# 深入C++探索--- 一 函数

## 零、声明

1. 这个系列《深入C++探索》，以汇编和内存角度对C++的一些基本的运行机制进行讨论；
2. 这个系列中出现的汇编代码，都是来自于VS2010的Debug环境下生成的；
3. 如果觉得文章对你有帮助而需要转载，也请阁下能够注明出处；
4. 如果觉得博文对问题的讨论有误，也可以给博主留言。

## 无参无返回值的函数

### 1.1.C++代码

函数：

void fun2()

{

int i = 0;

}

调用：

fun2();

### 1.2．汇编解释

void fun2()

{

012A17F0 push ebp

012A17F1 mov ebp,esp

012A17F3 sub esp,0CCh

012A17F9 push ebx

012A17FA push esi

012A17FB push edi

012A17FC lea edi,[ebp-0CCh]

012A1802 mov ecx,33h

012A1807 mov eax,0CCCCCCCCh

012A180C rep stos dword ptr es:[edi]

int i = 0;

012A180E mov dword ptr [i],0

}

012A1815 pop edi

012A1816 pop esi

012A1817 pop ebx

012A1818 mov esp,ebp

012A181A pop ebp

012A181B ret

//

在上一篇《开始》里讲过了其中大部分代码，所以我们这里把它们都去掉再看一下

void fun2()

{

int i = 0;

012A180E mov dword ptr [i],0 //将0赋值给i所指向的地方，这个i的位置是ebp-4的位置

}

## 有参无返回值的函数

### 2.1.C++代码

函数：

void fun3(int x)

{

x += 1;

}

调用：

fun3(10);

### 2.2.汇编代码

我们直接去掉函数的公共汇编代码，首先来看调用处的汇编

012A2D08 push 0Ah //压入参数0xA，就是fun3(10)中的10

012A2D0A call fun3 (12A11CCh) //调用fun3

012A2D0F add esp,4 //平衡栈，这句代码是否出现，还要看函数的调用方式

//函数的调用方式，后面篇章详细介绍，这里就不赘述了

看一下fun3内部的汇编。之后没有特殊说明，函数的公共汇编代码都会去掉用…替代

void fun3(int x)

{

...

x += 1;

012A205E mov eax,dword ptr [x] //将参数x的值赋给eax

//x的位置在ebp-4

-4

012A2061 add eax,1 //等价于eax = eax + 1

012A2064 mov dword ptr [x],eax //等价于x = eax

}

....

## 有参有返回值的函数

### 3.1.C++代码

函数：

int fun5(int x,int y)

{

int z = x+ y;

return z;

}

调用：

int z = fun5(30,40);

### 3.2.汇编代码

我们直接去掉函数的公共汇编代码，首先来看调用处的汇编

012A2D1E push 28h //压入参数0x28，就是fun5(30,40)中的40

012A2D20 push 1Eh //压入参数0x1E，就是fun5(30,40)中的30

//这里参数压入栈的方式也是和调用方式有关，

//这里C++使用的是\_\_stdcall，压栈是从右往左

012A2D22 call fun5 (12A11C7h) //调用fun5

012A2D27 add esp,8 //平衡栈

012A2D2A mov dword ptr [z],eax //函数的返回值在eax寄存中

看一下fun3内部的汇编。之后没有特殊说明，函数的公共汇编代码都会去掉用…替代

int fun5(int x,int y)

{

...

int z = x+ y;

012A267E mov eax,dword ptr [x] //将参数x赋给eax

012A2681 add eax,dword ptr [y] //将y和eax相加，相当于eax = eax + y

012A2684 mov dword ptr [z],eax //相当于z = eax

return z;

012A2687 mov eax,dword ptr [z] //将z赋给eax，函数的返回值是靠eax寄存器传递的

}

...

## 四、总结

去除函数的公共汇编代码，整个参数的传入方式和返回值如何返回，整个过程都比较清晰。这里没有去讨论临时变量的存放位置和传入参数如何取得，留在一下篇中详细介绍。