Практические занятия с применением C++

Online-компиляторы поддерживающий язык программирования C++:

http://cpp.sh/

http://www.pythontutor.com/visualize.html#mode=edit

http://rextester.com/l/cpp_online_compiler_visual

Занятие 7. Реализация сущности в виде класса

На этом занятии демонстрируется организация программ в соответствии с объектно-ориентированным стилем программирования, правила описания, объявления и использования классов.

Задание 1. Реализация точки как структуры

- До функции main() объявите структуру Point, реализующую сущность «точка». Точка описывается двумя координатами вещественного типа:

```
#include <iostream>
#include <math.h>

using namespace std;

struct Point
{
    double x;
    double y;
};
```

- При создании структурной переменной точка есть возможность сразу сообщить с какими координатами она будет создаваться. Для этого при объявлении структуры необходимо указать как минимум два конструктора (функции, которые автоматически вызываются при создании объекта для его инициализации), первый «по умолчанию» – для создания точки без передачи параметров, а второй «с параметрами» – для создания точки с конкретными параметрами:

```
struct Point
{
    double x;
    double y;

    Point()
```

```
    {
      x = y = 0;
    }
    Point(double xd, double yd)
    {
        x = xd;
        y = yd;
    }
};
```

- После объявления структуры и перед функцией main() объявите новую функцию для вывода информации о точке на экран:

```
void ShowPoint(Point p)
{
    cout << p.x << "-" << p.y << "\n";
}
```

- Обратите внимание, что эта функция (по сути, процедура) принимает в качестве параметра некую точку и отрабатывает вывод на экран.
- В функции main() объявите две точки – первую с передачей в конструктор требуемых значений, вторую – по умолчанию, а затем дважды вызовите функцию вывода данных о точках на экран:

```
int main()
{
    Point a(1, 10);
    Point d;

    ShowPoint(a);
    ShowPoint(d);
}
```

- Запустите программу. Проверьте, что значение координат отобразились как это реализовано в функции вывода.

Замечание. Такое разделение на объявление сущности и создание отдельной самостоятельной функции, работающей с переменной этой сущности, относится к процедурному стилю проектирования и программирования.

- Измените стиль мышления – представьте, что точка сама должна уметь выводить свои данные на экран.
- Для реализации этой самостоятельности объявите внутри структуры функцию, печатающую данные на экране (причем передавать параметр уже не надо – точка сама знает какие у нее координаты):

```
struct Point
{
```

```
    double x;
    double y;

    Point()
    {
        x = y = 0;
    }
    Point(double xd, double yd)
    {
        x = xd;
        y = yd;
    }

    void ShowPoint()
    {
        cout << x << "-" << y << "\n";
    }
};
```

- В функции main() добавьте новый вариант вызова функции печати на экране – функция вызывается для конкретной структурной переменной без передачи параметра – она сама реализует свое “умение”:

```
a.ShowPoint();
d.ShowPoint();
```

- Запустите программу. Проверьте, что значение координат отобразились как это реализовано в функции вывода.

Замечание. Такое совместное объявление сущности и функции, непосредственно работающей с переменной этой сущности, относится к объектно-ориентированному стилю проектирования и программирования – данные и функции, работающие с этими данными объявляются в одном модуле – структуре или классе.

Задание 2. Реализация отношения между структурой и классом

- После структуры объявите класс Triangle, реализующий геометрическую фигуру «треугольник».

Треугольник описывается тремя точками – объектами структуры Point. Такое отношение между классом треугольника и структурой точки называется композиция (частный случай ассоциации): треугольник – целое, точка – часть, входящая в целое.

```
class Triangle
{
    Point a;
    Point b;
    Point c;
```

- В класс Triangle добавьте конструктор, принимающий три точки и функцию расчета площади треугольника. Эти члены класса должны быть объявлены как открытые (публичные) с модификатором public чтобы их можно было бы вызвать из других функций и классов:

```
  public:
      Triangle (Point pa, Point pb, Point pc)
      {
          a = pa;
          b = pb;
          c = pc;
      }

      double getArea()
      {
         return abs((a.x - c.x) * (b.y - c.y) - (b.x - c.x) * (a.y -
c.y)) / 2;
      }
};
```

- В функции main() создайте еще одну точку b (в добавлении к точкам a и d):

```
int main()
{
   Point a(1, 10) , b(21, 25);
   Point d;
```

- Далее объявите переменную (объект tr1) типа Triangle (треугольник) с передачей в его конструктор имеющихся точек:

```
   Triangle tr1(a,b,d);
```

- Вычислите площадь созданного треугольника, вызвав его функцию:

```
   double Streug1 = tr1.getArea();
```

- Создайте еще один треугольник (объект-треугольник tr2), используя непосредственно значения координат точек (этот вариант применяется в версии C++11):

```
   Triangle tr2({1, 10}, {21, 25}, {0, 0});
```

- Аналогично вычислите площадь второго треугольника:

```
   double Streug2 = tr2.getArea();
```

- Выведите значения площадей на экран:

```
   cout << "S = " << Streug1 << endl;
   cout << "S = " << Streug2 << endl;
   return 0;
}
```

- Проверьте работу программы. Значения площадей двух треугольников должны быть равны.

Задание 3. Применение наследования для организации иерархии классов

Для реализации программ сложных систем часто создаются иерархические отношения между сущностями, основным таким отношением является наследование. В результате наследования класс-потомок получает (наследует) свойства и поведение класса-предка.

- Подумайте какая сущность может быть обобщающей для треугольника?

Ход рассуждений:

- Чем по своей сути является треугольник?
- Известно, что обобщенным понятием треугольника является понятие «Геометрическая фигура».
- Какие еще геометрические фигуры известны и могут ли они быть тоже обобщающими сущностями для треугольника?
- Можно предложить в качестве фигур: окружность, прямоугольник, трапеция, тетраэдр.
- В текущей задача треугольник – плоская фигура?
- Да.
- Какими свойствами и поведением обладает любая плоская фигура?
- У любой плоской фигуры есть хотя бы одна точка, она имеет некую площадь и название.

Таким образом, исходя из проведенных рассуждений можно сделать вывод о том, что базовой сущностью-классом для класса *Треугольник* будет класс *Фигура*, в которой следует объявить общие свойства – *Точка* и функции – *Расчет площади* и *Вывод названия фигуры*.

Согласно сделанным выводам внесите следующие изменения в программу.

- После структуры Point объявите новый класс Shape, описывающий сущность «Фигура».
- Любая фигура описывается хотя бы одной точкой, поэтому в новом классе объявите переменную типа Point.
- Класс фигуры рекомендуется сделать абстрактным – в таком случае нельзя будет создавать объекты этого класса, но для любой фигуры потребуется рассчитывать площадь. В этом случае требуется объявить абстрактную функцию (в C++ она называется чистой виртуальной функцией):

```
class Shape
{
public:
    Point a;
    virtual double getArea() = 0;
};
```

Таким образом, любой класс, который объявит себя наследником класса Shape, получит в наследство одну точку и обязанность (потому что функция объявлена абстрактной и не имеет тела) реализовать свою версию расчета площади.

- Измените класс Triangle, сделайте его наследуемым от класса фигуры и удалите первую точку в поле класса:

```
class Triangle : public Shape
{
    Point b;
    Point c;

    public:
    Triangle (Point pa, Point pb, Point pc)
        {
            a = pa;
            b = pb;
            c = pc;
        }
        double getArea()
        {
            return abs((a.x - c.x) * (b.y - c.y) - (b.x - c.x) *
(a.y - c.y)) / 2;
        }
};
```

- Проверьте работу программы, ее функциональность не должна измениться.

При использовании наследования можно не только расширять функциональность базового класса, но и пользоваться функциональностью, реализованной в базовом классе.

- Добавьте в базовый класс Shape новую функцию, возвращающую строку текста:

```
class Shape
{
public:
    Point a;
    virtual double getArea() = 0;
    string OutTitle(string name)
    {
        return "Shape: " + name;
    }
};
```

Класс Triangle остается без изменений, он наследует функциональность базового класса.

- В функции main() добавьте код для проверки функциональности объектов треугольников:

```
cout << tr1.OutTitle("First Triangle") << endl;
cout << "S = " << Streug1 << endl;

cout << tr2.OutTitle("Second Triangle ") << endl;
cout << "S = " << Streug2 << endl;
```

- Проверьте работу программы, ее функциональность должна измениться – появятся текстовые сообщения.