C++11 构造函数和复制赋值——第 2 部分

BrainlessLabs.com

2014 年 9 月 3 日 MPL

C++11 构造函数和复制赋值——第 2 部分

涵盖的主题

在本节中，我们将了解更多 C++ 构造函数。将涵盖以下内容：

1. 复制构造函数
2. 复制赋值运算符
3. 移动构造函数

复制构造函数

定义

一般定义

对于X类的非模板的构造是一个拷贝构造函数，如果它的第一个参数是`type X&`，`const X&`，`volatile X&`或者`const volatile X&`，并且或者有没有其他参数，否则所有其他参数默认参数。

例子：

C++ 复制代码

```
class X {
public:
  X ( const X& );  // Copy Constructor
  X ( const X&, int = 1 );  // Copy Constructor
};
```

这里的复制构造函数是`public`. 此类称为可复制构造。

在下面的情况下会发生什么？

C++ 复制代码

```
class X {
public:
  X ( X& x, int i );
  X ( ) { };
};

X::X ( X& x, int i = 0) {

}
```

当我们在另一篇文章中深入研究时，我们将讨论更多这些案例。

像普通的构造函数一样，它们也可以被删除或默认。

C++ 复制代码

```
X ( const X& ) = default; // Forcing a copy constructor to be generated by the compiler
X ( const X& ) = delete; // Avoiding implicit default constructor
```

现在的问题是；我们总是需要在复制构造函数中传递引用吗?

好吧，让我们看看下面的案例：

C++ 复制代码

```
class X {
public:
  X ( const X v);
};

// Somewhere else in code
X a;
X b = a;
```

现在会发生什么? 这里 `X b = a;` 的代码看起来像`b( X v = a)`. 现在这又是一个复制构造函数调用。这将像这样递归地发生，直到我们用完堆栈空间并且程序将崩溃。所以这是不允许的。这将抛出错误，告诉这是非法的复制构造函数。

一个类可以有多个复制构造函数，例如：

C++ 复制代码

```
class X {
public:
  X ( const X& );
  X ( X& );
};
```

永远不会实例化成员函数模板以生成复制构造函数签名。喜欢：

C++ 复制代码

```
class X {
public:
 template<typename T>
 X ( T );
 X ( );
};

X a;
void foo ( ) {
 X b ( a ); // does not instantiate the member template to produce X::X<X>(X);
}
```

在这种情况下，使用隐式声明的复制构造函数。

原因如下：

1. 模板可以被实例化以模仿复制构造函数，但根据标准语句，它不是复制构造函数（如果您还没有再次看到定义，它明确指出是非模板化的）。所以这意味着该类没有定义一个。
2. 编译器现在在正确的情况下生成一个复制构造函数。所以说编译器也实例化了这个模板化代码。Then when the time to choose comes, the over loading resolution chooses the non templated function rather than the templated one as it does for other functions.

隐式声明的复制构造函数

如果没有为类类型提供用户定义的复制构造函数，编译器将始终声明具有以下类型的复制构造函数：

C++ 复制代码

```
class X {
public:
  inline X ( const X& );  // It is public and inline
};
```

在`const`如果满足以下条件修改为仅适用于：

- “ **X”的**所有直接和虚拟基类 都有复制构造函数，它们的第一个参数是对`const`或对的引用`const volatile`。
- “ **X** ”的所有非静态成员都有复制构造函数，其中引用`const`或`const volatile`作为它们的第一个参数。

一些事实

- 非联合类的隐式定义复制构造函数X执行其基类和成员的成员方式复制。
- 初始化的顺序与用户定义的构造函数中基类和成员的初始化顺序相同。
- 联合的隐式定义的复制构造函数X复制 的对象表示X。

删除隐式声明的复制构造函数

如果满足以下条件，则类 X 的隐式声明或默认复制构造函数被定义为已删除或在更简单的 C++11 前术语未定义（编译器无法定义它们）：

- X 具有无法复制的非静态数据成员（具有已删除、不可访问或不明确的复制构造函数）
- X 具有无法复制的直接或虚拟基类（已删除、不可访问或不明确的复制构造函数）
- X 具有带有已删除或不可访问的析构函数的直接或虚拟基类。

普通复制构造函数

普通复制构造函数是一个构造函数，它创建参数的对象表示的逐字节副本，并且不执行其他操作。可以通过手动复制对象表示来复制具有普通复制构造函数的对象，例如使用`std::memmove`. 所有与 C 语言兼容的数据类型（POD 类型）都可以轻松复制。

X如果以下所有条件都为真，则类的复制构造函数是微不足道的：

- 不是用户提供的（即隐式定义或默认的），如果是默认的，则其签名与隐式定义相同
- X 没有虚成员函数
- X 没有虚拟基类
- 为每个直接基选择的复制构造函数X是微不足道的
- 为每个非静态类类型（或类类型数组）成员选择的复制构造函数X是微不足道的

隐式定义的复制构造函数

如果隐式声明的复制构造函数既不删除也不平凡，则由编译器定义（即生成并编译函数体）。

- 对于联合类型，隐式定义的复制构造函数复制对象表示（如 by `std::memmove`）。
- 对于非联合类类型（`class`和`struct`），构造函数使用直接初始化按初始化顺序执行对象的基类和非静态成员的完整成员复制。

用法

C++ 复制代码

```
class X {
public:
  X ( const X& );  // Copy Constructor
};

X a;
// General initialization
X b = a;
X b ( a );
// Function argument passing
void f ( const X v ); // Function definition
f ( a );
// Return
void f ( ) {
  X a;
  return a;
}
```

如果一个类 X 只有一个参数为 的复制构造函数`type X&`，一个初始化器`type const X` 或`volatile X`不能初始化类型（可能是 cv 限定的）的对象`X`。

C++ 复制代码

```
class X {
public:
  X ( ) ; // default constructor
  X ( X& ) ; // copy constructor with a non-const parameter
};
const X a;
X b = a;  // error: X::X(X&) cannot copy a into b
```

复制赋值运算符

定义

用户声明的复制赋值运算符`X::operator=`是类的一个非静态非模板成员函数，`X`只有一个类型为 `X`、`X&`、`const X&`、`volatile X&`或 的参数`const volatile X&`。与任何函数和构造函数一样，它们可以是默认的或删除的。

C++ 复制代码

```
class X {
public:
  X& operator=( X );
  X& operator=( const X& );
  X& operator=( const X& ) = default;
  X& operator=( const X& ) = delete;
};
```

本文最初发表于http://brainlesslabs.com/cpp_cxx_cxx11_cpp11-constructors-copy-assignment-p2?utm_source=rss&utm_medium=rss&utm_campaign=cpp_cxx_cxx11_cpp11-constructors-copy-assignment-p2

执照

分享

关于作者

BrainlessLabs.com

建筑师
印度

手表该会员

我喜欢探索技术的不同方面。尝试新事物，并获得快乐。我的兴趣是编程语言和成像。但在其他事情上工作也不难。算法让我在喝咖啡休息时感到高兴。

我基本上用 C++ 编写代码，但 JAVA 对我来说并不陌生。我也知道很少的脚本语言。基本上我觉得知道一门编程语言只是一个无...

展示更多

评论和讨论

添加评论或问题

电子邮件提醒 Search Comments

第一 页上一页 下一页

你的part1在哪里?
Southmountain 29-Jan-20 4:48

我的5票
kanalbrummer 4-Sep-14 12:33

回复：我的5票
BrainlessLabs.com 10-Sep-14 16:07

刷新 1

一般 新闻 建议 问题 错误 答案 笑话 赞美 咆哮 管理员

使用Ctrl+Left/Right 切换消息，Ctrl+Up/Down 切换主题，Ctrl+Shift+Left/Right 切换页面。