# 编程之道，自强之路！

更多分享尽在：https://imqiuyang.github.com



个人资料

Jofranks

访问： 118465次
积分： 2247
等级： BLOG 5
排名： 第8587名

原创： 83篇
转载： 0篇
译文： 0篇
评论： 108条







# [置顶] 【C/C++学习】之十七、C++11中我们需要关注的

分类： C/C++ C++11 2013-12-19 17:03 1358人阅读 评论(0) 收藏 举报

前面的博客一只在说c++，现在来看一下c++新标准，在维基百科中已经给出了详细说明，可以去看一下。

## C++11的设计目标是：

1.使得c++成为更好的适用于系统开发及库开发的语言。

2.使得c++成为更易于教学的语言。

3.保证语言的稳定性，以及和C++03及c语言的兼容性。

## c++11 的增强点，主要包括：

1.通过内存模型，线程，原子操作等来支持本地并行编程。

2.通过统一初始化表达式、auto、declytype、移动语义等来统一对泛型编程的支持。

3.通过constexpr，POD等更好地支持系统编程。

4.通过内联命名空间，集成构造函数和右值引用等，以更好地支持库的构建。

下面我们把特性里面涉及的名词来解释一下：

**1**、统一初始化表达式：

c++11用大括号统一了初始化。在c++11中，这种初始化方法叫做：初始化列表 initializer list

```cpp
int a[] {12, 22};
vector<int> a{ 1, 2, 3 };
map<int, float> a = { { 1, 1.0f }, { 2, 2.0f } };
```

在c++11中，我们可以通过以下几种形式来完成初始化的工作：

```cpp
int a = 1 + 1;
int a = { 1 + 1 };
int a{ 1 + 1 };
int a(1 + 1);
```

**2**、就地声明：

在类中，我们直接使用等号来初始化静态成员常量的方式，就是就地初始化。如：

```cpp
class a{
public:
    a() :b(0) {}
private:
    int b;
    const static int b = 0;
};
```

要注意的是，需要静态的常量成员。 还有一点就是需要整形或者枚举类型。

但是在我们c++11中，允许对非静态成员变量进行就地初始化：

```cpp
class a{
    int b = 1;
    int c{ 2 };
};
```

要注意的是，这里不能使用圆括号进行就地初始化。
在类中会涉及到构造函数初始化列表的问题，他跟就地初始化并不冲突，但是构造函数初始化列表的效果总是优先于就地初始化。

**3**、自定义类初始化列表：
可以通过#include<initializer_list>头文件中的initializer_list类模板支持初始化列表。
下面来看一下对类、函数和操作符进行初始化的例子：

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

class C{
public:
    C(initializer_list<pair<string, int>> l, initializer_list<int> m, initializer_list<int> n)
    {
        auto i = l.begin();
        for (; i != l.end(); ++i)
        {
            data.push_back(*i);
        }
        auto j = m.begin();
        for (; j != m.end(); ++j)
            idx.push_back(*j);

        auto s = n.begin();
        for (; s != n.end(); ++s)
            d.push_back(*s);
    }

    C & operator[](initializer_list<int> l)
    {
        for (auto i = l.begin(); i != l.end(); ++i)
        {
            idx.push_back(*i);
        }
        return *this;
    }

    C & operator = (int v)
    {
        if (idx.empty() != true)
        {
            for (auto i = idx.begin(); i != idx.end(); ++i)
            {
                d.resize((*i > d.size()) ? *i : d.size());
                d[*i - 1] = v;
            }
            idx.clear();
        }
        return *this;
    }

    void Fun(initializer_list<int> l)
    {
        for (auto i = l.begin(); i != l.end(); ++i)
        {
            cout << *i << " ";
        }
        cout << endl;
```

```
    }
private:
    vector<pair<string, int>> data;
    vector<int> idx;
    vector<int> d;
};

int main(void)
{
    C ctr{ { { "s", 1 }, { "e", 2 } }, { 3, 4 }, { 5, 6 } };
    ctr.Fun({ 1, 2 });
    ctr.Fun({});

    ctr[{2, 3, 4}] = 7;
    ctr[{1, 3, 4, 5}] = 2;
};
```

### 4、auto类型

先来看一下代码：

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    auto name = "hello world \n";
    cout << name;

    char* name1 = "hello world \n";
    cout << name1;

    return 0;
}
```

根据上面的代码可以知道，auto和string的效果是一样的，是的，auto关键字要求编译器对变量name的类型进行自动推导。

在之前版本的C++中，auto的意思是具有自动存储期的局部变量，然而，一般情况下在函数里没有声明为static的变量总是具有自动储存期的局部变量。

在c++11中，auto声明变量的类型必须由编译器在编译时期推导得到。

### 5、declytype

declytype与auto类似，也可以进行类型推导，但是使用方式有一定区别。

declytype以一个普通的表达式为参数，返回该表达式的类型。他也是作为一个类型指示符，在编译时进行推导。看代码：

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    int i;
    decltype(i) j = 0;
    cout << typeid(j).name() << endl;

    int a;
    double b;
    decltype(a + b) c;
    cout << typeid(c).name() << endl;
    return 0;
}
```

### 6、继承和委托构造函数

c++中，继承类无法使用基类的非虚函数，除非显式使用。

而在c++11中，允许使用using声明来声明继承类的构造函数。如：

```cpp
class A{
public:
    A() {}
    A(int i) {}
};

class B :public A{
public:
    using A::A;     //继承构造函数
    virtual void s(int i)
    {
        cout << "B:" << i << endl;
    }
};
```

在c++11中，标准继承构造函数被设计为跟派生类中的各种类默认函数（默认构造，析构，copy构造等）一样，都是隐式声明的。

c++11中构造函数可以调用同一个类的另一个构造函数，通过委派其他构造函数，过构造函数的类会更加容易编写：

```cpp
class A{
public:
    A(int a)
    {
        this->a = a;
    }
    A() :A(0) {}
    int geta()
    {
        return a;
    }
private:
    int a;
};
int main(void)
{
    A b;
    cout << b.geta() << endl;
    return 0;
}
```

委派构造函数：委派函数将构造的任务委派给了目标构造函数来完成这样一种类构造的方式。

**7**、右值引用

在c++11中，右值是由两个概念构成的，一个是将亡值，另一个则是纯右值。

在c++11中，右值引用就是对一个右值进行引用的类型。它能够以non-const值的方式传入，允许对象去改动他。

如：T&& a = returna();

这里a是右值引用，他的值等于returna返回的临时变量的值。

**8**、移动语义

来看一张图：

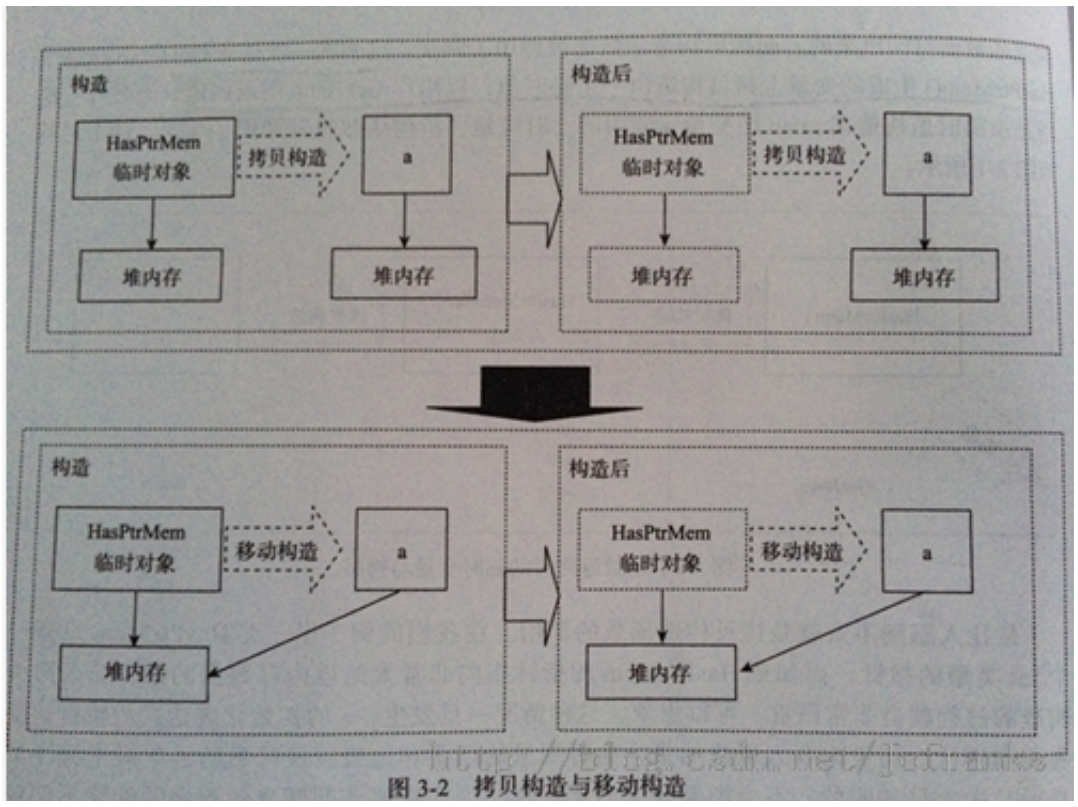

图 3-2 拷贝构造与移动构造

看到上图，应该可以看得出上半部分是copy构造函数咯。而下面的部分就是c++11中的新特性，叫做移动构造函数。

移动移动，移为己用，他偷走了临时变量中的资源。

```cpp
class string
{
    string (string&&); //move constructor
    string&& operator=(string&&); //move assignment operator
};
```

**9、pod**

c++11中，学习pod的概念是非常有用的。

plain old data。

plain表示了pod是一个普通的类型。

old体现了他与c的兼容性。

在pod中，有两个基本概念：平凡的+标准布局的。

**10**、内联命名空间

c++中，名字空间的目的是分割全局共享的名字空间。

在c++11中，引入了“内联命名空间” inline namespace 就可以声明一个内联命名空间。

在内联空间中，允许程序员在父命名空间定义或者特化名字空间的模板。

当你需要长期维护，发布不同的版本的时候，可以用一下内联名字空间。 后续还会介绍。

**11、nullptr**

先来看一下NULL宏的定义：

```cpp
#undef NULL
#if defined(_cplusplus)
#define NULL 0
#else
#define NULL ((void*)0)
#endif
```

从代码中可以看到NULL可以是0或者是((void*)0)，无论哪种都会出现这样那样的麻烦，所以在c++11中就出现了nullptr这个强类型。

```cpp
void f(int); //#1
void f(char *);//#2
//C++03
f(0); //二义性
```

```
//C++11
f(nullptr) //无二义性，调用f(char*)
```

在c++11用，我们要用nullptr来初始化指针。

### 12、override，final

final关键字的作用是使派生类不可覆盖它所修饰的虚函数。

override关键字的作用是保证编译器辅助检查。

### 13、lambda

他来源于函数式编程，也就是你在使用的地方定义，也就是“闭包”。

来看一下C++11中lambda的定义：

```cpp
[capture](parameters) mutable->return-type {statement}
```

[capture]：捕捉列表。

(parameters)：参数列表。

mutable：mutable修饰符。

->return-type：返回类型。

{statement}：函数体。

看一下实例：

```cpp
int main()
{
   char s[]="Hello World!";
   int Uppercase = 0; //modified by the lambda
   for_each(s, s+sizeof(s), [&Uppercase] (char c) {
    if (isupper(c))
     Uppercase++;
    });
 cout << Uppercase << " uppercase letters in: " << s <<endl;
}
```

c++11中，引入lambda有两个原因：

1）、可以定义匿名函数。

2）、编译器会把其转换成函数对象。

### 14、static_assert

c++中assert宏只有在程序运行时才能起作用，#error只有在编译器预处理时才能起作用。

在c++11中加入了static_assert宏来实现编译时期的断言：静态断言。

他的声明方式：

static_assert(编译时可推断出结果的表达式，一个简单的多字节的字符串)。

第一个参数：断言表达式，必须是编译时可知的。

第二个参数：警告信息。

比如：

```cpp
static_assert(sizeof(int) == 4, " 32-bit");
```

c++11中还有很多新变化，非常好用，c++11是一门新语言。

后续博客会进行详细解释。

——2013.12.19

上一篇 【啊哈，算法】之十、后缀数组，求最长重复子串

下一篇 【C\C++学习】之十八、C++11六大函数（构造函数，移动构造函数，移动赋值操作符，复制构造函数，赋值操作符，析构函数）