网名还没想好

posts - 366, comments - 52, trackbacks - 0, articles - 2

导航

博客园
首页
新随笔
联系
XML 订阅
管理

公告

昵称：网名还没想好
园龄：5年4个月
粉丝：100
关注：96
+加关注

< 2017年4月 >

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	**24**	25	26	27	28	29
30	1	2	3	4	5	6

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

C/C++(63)
Linux(62)
OBJC(13)
OPENGL(7)
SQL(15)
web(49)
网络(32)

随笔档案

2017年2月 (3)
2017年1月 (1)
2016年12月 (2)
2016年10月 (6)
2016年9月 (2)
2016年8月 (10)

C++ explicit关键字详解

Posted on 2014-03-29 16:03 网名还没想好 阅读(11174) 评论(4) 编辑 收藏

首先，C++中的explicit关键字只能用于修饰只有一个参数的类构造函数，它的作用是表明该构造函数是显示的，而非隐式的，跟它相对应的另一个关键字是implicit，意思是隐藏的,类构造函数默认情况下即声明为implicit(隐式).

那么显示声明的构造函数和隐式声明的有什么区别呢？我们来看下面的例子：

```
1.  class CxString  // 没有使用explicit关键字的类声明，即默认为隐式声明
2.  {
3.  public:
4.      char *_pstr;
5.      int _size;
6.      CxString(int size)
7.      {
8.          _size = size;                // string的预设大小
9.          _pstr = malloc(size + 1);    // 分配string的内存
10.         memset(_pstr, 0, size + 1);
11.     }
12.     CxString(const char *p)
13.     {
14.         int size = strlen(p);
15.         _pstr = malloc(size + 1);    // 分配string的内存
16.         strcpy(_pstr, p);            // 复制字符串
17.         _size = strlen(_pstr);
18.     }
19.     // 析构函数这里不讨论，省略...
20. };
21.
22.     // 下面是调用:
23.
24.     CxString string1(24);     // 这样是OK的，为CxString预分配24字节的大小的内存
25.     CxString string2 = 10;    // 这样是OK的，为CxString预分配10字节的大小的内存
26.     CxString string3;         // 这样是不行的，因为没有默认构造函数，错误为：“CxString”：没有合适的默认构造函数可用
27.     CxString string4("aaaa"); // 这样是OK的
28.     CxString string5 = "bbb"; // 这样也是OK的，调用的是CxString(const char *p)
29.     CxString string6 = 'c';   // 这样也是OK的，其实调用的是CxString(int size)，且size等于'c'的ascii码
30.     string1 = 2;              // 这样也是OK的，为CxString预分配2字节的大小的内存
31.     string2 = 3;              // 这样也是OK的，为CxString预分配3字节的大小的内存
32.     string3 = string1;        // 这样也是OK的，至少编译是没问题的，但是如果析构函数里用free释放_pstr内存指针的时候可能会报错，完整的代码必须重载运算符"="，并在其中处理内存释放
```

上面的代码中，"CxString string2 = 10;" 这句为什么是可以的呢？在C++中，如果的构造函数只有一个参数时，那么在编译的时候就会有一个缺省的转换操作:将该构造函数对应数据

2016年7月 (5)
2016年6月 (14)
2016年5月 (19)
2016年4月 (9)
2016年3月 (7)
2016年1月 (7)
2015年12月 (22)
2015年11月 (9)
2015年10月 (14)
2015年9月 (2)
2015年8月 (10)
2015年7月 (2)
2015年6月 (3)
2015年5月 (6)
2015年4月 (5)
2015年3月 (1)
2015年2月 (2)
2015年1月 (1)
2014年11月 (7)
2014年10月 (4)
2014年9月 (1)
2014年8月 (3)
2014年7月 (9)
2014年6月 (6)
2014年5月 (14)
2014年4月 (9)
2014年3月 (6)
2013年8月 (1)
2012年9月 (1)
2012年7月 (1)
2012年6月 (2)
2012年5月 (11)
2012年4月 (54)
2012年3月 (21)
2012年2月 (30)
2012年1月 (12)
2011年12月 (9)
2011年11月 (3)

最新评论

1. Re:C++ explicit关键字详解

在我这里，加上explicit后，string3 = string1;是没问题的。这里应该是调用了该类的默认拷贝构造函数。我又加上了一个CxString(float size) {}这样的话CxStr.....

--chaosink

2. Re:javaScript & jquery完美判断图片是否加载完毕

@网名还没想好　这个代码能在手机端运行吗？

--曲终人未散

类型的数据转换为该类对象. 也就是说 "CxString string2 = 10;" 这段代码, 编译器自动将整型转换为CxString类对象, 实际上等同于下面的操作:

```
CxString string2(10);
或
CxString temp(10);
CxString string2 = temp;
```

但是, 上面的代码中的_size代表的是字符串内存分配的大小, 那么调用的第二句 "CxString string2 = 10;" 和第六句 "CxString string6 = 'c';" 就显得不伦不类, 而且容易让人疑惑. 有什么办法阻止这种用法呢? 答案就是使用explicit关键字. 我们把上面的代码修改一下, 如下:

```
class CxString  // 使用关键字explicit的类声明, 显示转换
{
public:
    char *_pstr;
    int _size;
    explicit CxString(int size)
    {
        _size = size;
        // 代码同上, 省略...
    }
    CxString(const char *p)
    {
        // 代码同上, 省略...
    }
};

    // 下面是调用:

    CxString string1(24);     // 这样是OK的
    CxString string2 = 10;    // 这样是不行的, 因为explicit关键字取消了隐式转换
    CxString string3;         // 这样是不行的, 因为没有默认构造函数
    CxString string4("aaaa"); // 这样是OK的
    CxString string5 = "bbb"; // 这样也是OK的, 调用的是CxString(const char *p)
    CxString string6 = 'c';   // 这样是不行的, 其实调用的是CxString(int size), 且size等于'c'的ascii码, 但explicit关键字取消了隐式转换
    string1 = 2;              // 这样也是不行的, 因为取消了隐式转换
    string2 = 3;              // 这样也是不行的, 因为取消了隐式转换
    string3 = string1;        // 这样也是不行的, 因为取消了隐式转换, 除非类实现操作符"="的重载
```

explicit关键字的作用就是防止类构造函数的隐式自动转换.

上面也已经说过了, explicit关键字只对有一个参数的类构造函数有效, 如果类构造函数参数大于或等于两个时, 是不会产生隐式转换的, 所以explicit关键字也就无效了. 例如:

```
class CxString  // explicit关键字在类构造函数参数大于或等于两个时无效
{
public:
    char *_pstr;
    int _age;
    int _size;
    explicit CxString(int age, int size)
```

```
    {
        _age = age;
        _size = size;
        // 代码同上，省略...
    }
    CxString(const char *p)
    {
        // 代码同上，省略...
    }
};

    // 这个时候有没有explicit关键字都是一样的
```

但是，也有一个例外，就是当除了第一个参数以外的其他参数都有默认值的时候，explicit关键字依然有效，此时，当调用构造函数时只传入一个参数，等效于只有一个参数的类构造函数，例子如下：

```
class CxString  // 使用关键字explicit声明
{
public:
    int _age;
    int _size;
    explicit CxString(int age, int size = 0)
    {
        _age = age;
        _size = size;
        // 代码同上，省略...
    }
    CxString(const char *p)
    {
        // 代码同上，省略...
    }
};

    // 下面是调用:

    CxString string1(24);     // 这样是OK的
    CxString string2 = 10;    // 这样是不行的，因为explicit关键字取消了隐式转换
    CxString string3;         // 这样是不行的，因为没有默认构造函数
    string1 = 2;              // 这样也是不行的，因为取消了隐式转换
    string2 = 3;              // 这样也是不行的，因为取消了隐式转换
    string3 = string1;        // 这样也是不行的，因为取消了隐式转换，除非类实现操作符"="的重载
```

以上即为C++ explicit关键字的详细介绍.

分类: C/C++

« 上一篇：C++类够做函数初始化列表
» 下一篇：[C++]复制构造函数、赋值操作符与隐式类类型转换

Feedback

#1楼
2016-12-05 18:05 by 雪忆寒

可以 很详细

支持(0) 反对(0)

#2楼
2017-01-05 10:21 by Lijiajia0704

很透彻，赞一个

支持(0) 反对(0)

#3楼
2017-03-24 17:46 by 喜欢兰花山丘

赞
感觉上面代码段

```
CxString(const char *p)
{
    int size = strlen(p);
    _pstr = malloc(size + 1);    // 分配string的内存
    strcpy(_pstr, p);            // 复制字符串
    _size = strlen(_pstr);
}
```

换成下面 才可以，毕竟C++是"类型安全"的

```
CxString(const char *p)
{
    _size = strlen(p);
    _pstr = static_cast<char *>(malloc(_size + 1));    // 分配string的内存
    assert(_pstr);
    strcpy(_pstr, p);            // 复制字符串
}
```

支持(0) 反对(0)

#4楼
2017-04-20 23:46 by chaosink

在我这里，加上explicit后，
string3 = string1;
是没问题的。
这里应该是调用了该类的默认拷贝构造函数。

我又加上了一个
CxString(float size) {}
这样的话
CxString s = 1.1;
是没问题的，而且
CxString s = 1;
也可以了。

结论：explicit只是限制构造函数或者类型转换函数只能以括号的方式调用，对其他函数无影响。企图用隐式转换时，无视所有explicit函数，并调用一个最匹配的函数。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。