

个人资料



Eric_Jo

访问：189984次

积分：1418

等级：BLOG > 4

排名：第12879名

原创：23篇 转载：18篇
译文：0篇 评论：162条

文章搜索

文章分类

- C++/CLI (1)
- C/C++ 开发 (8)
- 形式语言 (1)
- 职场人生 (6)
- 职场技能 (6)
- 设计模式 (6)
- 调试&测试 (5)
- 面向对象 (3)

从零开始掌握iOS8开发技术（Swift版） 那些年我们追过的Wrox精品红皮计算机图书 CSDN学院--学习礼包大派送 CSDN JOB带你坐飞机回家过年

关于C++ const 的全面总结

分类：C/C++ 开发

2009-04-30 08:54

104591人阅读

评论(105)

收藏

举报

c++ function class 编译器 fun 存储

C++中的const关键字的用法非常灵活，而使用const将大大改善程序的健壮性，本人根据各方面查到的资料进行总结如下，期望对朋友们有所帮助。

Const 是C++中常用的类型修饰符,常类型是指使用类型修饰符const说明的类型，常类型的变量或对象的值是不能被更新的。

一、Const作用

如下表所示：

No.	作用	说明	参考代码
1	可以定义const常量		const int Max = 100;
2	便于进行类型检查	const常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查，而对后者只进行字符替换，没有类型安全检查，并且在字符替换时可能会产生意料不到的错误	<pre>void f(const int i) {} //对传入的参数进行类型检查，不匹配进行提示</pre>
3	可以保护被修饰的东西	防止意外的修改，增强程序的健壮性。	<pre>void f(const int i) { i=10;//error! } //如果在函数体内修改了i，编译器就会报错</pre>
4	可以很方便地进行参数的调整和修改	同宏定义一样，可以做到不变则已，一变都变	
5	为函数重载提供了一个参考		<pre>class A { void f(int i) {...} //一个函数</pre>

项目管理 (5)
项目管理-软件外包 (2)

文章存档

2011年09月 (7)
2010年08月 (2)
2010年03月 (1)
2010年02月 (5)
2009年11月 (1)

展开

阅读排行

关于C++ const 的全面总 (104559)
C++结构体实例和类实例 (17694)
C++/CLI简介（什
更多相关资源：
虚析构造函数解析

HP大中华区总裁孙振耀记 (5097)
OO设计原则 -- Liskov Su (4114)
C++经典书籍推荐 (3799)
OO设计原则 -- Depend (3402)
利用MAP文件精确定位代 (3320)
C/C++中“空语句”的说明 (2705)

评论排行

关于C++ const 的全面总 (105)
OO设计原则 -- Depend (8)
C++结构体实例和类实例 (5)
C++经典书籍推荐 (4)
虚析构造函数解析 (4)
利用MAP文件精确定位代 (4)
C++/CLI简介（什么是C+ (3)
最快速度找到内存泄漏 (3)
OO设计原则 -- Liskov Su (3)
C/C++中“空语句”的说明 (3)

推荐文章

			<pre>void f(int i) const {.....} //上一个函数的重载 };</pre>
6	可以节省空间，避免不必要的内存分配	const定义常量从汇编的角度来看，只是给出了对应的内存地址，而不是象#define一样给出的是立即数，所以，const定义的常量在程序运行过程中只有一份拷贝，而#define定义的常量在内存中有若干个拷贝	<pre>#define PI 3.14159 //常量宏 const doulbe Pi=3.14159; //此时并未将Pi放入ROM中 double i=Pi; //此时为Pi分配内存，以后不再分配！ double I=PI; //编译期间进行宏替换，分配内存 double j=Pi; //没有内存分配 double J=PI; //再进行宏替换，又一次分配内存！</pre>
7	提高了效率	编译器通常不为普通const常量分配存储空间，而是将它们保存在符号表中，这使得它成为一个编译期间的常量，没有了存储与读	
		高	

二、Const的使用

1、定义常量

(1)const修饰变量，以下两种定义形式在本质上是一样的。它的含义是：const修饰的类型为TYPE的变量value是不可变的。

```
TYPE const ValueName = value;
const TYPE ValueName = value;
```

(2)将const改为外部连接,作用于扩大至全局,编译时会分配内存,并且可以不进行初始化,仅仅作为声明,编译器认为在程序其他地方进行了定义.

```
extend const int ValueName = value;
```

2、指针使用CONST

(1)指针本身是常量不可变

```
(char*) const pContent;
const (char*) pContent;
```

(2)指针所指向的内容是常量不可变

```
const (char) *pContent;
(char) const *pContent;
```

(3)两者都不可变

```
const char* const pContent;
```

(4)还有其中区别方法，沿着*号划一条线：

如果const位于*的左侧，则const就是用来修饰指针所指向的变量，即指针指向为常量；

如果const位于*的右侧，const就是修饰指针本身，即指针本身是常量。

- * 浅析总结 Java 内部类的一些使用与梳理
- * Qt for iOS, Qt 与Objective C 混合编程
- * 教你写Android ImageLoader框架之基本架构
- * 三大运营商的游戏“刷金”漏洞解决方案
- * 百度地图开发（二）之添加覆盖物 + 地理信息和后地理信息

最新评论

- 关于C++ const 的全面总结
yokeyoke: 分享的很全面，也易读懂，赞一个！
- OO设计原则 — Single Responsi
fengtaijun: 那什么，有种醍醐灌顶的感觉，学习了。
- 关于C++ const 的全面总结
回首撒哈拉: 求下一句的注解：
const AP_HAL::HAL& hal = AP_HAL_BOARD_DRIV...
- OO设计原则 -- Dependency Inve
rush_09: 非常棒 原文解释到位
- 关于C++ const 的全面总结
小布: 楼主好心值得鼓励，只不过做技术已经要有严谨的作风，否则容易误导后来者，毕竟知识是先入为主，第一印象很...
- 关于C++ const 的全面总结
endlessbest: 非常不错哈，感谢分享
- 关于C++ const 的全面总结
eKeeper_Lirz: 作者写得很全面，整理得很不错。但有种怕被误导的心理。望楼主重新整理验证后，再@all.
- 关于C++ const 的全面总结
yk610900880: 要让我这么说你呢？拿来主义也是要赠别一下正确与否可以么？ ----编译器通常不为普通const常量分配...
- 关于C++ const 的全面总结
shallyhanlu: 菜鸟学习了
- 关于C++ const 的全面总结
cilin1991: 编译器通常不为普通const常量分配存储空间，而是将它们保存在符号表中，这使得它成为一个编译期间的常...

3、函数中使用CONST

(1)const修饰函数参数

- a. 传递过来的参数在函数内不可以改变(无意义，因为Var本身就是形参)

```
void function(const int Var);
```

- b. 参数指针所指内容为常量不可变

```
void function(const char* Var);
```

- c. 参数指针本身为常量不可变(也无意义，因为char* Var也是形参)

```
void function(char* const Var);
```

- d. 参数为引用，为了增加效率同时防止修改。修饰引用参数时：

```
void function(const Class& Var); //引用参数在函数内不可以改变
```

```
void function(const TYPE& Var); //引用参数在函数内为常量不可变
```

这样的 一个const引用传递和最普通的函数按值传递的效果是一模一样的, 他禁止对引用的对象的一切修改, 唯一不同的是按值传递会先建立一个类对象的副本, 然后传递过去, 而它直接传递地址, 所以这种传递比按值传递更有效. 另外只有引用的const传递可以传递一个临时对象, 因为临时对象都是const属性, 且是不可见的, 他短时间存在一个局部域中, 所以不能使用指针, 只有引用的const传递能够捕捉到这个家伙.

(2)const 修饰函数返回值

const修饰函数返回值其实用的并不是很多，它的含义和const修饰普通变量以及指针的含义基本相同。

- a. const int fun1() //这个其实无意义，因为参数返回本身就是赋值。

- b. const int * fun2() //调用时 const int *pValue = fun2();
//我们可以把fun2() 看作成一个变量，即指针内容不可变。

- c. int* const fun3() //调用时 int * const pValue = fun2();
//我们可以把fun2() 看作成一个变量，即指针本身不可变。

一般情况下，函数的返回值为某个对象时，如果将其声明为const时，多用于操作符的重载。通常，不建议用const修饰函数的返回值类型为某个对象或对某个对象引用的情况。原因如下：如果返回值为某个对象为const（const A test = A 实例）或某个对象的引用为const（const A& test = A实例） ，则返回值具有const属性，则返回实例只能访问类A中的公有（保护）数据成员和const成员函数，并且不允许对其进行赋值操作，这在一般情况下很少用到。

4、类相关CONST

(1)const修饰成员变量

const修饰类的成员函数，表示成员常量，不能被修改，同时它只能在初始化列表中赋值。

```
class A
{
    ...
    const int nValue; //成员常量不能被修改
    ...
    A(int x): nValue(x) { } ; //只能在初始化列表中赋值
}
```

(2) const修饰成员函数

const修饰类的成员函数，则该成员函数不能修改类中任何非const成员函数。一般写在函数的最后来修饰。

```
class A
{
    ...

    void function() const; //常成员函数， 它不改变对象的成员变量。

    //也不能调用类中任何非const成员函数。
}
```

对于const类对象/指针/引用，只能调用类的const成员函数，因此，const修饰成员函数的最重要作用就是限制对于const对象的使用。

- a. const成员函数不被允许修改它所在对象的任何一个数据成员。
- b. const成员函数能够访问对象的const成员，而其他成员函数不可以。

(3) const修饰类对象/对象指针/对象引用

- const修饰类对象表示该对象为常量对象，其中的任何成员都不能被修改。对于对象指针和对象引用也是一样。
- const修饰的对象，该对象的任何非const成员函数都不能被调用，因为任何非const成员函数会有修改成员变量的企图。

例如：

```
class AAA
{
    void func1();
    void func2() const;
}

const AAA aObj;
aObj.func1(); ×
aObj.func2(); 正确

const AAA* aObj = new AAA();
aObj-> func1(); ×
aObj-> func2(); 正确
```

三、将Const类型转化为非Const类型的方法

采用const_cast 进行转换。

用法：const_cast <type_id> (expression)

该运算符用来修改类型的const或volatile属性。除了const 或volatile修饰之外， type_id和expression的类型是一样的。

- 常量指针被转化成非常量指针，并且仍然指向原来的对象；
- 常量引用被转换成非常量引用，并且仍然指向原来的对象；

- 常量对象被转换成非常量对象。

四、使用const的一些建议

- 要大胆的使用const，这将给你带来无尽的益处，但前提是你必须搞清楚原委；
- 要避免最一般的赋值操作错误，如将const变量赋值，具体可见思考题；
- 在参数中使用const应该使用引用或指针，而不是一般的对象实例，原因同上；
- const在成员函数中的三种用法（参数、返回值、函数）要很好的使用；
- 不要轻易的将函数的返回值类型定为const；
- 除了重载操作符外一般不要将返回值类型定为对某个对象的const引用；
- 任何不会修改数据成员的函数都应该声明为const 类型。

五、补充重要说明

- 类内部的常量限制：使用这种类内部的初始化语法的时候，常量必须是被一个常量表达式初始化的整型或枚举类型，而且必须是static和const形式。
- 如何初始化类内部的常量：一种方法就是static 和 const 并用，在外部初始化，例如：

```
class A { public: A() {} private: static const int i; file://注意必须是静态的! };
```



```
const int A::i=3;
```

另一个很常见的方法就是初始化列表：

```
class A { public: A(int i=0):test(i) {} private: const int i; };
```

还有一种方式就是在外部初始化，
- 如果在非const成员函数中，this指针只是一个类类型的；如果在const成员函数中，this指针是一个const类类型的；如果在volatile成员函数中,this指针就是一个volatile类类型的。
- new返回的指针必须是const类型的。

[上一篇](#) [具备十五种让你成功的能力](#)

[下一篇](#) [日本开发担当者经常提出的6个问题---对日软件开发过程中的六个问题](#)

主题推荐

[C++](#)

[内存分配](#)

[编译器](#)

[数据](#)

[实例](#)

猜你在找

[cs硕士妹子找工作经历阿里人搜等互联网](#)

[割绳子的作者你如此歧视无视鄙视中国人这是何苦呢](#)

[Android如何完全调试framework层代码](#)

[C++学习之深入理解虚函数一虚函数表解析](#)

[moto & google笔试题目-STLC++面试题](#)

[asserth头文件之断言](#)

准备好了么？跳吧！

更多职位尽在 CSDN JOB

C++服务端开发工程师	我要跳槽	C++开发工程师	我要跳槽
欢聚时代（多玩YY）	面议	浙江大华技术股份有限公司	10-15K/月
C++工程师	我要跳槽	C++ / C#程序员(虚拟仿真平台系统软件)	我要跳槽
北京和勤联创技术发展有限公司	12-24K/月	北京东方仿真软件技术有限公司	面议

悠悠空间 - 自助式小仓库

提供家庭企业自助式迷你仓储服务,24小时运作,空间大小随意选 欢迎致电

查看评论

83楼 [yokeyoke](#) 昨天 23:06发表

[yokeyoke](#) 分享的很全面，也易读懂，赞一个！

82楼 [回首撒哈拉](#) 2014-12-10 23:01发表



求下一句的注解：
`const AP_HAL::HAL& hal = AP_HAL_BOARD_DRIVER;`
补充
AP_HAL是一个namespace，HAL是namespace AP_HAL中的一个class

81楼 [小布](#) 2014-11-02 10:14发表



楼主好心值得鼓励，只不过做技术已经要有严谨的作风，否则容易误导后来者，毕竟知识是先入为主，第一印象很重要。

80楼 [endlessbest](#) 2014-10-13 14:11发表



非常不错哈，感谢分享

79楼 [eKeeper_Lirz](#) 2014-08-27 16:17发表



作者写得很全面，整理得很不错。但有种怕被误导的心理。望楼主重新整理验证后，再@all.

78楼 [yk610900880](#) 2014-08-25 15:06发表



[cpp]

01.

02.

03.

04.
- 要让我这么说你呢？拿来主义也是要赠别一下正确与否可以么？
----编译器通常不为普通**const**常量分配存储空间，而是将它们保存在符号表中，这使得它成为一个编译期间的常量，没有了存储与读内存的操作，使得它的效率也很高
这个要看是什么编译器，有的就不会这样做。


```
05.      (char*) const pContent;  // const pointer
06.      const (char*) pContent;  // const value
07.      const (char) *pContent;  // const value
08.      (char) const *pContent;  // const value
09.      这和你加不加括号有个毛的关系？
10.      请不要误人子弟了好么？
```

77楼 [shallyhanlu](#) 2014-08-23 11:27发表



菜鸟学习了

76楼 [cilin1991](#) 2014-08-20 11:20发表



编译器通常不为普通**const**常量分配存储空间，而是将它们保存在符号表中，这使得它成为一个编译期间的常量，没有了存储与读内存的操作，使得它的效率也很高

完全的错误。

const能取地址的，根本就不是编译期间的常量。
如果是常量，那和**#define**还有什么区别？？

75楼 [cilin1991](#) 2014-08-20 10:58发表



```
void f(const int i) { .....}
```

这里的**const**一般而言就是代码错误。

74楼 [ccjoe](#) 2014-08-16 11:25发表



错误多多，楼主发之前是不是应该自己去实践一下，有无人子弟的嫌疑

73楼 [buaa_shang](#) 2014-08-06 21:16发表



不少错误吧

72楼 [xdlwd086](#) 2014-07-24 09:24发表



楼主的博文写的虽然有错误，但也很好，大家的评论也很受用，感谢大家！！！！

71楼 [Megatron_King](#) 2014-06-21 11:43发表



(1)指针本身是常量不可变
(char*) const pContent;
const (char*) pContent;

(2)指针所指向的内容是常量不可变
const (char) *pContent;
(char) const *pContent;

(3)两者都不可变
const char* const pContent;

(4)还有其中区别方法，沿着*号划一条线：
如果**const**位于*的左侧，则**const**就是用来修饰指针所指向的变量，即指针指向为常量；
如果**const**位于*的右侧，**const**就是修饰指针本身，即指针本身是常量。

（1）和（2）矛盾，（1）和（4）矛盾，

70楼 [xuexijun1992](#) 2014-06-11 16:54发表



mark~

69楼 [xujingSY](#) 2014-04-30 14:19发表



在const和非const类型转换问题上无尽的折腾，有的时候甚至是无解的，事实上如果一个代码傻逼到能在get函数中修改传入的参数的话，加了const也阻止不了这类程序员犯错。

68楼 [lion_kangaxx](#) 2014-04-01 10:49发表



const(char*)p 这个是c11标准的新代码么？在liunx下和windows下都不能这么写么？

67楼 [占占](#) 2014-02-09 15:52发表



so good!

66楼 [面神君](#) 2014-02-01 18:55发表



上面mark的傻逼，lz扯淡。

65楼 [Niteip](#) 2013-10-18 20:12发表



const(char*)pContent;
是错的

64楼 [daiyuancun](#) 2013-10-02 10:54发表



觉得还是@ Jingle转载的那篇文章对const的总结更精确，也更简洁！

63楼 [poppick](#) 2013-09-24 17:57发表



确实问题多多 大家要注意
严谨啊严谨

62楼 [chenishr](#) 2013-09-23 21:26发表



const(char*)pContent;
const(char)*pContent;
这两个有区别吗？

61楼 [青青子衿_悠悠我心](#) 2013-09-07 21:20发表



写得非常好！顶！

60楼 [牛肉圆粉不加葱](#) 2013-08-19 23:21发表



问题多多吧~

59楼 [xingzai2012](#) 2013-07-17 09:51发表



double i=Pi; //此时为Pi分配内存，以后不再分配！
double l=PI; //编译期间进行宏替换，分配内存
double j=Pi; //没有内存分配
double J=PI; //再进行宏替换，又一次分配内存！

这里进行宏替换，分配内存是什么意思？

58楼 [xiaocong1314](#) 2013-07-11 17:07发表



楼主整体还是写的蛮好的，不过有些瑕疵
`const doulbe Pi=3.14159;` //此时并未将Pi放入ROM中
`double i=Pi;` //此时为Pi分配内存，以后不再分配！
`double j=Pi;` //没有内存分配
此时我用`cout<<&Pi<<" "<<&i<<" "<<&j<<endl;`发现它们的3个地址是不同的

Re: [CitY_KilleR](#) 2013-08-27 15:30发表



回复xiaocong1314：这个应该是你自己理解错了吧，你那三个变量本身就在内存中存放的啊。

57楼 [Rainr](#) 2013-06-23 15:38发表



学习了 记录一下。

56楼 [love-xiao-forever](#) 2013-05-26 10:08发表



问题多多啊

55楼 [Hughen](#) 2013-05-11 17:34发表



楼主，你这篇文章有问题哈，小心误人子弟
`const`修饰的不能有括号的，比如`(char*) const p;`这是一种错误的语法结构，还希望楼主看看

Re: [jsnicky8888](#) 2013-08-20 05:35发表



回复gg513482543：其实博主的意思是如果用类型别名的话就是修饰对象本身了。例如`typedef int * int_t;const int_t p;`此时就表明`const`修饰的是`p`，相当于`int_t const p;`表示该类型对象是不可修改的。如果用宏定义的话就和平常本意一样了，例如`#define INT_T int * const INT_T p;`此时`const`便是修饰的`p`所指向的对象，表明该对象是不可修改的，但是`p`是可以修改的。因为宏定义只是一个替换。

54楼 [LTheMiracle](#) 2013-05-10 11:23发表



防止修改返回值有什么用呢？

53楼 [geekle](#) 2013-05-03 14:50发表



我觉得有些东西没有经过严格求证发出来，和散播谣言差不多，只会误导初学者。。。`#define`这种预编译命令不会分配内存的。`void f() const{}`，这种常量成员函数虽然可以作为重载，但更重要的是只能被`const`对象调用。

52楼 [micro_xzq](#) 2013-02-28 11:28发表



学习了，都很有道理，谢谢。

51楼 [hypercode](#) 2013-01-25 16:38发表



学习const

50楼 [Jingle](#) 2013-01-18 13:13发表



最后推荐另一个朋友对于`const`的整理，我觉得他说得比楼主要正确一些，转载如下，欢迎大家拍砖：

`const`的作用

`const`是C语言的一种关键字，起受保护，防止以外的变动的作用！可以修饰变量，参数，返回值,甚至函数体。`const`可以提高程序的健壮性，你只管用到你想要的任何地方。

(一)`const`修饰参数。`const`只能修饰输入参数。

1、如果输入参数是指针型的，用`const`修饰可以防止指针被意外修改。

2、如果参数采用值传递的方式，无需`const`，因为函数自动产生临时变量复制该参数。

3、非内部数据类型的参数，需要临时对象复制参数，而临时对象的构造，析构，复制较为费时，因此建议采用前加`const`的引用方式传递非内部数据类型。而内部数据类型

无需引用传递。

(二)**const**修饰函数返回值。

1、函数返回**const**指针，表示该指针不能被改动，只能把该指针赋给**const**修饰的同类型指针变量。

2、函数返回值为值传递，函数会把返回值赋给外部临时变量，用**const**无意义！不管是内部还是非内部数据类型。

3、函数采用引用方式返回的场合不多，只出现在类的赋值函数中，目的是为了实链式表达。

(三)**const**+成员函数。任何不修改数据成员的函数都应该声明为**const**类型，如果**const**成员函数修改了数据成员或者调用了其他函数修改数据成员，编译器都将报错！

```
class stack
```

```
{
```

```
public:
```

```
int GetCount(void) const ;
```

```
private:
```

```
int m_num;
```

```
};
```

```
int stack::GetCount(void) const
```

```
{
```

```
    m_num++;
```

```
}
```

编译器输出错误信息： error C2166: l-value specifies const object。

(四)**const** 修饰变量，表示该变量不能被修改。

1、**const char *p** 表示 指向的内容不能改变

2、**char * const p**，就是将**P**声明为常指针，它的地址不能改变，是固定的，但是它的内容可以改变。

3、这种**const**指针是前两种的结合,使得指向的内容和地址都不能发生变化.

```
const double pi = 3.14159;
```

```
const double *const pi_ptr = &pi;
```

49楼 [Jingle](#) 2013-01-18 11:52发表



这篇文章确实很应该感谢作者的收集和整理，不过谈技术问题，一定应该严谨，这篇文章里对**const**的描述确实有很多明显的错误。其实这些东西应该多用几种**C++**编译调试器实验正确了再总结，现在看来终归有点纸上谈兵。

48楼 [absorbguo](#) 2012-12-10 22:57发表



不错不错

47楼 [Gcache](#) 2012-11-06 10:09发表



不错，mark

46楼 [幸福摩天轮](#) 2012-09-29 13:32发表



"a. **const**成员函数不被允许修改它所在对象的任何一个数据成员。" 当数据成员为**mutable**时候，**const**成员函数是可以修改**mutable**数据成员。

45楼 [OOD-SONG](#) 2012-09-28 00:14发表



a.传递过来的参数在函数内不可以改变(无意义，因为**Var**本身就是形参)

```
void function(const int Var);
```

这里的**const**并不是没有意义，而是为了防止传进来的**Var**值在

function中被修改，是有意义的。

博文随写得有条理，不过很多误点，希望楼主修改一下，莫要误导了我们这些童鞋！

Re: [AaBb301](#) 2013-01-06 15:43发表



回复zajuan0810：那是无意义的，因为作为值传递，会自动创建临时副本，在函数体内你再怎么改变，也不可能倒传回实参，也就是值传递时实参一定是不变的，那么这时再加**const**不就是多余吗？

Re: [Jingle](#) 2013-01-18 11:12发表



回复AaBb301：当然不多余，参数只在函数体内被修改固然不会影响实参，不过函数体内的逻辑很可能由于把形参当成常量，却又无意间被程序逻辑修改，为了杜绝这种无心之失酿成大错，在函数体内让形参一直保持**const**是一种良好的做法。

Re: [cee zoo](#) 2013-03-27 23:53发表



回复dj0379：话说的有道理，不过有点过了，还是看个人编程习惯

44楼 [lovegetty](#) 2012-08-21 09:57发表



mark一下！

43楼 [dizhiling](#) 2012-07-24 16:02发表



还是很不错的。

42楼 [jzp12](#) 2012-06-22 11:56发表



不严谨。精神可嘉。
有多处明显错误，这些内容并未在编译器上调试。

41楼 [clazazu](#) 2012-03-22 07:20发表



那(char*) const pContent; 和const (char*) pContent;有什么区别呢？就是**const**放char前面和放后边有什么区别啊？

40楼 [cpp罗纳尔多](#) 2012-03-17 23:34发表



好文章，有个问题请教一下

```
struct rect {int x,y,w,h};
class Rect
{
rect r;

public:
rect getrect(){return r;};
}
```

调用时
Rect a;
a.getrect().x;会报错

但是把getrect()定义成getrect() const;则没有问题

看楼主的博文还未知所以然，望提点

39楼 [yi5971](#) 2012-03-09 12:38发表



学习了

38楼 [shang_1991](#) 2012-02-20 21:53发表



楼主说不全面
const int arrysize=10;//不分配内存
int array[arraysize];//合法

const int array1[]={1,2,3,4};//分配内存
int array2[array1[1]]//不合法

WHY? ? ?

Re: [lm_whales](#) 2013-07-23 16:50发表



回复shang_1991: 楼主说不全面
const int arrysize=10;//不分配内存
int array[arrysize];//合法

const int array1[]={1,2,3,4};//分配内存
int array2[array1[1]]//不合法

WHY? ? ?

array1[1] 是个变量，不是常量，**array1[1]**是 数组**array1**定义的4个变量中的一个，值为**2**；
尽管**array1**这个数组名，也是个指针常量，但不是一个常量表达式，同时也不是整型数常量。
array1[1] 则是是个变量，更不是常量表达式。

同意楼主说不全面。

Re: [deajosha](#) 2012-02-24 10:14发表



回复shang_1991: 数组名表示的是一个地址，这个就与**const** 修饰指针使用相同的，这里已经不是一个常量了

37楼 [loong460](#) 2012-01-10 15:25发表



Perfect

36楼 [ibmmicrosoft](#) 2011-12-04 18:42发表



nice~~

35楼 [dingkh](#) 2011-12-02 15:08发表



very good! QQQ!

34楼 [真嵐天下](#) 2011-10-03 21:28发表



谢啦~不会的继续请教

33楼 [tkggjyim](#) 2011-09-16 09:26发表



找到想要的了! thank you for your share.

32楼 [liyongjin2009](#) 2011-09-12 14:01发表



请博主再仔细验证一下，所讲的内容是否正确，有一些根本无法编译通过.在linux，g++环境下

31楼 [hai836045106](#) 2011-07-11 13:10发表



总结得很好

30楼 [tulipcaicai](#) 2011-06-13 22:52发表



[e01]挺好，看了有收获。

29楼 [ljt350740378](#) 2011-06-02 15:33发表



[e01]

28楼 [weizhuo1989](#) 2011-05-09 20:33发表



[e03] qiangren!

27楼 [Kang_Nian](#) 2011-05-09 19:44发表



[e01]

26楼 [yinyuanlin123](#) 2011-05-08 09:56发表



菜鸟2号来学习学习啦[e03]

25楼 [kanchangcheng](#) 2011-03-23 22:59发表



[e01]

24楼 [big_world](#) 2011-03-01 22:18发表



[e02]

23楼 [tt2com](#) 2011-02-21 16:14发表



mark

22楼 [zxzx1234321](#) 2010-12-24 20:29发表



[e03]很详细，很清晰

21楼 [liyaobinRyan](#) 2010-10-14 09:49发表



记录一下，方便学习

20楼 [liyang467](#) 2010-09-01 11:46发表



const (char*) pContent; 这个根本无法被编译通过。

```
1>d:/demo/conversion/conversion.cpp(43): error C4430: missing type specifier - int assumed. Note: C++ does not support default-int
1>d:/demo/conversion/conversion.cpp(43): error C2062: type 'char' unexpected
```

而且

const char* pContent; 应该是指向常量的指针。

19楼 [yinhanng](#) 2010-08-24 00:03发表



```
#define PI 3.14159 //常量宏
const doulbe Pi=3.14159; //此时并未将Pi放入ROM中
double i=Pi; //此时为Pi分配内存，以后不再分配！
double j=Pi; //没有内存分配
```

这里面Pi 是分配在符号表上, i和j都是分配在静态区, 不太博主写的没有内存分配 的意思, 望楼主指点.

Re: [Eric_Jo](#) 2010-08-25 23:43发表



回复 yinhanng: 这里的说没有内存分配是说, **const**常量Pi只分配一次内存的意思。

Re: [hulin0229](#) 2011-01-02 05:46发表



回复 Eric_Jo:

还有第7点, 你说“编译器通常不为普通**const**常量分配存储空间, 而是将它们保存在符号表中...”, 这个其实也是错误的。首先, 我查阅了msdn关于**const**的文档, 并没有发现存在提及你所说的“符号表”内容。其次, 我们假设有这个符号表, 那么我想问: 这个符号表是保存在哪里, 符号表本身占不占存储空间? 如果真有这个符号表, 那么不但会同样占用内存空间, 而且会额外增加检索开销, 因此你的这个假想不太合理。

Re: [godmessengers](#) 2012-08-29 23:54发表



回复hulin0229: 说的有理,楼主错误,大大的

Re: [lm_whales](#) 2013-07-11 11:30发表



回复godmessengers: 看下编译原理, 符号表, 是编译器, 在编译你的程序的过程中, 使用的数据, 不是编译后的存储在.exe,.obj 这种代码中的数据, 只会占用编译器的空间(内存, 文件), 你的代码中是不会有符号表的痕迹的, 除非必要---比如用解释器解释执行, 或者嵌入解释器---你的程序里, 没有符号表相关或者类似的东西。
但是, 确实总结常量的使用, 以及各种常量, 是不容易的, 难免会出错的。

Re: [hulin0229](#) 2011-01-02 05:24发表



回复 Eric_Jo:

对与楼主的**const**常量节约内存的观点不敢苟同。事实上宏定义是不占内存的, 而**const**常量是占用内存的。宏定义在编译时会被替换成字面值, 然后用这个字面值赋给变量, 当所有使用该宏的地方都被替换完毕之后, 宏本身就销毁了。宏是一种预处理机制, 它不会作为一个内存值被编译到程序中。另外, 变量在定义时就分配了内存空间, 跟它是被**const**常量赋值或是被宏替换赋值无关

18楼 [yangkui_happy](#) 2010-05-25 10:47发表



楼主有很多说的是错误的

你若不知道你写的那些是不是C++标准里面提到的, 但至少你要保证你粘出来前要找编译器试下嘛。

```
class AAA
{
void func1();
void func2() const;
}
const AAA aObj;
aObj.func1(); &#215;
aObj.func2(); 正确
```

```
const AAA* aObj = new AAA();
aObj-&gt; func1(); &#215;
aObj-&gt; func2(); 正确
```

Re: [Eric_Jo](#) 2010-06-01 23:47发表



回复 yangkui_happy:

你指出的错误是什么呢? 你写的**code**和我写的**code**应该是两个不同概念了。

Re: [qsdxt_123](#) 2011-12-21 11:04发表



回复Eric_Jo: class AAA

```
{
public:
```



```
void func1();
void func2() const;
};

int main()
{
    const AAA aObj;
    // aObj.func1();
    aObj.func2();

    const AAA* aObj1 = new AAA();
    // aObj1->func1();
    aObj1->func2();

    return 0;
}
```

17楼 [typedf_lei](#) 2010-05-15 15:28发表



[e01]

16楼 [stonan](#) 2010-05-06 21:28发表



thank you for your explanation

15楼 [匿名用户](#) 2010-04-22 21:56发表



[e03]

14楼 [匿名用户](#) 2010-04-11 18:08发表



[e03]

13楼 [wangweiwangchao](#) 2010-03-22 09:21发表



[e01][e03]。顶了。

12楼 [UC_砵哥](#) 2010-03-11 15:56发表



[e01]楼主，是个细致的好人啊。

11楼 [xjiang_92](#) 2010-02-08 12:10发表



同意楼上的，**const** 成员函数相当于是声明了
const class * this
于是乎便不能修改任何类成员了
而其它成员函数应该还是可以访问对象中的**const**成员的

10楼 [tt870906](#) 2009-11-03 09:16发表



thanks a lot !
It's really wonderful!

9楼 [qhmao](#) 2009-09-07 14:55发表

1)指针本身是常量不可变



(char*) const pContent;
const (char*) pContent;
-----const (char*) pContent; 有问题。。。。。

Re: [p_zzf000](#) 2010-03-25 16:34发表



回复 qhmao: (1)指针本身是常量不可变
(char*) const pContent;
const (char*) pContent;

(2)指针所指向的内容是常量不可变
const (char) *pContent;
(char) const *pContent;
确实有问题。

const (char*) pContent 与 const (char) *pContent是一样的，指向内容不变。

Re: [godmessengers](#) 2012-08-29 23:58发表



回复p_zzf000: 同识,
(1)指针本身是常量不可变
(char*) const pContent;
const (char*) pContent;
(2)指针所指向的内容是常量不可变
const (char) *pContent;
(char) const *pContent;

const (char*) pContent;同const (char) *pContent;到底有什么不同,差误纠结了.

8楼 [Amanda19810920](#) 2009-05-25 09:59发表



It seems good ,but I can not understand it at all .

7楼 [天空的期望](#) 2009-05-23 14:47发表



谢谢 good !you are very 强

6楼 [wbgxx](#) 2009-05-23 14:36发表



well! i like it!

5楼 [tjj023](#) 2009-05-22 20:55发表



Good!
Mark!
Thanks!

4楼 [freezgw1985](#) 2009-05-22 17:11发表



菜鸟学习来了

3楼 [hsf1002](#) 2009-05-22 12:18发表



mark

2楼 [hsf1002](#) 2009-05-22 12:18发表

mark



1楼 armzm 2009-05-16 19:07发表



wonderful! thank you for your share.

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题	Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack	VPN	Spark	ERP	IE10		
Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP	jQuery	BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS
Fedora	XML	LBS	Unity	Splashtop	UML	components	Windows Mobile	Rails	QEMU	KDE	Cassandra	CloudStack	FTC			
coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo	Compuware	大数据	apttech	Perl				
Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr	Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap				

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持
京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved 