一、解决问题

语句for(n=0;code[n]&&code[n]!=ch;n++);中为什么要使用code[n]&&code[n]!=ch;而不用code[n]!=ch？

为了防止n++的时候超出字符串“+-\*/”的范围。

二、提出问题：

1、重新总结针对12的关于共性和个性的问题

研究12中b.c程序的共性和个性的思考可以分成几个部分。

第一部分：通过函数指针数组func[n]中选择n的值来实现调用不同的函数，其中func[]是共性，通过变量n来调用不同的add（），sub（）等函数，参数不同，这是个性；

第二部分：几个个性函数的接收参数都是两个，这是共性，但是对于两个参数的操作是不同的，这是个性；

第三部分：字符串code中的“+-\*/”等都是可以增减变化的，而且各自的意义不同，这是个性，但是函数中对于code[n]数组元素的遍历及判断选择操作并不会因为code中元素内容的改变而更改，这是共性。

2、将加减乘除的运算加入到13的程序中

放在综合研究13里了。

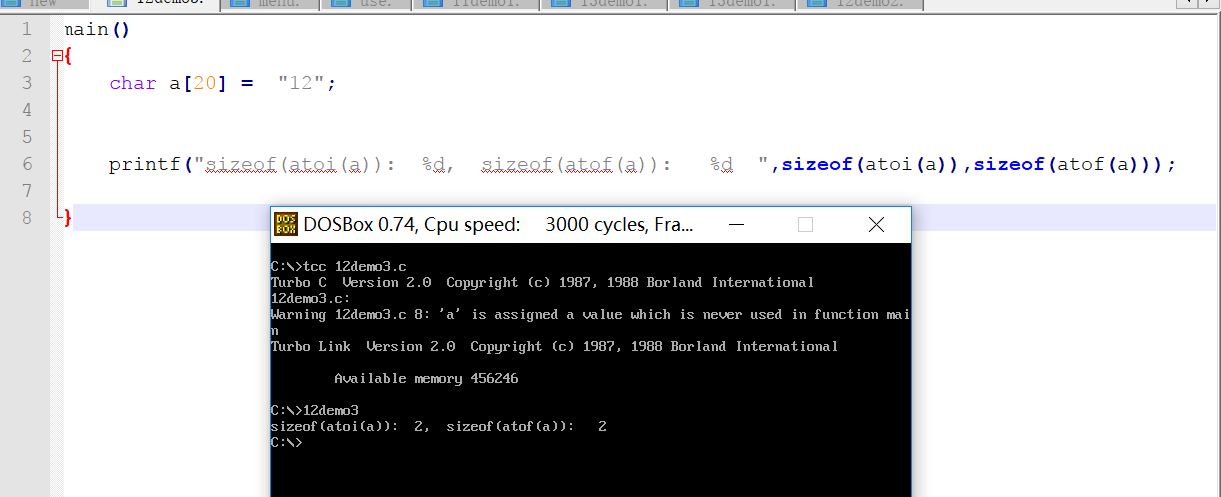
3、（1）atof()为什么不声明也能使用？ （会到库中查找，maths.lib）

（2）不声明为什么得到的结果不对？

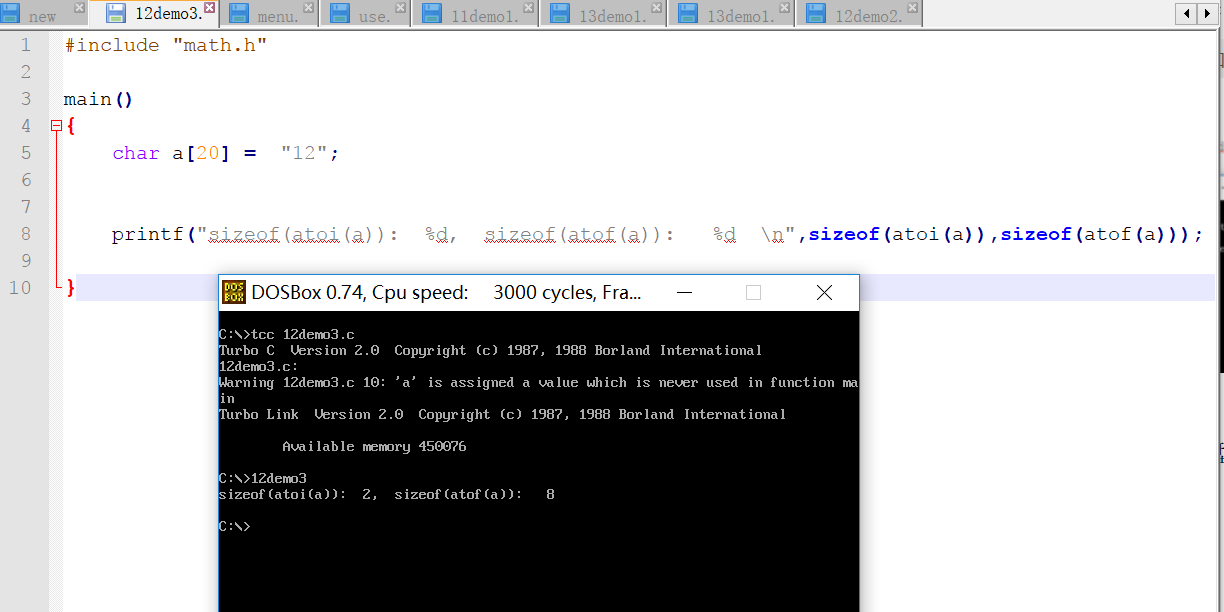
Atof（）的声明在math.h文件中：

“double \_Cdecl atof (const char \*s);”

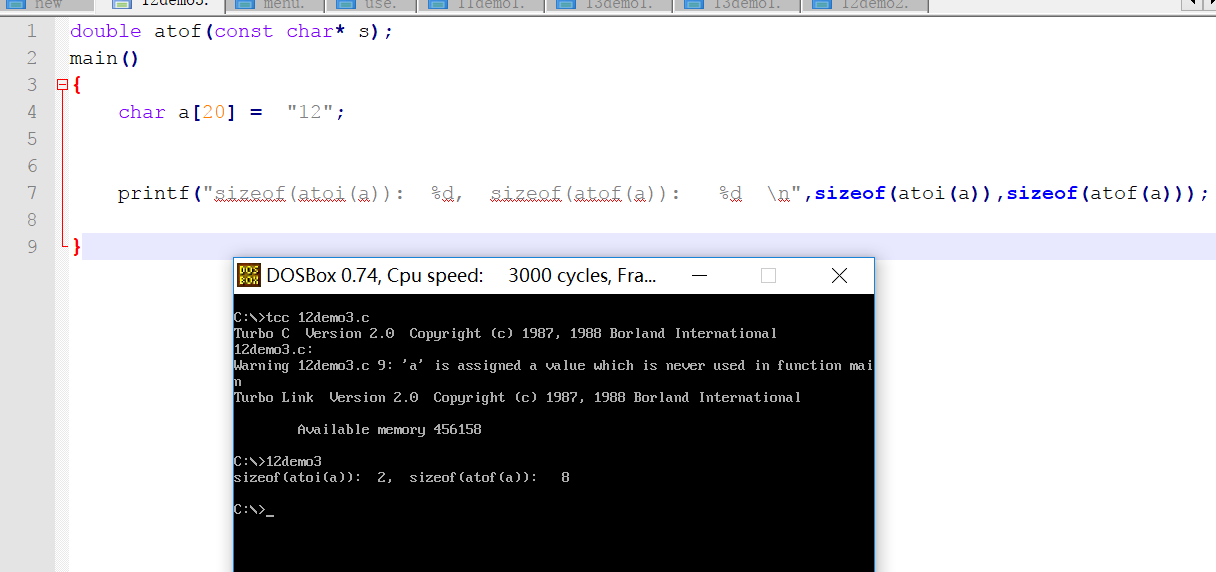
返回值为double型。



正常情况下，调用成功atof（）之后，返回值类型应该是double类型，用sizeof答应出的结果应该是8.引入math.h头文件，可以确认一下：



这说明在调用的过程中，atof（）的返回值类型由原本的double型变为了int型，而在这之前我们编写的程序中并没有对函数atof（）的正确声明，但是编译器会自动生成一个返回值为int型的声明 ，由此出错。下面可以不引入math.h，在此程序中自己写出atof（）的正确声明看一下：



还是会显示为8。

原因应该就是这样。

再用debug看一下：

自己编写atof（）函数会在编译的时候就出错，但是原本在库中的原本的atof（）函数只有在连接的时候才会出现。编译器对函数的默认处理是返回值是int类型，参数是不定参数。这样做的意义？返回值类型决定了调用的时候，入栈进行操作的不同，参数的话，如果未声明，则会出现多出参数的警告，但是不会有错误。

4、什么是函数原型？什么是函数体？

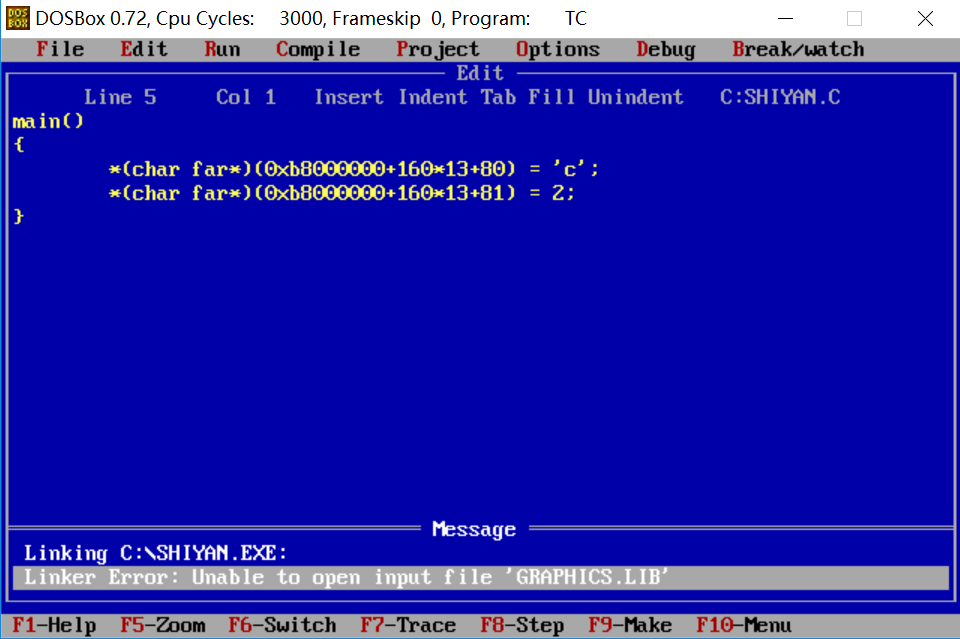
5、graphic.lib在tc中要用到，在tc中起作用吗？起什么作用？

在原来用tc图形化编程文件夹C:\minic目录下的graphics.lib文件删掉，运行tc2.0试一下：



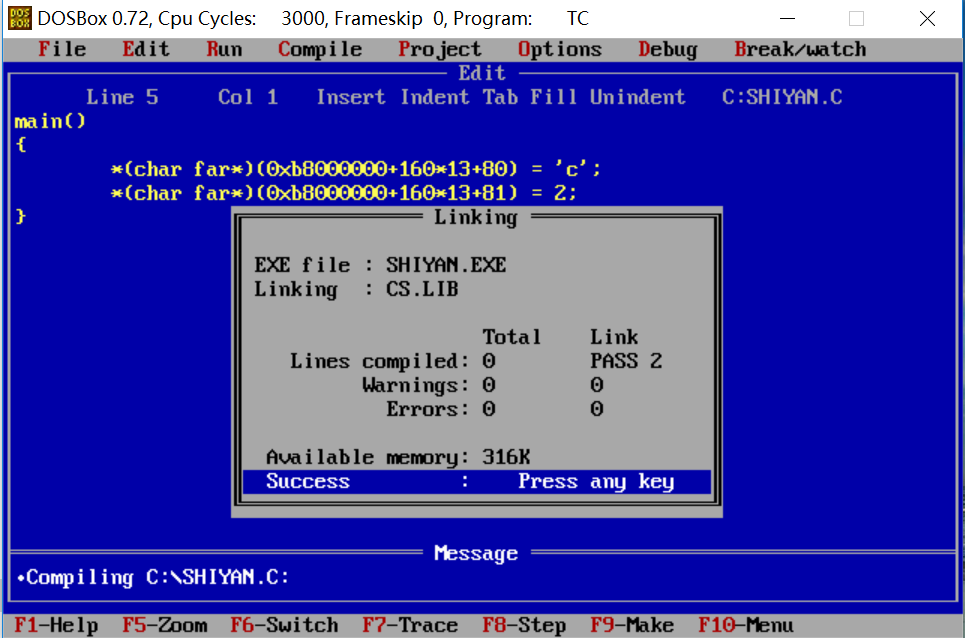
发现可以正常运行。

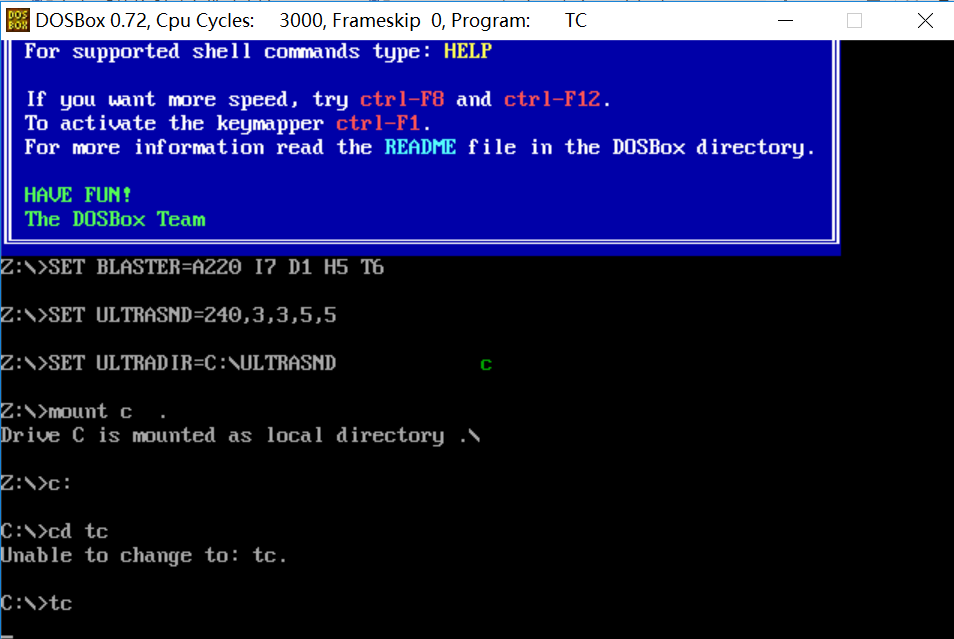
但是再编写一个直接操作显存用于屏幕显示的程序：



可以编译通过，但是出现连接错误，错误原因正是缺少了graphics.lib文件。

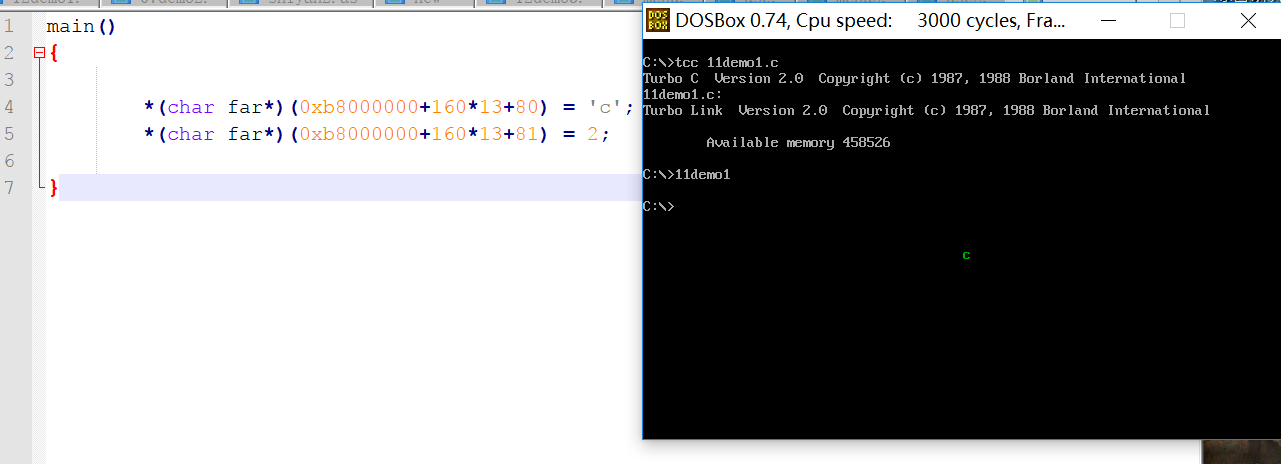
将这个文件重新放到C:\minic目录下,再看一下：





链接成功，显示正常。

再到C:\c目录下，重复上述操作，比较一下二者的异同：

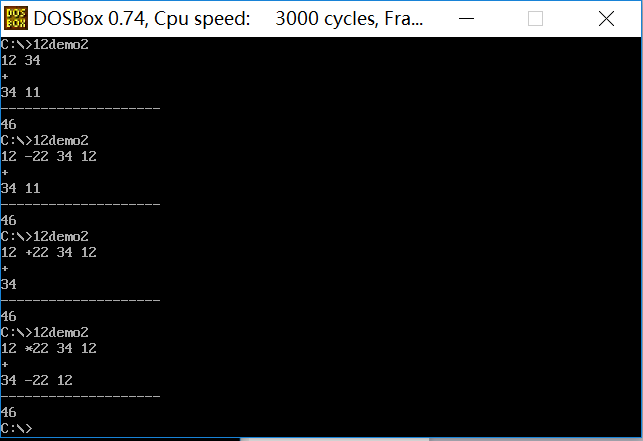


结果tcc下尽管移除了graphics.lib文件仍然可以正常打印结果。

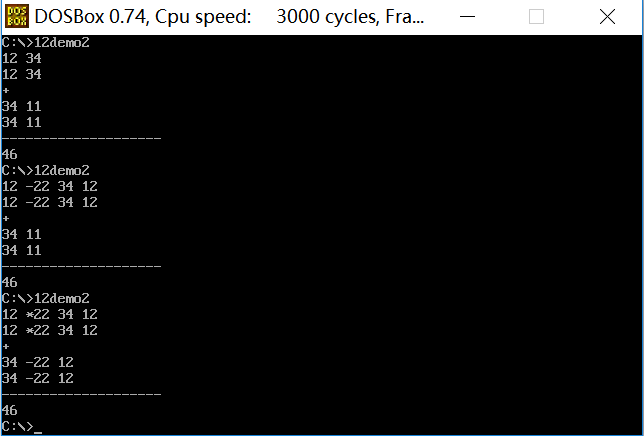
这又是为什么呢？

6、（1）为什么获取字符输入用gets()不用scanf()？

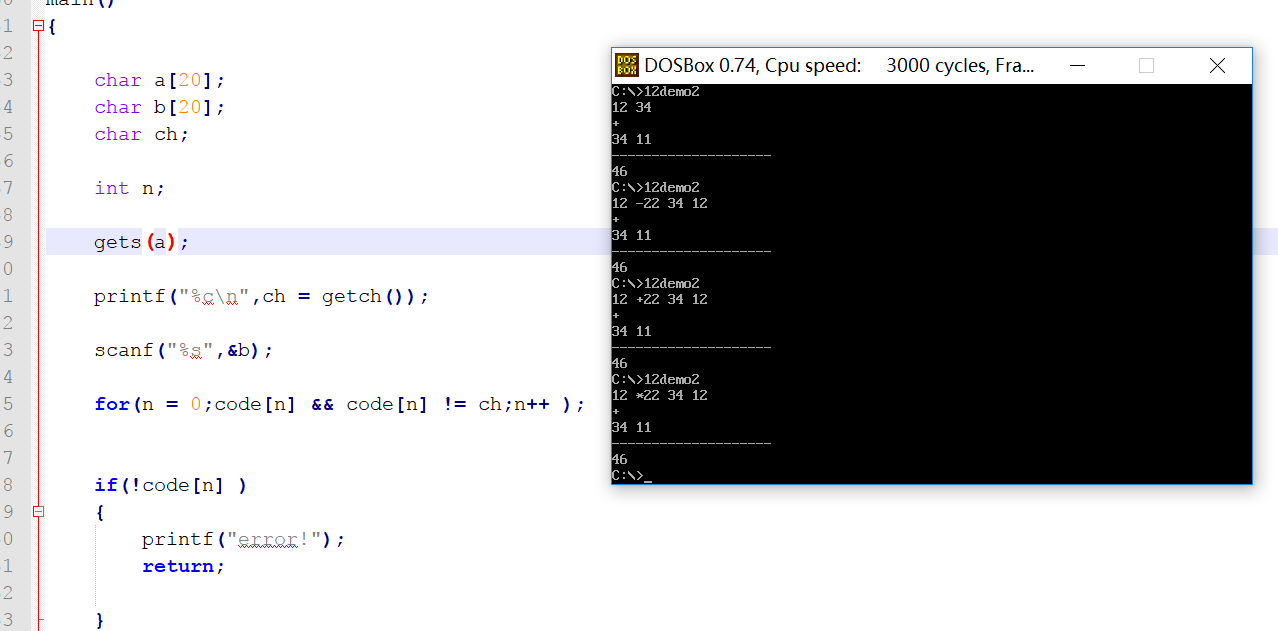
两个都为gets():



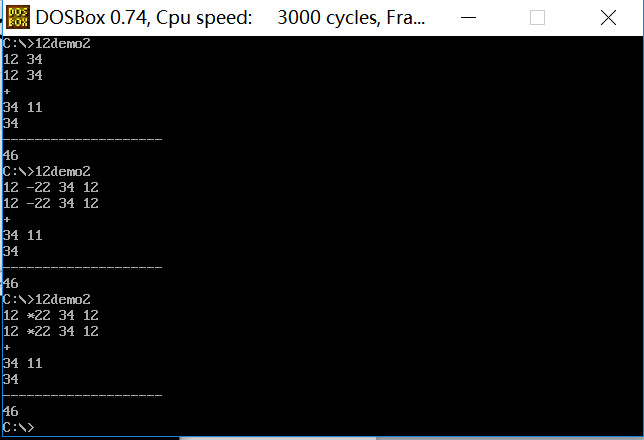
/////////////////////////////////////////////////////////////////



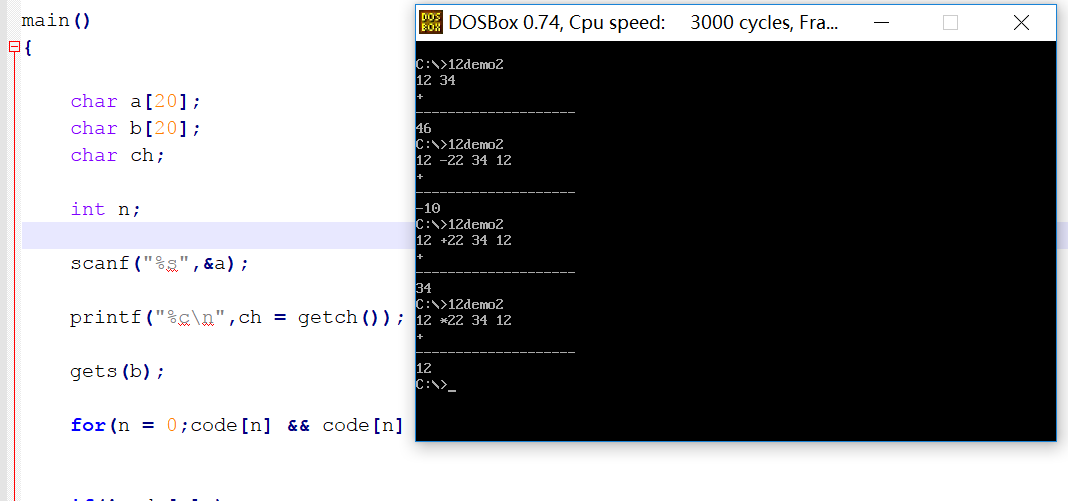
前一个为gets()，后一个为scanf()：



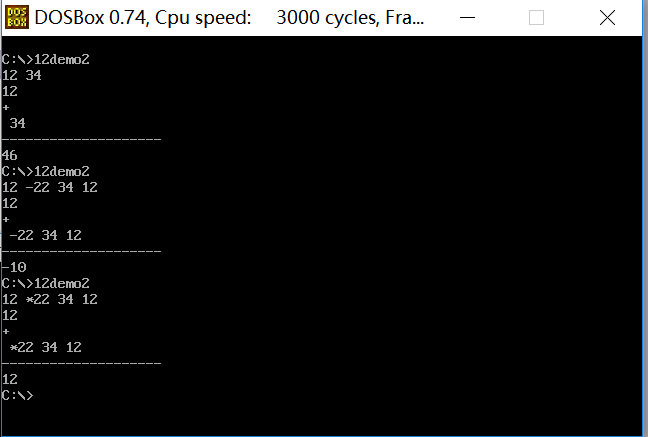
////////////////////////////////////////////////////////////////////////////////////////////////



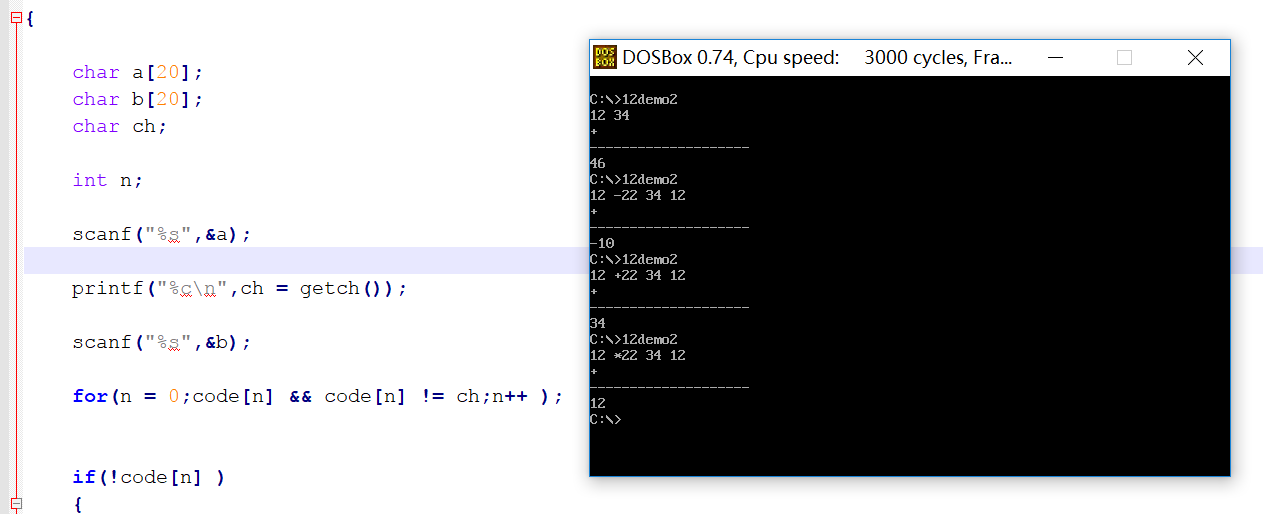
前一个为scanf()，后一个为gets()：



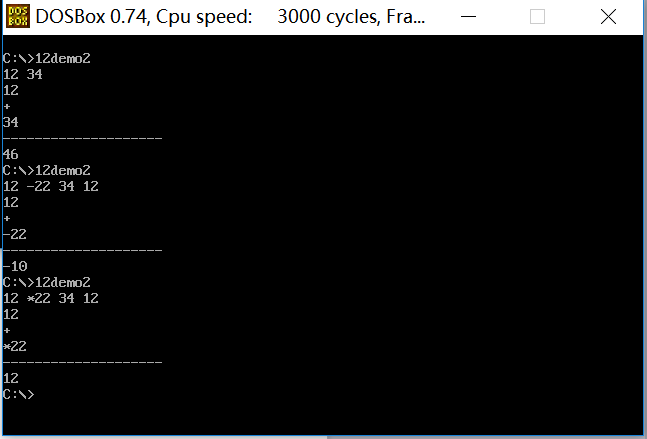
///////////////////////////////////////////////////////////////////////



两个都为scanf()：



///////////////////////////////////////////////////////////////////////

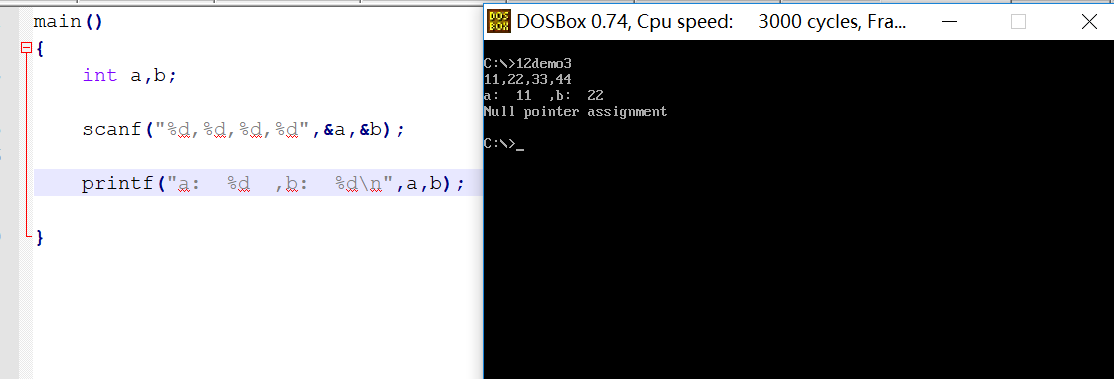


可以看到前两种情况和后两种情况分别有很高的相似性，在使用多个scanf（）时，如果输入缓冲区还有数据的话，那么scanf（）就不会询问用户输入，而是直接就将输入缓冲区的内容拿出来用了。

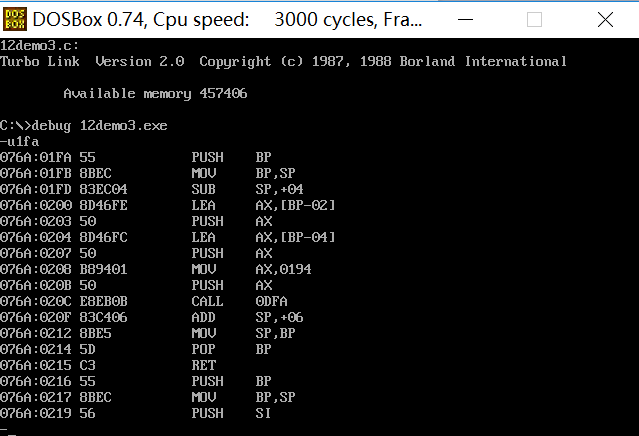
所以下面的第二问，很有可能就是33,44存放在缓冲区了。

（2）scanf(“%d,%d,%d,%d”,&a,&b)

测试：1 2 3 4 3和4存放到哪里了？



删去printf语句，debug反汇编看一下：



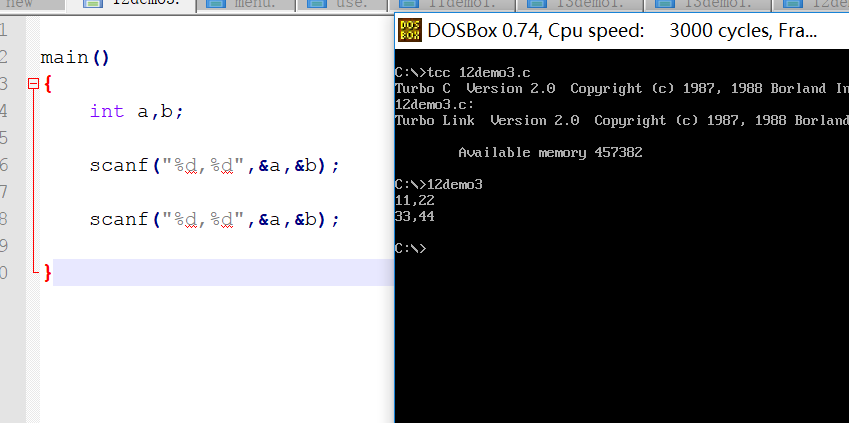
看不出所以然，但是猜测是存放在输入缓冲区了，一种是直接查找有没有可以查看输入缓冲区内容的函数，另一种就是用提出来，看看是不是33,44。

但是用gets（）(从流中提取一个字符串)和getchar(从stdin流中提取字符)，输出的都是0，这是为什么？是不是出发点偏了。

（3） scanf(“%d,%d”,&a,&b)

scanf(“%d,%d”,&a,&b)

分析这种奇怪现象



什么奇怪现象？

int far (\*p1)();

int (far\*p2)();

int far(far\* p3)();

区别：

程序：

int far (\*p1)();

int (far\*p2)();

int far(far\* p3)();

int\* p4;

main()

{

printf("%d %d %d %d\n",sizeof(p1),sizeof(p2),sizeof(p3),sizeof(p4));

}

现象：

