, x+dx,y-b);
    Painter->drawLine(x+dx,y-b, x+dx+halflen,y);
}
```

Код модуля widget.h:

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include <QtGui>
#include <QPushButton>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include "area.h"
class Window : public QWidget
{
protected:
    Area * area;        // область отображения рисунка
    QPushButton * btn;
public:
    Window();
};

#endif // WIDGET_H
```

Код модуля widget.cpp:

```
#include "widget.h"
Window::Window()
{
    this->setWindowTitle("Обработка событий");
    area = new Area( this );
    btn = new QPushButton("Завершить",this );
    QVBoxLayout *layout = new QVBoxLayout(this);
    layout->addWidget(area);
    layout->addWidget(btn);
    connect(btn, SIGNAL(clicked(bool)),this,SLOT(close()));
};
```

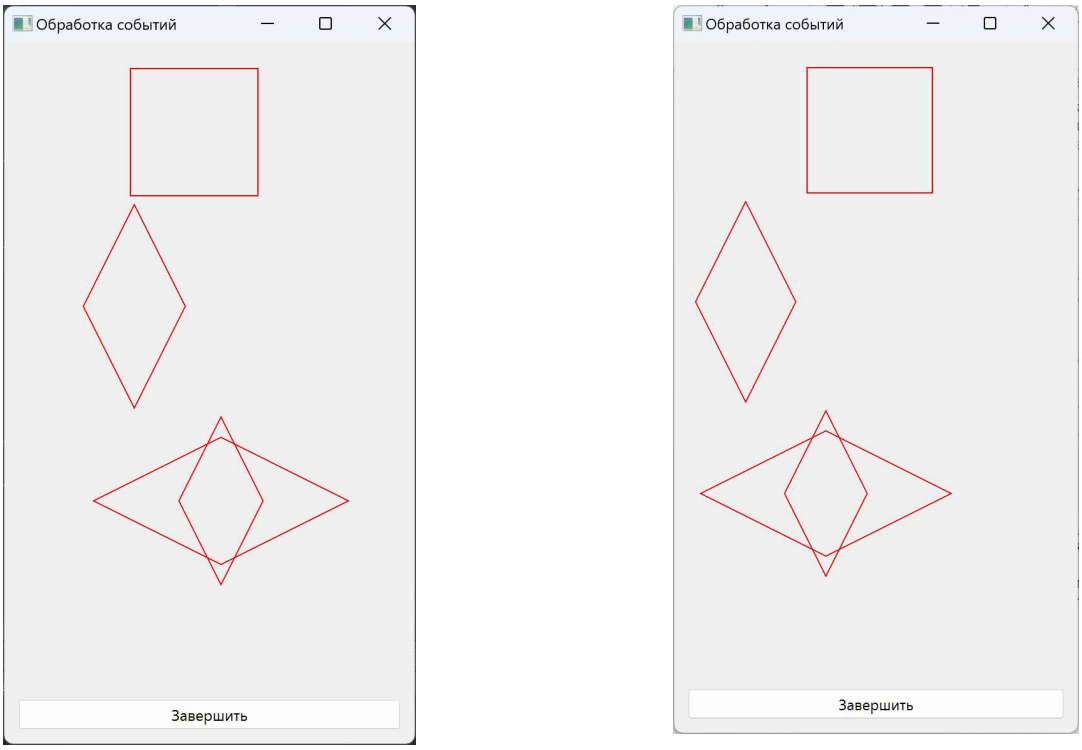

Рисунок 1 – работающая версия программы

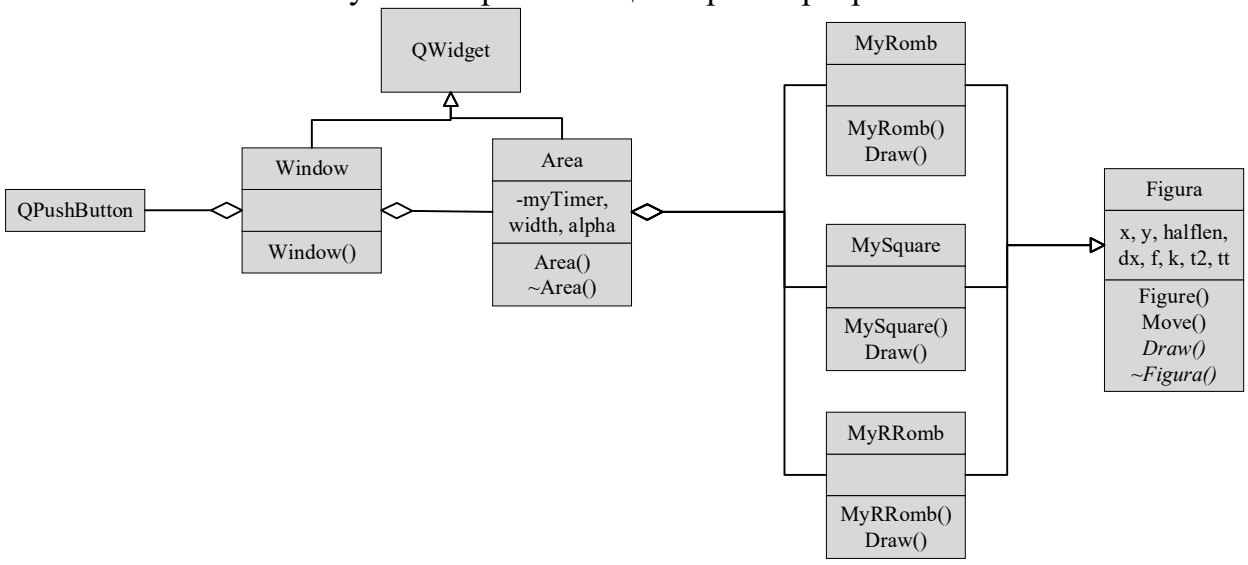

Рисунок 2 – диаграмма классов

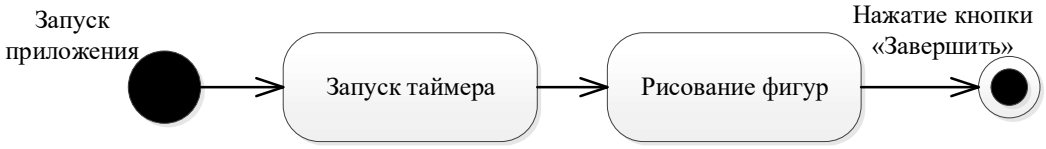

Рисунок 3 – диаграмма состояний интерфейса

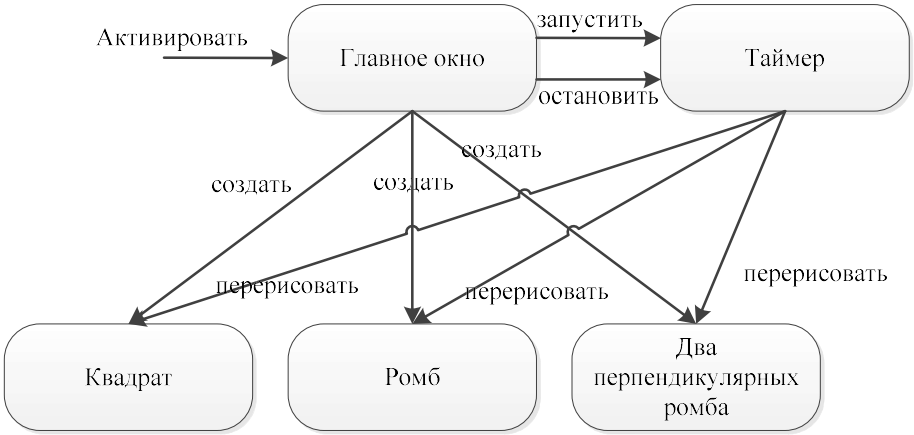

Рисунок 4 – объектная декомпозиция

Вывод: мы научились создавать при помощи среды QtCreator несколько движущихся по определенной траектории фигур, отличающихся друг от друга скоростью движения.