成员函数重载及其缺省参数（09-02）

1.前面我们已经介绍函数可以重载，函数重载有一个要求就是：相同的作用域。成员函数也可以重载，也需要在相同的作用域里面。也就是在类的作用域当中。

2.这里假设我们定义三个数据成员，没有实际定义，

```
(全局范围)
#ifndef _TEST_H_
#define _TEST_H_
class Test
{
    public: //定义成内联的有两种方式
        int add(int a,int b)//假设我们要将该函数定义为内联的，声明的时候可以不给出inline关键字，可以在实现的时候给出inline关键字
        {                   //第一种内联函数的方式在类外给出实现代码，并加关键字inline,这一种是在类里给出实现，不用加inline关键字
            return a+b;    //即使我们没有加inline关键字，它也是内联的
        }
    private:      //定义三个数据成员，没有实际的意义，举例而已
        int x_;
        int y_;
        int z_;
};
#endif //_TEST_H_
```

需要对这三个数据成员进行初始化：

```
#ifndef _TEST_H_
#define _TEST_H_
class Test
{
    public: //定义成内联的有两种方式
        int add(int a,int b)//假设我们要将该函数定义为内联的，声明的时候可以不给出inline关键字，可以在实现的时候给出inline关键字
        {                   //第一种内联函数的方式在类外给出实现代码，并加关键字inline,这一种是在类里给出实现，不用加inline关键字
            return a+b;    //即使我们没有加inline关键字，它也是内联的
        }
        //定义成员变量的初始化函数
        void Init();
        void Init(int x);
        void Init(int x,int y);
        void Init(int x,int y,int z);
    private:      //定义三个数据成员，没有实际的意义，举例而已
        int x_;
        int y_;
        int z_;
};
#endif //_TEST_H_
```

那么这四个初始化函数就构成了重载，因为他们的作用域是相同的，函数名相同，但是参数不同，并且他们都是在相同的类中。

接下来我们实现一下这四个函数：

Test.cpp* | Test.h

(全局范围)

```cpp
#include "Test.h"
//inline int Test::add(int a,int b)  //这是一种内联函数的定义方式  //第二种定义方式是直接在类中给出实现代码
//{
//  return a+b;
//}
//以下是四个成员变量初始化函数的实现
void Test::Init()
{
    x_=0;
    y_=0;
    z_=0;
}
void Test::Init(int x)
{
    x_=x;
    y_=0;
    z_=0;
}
void Test::Init(int x,int y)
{
    x_=x;
    y_=y;
    z_=0;
}
void Test::Init(int x,int y,int z)
{
    x_=x;
    y_=y;
    z_=z;
}
```

增加显示成员变量函数和实现函数：

01.cpp* | Test.cpp* | Test.h*

Test

```cpp
#ifndef _TEST_H_
#define _TEST_H_
class Test
{
    public: //定义成内联的有两种方式
        int add(int a,int b)//假设我们要将该函数定义为内联的，声明的时候可以不给出inline关键字，可以在实现的时候给出inline关键字
        {                   //第一种内联函数的方式在类外给出实现代码，并加关键字inline,这一种是在类里给出实现，不用加inline关键字
            return a+b;    //即使我们没有加inline关键字，它也是内联的
        }
        //定义成员变量的初始化函数
        //这四个初始化函数就构成了重载（函数名相同，参数不同），因为他们的作用域是相同的，因为他们都是在相同的类中
        void Init();
        void Init(int x);
        void Init(int x,int y);
        void Init(int x,int y,int z);
        //显示三个成员变量
        void Display();
    private:      //定义三个数据成员，没有实际的意义，举例而已
        int x_;
        int y_;
        int z_;
};
#endif //_TEST_H_
```

```cpp
#include "Test.h"
#include "iostream"
using namespace std;
//inline int Test::add(int a,int b)  //这是一种内联函数的定义方式  //第二种定义方式是直接在类中给出实现代码
//{
//  return a+b;
//}
//以下是四个成员变量初始化函数的实现
void Test::Init()
{
    x_=0;
    y_=0;
    z_=0;
}
void Test::Init(int x)
{
    x_=x;
    y_=0;
    z_=0;
}
void Test::Init(int x,int y)
{
    x_=x;
    y_=y;
    z_=0;
}
void Test::Init(int x,int y,int z)
{
    x_=x;
    y_=y;
    z_=z;
}
void Test::Display()
{
    cout<<"x_="<<x_<<"y_="<<y_<<"z_"<<z_<<endl;
}
```

函数的调用和最终的效果如下：

```cpp
#include "Test.h"
#include "iostream"
int main(void)
{
    Test t; //定义一个Test对象
    t.Init();//成员变量初始化,默认值成员变量
    t.Display();//显示成员变量
    t.Init(5); //初始化一个参数的初始化函数
    t.Display();
    t.Init(2,5);//初始化两个参数的初始化函数
    t.Display();
    t.Init(6,3,7);//全部参数都初始化的函数
    t.Display();
    return 0;
}
```

```
C:\Windows\system32\cmd.exe
x_=0 y_=0 z_=0
x_=5 y_=0 z_=0
x_=2 y_=5 z_=0
x_=6 y_=3 z_=7
请按任意键继续. . .
```

这就是成员函数的重载。

接下来我们可以下成员函数的缺省参数：

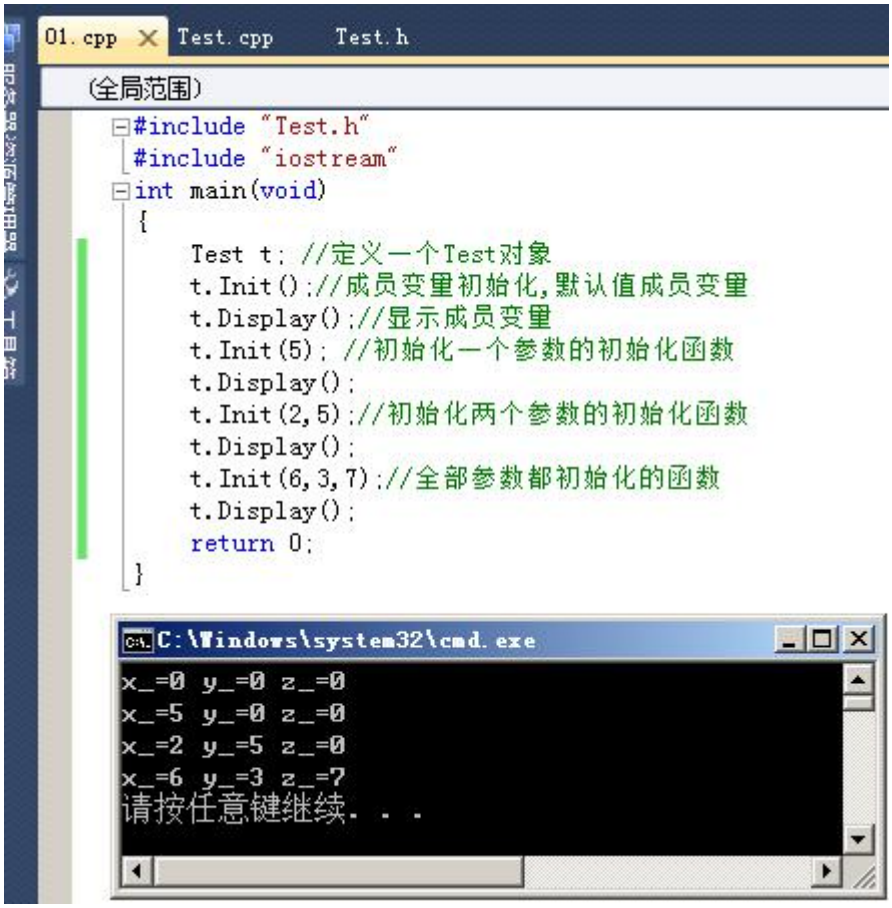
实际上我们重载的例子可以用另外一种方式编写：声明修改为

```
//定义成员变量的初始化函数
//这四个初始化函数就构成了重载（函数名相同，参数不同），因为他们的作用域是相同的，因为他们都是在相同的类中
/*void Init();
void Init(int x);
void Init(int x,int y);
void Init(int x,int y,int z);*/
//前面重载的四个初始化函数，可以写为下面这句
void Init(int x=0,int y=0,int z=0);
```

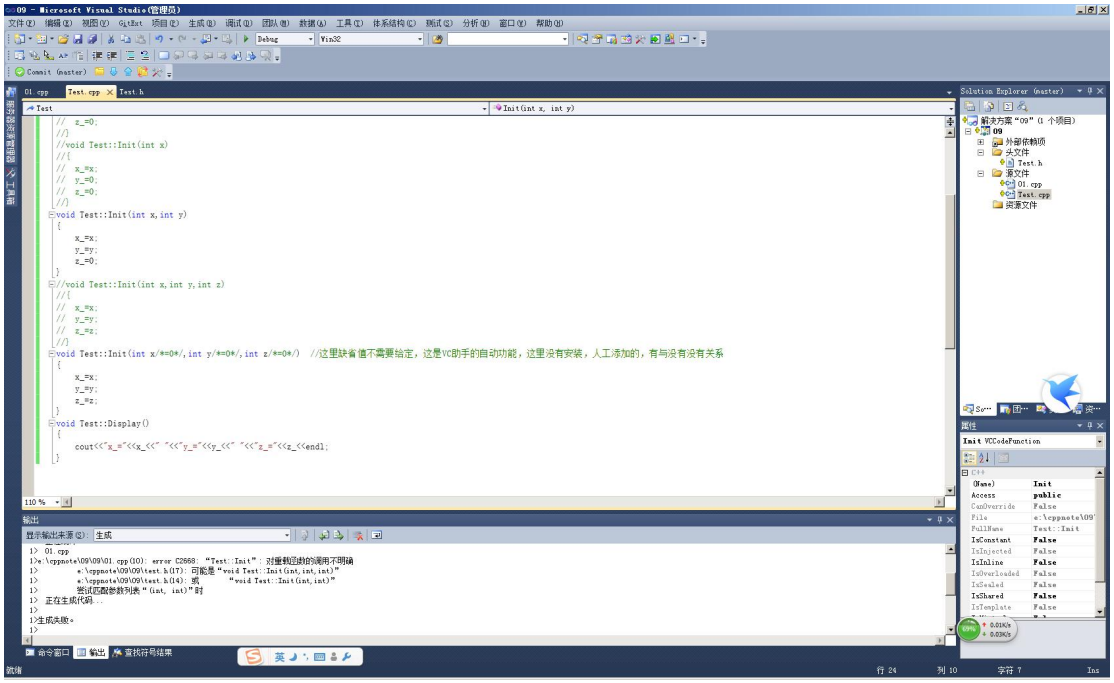
实现修改为：

```
void Test::Init(int x/*=0*/,int y/*=0*/,int z/*=0*/)  //这里缺省值不需要给定，这是VC助手的自动功能，这里没有安装，人工添加的，有与没有没有关系
{
    x_=x;
    y_=y;
    z_=z;
}
```

调用代码不需要修改，最后的效果和之前的一模一样，但是实现的形式简单了很多：



如果将之前的某个重载函数打开：


09 - Microsoft Visual Studio(管理员)
Test.cpp
Test.h
Init(int x, int y)
void Test::Init(int x,int y)
{
x_=x;
y_=y;
z_=0;
}
void Test::Init(int x/*=0*/,int y/*=0*/,int z/*=0*/) //这里缺省值不需要给定，这是VC助手的自动功能，这里没有安装，人工添加的，有与没有没有关系
{
x_=x;
y_=y;
z_=z;
}
void Test::Display()
{
cout<<"x_="<<x_<<" "<<"y_="<<y_<<" "<<"z_="<<z_<<endl;
}
输出
1>01.cpp
1>e:\cppnote\09\09\01.cpp(10): error C2668: "Test::Init": 对重载函数的调用不明确
1> e:\cppnote\09\09\test.h(17): 可能是"void Test::Init(int,int,int)"
1> e:\cppnote\09\09\test.h(14): 或 "void Test::Init(int,int)"
1> 尝试匹配参数列表"(int, int)"时
1> 正在生成代码...
1>
1>生成失败。
1>
Solution Explorer (master)
解决方案"09"(1 个项目)
09
外部依赖项
头文件
Test.h
源文件
01.cpp
Test.cpp
资源文件
属性
Init VCCodeFunction
(Name) Init
Access public
CanOverride False
File e:\cppnote\09
FullName Test::Init
IsConstant False
IsInjected False
IsInline False
IsOverloaded False
IsSealed False
IsShared False
IsTemplate False
命令窗口
输出
查找符号结果
就绪
行 24
列 10
字符 7
Ins

就会产生二义性。