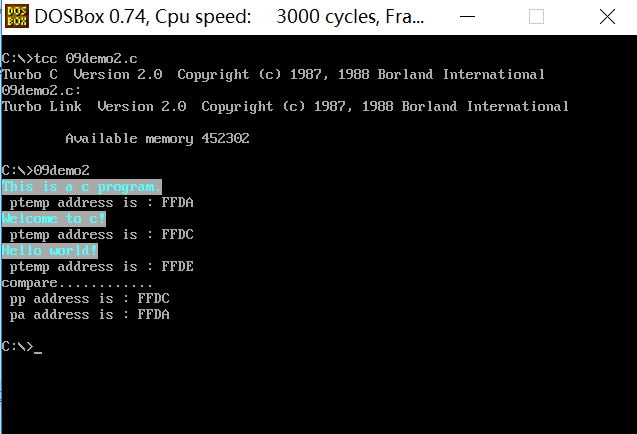
**九**

三种方法实现程序：二级指针、stdarg.h、汇编（\_BP）

1. 对#define va\_start (ap,parmN)

(ap=...)的理解

运行：



09demo2.c程序部分：

#include <stdarg.h>

void showstr(char \*,...);

char \*p1 = "This is a c program.";

char \*p2 = "Welcome to c!";

char \*p3 = "Hello world!";

main()

{

showstr(p1,p2,p3,0);

}

void showstr(char\* p,...)

{

va\_list pp;

int\* ptemp;

unsigned int i;

char \*\* pa;

pa = &p;

ptemp = (int\*)(&p);

while(1)

{

if( 0 == \*ptemp )

{

break;

}

else

{

for(i = 0; i < strlen(\*ptemp) ; i++)

{

printf("\33[1m\033[47;36m%c\033[0m",\*( (char\*)(\*ptemp) + i) );

}

printf("\n");

printf(" ptemp address is : %p \n",ptemp);

ptemp++;

}

}

pp = ...;

printf("compare............\n");

printf(" pp address is : %p \n",pp);

printf(" pa address is : %p \n",pa);

}

分析部分：

可以看到，“...”打印出来的地址是第二个字符串的首地址，也就是第二个参数的首地址，同时也是第一个不定参数的首地址。

1. 说明char \*p=\*ap++; 的运算过程，写程序验证

程序1：

main()

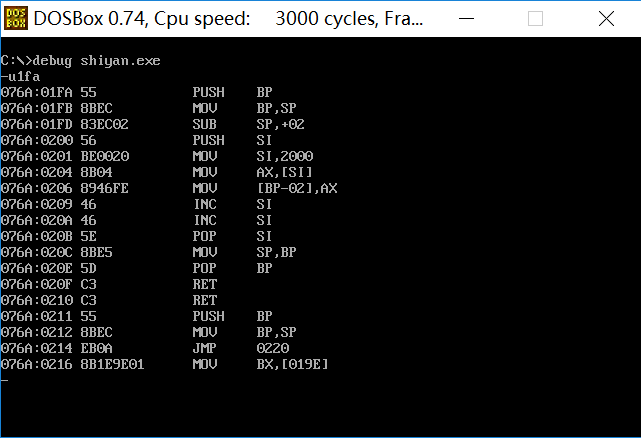
{

char\*\* ap = (char\*)0x2000;

char\* p = \*ap++;

}

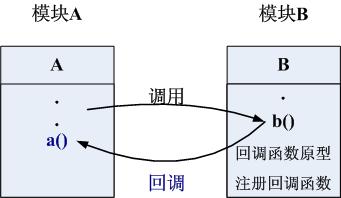
Debug程序1：



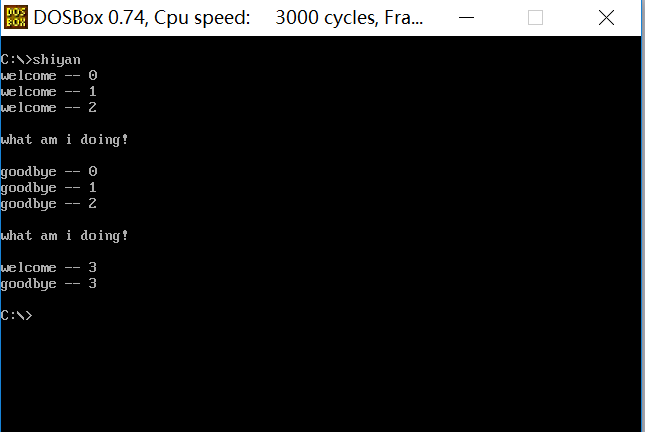
看来平常最好还是加括号，易理解，也不容易出错。

**十**

1. 理解回调函数



运行一下：



程序部分：

#include <stdio.h>

void printWelcome(int len)

{

printf("welcome -- %d\n", len);

}

void printGoodbye(int len)

{

printf("goodbye -- %d\n", len);

}

void callback(int times, void (\* print)(int))

{

int i;

for (i = 0; i < times; ++i)

{

print(i);

}

printf("\nwhat am i doing!\n\n");

}

void main(void)

{

callback(3, printWelcome);

callback(3, printGoodbye);

printWelcome(3);

printGoodbye(3);

}

这里，callback（）是可以调用两个回调函数（ printWelcome（）、printGoodbye（） ）的函数，这两个回调函数“类型”是一样的。Callback（）这里就像普通的整型变量一样用“这两个回调函数的类型”定义了一个形参变量给自己，另外还有一个变量，是在main（）函数中可以提供的。因为打印时候的i本质上是受传入的参数times影响和控制的。

所以这个有什么用呢？网上对这个讲的真的是五花八门，看着都能睡着了。我就觉得他让我最感兴趣的就俩点儿：

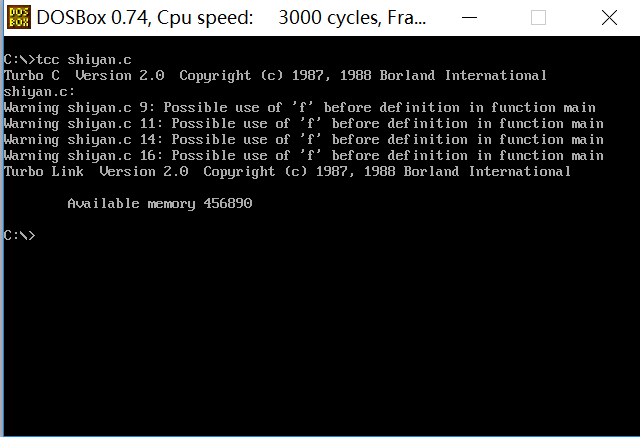
1. 把函数像变量来使用，通过使用函数的地址、函数指针等；
2. 在最外面的函数，这里就是main（），通过在main（）函数里面进行操作，也能很轻松按照我们想要的方式控制和影响最里面的函数的变化，从而实现最终结果变量化的功能。

表面上看，中间函数callback（）可以通过函数指针形参来选择回调函数，最外层函数又可以通过times传参影响callback函数，但是这个times参数还会对回调函数产生影响，而且其实最终目的就是为了影响最内层的回调函数。

我该不会越讲越乱了吧。

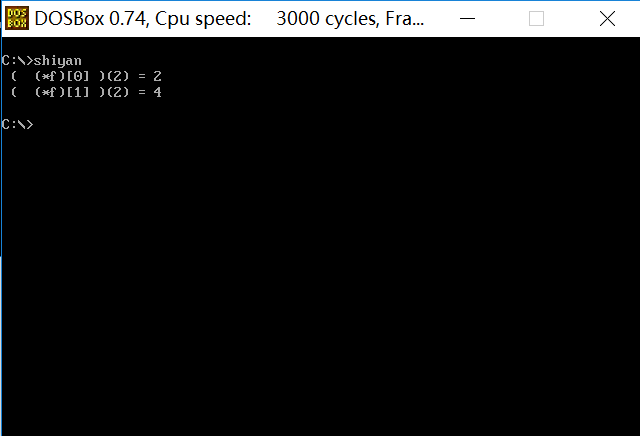
1. 定义并解释 函数指针数组指针：

运行一下：



可以看到有警告。

可是能运行呀：



不过还没有看那些书呢，就只是自己加了括号，肯定有地方做的不合适，得好好比对一下声明规则。

程序部分：

int f1(int);

int f2(int);

main()

{

int (\*( (\*f)[2]) )(int);

(\*f)[0] = f1;

(\*f)[1] = f2;

printf(" ( (\*f)[0] )(2) = %d \n",( (\*f)[0] )(2));

printf(" ( (\*f)[1] )(2) = %d \n",( (\*f)[1] )(2));

}

int f1(int a)

{

return a;

}

int f2(int a)

{

return 2\*a;

}

分析部分：

运算符了，复杂声明了，难道不是加个括号的事么？还是很有必要弄懂呢？