**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Наследование, полиморфизм

Студент: Баранников Степан Алексеевич

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата: 19.10.2019

Оценка:

Москва, 2019

1. Постановка задачи

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов:

1.) Вычисление геометрического центра фигуры;

2.) Вывод в стандартный поток вывода std::cout координат вершин фигуры;

3.) Вычисление площади фигуры;

Вариант задания

|  |  |  |  |
| --- | --- | --- | --- |
| 18. | Квадрат | Прямоугольник | Трапеция |

1. Описание программы

Программа позволяет:

• Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.

• Сохранять созданные фигуры в динамический массив std::vector<Figure\*>

• Вызывать для всего массива общие функции.Т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.

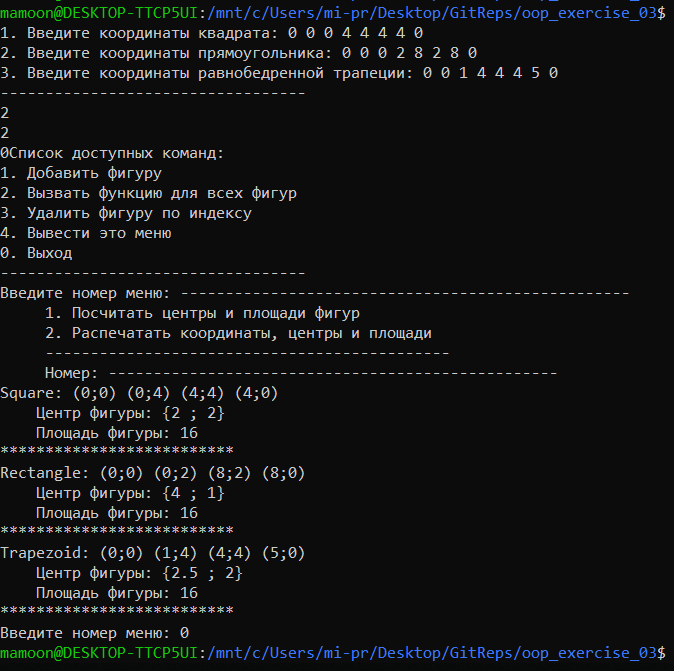
• Удалять из массива фигуру по индексу;

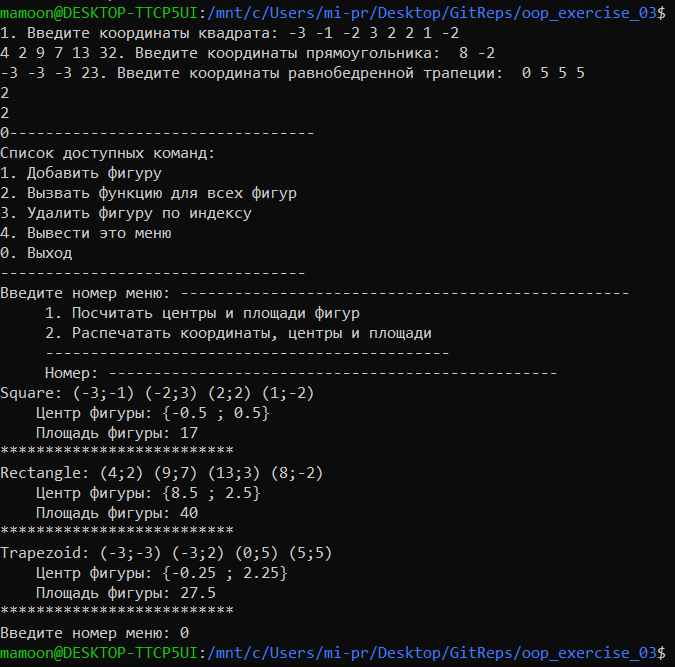
1. Набор testcases

test\_01.txt  
0 0 0 4 4 4 4 0  
0 0 0 2 8 2 8 0  
0 0 1 4 4 4 5 0  
2  
2  
0

test\_02.txt  
-3 -1 -2 3 2 2 1 -2  
4 2 9 7 13 3 8 -2  
-3 -3 -3 2 0 5 5 5  
2  
2  
0

1. Результаты выполнения тестов.

./exercise < test\_01.txt  


./exercise\_02 < test\_02.txt  


1. Листинг программы

|  |
| --- |
| //Barannikov Stepan M8o-201b /\* \*Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. \*Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов: \*1. Вычисление геометрического центра фигуры; \*2. Вывод в стандартный поток вывода std::cout координат вершин фигуры; 3. Вычисление площади фигуры; \*Вариант задания - 18: Квадрат, Прямоугольник, Трапеция. \*/  #include <iostream> using namespace std; #include <vector> #include <cmath>  class Figure{  protected:  double Area;  double xCent, yCent;  vector<double> X;  vector<double> Y;  public:  Figure(){}  virtual ~Figure(){}   virtual void calcCent() = 0;  virtual void calcArea() = 0;  virtual void printCords() = 0;   void printCent(){  cout << "Центр фигуры: " << "{" << xCent << " ; " << yCent << "}" << endl;  }  void printArea(){  cout << "Площадь фигуры: " << Area << endl;  }  double squareSide(double x1, double y1, double x2, double y2){  return (x1-x2)\*(x1-x2) + (y1-y2)\*(y1-y2);  } }; class Square: public Figure{  public:  Square(){};  Square(double X1, double Y1, double X2, double Y2, double X3, double Y3, double X4, double Y4){  X.push\_back(X1); X.push\_back(X2); X.push\_back(X3); X.push\_back(X4);  Y.push\_back(Y1); Y.push\_back(Y2); Y.push\_back(Y3); Y.push\_back(Y4);  }  ~Square(){};   void calcArea()override{  if(X.size() < 4)  return;  Area = squareSide(X[0],Y[0],X[1],Y[1]);  }  void calcCent()override{  if(X.size() < 4)  return;  xCent = yCent = 0;  for(int i = 0; i < X.size(); i++){  xCent = xCent + X[i];  yCent = yCent + Y[i];  }  xCent = xCent / X.size();  yCent = yCent / Y.size();  }  void printCords(){  cout << "Square: ";  for(int i = 0; i < X.size(); i++)  cout << "(" << X[i] << ";" << Y[i] << ") ";  cout << endl;  } }; class Rectangle: public Figure{  public:  Rectangle(){}  Rectangle(double X1, double Y1, double X2, double Y2, double X3, double Y3, double X4, double Y4){  X.push\_back(X1); X.push\_back(X2); X.push\_back(X3); X.push\_back(X4);  Y.push\_back(Y1); Y.push\_back(Y2); Y.push\_back(Y3); Y.push\_back(Y4);  }  ~Rectangle(){}   void calcArea()override{  Area = 0;  if(X.size() < 4)  return;  Area = sqrt(squareSide(X[0],Y[0],X[1],Y[1])\*squareSide(X[1],Y[1],X[2],Y[2]));  }  void calcCent()override{  if(X.size() < 4)  return;  xCent = yCent = 0;  for(int i = 0; i < X.size(); i++){  xCent = xCent + X[i];  yCent = yCent + Y[i];  }  xCent = xCent / X.size();  yCent = yCent / Y.size();   }  void printCords(){  cout << "Rectangle: ";  for(int i = 0; i < X.size(); i++)  cout << "(" << X[i] << ";" << Y[i] << ") ";  cout << endl;  } }; class Trapezoid: public Figure{  private:  vector<double> side;  public:  Trapezoid(){}  Trapezoid(double X1, double Y1, double X2, double Y2, double X3, double Y3, double X4, double Y4){  X.push\_back(X1); X.push\_back(X2); X.push\_back(X3); X.push\_back(X4);  Y.push\_back(Y1); Y.push\_back(Y2); Y.push\_back(Y3); Y.push\_back(Y4);  side.push\_back(squareSide(X1, Y1, X2, Y2));  side.push\_back(squareSide(X2, Y2, X3, Y3));  side.push\_back(squareSide(X3, Y3, X4, Y4));  side.push\_back(squareSide(X1, Y1, X4, Y4));  }  ~Trapezoid(){}    void calcArea()override{  Area = 0;  if(X.size() < 4)  return;  double a,b,c,part,sum;  if(side[0] == side[2]){  c = side[0];  if(side[1] > side[3]){  a = side[1]; b = side[3];  }  else  a = side[3]; b = side[1];  }  if(side[1] == side[3]){  c = side[1];  if(side[0] > side[2]){  a = side[0]; b = side[2];  }  else  a = side[2]; b = side[0];  }  part = sqrt(a) - sqrt(b);  sum = (sqrt(a) + sqrt(b))/2;  Area = sum\*sqrt(c-part\*part/4);  }  void calcCent()override{  if(X.size() < 4)  return;  xCent = yCent = 0;  if(side[0] == side[2]){  xCent = (((X[1]+X[0])/2)+((X[2]+X[3])/2))/2;  yCent = (((Y[1]+Y[0])/2)+((Y[2]+Y[3])/2))/2;  }  if(side[1] == side[3]){  xCent = (((X[1]+X[2])/2)+((X[3]+X[0])/2))/2;  yCent = (((Y[1]+Y[2])/2)+((Y[3]+Y[0])/2))/2;  }  }  void printCords(){  cout << "Trapezoid: ";  for(int i = 0; i < X.size(); i++)  cout << "(" << X[i] << ";" << Y[i] << ") ";  cout << endl;  } };  void printMenu(){  cout << "----------------------------------" << endl;  cout << "Список доступных команд:" << endl;  cout << "1. Добавить фигуру" << endl;  cout << "2. Вызвать функцию для всех фигур" << endl;  cout << "3. Удалить фигуру по индексу" << endl;  cout << "4. Вывести это меню" << endl;  cout << "0. Выход" << endl;  cout << "----------------------------------" << endl; }  int main(){  vector<Figure\*> v;  double x1, y1, x2, y2, x3, y3, x4, y4;  cout << "1. Введите координаты квадрата: ";  cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;  Figure \*s;  s = new Square(x1, y1, x2, y2, x3, y3, x4, y4);  v.push\_back(s);  cout << "2. Введите координаты прямоугольника: ";  cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;  s = new Rectangle(x1, y1, x2, y2, x3, y3, x4, y4);  v.push\_back(s);  cout << "3. Введите координаты равнобедренной трапеции: ";  cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;  s = new Trapezoid(x1, y1, x2, y2, x3, y3, x4, y4);  v.push\_back(s);  for(int i = 0; i < v.size(); i++)  (\*v[i]).calcArea();  for(int i = 0; i < v.size(); i++)  (\*v[i]).calcCent();  printMenu();  while(true){  cout << "Введите номер меню: ";  int k;  cin >> k;   switch(k){  case 0:{  for(size\_t i = 0; i < v.size(); i++){  delete v[i];  }  return 0;  }  case 1:{  cout << "--------------------------" << endl;  cout << " 1. Квадрат" << endl;  cout << " 2. Прямоугольник" << endl;  cout << " 3. Трапеция" << endl;  cout << " ---------------------" << endl;  cout << " Выберите тип фигуры: ";   int a;  cin >> a;  cout << "--------------------------" << endl;    if(a < 1 || a > 3){  cout << "Неверный номер" << endl;  break;  }   cout << "Введите координаты: ";  cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;   Figure \*s;  switch(a){  case 1:  s = new Square(x1, y1, x2, y2, x3, y3, x4, y4);  break;  case 2:  s = new Rectangle(x1, y1, x2, y2, x3, y3, x4, y4);  break;  case 3:  s = new Trapezoid(x1, y1, x2, y2, x3, y3, x4, y4);  break;  }  v.push\_back(s);  (\*v[v.size()-1]).printCords();  break;  }  case 2:{  cout << "--------------------------------------------------" << endl;  cout << " 1. Посчитать центры и площади фигур" << endl;  cout << " 2. Распечатать координаты, центры и площади" << endl;  cout << " ---------------------------------------------" << endl;  cout << " Номер: ";  int a;  cin >> a;  cout << "--------------------------------------------------" << endl;  if(a < 1 || a > 2){  cout << " Неверный номер" << endl;  break;  }  switch(a){  case 1:{  for(int i = 0; i < v.size(); i++)  (\*v[i]).calcArea();  for(int i = 0; i < v.size(); i++)  (\*v[i]).calcCent();  break;  }  case 2:{  for(int i = 0; i < v.size(); i++){  (\*v[i]).printCords();  cout << " ";  (\*v[i]).printCent();  cout << " ";  (\*v[i]).printArea();  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  }  break;  }  }  break;  }  case 3:{  cout << "Индекс: ";  size\_t id;  cin >> id;   if(id < 0 || id > v.size()-1){  cout << "Индекс выходит за границы массива" << endl;  break;  }   vector<Figure\*> next;  for(int i = 0; i < v.size(); i++){  if(i == id)  continue;  next.push\_back(v[i]);  }  swap(v, next);  break;  }  case 4:{  printMenu();  break;  }  }  } } |

1. Вывод:

Научился использовать наследование в C++ для реализации иерархии классов.

## Список литературы

1. Наследование в C++ [электронный ресурс].

URL: <https://habr.com/ru/post/445948/>

1. std::vector [электронный ресурс].

URL: <https://ru.cppreference.com/w/cpp/container/vector>