Указатель this. Ключевое слово explicit

25 мая 2017 г.

Ключевое слово this

- константный указатель на объект, для которого вызван метод
- неявно передается во все нестатические методы класса
- инициализируется значением адреса объекта, для которого вызван метод
- нельзя получить адрес
- нельзя присвоить что-нибудь
- в константном методе указывает на константный объект

Ключевое слово this: пример 1

```
myObj.setX(3);
```

```
setX(&myObj, 3);
```

```
setX(this, 3);
```

```
void MyClass ::setX(int x)
{
  x_ = x;
  this->x_ = x;
  (*this).x_ = x;
}
```

this->имя_поля/имя_метода
*(*this).имя_поля/имя_метода*

Ключевое слово this: пример 2

```
class MyClass {
    /* … */
};

void MyClass::setX(int x_)
{
  x_ = x_;                  // Error и плохой стиль…
  this->x_ = x_;            // Ok
  (*this).x_ = x_;          // Ok
}
```

Задание

Создайте класс Line для хранения прямой, заданной уравнением вида y = kx + b. В классе должны быть следующие методы:

- конструктор, принимающий два числа типа float (коэффициенты k и b соответственно)
- конструктор, принимающий две точки типа Point
- аксессоры
- два метода, проверяющих, принадлежит ли некоторая точка данной прямой (точку можно задать отдельными координатами или объектом типа Point)
- функцию, которая проверяет, пересекаются ли две прямые

Ключевое слово explicit (C++11)

Запрещает неявное преобразование типов

в конструкторе:

- с одним параметром
- с несколькими параметрами, из которых только один не имеет значения по умолчанию

```
class MyClass {
        int x;
public:
        explicit MyClass(int newX) : x(newX) { };
};
```

Явный (конвертирующий) конструктор: инициализация

```
class Box {
    int side_;
public:
    Box(int side) : side_(side) { }
    ~Box(){ }
};

int main(int argc, char* argv[]){
    Box myBox(3);
    Box myBigBox1 = 100;  // Box myBigBox1(100)
    Box myBigBox2 = 'a';  // Box myBigBox2(97)
    return 0;
}
```

Явный (конвертирующий) конструктор: инициализация

```
class Box {
    int side1_, side2_;
public:
    Box(int side1, int side2 = 1) :
       side1_(side1), side2_(side2) { }
    ~Box(){ }
};

int main(int argc, char* argv[]){
    Box myBox(1, 7);
    Box myBigBox = 9.98;  // Box myBigBox(9, 1)
    return 0;
}
```

Явный (конвертирующий) конструктор: аргумент функции

```
class Box {
    int side_;
public:
    Box(int side) : side_(side) { }
    int getSide() const { return side_; }
};

void printBaseArea(Box b){
    std::cout << b.getSide() * b.getSide() << std::endl;
}

int main(int argc, char* argv[]){
    printBaseArea(2.99);  // 4
    return 0;
}
```

Явный (конвертирующий) конструктор

```
class Box {
    int side_;
public:
    explicit Box(int side) : side_(side) { }
    ~Box(){ }
};

int main(int argc, char* argv[]){
    Box myBox(3);
    Box myBigBox = 100;  // Error
    printBaseArea(2.99);  // Error
    return 0;
}
```

Какие конструкторы следовало бы объявить как explicit?

1) MyClass(int x, int y);
2) MyClass(int x);
3) MyClass(const int x);
4) MyClass(int x, int y = 10);
5) MyClass(int x, int y, int z = 0);
6) MyClass(int x = 0, int y = 0, int z = 0);
7) MyClass (char c, std::string s = "");
8) MyClass (std::string str, std::string s = "");

Задание

(1) Измените класс Line так, чтобы вызовы вида

```
Line line1 = 10;
checkIntersection(1, 99);
checkIntersection ('k', 't');
```

были невозможны.

(2) Добавьте функцию findCommonPoint(), которая для двух прямых, лежащих в одной плоскости, вычисляет точку их пересечения.

Вопросы?