Практическое занятие 3.10

Классы. Конструкторы. Деструкторы.

Класс

Класс - это некоторый шаблон (абстракция) еще не существующих объектов, в котором собраны все детали, все свойства и все нужные действия необходимые для этих объектов. В первую очередь класс является **собственным типом данных**.

Класс – основное понятие объектно-ориентированного подхода к программированию.

Основные принципы:

– абстракция,
– инкапсуляция,
– полиморфизм,
– наследование.

Класс

```
#include <math.h>
class Point {
    int x, y; //по умолчанию private – следуем принципу инкапсуляции
  public:
    void getXY() {//Функция доступна из любого места программы.
        std::cout << "Input X:  ";
        std::cin >> x;
        std::cout << "Input Y:  ";
        std::cin >> y;
    }
    double getLength() {  //Функция суммирования двух чисел
        return sqrt(x*x + y*y);
        //или так, обратите внимание, полезно ставить пробелы
        //return sqrt((*this).x * (*this).x + this->y * this->y);
    }
} p1; //p1 - это объект класса Point.
```

Конструкторы класса

```
class MyClass {
  private:
      static int counter;
      int test;
  public:
      const int id;
  public:
      MyClass(int a) : test(a), id(0) { }
// Задание значений

      // Можно и так (медленнее), кроме того,
id - const
      MyClass() : id(0) {
            test = 0;
      }
```

Конструкторы копирования

```
// конструктор копирования
Point(const Point& other) : id(++count) {
      this->x = other.x;
      this->y = other.y;
      cout << "new copy object Point id = " << id
<< "\n";
}

Point& operator = (const Point& other) {
     this->x = other.x;
     this->y = other.y;
     std::cout << "!!! operator = (const Point&)
" << id << " " << other.id << "\n";
return *this;
}
```

Деструктор

Деструктор класса вызывается при уничтожении объекта. Имя деструктора аналогично имени конструктора, только в начале ставится знак тильды «**~**». Деструктор не имеет входных параметров.

```
~Point() {
      std::cout << "!!! delete object Point() id =
" << id << "\n";
}
```

Префиксный и постфиксный ++

```
// Define prefix increment operator.
Point& operator ++ () {
     this->x++;
     this->y++;
     return *this;
}

// Define postfix increment operator.
Point operator ++ (int) {
     Point temp = *this;
     this->x++;
     this->y++;
     return temp;
}
```

Ключевое слово const

Объект класса, не модифицируется программой напрямую (инкапсуляция!). Вместо этого используется та или иная открытая функция-член класса. Компилятор имеет возможность различать безопасные (те, которые не изменяют объект) и небезопасные (те, которые пытаются это сделать) функции-члены:

const Screen blankScreen;

blankScreen.display(); `// читает объект класса`

blankScreen.set('');* `// ошибка: модифицирует объект класса`

Можно указать, какие функции-члены не модифицируют объект, объявив их константными с помощью спецификатора **const.**

```
class MyClass {
public:
   int getX() const {
      return x;
   }
};
```

Задача

Требуется определить классы «**Точка на плоскости**», и класс «**Геометрический вектор**» и проверить их работу в main().

Для «**Точки на плоскости**» должны быть доступны:

- –Координаты по осям
- –Возможность копирования одной токи в другую
- –Переопределенный оператор присваивания
- –Методы записи/получения координат
- –Номер точки в программе по порядку и количество точек в настоящий момент времени

Для «**Вектора**»:

- –Создание из 2х точек на плоскости
- –Получение этих точек
- –Распечатка информации в формате (x1,y1)-(x2,y2)
- –Умножение вектора на число
- –Скалярное произведение векторов