ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.8, No.16, June 2012.
E-mail: xsjl@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

关于C++引用底层实现机制的研究与分析

和力,吴丽贤

(韩山师范学院 数学与信息技术学院,广东 潮州 521041)

摘要:为了澄清C++中引用的本质,通过反汇编手段揭示了引用的底层实现机与指针实质是一样的,也是属于间接访问,并对实体名、指针和引用三种访问方式进行了比较。

关键词:C++;引用; 底层实现机制;反汇编

中图分类号:TP312 文献标识码:A 文章编号:1009-3044(2012)16-3854-02

Study and Analysis on Underlying Mechanism of Reference in C++

HE Li, WU Li-xian

(Department of Mathematics and Information Technology, Hanshan Teachers' College, Chaozhou 521041, China)

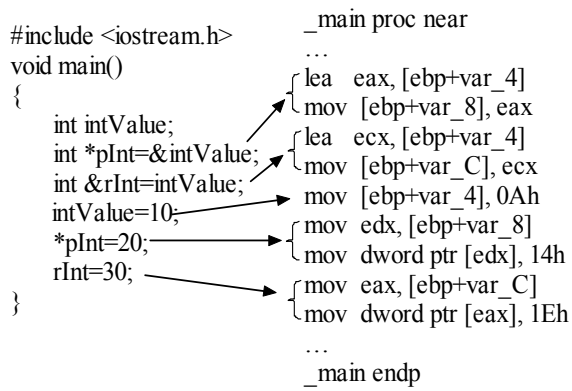
Abstract: In order to clarify the Essence of reference in C++, the underlying mechanism of reference in C++ is revealed by means of disassembling. References and pointers are indirect access. Entity name, pointer and references-three access methods are compared.

Key words: C++; reference; underlying mechanism; disassembling

C++语言中访问实体(包括变量和类对象)可通过实体名(包括变量名和对象名)、指针和引用三种方式。通过学习,学生一般能较好地理解前两种访问方式,但由于很多文献和教材只是简单地把引用看作是实体的别名,对于它的底层实现机制和工作原理介绍较为模糊,所以学生对于引用的本质,以及它与指针的异同大都显得一知半解。下面以Visual C++为例,对于引用的底层实现机制进行深入分析,以揭示引用的本质。

1 引用的实现机制

下面给出程序test1.cpp,并用反汇编工具IDA Pro对其进行反汇编,结果如图1所示。



注:ebp+var_4为intValue, ebp+var_8为pInt, ebp+var_C为rInt的地址

图1 三种访问方式的底层实现机制

通过分析汇编代码可知,访问实体的三种方式中,实体名属于直接访问,指针和引用都属于间接访问,而且引用与指针的底层实现机制是一样的,三者的联系和区别可用图2表示。

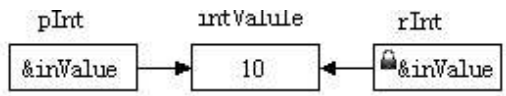


图2 三种访问方式的区别与联系

收稿日期:2012-04-05
基金项目:2011年度广东省高等学校本科特色专业建设点:计算机科学与技术(师范类)资助
作者简介:和力(1971-),男(纳西族),云南丽江人,副教授,硕士,主要研究方向为软件工程、中间件技术和分布式应用;吴丽贤(1974-),女,广东潮州人,讲师,硕士,主要研究方向为中间件技术和分布式应用。

从上可知,引用变量和指针变量分配相同字节的存储单元,保存的都是所指实体的地址。两者不同的地方是指针既可以改变自身的值,也可以改变所指实体的值,而引用只可改变所指实体的值,它与关联实体的联系是固定的,不可改变。引用与实体的关系,看似直接访问,实为间接访问,编译器会自动将引用rInt转换成对*rInt的操作。从这点上讲,可以把引用看作是指针常量的优化版。

2 不同方式的访问效率比较

在上例中,当对实体进行赋值操作时,实体名方式属于直接访问方式,对应于一条指令,而引用和指针方式属于间接访问方式,对应于两条指令,所以通过引用和指针访问实体的效率应低于实体名方式。在test1.cpp程序中分别增加下列代码段,并测试执行时间,整理后得表1所示。

表1 不同访问方式的效率比较

调用语句	时/ms	调用方式
for(int i=1000000000;i>0;i--) intValue=-intValue;	3937	直接访问
for(int i=1000000000;i>0;i--) *pInt=-*pInt;	4562	间接访问
for(int i=1000000000;i>0;i--) rInt=-rInt;	4562	间接访问

表1的结果验证了先前的分析,引用不能简单的看作是实体的别名,它的底层实现机制与指针是一样的,本质上也是间接访问。

既然引用和指针访问实体的效率低于实体名,哪为什么还要引入引用和指针呢?引用和指针的价值在于在函数调用中传递参数,传值调用需在被调函数的栈空间复制实体的副本,如果传递的是大对象,不仅要求消费大量的栈空间,而且要花费额外的时间,这将降低参数传递的效率。而如果使用引用或指针来传递参数,则是将实体的地址值赋给形参,而不用拷贝副本,这样减少了时间和空间上的花费,大大提高了效率,而且在被调函数中可通过引用和指针操控所指的实体。

函数返回实体时,会在函数中生成实体的一个临时副本,然后返回调用函数,而返回引用时,将不会在栈空间中生成实体的临时副本,这将使程序的运行效率和空间利用率都得到提高。

以下的代码声明了一个名为BigClass的类和类对象a,定义了参数分别为对象名、指针和引用的三个空函数,以及返回对象和返回引用的两个函数,并在主函数中用表2中的代码调用它们,并测试执行时间。

```
class BigClass
{ private:
      double a,b,c,d,e,f,g,h;
};
BigClass a;
void f1(BigClass b){}
void f2(BigClass *p){}
void f3(BigClass &r){}
BigClass f4(){ return a;}
BigClass& f5(){ return a;}
```

表2 不同调用和返回方式的效率比较

调用语句	时/ms	调用/返回方式
for(int i=1000000000;i>0;i--) f1(a);	69171	传值调用
for(int i=1000000000;i>0;i--) f2(&a);	39359	传地址调用
for(int i=1000000000;i>0;i--) f3(a);	39365	传引用调用

BigClass b;
for(int i=1000000000;i>0;i--)
b=f4();
	103625	返回值

BigClass b;
for(int i=1000000000;i>0;i--)
b=f5();
	70768	返回引用

(下转第3874页)

本栏目责任编辑:谢媛媛

4.2 补码溢出的判断

计算机机器字长决定了所能表示的数据范围。一旦运算结果超出能表示的数据范围,就会产生溢出。发生溢出会导致运算结果错误,计算机必须判断是否产生溢出。异号数相加时,实际是两数的绝对值相减,不可能产生溢出,但有可能出现正常进位;同号数相加时,实际上是两数的绝对值相加,既可能产生溢出,也可能出现正常进位。

无论采用何种机器数,只要运算的结果大于计算机字长所能表示数的范围,就会产生溢出。如对于字长为n的机器能表示有符号整数范围为$-2^{n-1}\sim 2^{n-1}-1$。如上面示例,补码运算存在符号位进位自然丢失而运算结果正确的情况,因此,应区分补码的溢出与正常进位。机器是正常进位还是溢出的判定依据:当补码运算结果中最高位和次高位均无进位产生,或均产生进位,则结果正确;当最高位和次高位中只有一位产生了进位则结果是错误的。在微机中可用异或门电路来实现这种判断。

5 结束语

该文在阐述机器数的传统计算方法的基础上,提出了一种快速准确的原码与补码相互变补的速算方法,实际应用中,用传统方法计算机器数易出错,采用速算方法,效率和准确度都会有所提高。该文还根据机器数的速算算法,设计了计算真值数的三种机器数的C++源代码。

参考文献:

[1] 王玉龙,付晓玲,方英兰.计算机导论 [M]. 3版.北京:电子工业出版社,2010.
[2] 中国高等院校计算机基础教育改革课题研究组.中国高等院校计算机基础教育课程体系[M]. 北京:清华大学出版社,2004.
[3] 谭浩强.C程序设计 [M]. 4版.北京:清华大学出版社,2010.
[4] 邓超成,赵勇.大学计算机基础－Office 2003+VFP 6.0版[M]. 北京:科学出版社,2009.
[5] 王爱英.计算机组成与结构[M]. 北京:清华大学出版社,2001.
[6] 徐孝凯.C++语言.基础教程 [M]. 2版.北京:清华大学出版社,2007.

(上接第3855页)

表2的结果支持了上面的分析,引用和指针作为函数参数,在参数传递环节效率要远远高于传值调用。而传地址调用和传引用调用虽然使用形式不同,但底层实现机制是相同的,都只是把实体的地址传给形参,所以它们的调用效率也是相同的。调用函数f4时,语句return a将创建一个临时对象,并将a赋给该临时对象,返回主函数后,语句b=f4()又把该临时对象赋给b,在这过程中生成了临时副本,并进行了两次赋值操作。而调用函数f5时,语句return a不产生临时副本,主函数的赋值语句b=f5()中,对象b直接从对象a得到拷贝,这样避免了临时对象的产生,而且只需一个赋值操作,提高了程序运行效率。

既然引用和指针的底层实现机制和实质是一样的,那么在已经有了指针后,C++中为什么还要引入引用呢?原因在于指针虽然为编程提供了极大的灵活性,但是如果运用不当,就有可能变为悬空指针,出现程序运行错误、死机等现象。C++引入的引用,须依附于固定的实体,而不能随意指向其它地方,减少了出错的机会,并且在使用引用时,语法更简洁,程序的可读性更好。

3 结束语

在C++程序设计语言的教学过程中,很多师生对于引用的实现机制、它的本质,以及它与指针的区别与联系感到困惑。该文通过反汇编手段揭示了引用的底层实现机与指针一样的,也是属于间接访问,并通过实例验证了我们的分析。引用具有指针一样的高效与便捷性,使用上又比指针安全、简洁。正是因为引用的这些特点,在随后出现的Java和C#语言中引用已取代指针在C\C++中原有的地位,而且它的功能也得到了进一步的丰富。

参考文献:

[1] 钱能. C++程序设计教程[M]. 2版.北京:清华大学出版社, 2005.
[2] Echel B, Allison C. Thinking in C++ (Second Edition), Volume 1: Introduction to Standard C++[M]. Upper Saddle River: Prentice Hall, 2000.
[3] Echel B, Allison C. Thinking in C++, Volume 2:Practical Programming[M]. Upper Saddle River: Prentice Hall, 2004.
[4] 胡巧多.面向对象程序设计C++语言引用机制的剖析[J]. 长春工程学院学报, 2003,4(1):60-62.
[5] 张鸿. C++面向对象程序设计中引用的使用[J]. 河南教育学院学报, 2005,14(2):59-61.