问题

函数指针和指针函数有什么区别？

指针函数

指针函数：**本质上是一个函数，其返回值是一个指针。**

声明形式：（以下几种都可以）

```
int *fun(int x,int y);
int * fun(int x,int y);
int* fun(int x,int y);      //这种写法相对容易理解一些，函数的返回值是一个 int* 指针
```

指针函数代码示例：

```
#include<iostream>
using namespace std;
struct Data {
    int a;
    int b;
};
//指针函数
Data* fun(int a, int b) {
    Data* data = new Data;
    data->a = a;
    data->b = b;
    return data;    //返回的是一个指针
}
int main(){
    Data* myData = fun(4, 5);    //调用指针函数
    cout << "a = " << myData->a << endl;
    cout << "b = " << myData->b << endl;
    return 0;
}
```

函数指针

函数指针：**本质上是一个指针，一个指向函数的指针变量。**

声明格式：类型说明符 (*函数名) (形参)

```
int (*fun)(int x,int y);    //声明一个函数指针，即一个指向函数的指针
```

函数赋值：（即函数指针需要把一个函数的地址赋值给它）

```
fun = &Function;    //假设Function是一个函数名，&取函数的地址
fun = Function;     //&不是必需的，因为函数名本身就表示了它的地址，类似数组名一样
```

函数指针的调用：（以下两种都可以，但都必须包含一个圆括号括起来的参数列表）

```
a = (*fun)();    //这种写法更加直观一点，可以直接看出是通过只针的方式调用函数的
a = fun();
```

函数指针代码示例：

```
#include<iostream>
using namespace std;

int add(int x, int y) { return x + y; }
int sub(int x, int y) { return x - y; }

int main(){
    int (*fun)(int x, int y);    //声明一个函数指针

    fun = &add;      //函数赋值第一种方式
    cout << "(*fun)(1,2) = " << (*fun)(1, 2) << endl;    //通过函数指针调用函数

    fun = sub;       //函数赋值第二种方式
    cout << "(*fun)(5,3) = " << (*fun)(5, 3) << endl;
    return 0;
}
```

总结

总结一下函数指针和指针函数的区别：

定义不同

指针函数：**本质上是一个函数，其返回值是一个指针**。

函数指针：**本质上是一个指针，一个指向函数的指针**。

写法不同

指针函数： int* fun(int x,int y);

函数指针： int (*fun)(int x,int y);

区分方法：函数名带括号的就是函数指针，不带括号的就是指针函数。

> 补充：还可以通过从右到左结合的方式区分，如
>
> ①对于指针函数 int* fun(int x,int y)，形参列表先和函数名fun结合，也就是说fun首先是一个函数，剩下的为函数的返回类型int*。
>
> ②对于函数指针 int (*fun)(int x,int y)，因为括号优先级高，所以*号会先和函数名fun结合，也就是说fun此时先是一个指针，再与形参列表结合，成为一个指向函数的指针，剩下的int为函数的返回类型。

用法不同

具体请看上面代码示例。

参考资料

函数指针和指针函数用法和区别