# 一、最简单的函数指针

　　变量都包括声明和赋值，指针不例外，函数指针也不例外。我们来看一个简单的函数：

void add(int a, int b){

cout << a + b << endl;

}

　　一个简单的加法计算并输出到命令行的函数。

　　那么如何通过函数指针来调用它呢？

**1、声明：**

void (\*p1)(int a, int b);

　　函数指针的声明很简单，基本就是通过一个指针把函数名替换。指针p1的类型为void (\*) (int a,int b)，表明指针是一个指向某个函数的指针，指针指向的类型为void () (int a,int b)

**2、赋值：**

p1 = add;

**3、也可以直接定义：**

void (\*p1)(int a, int b) = add;

　　注意，函数void add(int a,int b)的函数名add就是函数的地址。将地址add赋值给指针p1，那么就可以通过函数指针p1直接调用函数了。

**4、调用：**

(\*p1)(1, 2);

p1(1, 2);

　　注意！出于历史原因以上2种方式都是可以调用函数的。

**二、包含多个函数指针的数组**

　　有时候有这种情况，有一个数组，数组中的每个元素都是一个函数指针，该怎么定义这个数组呢？

**1、解释\*p[n]和(\*p)[n]**

　　我们知道，[]运算符的优先级要高于\*，所以，p[3]表示含有3个元素的数组，而\*p[3] 前面的 " \* " 指明了数组中元素的类型，即\*p[3]表示一个指向3个指针的数组。

　　p[3]表示含有3个元素的数组，那么(\*p)[3]就是用 \*p 替换了 p，很容易想到，(\*p)[3] 表示指向一个包含3个元素的数组的指针。

**2、声明：**

void (\*p2[2])(int a, int b);

　　数组名为p2，数组大小为2，数组中元素类型为void (\*)(int a, int b)，表明元素是一个指向某个函数的指针，指针指向的类型为void () (int a,int b)。

**3、赋值：**

p2[1] = add;

　　理解上跟上面是一样的。

**4、调用：**

p2[1](2,3);

(\*p2[1])(3,4);

　　同样是2种方式都可以。

**三、指向“包含多个函数指针的数组“的指针**

　　这个标题好像有点拗口。简而言之，这个指针指向上文中的 “包含多个函数指针的数组” 。其实很简单，说白了，就是把上文中的p2用一个指针来代替。

**1、声明：**

void (\*(\*p3)[2])(int a, int b);

 　　可以看到，无非就是把p2用\*p3代替。

**2、赋值，注意，既然是指针，使用前必须初始化：**

p3 = &p2;  
(\*p3)[1] = add;

　　注意！既然实质上就是把p2用\*p3代替，c++11可以很简单的这样直接定义：auto p3 = &p2; 代替了void (\*(\*p3)[2])(int a, int b)= &p2;

**3、调用：**

(\*p3)[1](1, 2);

(\*(\*p3)[1])(1, 2);