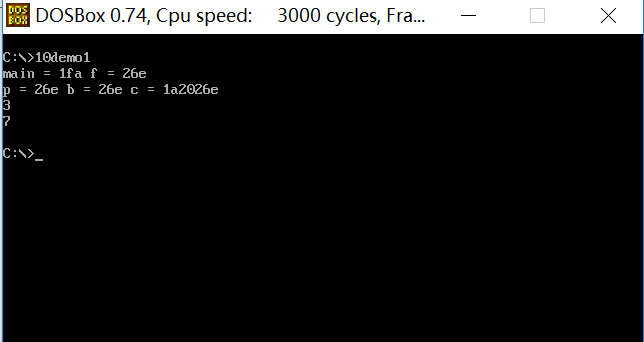
**c语言综合研究十报告**

1. **研究过程展示**

**1>.程序1部分：**

第一个程序修改了一下参数。

运行一下：



结果表明用函数指针p和用变量b强转，都实现了调用函数的效果。

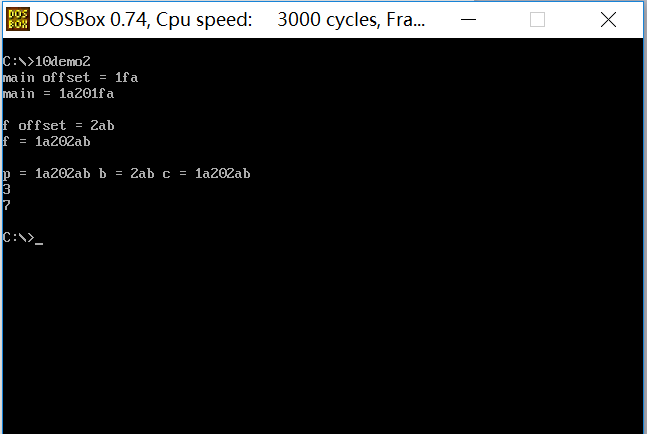
这里最令我感兴趣的就是“a = ( (int (\*)(char,char) )b )(3,4);”这一句，因为他又让我想起了自己之前那段逻辑不清的表述，这里也是可以归类于其中的另一种“强制类型转换”，尽管书上没有教过我们这个道理，但它的确是存在的，这个程序恰恰验证了。“(int (\*)(char,char) )b”这里就是将整型变量b强制类型转换为“返回值为int，参数为char、char的函数的函数指针变量类型”。而之所以可以实现f（）函数的功能，是因为在这之前b的值已经通过“b =（int）f”赋值为f（）函数的首地址了。

看到这里，感觉之前自己认为的“（&）a”，这里a是一个普通变量，假设是整型吧，可以有另一种自己觉得更好的解释，本来a的地址与变量a以及变量a的值就是一体的，但是我们平时默认的就是正面的角色-》a的值，而（&）a自然就是反面角色了。但是正反从来都是一体的。看过不少武侠书籍之中，总会提到一种常见的机关门，按下机关，门就会翻转，主人公就能抵达暗室了。在这里，a的值，以及（&）a的值，就像是这扇会翻转的门的正反两面，而他们从来都不是分割的，因为他们都在a这扇门上。嘿嘿，感觉好像是在“翻牌子”；那么“\*p”呢？其实我更喜欢“->”的标记来描述。这里p是一个普通的指针变量，假设是整型指针吧，它又有什么自己的解释呢？它更像个穿越门，穿越空间，不穿越时间。到了链表中就更好说了，通过p可以穿越到哪里，制定目的地的主人，就是我们了。多个穿越门可以到达同一个地方，一个穿越门可以修改到达的目的地，穿越了之后可以再次穿越。。。。。。我去 ，怎么有点哈利波特的感觉了？哈哈。希望以后现实中也可以有这种超级黑科技。应该，可以实现吧？

程序部分：

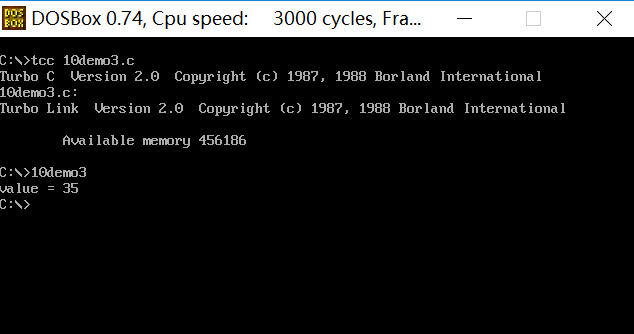
**2>.程序2部分：**

运行一下：



**3>.程序3部分：**

构造程序来描述函数指针数组的用法和向函数传函数指针的方法。



程序部分：

int f1(int,int);

int f2(int,int);

int f( int (\*)(int ,int),int (\*)(int ,int) );

void main(void)

{

int value = 0;

int (\*(p[2]))(int ,int);

p[0] = f1;

p[1] = f2;

value = f( (p[0]),(p[1]) );

printf("value = %d",value);

}

int f1(int a,int b)

{

return a+b;

}

int f2(int a,int b)

{

return a\*b;

}

int f( int (\*a)(int ,int ), int (\*b)(int ,int) )

{

return a(1,2)+b(8,4);

}

1. **已思考研究并已解决问题汇总**

函数指针数组及想函数传递函数指针的方法。可以见第3个程序

1. **已思考研究并未解决问题汇总**
2. 怎么自己提不出问题了？是提不出还是不想提了？
3. 自己的精力是不是不够做自己另外要做的、想做的东西了，怎么解决？自己想办法吧。
4. **研究感想（心得体会）**

自己总是忍不住对研究过程中的对象联想，比如什么翻转门，穿越门，可是又怕联想会破坏自己对对象本身的正确认识。不管怎样，尽力联想吧，年轻人不要怕犯错。