


内联成员函数

1. **内联函数能够提高效率**：因为内联函数在编译的时候，直接将代码嵌入到调用的地方。从而减少了函数调用的开销，我们知道函数调用有一定的开销，他需要建立**系统堆栈**，**保护现场**，**将参数压入栈当中**，并且**控制程序的跳转**。这些需要一定的开销。而内联函数，直接将调用的代码直接嵌入到调用的地方，就不需要这些开销，所以说它能提高效率。既然它是嵌入到调用的地方，那么每调用一次，就会增大一次，也就是说程序的体积会增大，这实际上是以**空间换时间**的例子。
2. 程序的体积膨胀了，但是运行的效率提高了。所以说一般你内联函数一般都比较**短小**，否则编译器不会把它指示为内联的，你连函数仅仅是给编译器一个提示而已，如果函数中有 `switch`, `for` 这些语句的话，那么编译器很可能就不会将他以内联的方式来解析。
3. 类当中的成员函数也可以做成内联的，接下来我们编写一个类做一个说明：类中的内联函数有两种定义方式，具体看代码：



```
01.cpp* Test.h Test.cpp*
Test
#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
    }
};
#endif // _TEST_H_
```



```
01.cpp* Test.h Test.cpp*
(全局范围)
#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;
//}
```