挑战30天C 入门极限：C_C 中函数指针的含义 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/134/2021_2022__E6_8C_91_E6_88_9830_E5_A4_c97_134526.htm 函数存放在内存的代码区域内，它们同样有地址，我们如何能获得函数的地址呢？ 如果我们有一个int test(int a)的函数，那么，它的地址就是函数的名字，这一点如同数组一样，数组的名字就是数组的起始地址。 定义一个指向函数的指针用如下的形式，以上面的test()为例： int (*fp)(int a).//这里就定义了一个指向函数的指针 函数指针不能绝对不能指向不同类型，或者是带不同形参的函数，在定义函数指针的时候我们很容易犯如下的错误。int *fp(int a).//这里是错误的，因为按照结合性和优先级来看就是先和()结合，然后变成了一个返回整形指针的函数了，而不是函数指针，这一点尤其需要注意！ 下面我们来看一个具体的例子：#include iostream> #include string> using namespace std. int test(int a). void main(int argc,char* argv[]) { cout//显示函数地址 int (*fp)(int a). fp=test.//将函数test的地址赋给函数学指针fp cout//上面的输出fp(5),这是标准c 的写法,(*fp)(10)这是兼容c语言的标准写法,两种同意,但注意区分,避免写的程序产生移植性问题! cin.get(). } int test(int a) { return a. } typedef定义可以简化函数指针的定义，在定义一个的时候感觉不出来，但定义多了就知道方便了，上面的代码改写成如下的形式：#include iostream> #include string> using namespace std. int test(int a). void main(int argc,char* argv[]) { cout typedef int (*fp)(int a).//注意,这里不是生命函数指针,而是定义一个函数指针的类型,这个类型是自己定义的,类型名为fp fp

fpi.//这里利用自己定义的类型名fp定义了一个fpi的函数指针! fpi=test. cout cin.get(). } int test(int a) { return a. } 函数指针同样是可以作为参数传递给函数的，下面我们看个例子，仔细阅读你将会发现它的用处，稍加推理可以很方便我们进行一些复杂的编程工作。//------------------该例以上一个例子作为基础稍加了修改---------- #include iostream> #include string> using namespace std. int test(int). int test2(int (*ra)(int),int). void main(int argc,char* argv[]) { cout typedef int (*fp)(int). fp fpi. fpi=test.//fpi赋予test 函数的内存地址 cout//这里调用test2函数的时候,这里把fpi所存储的函数地址(test的函数地址)传递了给test2的第一个形参 cin.get(). } int test(int a) { return a-1. } int test2(int (*ra)(int),int b)//这里定义了一个名字为ra的函数指针 { int c=ra(10) b.//在调用之后,ra已经指向fpi所指向的函数地址即test函数 return c. } 利用函数指针，我们可以构成指针数组，更明确点的说法是构成指向函数的指针数组，这么说可能就容易理解的多了。#include iostream> #include string> using 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com