Міністерство освіти і науки України

Одеський національний університет імені І. І. Мечникова

ФМФІТ

**КУРСОВА РОБОТА**

**на тему: “Об’єктно орієнтоване програмування. Геометричні фігури”**

Виконано: студентом

І курсу групи №1

Колєсником Олексієм Олексійовичем

Науковий керівник:

Антоненко Олександр Сергійович

Одеса, 2019

**Геометрия**

1. Создать абстрактный класс «геометрическая фигура».
2. Создать классы-наследники: «круг», «треугольник»
3. Создать класс «четырехугольник»
4. Создать классы-наследники «четырехугольника»: «ромб», «трапеция», «прямоугольник».
5. Для всех классов наследников реализовать вычисление площади, периметра, смещения фигур
6. Реализовать возможность вывода информации о фигурах на экран
7. Реализовать возможность добавления новых фигур
8. Реализовать возможность сохранения информации о фигурах в файлы.
9. Абстрактный класс «геометрическая фигура». Код:

class GeometricFigure

{

public:

 virtual double Perimetr() const = 0; // чисто виртуальный класс периметра

 virtual double Square() const = 0;

 virtual void Bias(string direction, int HowMuch) = 0;

};

};

У абстрактного класса есть три чисто виртуальных метода, который в дальнейшем будут переопределены в дочерних классах: периметр, площадь, сдвиг. Метод «сдвиг» имеет два параметра: направление сдвига, на сколько сдвинуть, об этом ниже.

1. **Классы наследники от абстрактного класса «геометрическая фигура»**

а) Круг

У класса «круг» есть три поля: координаты центра и радиус.

private:

 int x; // координаты центра

 int y;// координаты центра

 int R; // радиус

Два конструктора, с параметрами, без параметров.

public:

 Circle() //конструктор без параметров

 {

  x = 0;

  y = 0;

  R = 2;

 }

 Circle(int x, int y, int R) // конструктор с параметрами

 {

  this->x = x;

  this->y = y;

  this->R = R;

 }

Геттеры, для получения информации о полях класса с main().

  int GetX() // геттеры

 {

  return x;

 }

 int GetY()

 {

  return y;

 }

 int GetR()

 {

  return R;

 }

Переопределенный метод для вычисления площади:

 double Square() const override

 {

  return PI \* sqr(R);

 }

И периметра (длины окружности):

double Perimetr() const override

 {

  return 2 \* PI\*R;

 }

Метод сдвига:

 void  Bias(string direction, int HowMuch) override  // смещение фигур

 {

  int dir;

  if (direction == "right") // right - 1, left - 2, up - 3, down - 4

   dir = 1;

  else if (direction == "left")

   dir = 2;

  else if (direction == "up")

   dir = 3;

  else if (direction == "down")

   dir = 4;

  // cout << "DR = " << dir << endl;

  switch (dir)

  {

  case 1:

  {

   x += HowMuch;

   break;

  }

  case 2:

  {

   x -= HowMuch;

   break;

  }

  case 3:

  {

   y += HowMuch;

   break;

  }

  case 4:

  {

   y -= HowMuch;

   break;

  }

  default:

   break;

  }

 }

Описание работы метода сдвига.

Пользователь задаёт одно из четырех направлений, в какую сторону он будет сдвигать геометрическую фигуру: right, left, up, down (строковый параметр direction), и задаёт шаг, на сколько будет сдвинута фигура (целочисленный параметр HowMuch). После этого координаты точек (в данном случае только координаты одной точки — центра, в случае с треугольником, ромбом, трапецией, прямоугольником — координаты всех точек) смещаются в сторону, заданную пользователем. Реализовано через switch.

б) Класс треугольник

В классе треугольник существует 9 полей

private:

 int x1; // координаты первой точки

 int y1;// координаты первой точки

 int x2; // координаты второй точки

 int y2;// координаты второй точки

 int x3; // координаты третьей точки

 int y3;// координаты третьей точки

 double AB; // первая сторона

 double BC; // вторая сторона

 double CD; // третья сторона

6 полей — координаты трех точек

3 поля — три стороны треугольник

Два конструктора — с параметрами и без параметров.

public:

 Triangle() //конструктор без параметров

 {

  x1 = 4;

  y1 = 3;

  x2 = 5;

  y2 = 1;

  x3 = 1;

  y3 = 2;

  AB = sqrt(sqr(x2 - x1) + sqr(y2 - y1)); // три стороны

  BC = sqrt(sqr(x3 - x2) + sqr(y3 - y2));

  CD = sqrt(sqr(x3 - x1) + sqr(y3 - y1));

  int a;

  if ((AB + BC) > CD && (AB + CD) > BC && (CD + BC) > AB)

   a = 1;

  else

   throw exception("Введенная фигура не является треугольником");

 }

Triangle(int x1, int y1, int x2, int y2, int x3, int y3) // конструктор с параметрами

 {

  this->x1 = x1;

  this->y1 = y1;

  this->x2 = x2;

  this->y2 = y2;

  this->x3 = x3;

  this->y3 = y3;

  AB = sqrt(sqr(x2 - x1) + sqr(y2 - y1)); // три стороны

  BC = sqrt(sqr(x3 - x2) + sqr(y3 - y2));

  CD = sqrt(sqr(x3 - x1) + sqr(y3 - y1));

  int a;

  if ((AB + BC) > CD && (AB + CD) > BC && (CD + BC) > AB)

   a = 1;

  else

   throw exception("Введенная фигура не является треугольником");

 }

В конструкторах происходит проверка на то, является ли созданный объект треугольником. В main() создание экземпляра класса выглядит следующим образом:

int x1; int y1; int x2; int y2; int x3; int y3;

    cout << "Введите координаты трех точек вашего ТРЕУГОЛЬНИКА" << endl;

    cout << "Первой точки (х1, у1): ";

    cin >> x1;

    cin >> y1;

    cout << "Второй точки (х2, у2): ";

    cin >> x2;

    cin >> y2;

    cout << "Третьей точки (х3, у3): ";

    cin >> x3;

    cin >> y3;

    try

    {

     Triangle TMP(x1, y1, x2, y2, x3, y3);

     T.push\_back(TMP);

     break;

    }

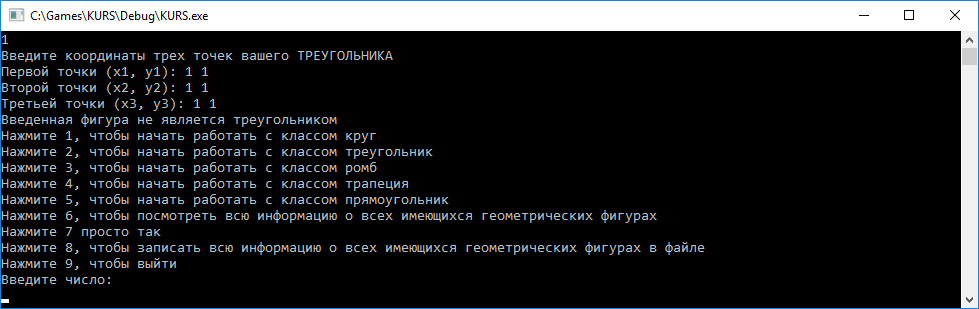
    catch (const exception &c)

    {

     cout << c.what() << endl;

     break;

    }

В случае, если данный объект не является треугольником (сумма двух сторон должна быть больше третьей стороны), то экземпляр класса не создается, и на экран выводится сообщение о том, что введенная фигура не является треугольником. 

Геттеры, для получения информации о полях класса с main().

 int GetX1() // геттеры

 {

  return x1;

 }

 int GetX2()

 {

  return x2;

 }

 int GetX3()

 {

  return x3;

 }

 int GetY1()

 {

  return y1;

 }

 int GetY2()

 {

  return y2;

 }

 int GetY3()

 {

  return y3;

 }

Переопределенный метод для вычисления периметра:

 double Perimetr() const override

 {

  return (AB + BC + CD);

 }

И площади:

 double Square() const override

 {

   double PoluP = Perimetr() / 2;

   return sqrt(PoluP\*(PoluP - AB)\*(PoluP - BC)\*(PoluP - CD));

 }

Метод Bias работает так же, как и в круге. Единственное изменения — кол-во точек.

 void  Bias(string direction, int HowMuch) // смещение фигур

 {

  int dir;

  if (direction == "right") // right - 1, left - 2, up - 3, down - 4

   dir = 1;

  else if (direction == "left")

   dir = 2;

  else if (direction == "up")

   dir = 3;

  else if (direction == "down")

   dir = 4;

  //  cout << "DR = " << dir << endl;

  switch (dir)

  {

  case 1:

  {

   x1 += HowMuch;

   x2 += HowMuch;

   x3 += HowMuch;

   break;

  }

  case 2:

  {

   x1 -= HowMuch;

   x2 -= HowMuch;

   x3 -= HowMuch;

   break;

  }

  case 3:

  {

   y1 += HowMuch;

   y2 += HowMuch;

   y3 += HowMuch;

   break;

  }

  case 4:

  {

   y1 -= HowMuch;

   y2 -= HowMuch;

   y3 -= HowMuch;

   break;

  }

  default:

   break;

  }

 }

3. Класс «Четрырехугольник»

Наследник класса «геометрическая фигура». Имеет 14 полей, что будут наследоваться в дочерних классах.

protected:

 int x1; // координаты первой точки

 int y1;// координаты первой точки

 int x2; // координаты второй точки

 int y2;// координаты второй точки

 int x3; // координаты третьей точки

 int y3;// координаты третьей точки

 int x4; // координаты третьей точки

 int y4;// координаты третьей точки

 double AB; // первая сторона

 double BC; // вторая сторона

 double CD; // третья сторона

 double AD; // четвертая сторона

 double AC; // диагонали

 double BD;

Два конструктора:

 Quadrangle()

 {

  x1 = 5;

  y1 = 7;

  x2 = 3;

  y2 = 3;

  x3 = 5;

  y3 = 2;

  x4 = 7;

  y4 = 3;

  AB = sqrt(sqr(x2 - x1) + sqr(y2 - y1)); // четыре  стороны

  BC = sqrt(sqr(x3 - x2) + sqr(y3 - y2));

  CD = sqrt(sqr(x4 - x3) + sqr(y4 - y3));

  AD = sqrt(sqr(x4 - x1) + sqr(y4 - y1));

   AC = sqrt(sqr(x3 - x1) + sqr(y3 - y1)); // диагонали

   BD = sqrt(sqr(x4 - x2) + sqr(y4 - y2));

 }

 Quadrangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4)

 {

  this->x1 = x1;

  this->y1 = y1;

  this->x2 = x2;

  this->y2 = y2;

  this->x3 = x3;

  this->y3 = y3;

  this->x4 = x4;

  this->y4 = y4;

  AB = sqrt(sqr(x2 - x1) + sqr(y2 - y1)); // четыре  стороны

  BC = sqrt(sqr(x3 - x2) + sqr(y3 - y2));

  CD = sqrt(sqr(x4 - x3) + sqr(y4 - y3));

  AD = sqrt(sqr(x4 - x1) + sqr(y4 - y1));

  AC = sqrt(sqr(x3 - x1) + sqr(y3 - y1)); // диагонали

  BD = sqrt(sqr(x4 - x2) + sqr(y4 - y2));

 }

Геттеры:

public:

 int GetX1() // геттеры

 {

  return x1;

 }

 int GetX2()

 {

  return x2;

 }

 int GetX3()

 {

  return x3;

 }

 int GetX4()

 {

  return x4;

 }

 int GetY1()

 {

  return y1;

 }

 int GetY2()

 {

  return y2;

 }

 int GetY3()

 {

  return y3;

 }

 int GetY4()

 {

  return y4;

 }

Метод смещения фигур:

void  Bias(string direction, int HowMuch) // смещение фигур

 {

  int dir;

  if (direction == "right") // right - 1, left - 2, up - 3, down - 4

   dir = 1;

  else if (direction == "left")

   dir = 2;

  else if (direction == "up")

   dir = 3;

  else if (direction == "down")

   dir = 4;

  //cout << "DR = " << dir << endl;

  switch (dir)

  {

  case 1:

  {

   x1 += HowMuch;

   x2 += HowMuch;

   x3 += HowMuch;

   x4 += HowMuch;

   break;

  }

  case 2:

  {

   x1 -= HowMuch;

   x2 -= HowMuch;

   x3 -= HowMuch;

   x4 -= HowMuch;

   break;

  }

  case 3:

  {

   y1 += HowMuch;

   y2 += HowMuch;

   y3 += HowMuch;

   y4 += HowMuch;

   break;

  }

  case 4:

  {

   y1 -= HowMuch;

   y2 -= HowMuch;

   y3 -= HowMuch;

   y4 -= HowMuch;

   break;

  }

  default:

   break;

  }

 }

4. а Классы-наследники «четырехугольника» - «ромб», «прямоугольник», «трапеция».

Класс ромб наследует конструктор класса «четырехугольник», при этом в конструкторе класса «ромб» проходит проверка на то, является ли заданная фигура ромбом.

 Rhombus(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4): Quadrangle(x1, y1, x2, y2, x3, y3, x4, y4) // конструктор с параметрами

 {

  int a;

  if ((AB == BC) || (BC == CD) || (CD == AD))

   a = 1;

  else

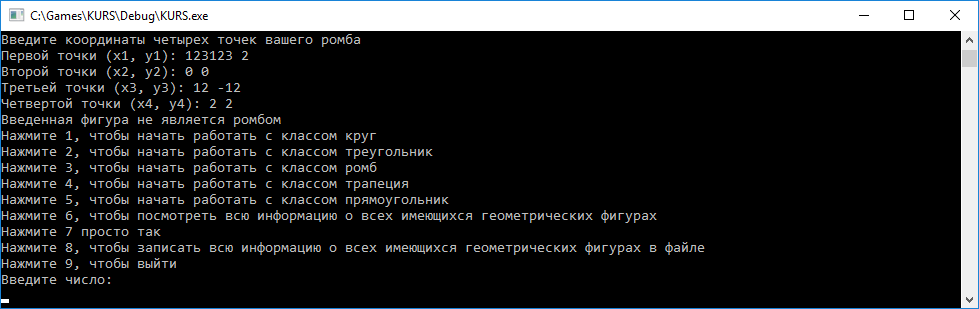
  {

   throw exception("Введенная фигура не является ромбом");

  }

 }

Объект можно создать только в случае, если все стороны фигуры будут равны. В противном случае, объект не создастся, а на экране пользователь будет видеть следующее:



5. а Методы нахождения площади и периметра:

 double Perimetr() const override

 {

  if ((AB == BC == CD == AD))

   return AB \* 4;

  else

   return 0;

 }

 double Square() const override

 {

  double d1 = sqrt(sqr(x3 - x1) + sqr(y3 - y1));

  double d2 = sqrt(sqr(x2 - x4) + sqr(y2 - y4)); // нахождение площади через диагонали

   return (d1 + d2) / 2;

 }

Метод сдвига наследуется у «Четырехугольника».

4. б

Класс «трапеция» отличается от класса «ромб» формулами нахождения периметра и площади, а так же проверкой в конструкторе на то, является ли данная фигура трапецией.

Trapezium(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4): Quadrangle(x1, y1, x2, y2, x3, y3, x4, y4) // конструктор с параметрами

 {

  int a;

  if (((y1 == y2) && (y3 == y4)) && (x1 != x3) && (x2 != x4)) // если две стороны - параллельны, а две другие - нет

   a = 1;

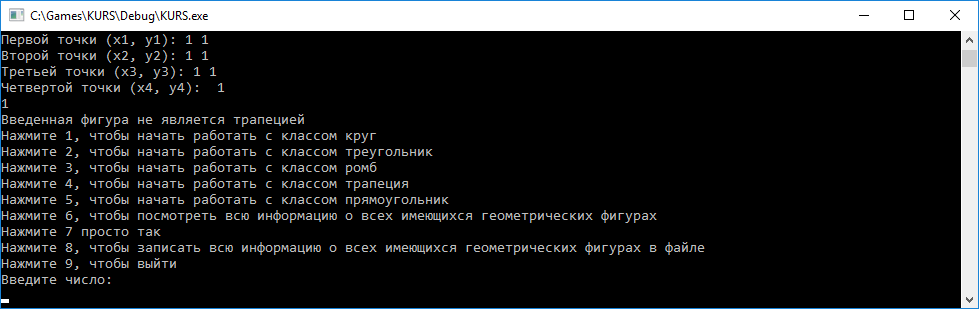
  else

  {

   throw exception("Введенная фигура не является трапецией");

  }

 }



5. б

 double Perimetr() const override

 {

  return (AB + BC + CD + AD);

 }

 double Square() const override

 {

  double a = AB;

  double b = BC;

  double c = CD;

  double d = AD;

  double first = (a + b) / 2;

  double second = sqr(c);

  double third = sqr(b - a);

  double fourth = c \* c - d \* d;

  double chislitel = third + fourth;

  double znamenatel = 2 \* (b - a);

  double fifth = sqr(chislitel / znamenatel);

  double SEC = sqrt(second - fifth);

  double result = first \* SEC;

  return result;

4. в

Класс «прямоугольник» отличается от прочих четырехугольников формулами расчета площади/периметра и проверкой на то, является ли заданная фигура прямоугольником.

 Rectangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) : Quadrangle(x1, y1, x2, y2, x3, y3, x4, y4) // конструктор с параметрами

 {

  int a;

  if ((sqrt(sqr(AD) + sqr(CD)) == AC) && (sqrt(sqr(BC) + sqr(CD)) == BD))

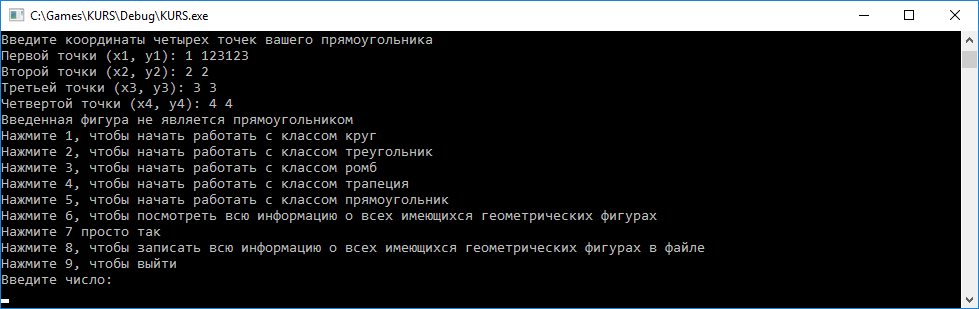
   a = 1;

  else

  {

   throw exception("Введенная фигура не является прямоугольником");

  }

 }

Проверка, является ли фигура прямоугольником проверяется через теорему Пифагора.

5. в

Вычисление площади и периметра:

 double Perimetr() const override

 {

  return (AB + BC + CD + AD);

 }

 double Square() const override

 {

  return AB \* BC;

 }

6, 7, 8.

В main() идет интерактивное взаимодействие с пользователем. Реализованы вектора объектов каждого из классов (за исключением абстрактных), в которые пользователи могут добавлять объекты, с помощью которых они могут высчитывать площадь, периметр (используя методы объектов класса).

Всё это происходит посредством нескольких switch, которые находятся в бесконечном цикле (While (true)). Выход из программы возможен, если пользователь вводит число 9 в одном из интерактивных диалогов.

Пример работы с классом «круг»

int FirstCase;

  cout << "Нажмите 1, чтобы начать работать с классом круг" << endl;

  cout << "Нажмите 2, чтобы начать работать с классом треугольник" << endl;

  cout << "Нажмите 3, чтобы начать работать с классом ромб" << endl;

  cout << "Нажмите 4, чтобы начать работать с классом трапеция" << endl;

  cout << "Нажмите 5, чтобы начать работать с классом прямоугольник" << endl;

  cout << "Нажмите 6, чтобы посмотреть всю информацию о всех имеющихся геометрических фигурах" << endl;

  cout << "Нажмите 7 просто так" << endl;

  cout << "Нажмите 8, чтобы записать всю информацию о всех имеющихся геометрических фигурах в файле" << endl;

  cout << "Нажмите 9, чтобы выйти" << endl;

  cout << "Введите число: " << endl;

  cin >> FirstCase;

  switch (FirstCase)

  {

  case 1:

  {

   cout << "Вы начали работу с классом круг" << endl;

   cout << "Нажмите 1, чтобы добавить объект класса круг" << endl;

   cout << "Нажмите 2, чтобы вычислить площадь объекта класса круг" << endl;

   cout << "Нажмите 3, чтобы вычислить периметр объекта класса круг" << endl;

   cout << "Нажмите 4, чтобы сделать сдвиг объекта класса круг" << endl;

   cout << "Нажмите 5, чтобы вернуться в предыдущее меню" << endl;

   int SecondCase;

   cin >> SecondCase;

   switch (SecondCase)

   {

   case 1:

   {

    int x;

    int y;

    int R;

    cout << "Введите координаты х, у и радиус круга" << endl;

    cin >> x;

    cin >> y;

    cin >> R;

    Circle TMP(x, y, R);

    C.push\_back(TMP);

    break;

   }

   case 2:

   {

    cout << "Введите индекс объекта круга (отсчет начинается с нуля), площадь которого вы хотите вычислить" << endl;

    cout << "На данный момент существуют " << C.size() << " объектов класса круг" << endl;

    int i;

    cin >> i;

    cout << "Площадь " << i << "-го элемента: " << C.at(i).Square() << endl;

    break;

   }

   case 3:

   {

    cout << "Введите индекс объекта круга (отсчет начинается с нуля), периметр которого вы хотите вычислить" << endl;

    cout << "На данный момент существуют " << C.size() << " объектов класса круг" << endl;

    int i;

    cin >> i;

    cout << "Периметр " << i << "-го элемента: " << C.at(i).Perimetr() << endl;

    break;

   }

   case 4:

   {

    string direction;

    int HowMuch;

    cout << "Введите индекс объекта круга (отсчет начинается с нуля),  который вы хотите сдвинуть" << endl;

    cout << "На данный момент существуют " << C.size() << " объектов класса круг" << endl;

    int i;

    cin >> i;

    cout << "В какую сторону вы хотите сдвинуть объект? (right, left, up, down)" << endl;

    cin >> ws; getline(cin, direction);

    cout << "На сколько вы хотите сдвинуть объект?" << endl;

    cin >> HowMuch;

    cout << "Координаты  точек вашего круга до сдвига: х = " << C.at(i).GetX() << " y = " << C.at(i).GetY() << endl;

    C[i].Bias(direction, HowMuch);

    cout << "Координаты  точек вашего круга после сдвига: х = " << C.at(i).GetX() << " y = " << C.at(i).GetY() << endl;

    break;

   }

   case 5:

   {

    break;

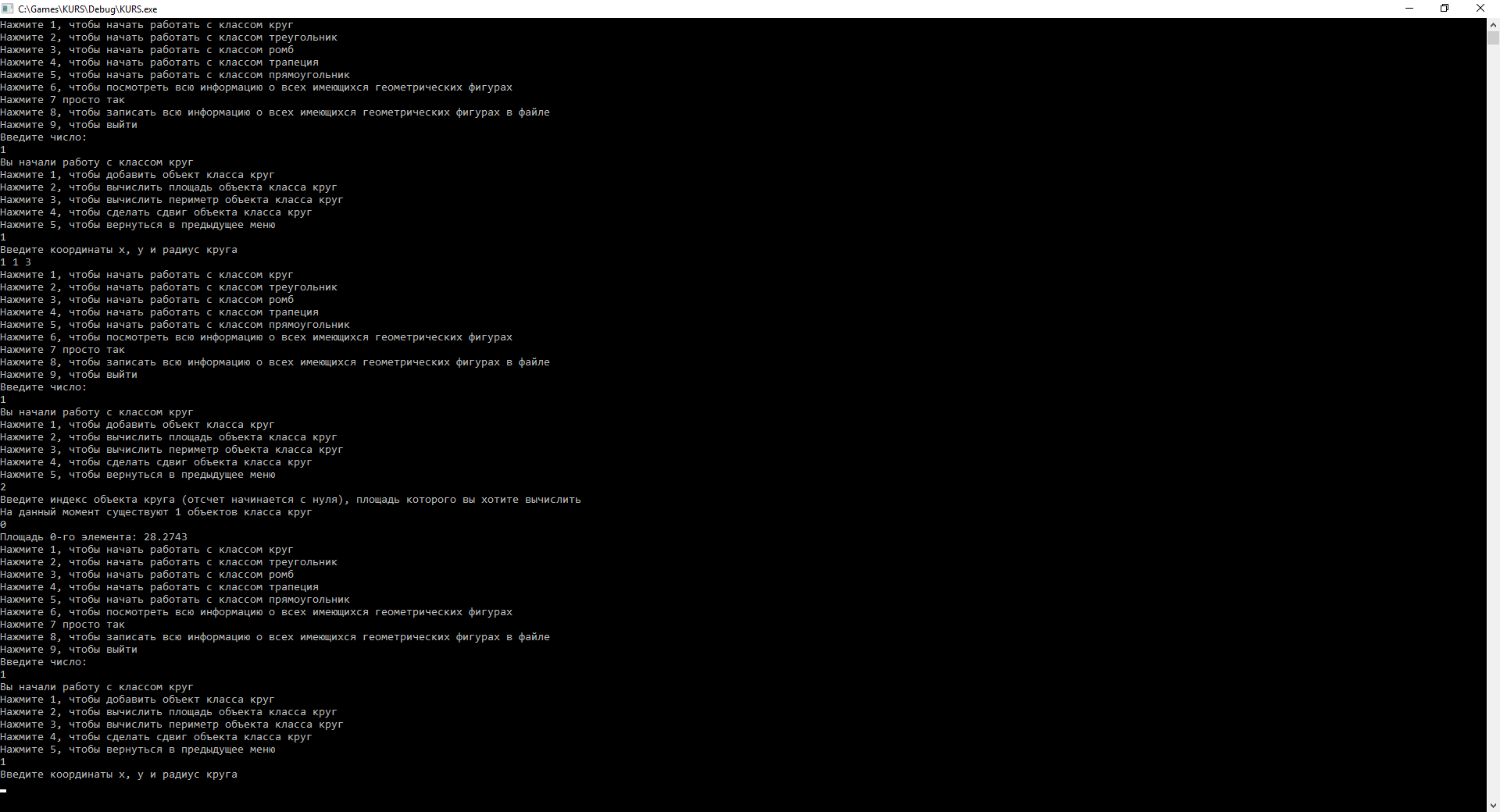
   }

   default:

    break;

   }

   break;

  }

Код, который выводит информацию о фигурах на экран

 case 6:

  {

   cout << "Всего объектов типа круг: " << C.size() << endl;

   for (int j = 0; j < C.size(); j++)

   {

    cout << "Круг под номером " << j << endl;

    cout << " x = " << C.at(j).GetX() << ", y = " << C.at(j).GetY() << endl;

    cout << "Площадь: " << C.at(j).Square() << ", периметр: " << C.at(j).Perimetr() << endl;

   }

   cout << "Всего объектов типа треугольник: " << T.size() << endl;

   for (int j = 0; j < T.size(); j++)

   {

    cout << "Треугольник под номером: " << j << endl;

    cout << " x1 = " << T.at(j).GetX1() << ", y1 = " << T.at(j).GetY1() << ", x2 = " << T.at(j).GetX2() << " , y2 = " << T.at(j).GetY2() << ", x3 = " << T.at(j).GetX3() << ", y3 = " << T.at(j).GetY3() << endl;

    cout << "Площадь: " << T.at(j).Square() << ", периметр: " << T.at(j).Perimetr() << endl;

   }

   cout << "Всего объектов типа ромб: " << R.size() << endl;

   for (int j = 0; j < R.size(); j++)

   {

    cout << "Ромб под номером: " << j << endl;

    cout << " x1 = " << R.at(j).GetX1() << ", y1 = " << R.at(j).GetY1() << ", x2 = " << R.at(j).GetX2() << " , y2 = " << R.at(j).GetY2() << ", x3 = " << R.at(j).GetX3() << ", y3 = " << R.at(j).GetY3() <<

     ", x4 = " << R.at(j).GetX4() << ", y4 = " << R.at(j).GetY4() << endl;

    cout << "Площадь: " << R.at(j).Square() << ", периметр: " << R.at(j).Perimetr() << endl;

   }

   cout << "Всего объектов типа трапеция: " << Tr.size() << endl;

   for (int j = 0; j < Tr.size(); j++)

   {

    cout << "Трапеция под номером: " << j << endl;

    cout << " x1 = " << R.at(j).GetX1() << ", y1 = " << Tr.at(j).GetY1() << ", x2 = " << Tr.at(j).GetX2() << " , y2 = " << Tr.at(j).GetY2() << ", x3 = " << Tr.at(j).GetX3() << ", y3 = " << Tr.at(j).GetY3() <<

     ", x4 = " << Tr.at(j).GetX4() << ", y4 = " << Tr.at(j).GetY4() << endl;

    cout << "Площадь: " << Tr.at(j).Square() << ", периметр: " << Tr.at(j).Perimetr() << endl;

   }

   cout << "Всего объектов типа прямоугольник: " << Rect.size() << endl;

   for (int j = 0; j < Rect.size(); j++)

   {

    cout << "Прямоугольник под номером: " << j << endl;

    cout << " x1 = " << Rect.at(j).GetX1() << ", y1 = " << Rect.at(j).GetY1() << ", x2 = " << Rect.at(j).GetX2() << " , y2 = " << Rect.at(j).GetY2() << ", x3 = " << Rect.at(j).GetX3() << ", y3 = " << Rect.at(j).GetY3() <<

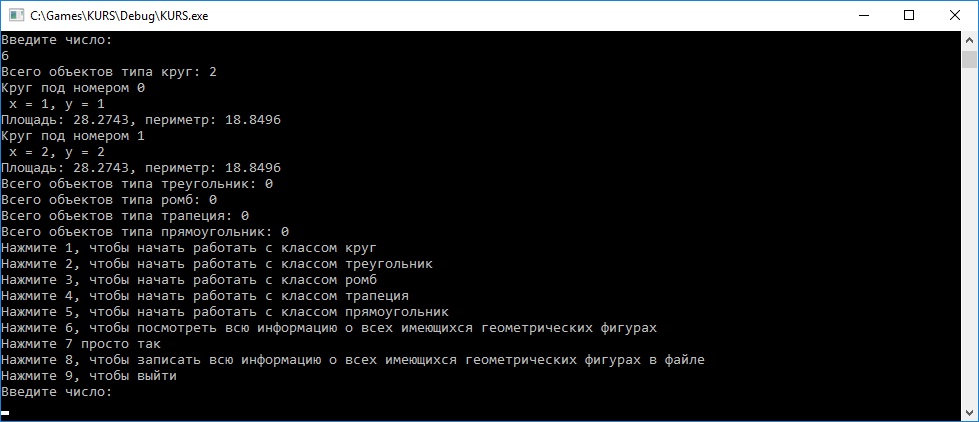
     ", x4 = " << Rect.at(j).GetX4() << ", y4 = " << Rect.at(j).GetY4() << endl;

    cout << "Площадь: " << Rect.at(j).Square() << ", периметр: " << Rect.at(j).Perimetr() << endl;

   }

  }

  break;



Код, который выводит эту же информацию в файл:

 case 8:

  {

   ofstream fout;

   fout.open("I NEVER ASK FOR THIS.txt");

   fout << "Всего объектов типа круг: " << C.size() << endl;

   for (int j = 0; j < C.size(); j++)

   {

    fout << "Круг под номером " << j << endl;

    fout << " x = " << C.at(j).GetX() << ", y = " << C.at(j).GetY() << endl;

    fout << "Площадь: " << C.at(j).Square() << ", периметр: " << C.at(j).Perimetr() << endl;

   }

   fout << "Всего объектов типа треугольник: " << T.size() << endl;

   for (int j = 0; j < T.size(); j++)

   {

    fout << "Треугольник под номером: " << j << endl;

    fout << " x1 = " << T.at(j).GetX1() << ", y1 = " << T.at(j).GetY1() << ", x2 = " << T.at(j).GetX2() << " , y2 = " << T.at(j).GetY2() << ", x3 = " << T.at(j).GetX3() << ", y3 = " << T.at(j).GetY3() << endl;

    fout << "Площадь: " << T.at(j).Square() << ", периметр: " << T.at(j).Perimetr() << endl;

   }

   fout << "Всего объектов типа ромб: " << R.size() << endl;

   for (int j = 0; j < R.size(); j++)

   {

    fout << "Ромб под номером: " << j << endl;

    fout << " x1 = " << R.at(j).GetX1() << ", y1 = " << R.at(j).GetY1() << ", x2 = " << R.at(j).GetX2() << " , y2 = " << R.at(j).GetY2() << ", x3 = " << R.at(j).GetX3() << ", y3 = " << R.at(j).GetY3() <<

     ", x4 = " << R.at(j).GetX4() << ", y4 = " << R.at(j).GetY4() << endl;

    fout << "Площадь: " << R.at(j).Square() << ", периметр: " << R.at(j).Perimetr() << endl;

   }

   fout << "Всего объектов типа трапеция: " << Tr.size() << endl;

   for (int j = 0; j < Tr.size(); j++)

   {

    fout << "Трапеция под номером: " << j << endl;

    fout << " x1 = " << Tr.at(j).GetX1() << ", y1 = " << Tr.at(j).GetY1() << ", x2 = " << Tr.at(j).GetX2() << " , y2 = " << Tr.at(j).GetY2() << ", x3 = " << Tr.at(j).GetX3() << ", y3 = " << Tr.at(j).GetY3() <<

     ", x4 = " << Tr.at(j).GetX4() << ", y4 = " << Tr.at(j).GetY4() << endl;

    fout << "Площадь: " << Tr.at(j).Square() << ", периметр: " << Tr.at(j).Perimetr() << endl;

   }

   cout << "Всего объектов типа прямоугольник: " << Rect.size() << endl;

   for (int j = 0; j < Rect.size(); j++)

   {

    fout << "Прямоугольник под номером: " << j << endl;

    fout << " x1 = " << Rect.at(j).GetX1() << ", y1 = " << Rect.at(j).GetY1() << ", x2 = " << Rect.at(j).GetX2() << " , y2 = " << Rect.at(j).GetY2() << ", x3 = " << Rect.at(j).GetX3() << ", y3 = " << Rect.at(j).GetY3() <<

     ", x4 = " << Rect.at(j).GetX4() << ", y4 = " << Rect.at(j).GetY4() << endl;

    fout << "Площадь: " << Rect.at(j).Square() << ", периметр: " << Rect.at(j).Perimetr() << endl;

   }

   fout << "© все права не защищены, 2019 \n";

   fout << "© all rights are not reserve, 2019";

   fout.close();

  }

