

## 1.什么是内联函数

用关键字inline修饰的函数就是内联函数。关键字在函数声明和定义的时候都要加上，不写系统还是会当成常规函数

## 2.内联函数与一般函数的区别

1) 内联含函数比一般函数在前面多一个inline修饰符

2) 内联函数是直接复制“镶嵌”到主函数中去的，就是将内联函数的代码直接放在内联函数的位置上，这与一般函数不同，主函数在调用一般函数的时候，是指令跳转到被调用函数的入口地址，执行完被调用函数后，指令再跳转回主函数上继续执行后面的代码；而由于内联函数是将函数的代码直接放在了函数的位置上，所以没有指令跳转，指令按顺序执行

3) 一般函数的代码段只有一份，放在内存中的某个位置上，当程序调用它是，指令就跳转过来；当下一次程序调用它是，指令又跳转过来；而内联函数是程序中调用几次内联函数，内联函数的代码就会复制几份放在对应的位置上

4) 内联函数一般在头文件中定义，而一般函数在头文件中声明，在cpp中定义

## 3.利与弊

利：避免了指令的来回跳转，加快程序执行速度

弊：代码被多次复制，增加了代码量，占用更多的内存空间

## 4.什么时候使用内联函数

1) 函数本身内容比较少，代码比较短，函数功能相对简单

2) 函数被调用得频繁，不如循环中的函数

## 5.什么时候不能使用内联函数

1) 函数代码量多，功能复杂，体积庞大。对于这种函数，就算加上inline修饰符，系统也不一定会相应，可能还是会当成一般函数处理

2) 递归函数不能使用内联函数

## 6.内联函数比宏更强大

看一段代码：

```
#include <iostream>
using namespace std;
#define SUM(x) x*x
inline int fun(int x)
{
    return x * x;
}
```

```

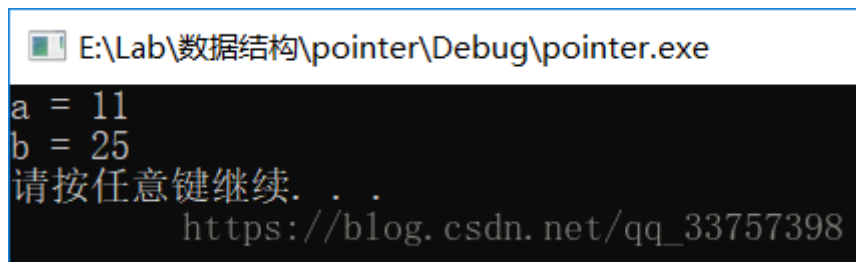
int main()
{
    int a = SUM(2 + 3);
    int b = fun(2 + 3);

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    system("pause");
    return 0;
}

```

执行结果：



```

E:\Lab\数据结构\pointer\Debug\pointer.exe
a = 11
b = 25
请按任意键继续. . .

```

为什么通过宏执行的结果是11呢，宏比较机械和简单，只是将传入的参数直接放上去就执行，所以 `int a = SUM(2 + 3);` 就相当于 `int a = 2 + 3 * 2 + 3;` 由于乘法优先级更高，所以得到a的值为11；而在内联函数中，传入的参数是5，所以得到25

为了得到正确的结果，我们应该将宏改变为：

```
#define SUM(x) ((x)*(x))
```

## 7.类与内联函数

- 1) 类内定义的函数都是内联函数，不管是否有inline修饰符
- 2) 函数声明在类内，但定义在类外的看是否有inline修饰符，如果有就是内联函数，否则不是