Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №3**

«Наследование»

Выполнил:
студент группы 213 Тихонов Е.Е.

Проверил:
старший преподаватель кафедры ПОиАИС Ураева Е.Е.

Курск, 2021

***Цель работы:*** изучить особенности реализации наследования классов в языке C++.

***Задание***

*Задача 1.* Самостоятельно разработать иерархию классов их полей и методов для предметной области «Геометрическая фигура на плоскости». Создать приложение, в котором создаются все объекты разработанной иерархии классов, для каждого из которых вызывается метод, возвращающий информацию об объекте в текстовом представлении. Обязательные условия к реализации:

- наличие по крайней мере трех уровней иерархии;
- описание двух различных типов связей между классами;
- использование множественного наследования;
- наличие конструкторов в каждом классе;
- наличие метода формирования текстовой информации об объекте в каждом классе.

***Разработка алгоритма***

*Задача 1*

Входные данные: нет.

Выходные данные: текстовые данные в консоли.

***Текст программы***

*Текст программы для решения задачи 1*

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
#include <stdexcept>
```

```
using namespace std;

class Figure;

class Flat
{
	public:
	vector<Figure*> ChildFigures;
};

class Dot
{
	public:
	float x;
	float y;
	Dot(int x, int y)
	{
	this->x = x;
	this->y = y;
	}
};

Dot operator +(Dot &Dot1, Dot &Dot2){
	return Dot(Dot1.x+Dot2.x, Dot1.y+Dot2.y);
}
Dot operator -(Dot &Dot1, Dot &Dot2){
	return Dot(Dot1.x-Dot2.x, Dot1.y-Dot2.y);
}
float operator *(Dot &Dot1, Dot &Dot2){
	return Dot1.x*Dot2.x + Dot1.y*Dot2.y;
}
```

```
Dot GetVector(Dot Dot1, Dot Dot2){
	return Dot2-Dot1;
}
float VectorAbs(Dot Vector){
	return sqrt(Vector.x*Vector.x + Vector.y*Vector.y);
}


class Figure
{
	private:
	int Id;

	protected:
	Figure()
	{
	//...
	}

	public:
	Flat* ParentFlat;
	vector<Dot> Dots;
	Figure(vector<Dot> Dots, Flat* ParentFlat)
	{
	this->Dots = vector<Dot>(Dots);
	this->ParentFlat = ParentFlat;
	this->Id = ParentFlat->ChildFigures.size();
	this->ParentFlat->ChildFigures.push_back(this);
	}
	~Figure()
	{
```

```
this->ParentFlat->ChildFigures.erase(this->ParentFlat->ChildFigures.be
gin()+this->Id);
	}

	protected:
	virtual void Print(string FigureName)
	{
	cout << FigureName <<"; Точки:";
	for (size_t i = 0; i < this->Dots.size(); i++)
	{
		cout<<" ("<<Dots[i].x<<", "<<Dots[i].y<<")";
	}
	cout<<endl;
	}
};

class Parallelogram: virtual protected Figure
{
	protected:
	Parallelogram()
	{
	//...
	}

	public:
	Parallelogram(Dot Dot1, Dot Dot2, Dot Dot3, Flat* ParentFlat):
Figure({Dot1,Dot2, Dot3}, ParentFlat)
	{
	Dot DiagonalCenter = Dot(0, 0);

	DiagonalCenter.x = (Dot1.x+Dot3.x)/2;
	DiagonalCenter.y = (Dot1.y+Dot3.y)/2;
```

```
	Dot Dot4 = Dot(0,0);

	Dot4.x = DiagonalCenter.x*2 - Dot2.x;
	Dot4.y = DiagonalCenter.y*2 - Dot2.y;

	this->Dots.push_back(Dot4);
	}

	virtual void Print()
	{
	Figure::Print("Параллелограмм");
	}
};

class Rectangle: virtual protected Figure
{
	protected:
	Rectangle()
	{
	//...
	}

	public:
	float Width;
	float Height;
	Rectangle(Dot LeftTopDot, float Width, float Height, Flat*
ParentFlat): Figure({LeftTopDot}, ParentFlat)
	{
	Dot RightTopDot = Dot(LeftTopDot.x+Width, LeftTopDot.y);
	Dot LeftBottomDot = Dot(LeftTopDot.x, LeftTopDot.y - Height);
	Dot RightBottomDot = Dot(LeftTopDot.x+Width, LeftTopDot.y -
Height);
```

```
	this->Dots.push_back(RightTopDot);
	this->Dots.push_back(RightBottomDot);
	this->Dots.push_back(LeftBottomDot);
	this->Width = Width;
	this->Height = Height;
	}

	virtual void Print()
	{
	Figure::Print("Прямоугольник");
	}
};

class Square: protected Rectangle, protected Parallelogram
{
	public:
	Square(Dot LeftTopDot, float Width, Flat* ParentFlat):
Rectangle(LeftTopDot, Width, Width, ParentFlat)
	{
	//...
	}

	Square(Dot Dot1, Dot Dot2, Dot Dot3, Flat* ParentFlat):
Parallelogram(Dot1, Dot2, Dot3, ParentFlat)
	{
	Dot Vector1 = GetVector(Dot1, Dot2);
	Dot Vector2 = GetVector(Dot2, Dot3);

	this->Width = VectorAbs(Vector1);
	this->Height = VectorAbs(Vector2);

	if (this->Width != this->Height)
	{
```

```
            throw std::invalid_argument("Width and hight of square not
equal");
      }

      if (Vector1 * Vector2 != 0)
      {
            throw std::invalid_argument("Square hasn't right angle");
      }
      }

      void Print()
      {
      Figure::Print("Квадрат");
      }

};

int main()
{
      Flat A;

      Parallelogram p1(Dot(4,1), Dot(1,1), Dot(2,3), &A);
      p1.Print();

      {
      Rectangle r1(Dot(1,1), 4, 2, &A);
      r1.Print();
      }

      Square s1(Dot(1,1),4, &A);
      s1.Print();

      Square s2(Dot(1,1), Dot(5, 1), Dot(5,-3), &A);
```

```
    s1.Print();

    return 0;
}
```

***Тестирование программы***

Тестирование задачи 1 представлено на рисунках 1, 2.

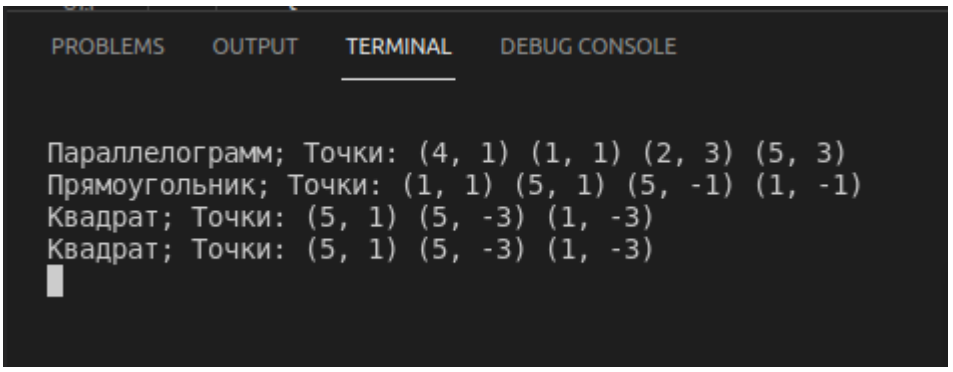


Рисунок 1 - Тест 1 задачи 1

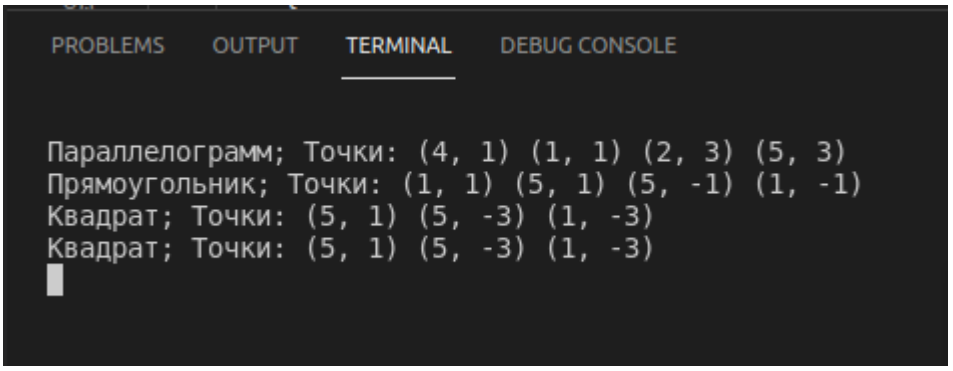


Рисунок 2 - Тест 2 задачи 1