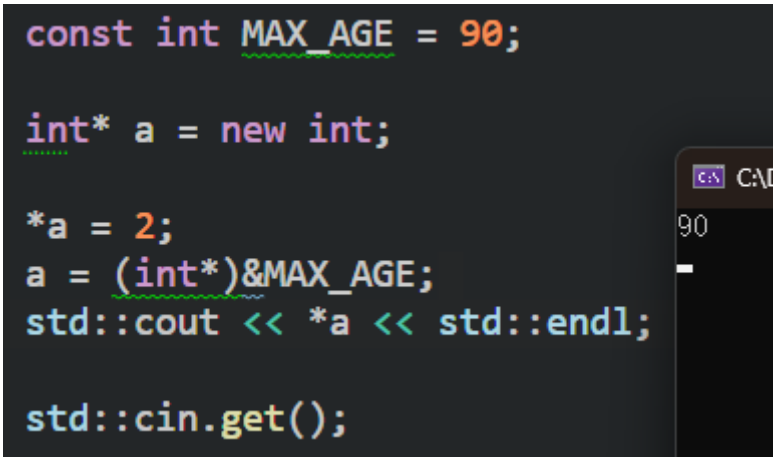
34 CONST in C++

1. const

我比较喜欢把`const`叫做一个"fake keyword"，因为它实际上在生成代码的时候并没有做什么。
它有点像类和结构体的可见性，是一种针对开发人员写代码的强制规则，为了让代码保持整洁的机制。
基本上const就是你做出承诺，某些东西是不变的，是不会改动的。但是它只是个承诺，而且你可以绕过或不遵守这个承诺，就像在现实生活中一样。



`*a = 2`为前面提过的解引用

```C++
const int* a = new int;
// 等同于int const* a = new int;
*a = 2;                                 // 报错，const使你不能修改a指针指向的内容
a = (int*)&MAX_AGE;
```

```C++
int* const a = new int;
*a = 2;
a = (int*)&MAX_AGE;                     // 报错，可以改变指针指向的内容，但是不能把指针自身重新赋值，让它指向其它东西
```

```C++
const int* const a = new int;       // 既不能改变指针的内容，也不能改变指针本身让它指向别处
*a = 2;
a = (int*)&MAX_AGE;
```

重点：看`const`在`*`的左边还是右边。

2. 类和方法中使用const

```C++
class Entity
{
private:
    int m_X, m_Y;
public:
    int GetX() const[ // 承诺了这个方法不会改变类，是一个只读方法
    {
        m_X = 2;        // 无法修改
        return m_X;
    }
};
```

`int* m_X, m_Y`，m_Y仍是一个int类型，要让两个都是指针需要写成`int* m_X, *m_Y`

const对象不能调用非const的成员函数，因为后者存在修改对象的可能

可以把变量设为`mutable int var`，这样即使在const方法中也可以修改