C++ 直接初始化和拷贝初始化

直接初始化和拷贝初始化

> 参考来源：《C++ primer（第五版）》

关于这两个概念，我相信有不少人会感到疑惑，今天我就带着大家一起弄清楚他们的区别。

首先我们介绍**直接初始化**：编译器使用普通的函数匹配来选择与我们提供的参数最匹配的构造函数。文字描述可能会让你们云里雾里，那我们直接看代码：

```
//先设计这样的一个类
class A{
public:
    A(){ cout << "A()" << endl; }
    A(int a){ cout << "A(int a)" << endl; }
    A(const A&){ cout << "A(const A&)" << endl; }
};
```

```
A a1(10);           //直接初始化
A a2(a1);           //直接初始化
```

结果为：

```
A(int a)
A(const A&)
```

通过上述的例子，我们可以发现直接初始化其实就是根据传入的参数选择最匹配的构造函数来进行初始化。但是其中有一个误区需要指出：直接初始化有可能调用拷贝构造函数！！！不能因为直接初始化不是拷贝初始化，就想当然认为其不可能调用拷贝构造函数。

A a = 1 <====> A a(1)
A a = b <====> A a(b)

接下来，让我们再看看拷贝初始化。


```
A a3 = a2;          //拷贝初始化1
A a4 = 2;           //拷贝初始化2
A a5 = A(3);        //拷贝初始化3
```

A a3(a2)
A a4(A(2))

结果为：

```
A(const A&)
A(int a)
A(int a)
```

上面介绍了在用=定义变量时发生的拷贝初始化，让我们仔细地来分析一下：

- 第1种情况：很好理解，是用a2去初始化a3，其实是`A a3(a2);`所以本次拷贝初始化是用了拷贝构造函数。
- 第2种情况：其实是相当于进行了两步：
  - `A a4 = A(2);`这一步称之为隐式类型转化。
  - `A a4(A(2));`其实也是调用了拷贝构造函数。
  - 但需要注意的是：有的编译器会做一定的优化，即在vs2013环境下，会将该种情况直接变为`A a4(2);`
- 第3种情况：其实就是`A a5(A(3));`同样在vs2013环境下会被优化为`A a5(3);`

拷贝构造初始除了用等号定义的情况，还有以下几种情况：

- 将一个对象作为实参传递给一个非引用类型的形参
  `void test(A a);`
- 从一个返回类型为非引用类型的函数返回一个对象
  `A test();`
- 用花括号列表初始化一个数组中的元素或一个聚合类中的成员
  `string str[2]{"lll","ttt"};`
- 某些类类型还会对他们所分配的对象使用拷贝初始化。如vector调用其insert或push成员。