

C++中引用、指针与const之间的爱恨情愁

原创

~~~~~08月30日 21:59:59

2188



学过C语言基础的肯定都知道变量和数据类型是再简单不过的知识，然而这个基础中确有几个泥潭，稍不注意就粉身碎骨——编程受阻，面试被刷。其中一个就是引用、指针和const，以及相互之间剪不断理还乱的关系。今天我们就来理一理。



1.引用是个什么鬼



1.1引用的概念

引用是为对象另外起的一个名字，也就是别名而已。那什么是对象呢？注意这里说的对象和面向对象里的对象不是一回事。这里的对象是内存的一块区域，它具有某种类型，变量是命名了的对象。可以这么认为，引用与对象简单的关系就像姓名和本人。姓名可以多换几个，但是必须和一个实在的人对应起来，不管它是男是女，是死是生，甚至可以是某个小说里杜撰的，但是一定要有个显示的人与之对应。

一般在初始化变量时，初始值会被拷贝到新建的对象中，然而在定义引用时，程序把引用和它的初始值绑定在一起了，而不是将初始值直接拷贝给引用，也就是所，引用指向对象，但数据就一份。一旦初始化完成，引用将和它的初初识值对象绑定在一起，因为引用不能重新绑定另一个对象，所以**引用必须初始化（要点一）**。

看下面的程序：

```
[cpp]
1. int ival=1024;
2. int &refVal=ival;
3. int &refval2;
```

第二行的定义了ival的引用refVal。第三行的就不对了，原因就是上面的要点一：未初始化。

需要注意的是，不能解绑、必须初始化不代表着引用类型的值就不能变了，事实上其改变更加多样。**定义了一个引用之后，对其进行的所有操作都是在与之绑定的对象上进行的（要点二）**。例如下面的程序：

```
[cpp]
1. int i,&ri=i;
2. i=5;
3. ri=10;
```

这里将ri与i绑定了，这里用i来初始化ri，虽然在第一行i也并未初始化，这没关系，以后给i赋值即可。就像新婚夫妻给自己的子取了名字，尽管孩子还没有，但是这没关系，名字属于未来出生的孩子。

中间两行是赋值，如果按照这个顺序，最后i和ri的值都为10。反过来，如果将中间两行反过来呢？那i和ri的值都是5.原因同样是要点二。

1.2引用的定义和使用

引用的符号是“&”，对于引用的定义，记住两个原则：**引用的初始值必须是一个对象（要点三），引用类型必须与初始值对象类型要一样（要点四）**。

例如：① `&refVal4=10;` 是错误的，原因是要点三。

② `doub al=3.14;`

`int &refVal5=dval;` 是错误的，原因是要点四。

③ `int ival=1.01; int &rval2=ival;` 这个代码是正确的，原因是ival的值是整数类型1，而不是1.01，但是这种代码有隐患，容易产生意想不到的问题。



纵横千里

|    |    |    |
|----|----|----|
| 原创 | 粉丝 | 喜欢 |
| 64 | 42 | 45 |



等级： **博客** 访问量： 15  
积分： 2293 排名： 1万+



注册一个小公司



### 他的最新文章

live555中的openRTSP如何通过收到的视频保存成mp4文件

如何在Ubuntu下安装Notepad++ add-apt-repository无法识别的

好未来2017年秋季校招面试题一

暴风影音2017年校招笔试题-选择

C/C++面试常考题目讨论之二：函数来分配空间

### 文章分类

杂七杂八

图像处理与检索

Linux研究

学习心得与计算机哲学

视频处理

C/C++编程

展开

### 文章存档

2018年3月

2018年2月

2018年1月

2017年6月

2017年4月

再看一个比较特殊的例子：

```
int i, &ri=i, &r2=ri;

i=5;
```

在这里，&r2=r1，在gcc下编译是正确的，但是否是说可以为引用定义引用呢？按照《C++ Primer》的说明，引用本身不是一个对象，因此不能定义引用的引用，但是最后输出的结果是三个变量值都为5。感觉这里ri应该指向了i，这里其实还是为i定义的引用。

我测试了功能用的两个完整代码：

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     int i=1,&r1=i;
6.     double d=2.1,&r2=d;
7.     //r2=3.1415;
8.     //cout<<"r2:"<<r2<<"d:"<<d<<endl;
9.     // r2=r1;
10.    // cout<<"r1:"<<r1<<"r2:"<<r2<<endl;
11.    // i=r2;
12.    // cout<<"i:"<<i<<"r2:"<<r2<<endl;
13.    r1=d;
14.    cout<<"r1:"<<r1<<"d:"<<d<<endl;
15.
16.
17.    return 0;
18. }
```

第二个：

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     int i,&ri=i,&r2=ri;
6.
7.     ri=10;
8.     i=5;
9.     cout<<i<<" "<<ri<<" "<<r2<<endl;
10.    return 0;
11. }
```

## 2.指针是个什么鬼

指针可不是闹着玩的，一本书都解释不完（还真有大牛的书，叫《C与指针》）。可以简单认为，指针就是某个对象的地址。指针的细节非常多，大部分人应该都知道，这里我们重点分析与引用的区别。

指针定义也很简单：

```
int ival=42;

int *p=&ival。
```

比较要命的就是这里有个“&”，指针里表示某个对象的地址，初学时容易与引用混在一起。上面的代码可以写成下面的形式,更容易理解：

```
int ival=42;

int *p;

p=&ival;
```

通过指针可以访问对象，例如

cout<<\*p;就可以输出42.效果与cout<<ival一样的。

其实指针和引用容易混淆的关键之一就是“&”和“\*”这种符号可以有多种含义。

### 他的热门文章

Ubuntu在vmware虚拟机无法上网方法

9729

Linux C编程下没有 itoa()函数的解决方法

8696

色度抽样（4:2:0）到底是什么意思

7661

qt5实现tftp和ftp的方法之五：实现ftp的方法

6503

ubuntu 下没有pthread库以及报undefined reference to 'pthread\_create'的错误

6162

程序员的自我修养\_之三\_曾国藩与

5414

STM32F4板子使用LWIP进行组网数据的完整过程，附代码

5353

C/C++程序设计学习笔记二：C语言中，如何使用指针交换两个数的值

5241

CTeX cannot open TrueType font for reading 简单的解决方法

5125

64位Ubuntu12.04下安装arm-linux-gcc，以及解决no termcap library 1

4232



注册一个小公司



联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

在等号左边，符号随着类型名出现，是声明的一部分(要点5.1)。如果在等号右边，符号在一个表达式中，因此是取地址 (&) 或者解析的 (\*) 。（要点5.2）

```
[cpp]
1. int i=42;
2. int &r=i; //要点5.1
3. int *p; //要点5.1
4. p //要点5.2
5. * //这里其实就是用常量i给*p赋值，《C++ Primer》里说这是解引用，不懂，请高手指点
6. int4&r2=*p; //这里与int &r2=42;是一样的，从地址p所指向的对象给引用r2赋值。</span>
```

指向指针的引用

引用本身不是一个对象，因此不能定义指向引用的指针，但是指针是对象，所以存在对指针的引用。

例如

```
int i=42;
int *p;
int *&r=p; //r是一个对指针的引用。这种代码很难理解，要点六：复杂的表达式从右向左一个个阅读。离变量名最近的符号对变量的类型有直接影响，这里* (&r) ,因此r是一个引用，然后(&r)是一个指针。
r=&i; //，上面要点5.2，r是一个引用，指向的是i的地址，因此给r赋值就是另P指向i。
*r=0; //解引用r得到i，也就是p指向的对象，将i的值改为0。小心上面这两种代码。个人感觉理解大型软件的主线就是理清楚数据从头到尾是怎么走的，如果遇到这种代码有时候会发现程序突然断掉了，大伤脑筋。
```

3 const限定符

const是一种类型，用于说明永不改变的对象，这也意味着const对象一旦定义就不能改变了，自然定义的时候就必须初始化。const的定义和使用啥的就不提了，重点介绍const与引用和const与指针。

当执行对象的拷贝操作时，拷入和拷出的对象必须具有相同的底层（指针所指向的对象）const资格，或者两个对象的数据类型必须能够转换。这里只有单向转换：非常量可以转换成常量，反之不可以。

3.1const与引用

将引用绑定到const上，就叫对常量的引用了。考虑到const的特征，对常量的引用不能用作修改它所绑定的对象。

```
[cpp]
1. const int ci=1024;
2. const int &r1=ci; //正确，引用及其对应的对象都是常量
3. int &r2=ci; //c错误，试图让一个非常量引用指向常量引用。注意这两行代码的区别

[cpp]
1. r1=42; //错误，r1是对常量的引用
```

const的要求比较严，不能用对常量的引用给普通的引用初始化。例如：

```
[cpp]
1. int i42;
2. const int &r1=i;
3. const int &r2=42;
4. const int &r3=r1*2;
5. int &r4=r1*2; //r4是非常量引用，不能用常量引用来初始化。</span></span>
```

在普通的引用中，引用的类型必须与其所引用对象的类型一致。但是在初始化的时候只要类型能转换过去就可以用任意表达式作为初始值。嘛意思呢？还是看代码：


```
[cpp]
1. double dval1=3.14;
2. const int &r1=dval; //正确
```

其实，这里可以结合强制类型转换来理解。

由于ri引用了一个int型的数，但是dval是一个双精度浮点数，所以这里其实进行了强制转换：

```
const int temp=dval;



const int &ri=dval;
```

也即这里  绑定了一个临时量对象。不过这种方式虽然能编译过去，但是没什么意义，也存在隐患。C++代码还是严谨一点好。



3.2 const 指针



const指针  解起来就容易多了，就是把一个指针定义成了不可改变的常量嘛。我们知道一个指针能够表示  对象以及对象的地址，很显然，const指针就可以表示地址本身是常量还是表示的数据是常量，《C++ Primer》中前者叫顶层const，后者叫底层const。

首先看一个代码：

[cpp]

```
1. const double pi=3.14;
2. double *ptr=&pi; //错误，用常量的地址给普通指针初始化，类似于用白马代替所有马了，以偏概全了。</span>
```

关于const指针，指向常量的指针不能用于改变其所指对象的值。要存放常量对象的地址，只能使用指向常量的指针。

例如：

[cpp]

```
1. const double pi=3.14;</span>
```

[cpp]

```
1. const double *cptr=&pi;
2. *cptr=42; //错误。不能给常量地址赋值。这里不管使用*p=42还是其它方式，都不可以了。
```

一般而言，指针的类型与其所指对象的类型要一致。但是一个指向常量的指针可以指向一个非常量对象（反之不行），从而间接的改变常量指针的值。例如：

[cpp]

```
1. const double pi=3.14;
2. const double *cptr=&pi;
3. double dval=3.333;
4. cptr=&dval;
```

这里cptr指向的就是dval的地址，\*cptr就是3.333了。但是不能通过cptr改变dval的值，指向常量的指针仅仅要求不能通过当前指针改变对象的值，但是可以通过其他途径间接的改变。

假如将\*放在const前面，会怎么样呢？那就是const指针了，作用是说明指针是一个常量，其深层含义是地址本身是常量而所指向的值可能被改变了。

例如：

[cpp]

```
1. int errNumb=0;
2. int *const curErr=&errNumb; //curErr将一直指向errNumb的地址
3. int errNumb2=2;
4. curErr=&errNumb2; //错误，不能再绑定其他值了
```

如何分清上述两种声明呢？根据前面的要点六，从右向左阅读。这里离currErr最近的符号是const，意味着curErr本身是一个常量对象，也就是地址不会变，然后再看\*就容易多了。

再来看一个更复杂一点的代码：

[cpp]

```
1. const double *const pip=&pi;
```

首先pip是一个常量指针，其次，其指向的对象也是常量，类型为双精度浮点型。

引用、指针和const之间的关系是C++基础语法里绕不过的关键主题。解决问题的思路就是先记住每个点的特征，也就是上面总结的几个要点，然后按照从右向左的顺序逐步拆分。如果有等号，记得符号在等号左右表示的含义不一样就能解决大部分的问题。


将上述几个要点稍作完善：

- 要点一：引用和const定义对象时必须初始化，而指针不必。
- 要点二：引用的引用，对其进行的所有操作都是在与之绑定的对象上进行的。而指针其实就是某个对象的地址，可以通过指针来访问对象，也可以通过某个对象来访问指针（地址）。
- 要点三：引用的初始值必须是一个对象，引用类型的定义类型与初始值对象类型要一样。
- 要点四：符号在等号左边，符号随着类型名出现，是声明的一部分。如果在等号右边，符号在一个表达式中，则取地址（&）或者解析的（\*）。
- 要点五：复杂的表达式从右向左一个阅读。离变量名最近的符号对变量的类型有直接影响。

上面大部分素材都来自《C++ Primer》第二章的2.3和2.4。不过感觉我还是没彻底将问题说清楚，甚至有些地方有漏洞。虽然博客写了大半天，感觉还是挺失败。以后找几个面试中遇到的例子整理再慢慢打磨这篇“爱恨情愁”。



版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/xueyushenzhou/article/details/52372499>

 目前您尚未登录，请 [登录](#) 或 [注册](#) 后参与评论

- 

qq\_37059483 2017-11-24 09:08 #4楼

1条回复  [回复](#)

写的好，收藏
- 

zgc187 2016-08-31 20:48 #3楼

2条回复  [回复](#)

顶一下
- 



zgc187 2016-08-31 20:45 #2楼

1条回复  [回复](#)

我真写的感受到了大神的存在

[查看 9 条热评](#)

C++ 使用const指针来传递对象

 helainthus 2016年05月14日 16:17  726

采用指针来传递对象，虽然可以避免重复调用复制构造函数和析构函数，但是由于它得到了该对象的内存地址，可以随时修改对象的数据，因此它实际上是破坏了按值传递的保护机制。 ...

c++const指针与函数调用



caoyan\_12727

2016年09月10日 10:50



564

在我的博客[http://blog.csdn.net/caoyan\\_12727/article/details/5206](http://blog.csdn.net/caoyan_12727/article/details/5206)

4958中，已经讨论了动态绑定和静态绑定，以及在缺省参数情况下虚函数的绑定情况。一...

80万走，AI程序员扔缺口百万，普通程序员的我该如何入门？




从小白到人工智能工程师，我只花了4个月！






C++指向 st对象的指针和const指针



zhy\_cheng

2012年10月08日 13:30




6133

1.指向const指针 我们可以通过指针来修改其所指对象的值，但如果指针指向的是const对象，则不允许使用指针来改变其所指的const值


+要求指向const对象的指针也是const类型...

C++中const修饰指针探讨



CHEN\_JP

2012年04月18日 16:44



3362

在C++里，const修饰指针有以下三种情况 （1）const int \*p; const在类型前，称指向常量的指针，可以这样理解它的功能，因为const在int前，所以p指向的这个in...

指针常量与常量指针（const用法总结）



Melody\_FHM


2011年08月13日 08:27



17462


const是一个C语言的关键字，它限定一个变量不允许被改变。使用const在一定程度上可以提高程序的安全性和可靠性 指向常量的指针： const int \*pa; int const \*pa; ...

一秒创造无法计算的价值




每满2000返200，最高返5000元代金券

C++中const修饰基本数据类型、指针、引用、对象



iamgaowei

2014年04月03日 14:49



1135

const修饰基本数据类型 #include using namespace std; void main(){ const int a = 1; const char b = 'k'; c...

C语言中关于const与指针结合的理解



u011408809


2016年07月05日 19:17



2250


C语言中const的可变与不可变的关系总结为： 只有被const直接修饰的变量最终指向的内容不可变（忽略变量类型修饰符，即int，long等）。 ...

const指针 与指向const的指针的区别



z1026544682

2016年08月09日 09:43



1789

const 指针与指向const的指针 当使用带有const的指针时其实有两种意思。一种指的是你不能修改指针本身的内容， 另一种指的是你不能修改指针指向的内容。听起来有点混淆一会放个例子上来就明白...


深入理解const 与指针



2012年08月26日 15:59 21KB


下载

const修饰指针的用法：常量指针和指针常量



oguro


2016年09月28日 21:49



2771


const通常用来修饰变量不能够更改值，多用来保护变量或参数。 const int b = 100; //b = 0;错误 当const修饰指针时，由于const的位置不同，它...

含const的双重指针解析



qq\_33826977

2017年03月28日 23:25



158

由于有2个const,我们可以先理清清楚关系，画好图，得知这个是指针的指针，也就是一个char类型的双重指针，具体关系如下图。然后再根据const 修饰的最近的内容确定哪个是常量即可。 ...

恭喜：一个公式教你秒懂天下英语



加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

const 指针

2013年03月08日 22:30   4KB   

下载

TXT

C语言中的const用法以及常量指针与指针常量（有口诀！）

编了这样的口诀，记住，应该不难：const（\*号）左边放，我是指针变量指向常量；const（\*号）右边放，我是指针常量指向变量；const（\*号）两边放，我是指针常量指向常量；指针变量...

xiaoch

3810

2014年11月27日 10:19

3380

关于常量、const指针以及const引用

这篇博客从我的角度讲一下关于const的一些问题。1.const常量与变量不同，有const修饰的量是常量，常量的值不可改变，在定义的时候必须初始化，使用未初始化的常量是错误的。以前还经常弄不清楚...

Mysunshinetbg

2015年09月10日 17:12

5515

C++：const和引用

上面一篇博客我们已经介绍了基本的const，对于这篇博客，我先说下引用是什么。引用说白了就是为对象另外起了一个名字，引用类型引用另一种类型。定义方式：将声明符写成&d的形式来定义引用类型其中d是声...

lishuzhai

2016年01月28日 18:24

2647

C++ 使用const 引用传递参数

类似const & int 的形式是C++的常量引用，在函数参数参数列表中常使用const的引用。...

zhangxiao93

2016年01月22日 15:18

4041

C++中const和引用修饰变量和函数的总结

一、对于修饰变量的用法对于const和&的基础用法就不说了。下面说点之前有误区和容易错的地方const修饰变量的误区关于const类型，这里有一个我之前的误区，我以为const定义的时候只能用常量初始...

haolexiao

2016年12月05日 22:32

3339

开发一个app大概需要多少钱呢

开发一个app多少钱

百度广告

C++ – const常量与指针和引用之间的关系

const常量1、const常量必须在定义的时候初始化，且不能修改。2、const常量的默认类型为int类型。3、#define是一个预处理器编译指令。该编译指令告诉预处理器，在程序中来查找并...

ko\_tin

2016年11月15日 23:36

752

C++智能指针简单剖析

智能指针是一个类，这个类的构造函数中传入一个普通指针，析构函数中释放传入的指针。智能指针的类都是栈上的对象，所以当函数（或程序）结束时会自动被释放。1. 智能指针背后的设计思想 auto\_ptr...

liuyanfeier

2016年10月16日 13:22

305

C++引用——const引用与非const引用

引用：是一种复合类型（指用其他类型定义的类型），不可以定义引用类型的引用，只是它绑定对象的另一个名字。eg: int a=1024; int &b=a; // a...

xiaobote\_

2015年06月13日 17:38

655