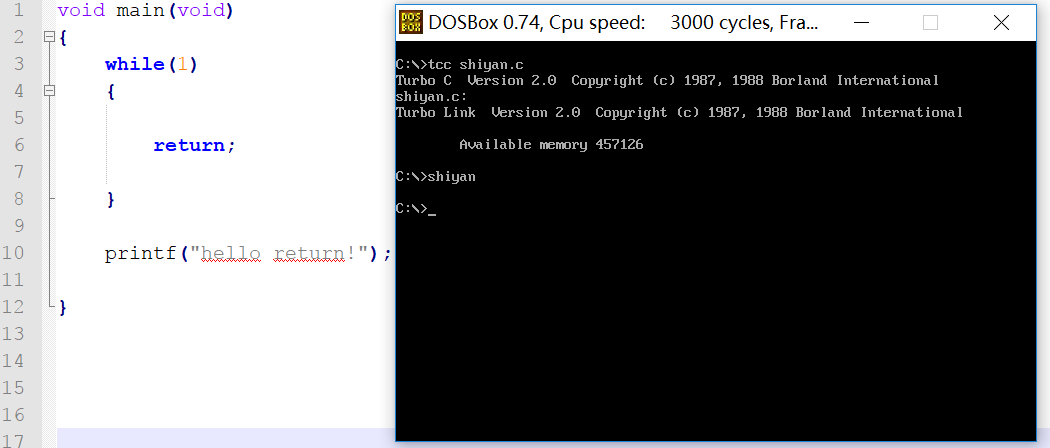
**c语言研究七、八研讨会报告**

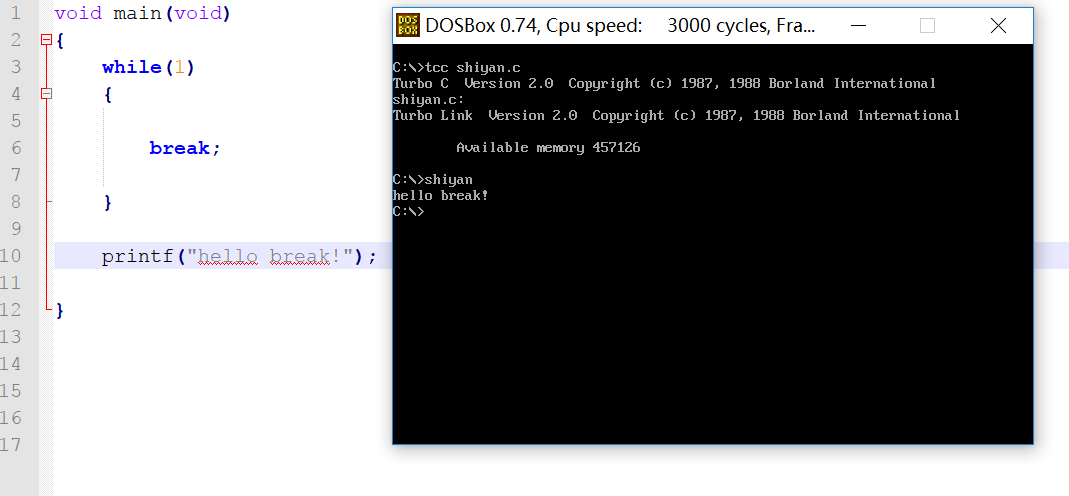
# 七

1. 循环中break与return退出循环的区别。

Return 直接程序结束：

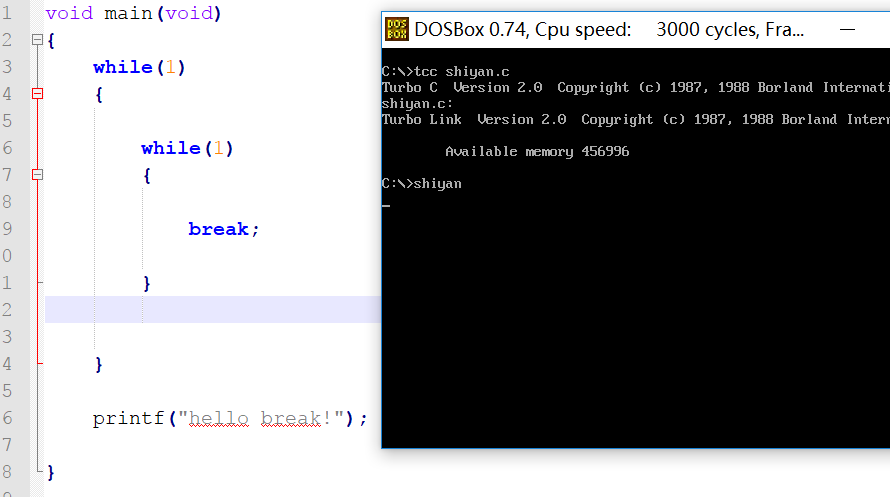


Break结束当前循环



加个双层循环，再比较一下：

Break 跳不出外层的while循环了：



这样看来，粗浅的个人理解break,continue,return ,只相较循环这来说：

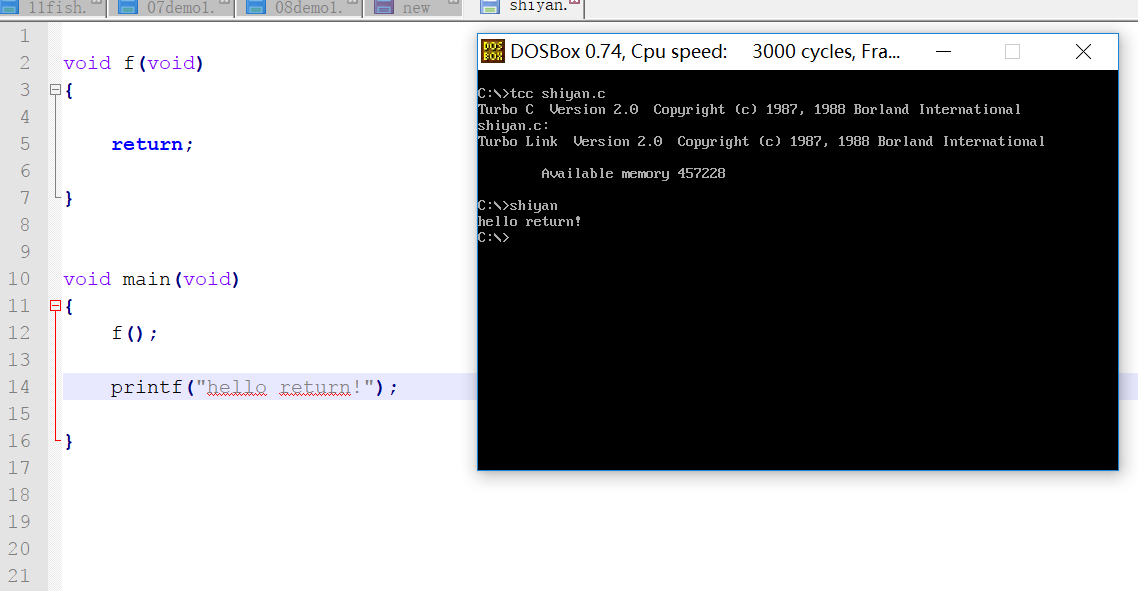
Continue<break<return.

Continue这个小伙子比较胆小，只是偶尔的时候胆子大那么一回脑门一热就翘了一门无聊的课，但是第二天还得低着脑袋灰溜溜地继续去上自己不喜欢的课，可能他怕点名。。。。。。Break这个人呢，胆子就比较大了，这门课不喜欢，那我这学期一次都不去了，我才不care你的什么学分、点名呢，什么纸老虎，我接下来该做啥我自己做去喽；但是这里面最厉害的还要看return 。中国有句古话，“光脚的不怕穿鞋的”，return就是这号人。他开心的时候可以return,他不开心还可以return。而且颇有哪吒对李靖说的“父亲，你给我的命，我还给你”的架势，可能留给函数世界里的众人一脸茫然，随后人群中炸起了“return真不负责任”、“return你怎么走的这么突然啊”、“return你让大家白发人送黑发人”，但是return呢，已经自认无愧于心地return到另一个函数世界了，额，转世了。

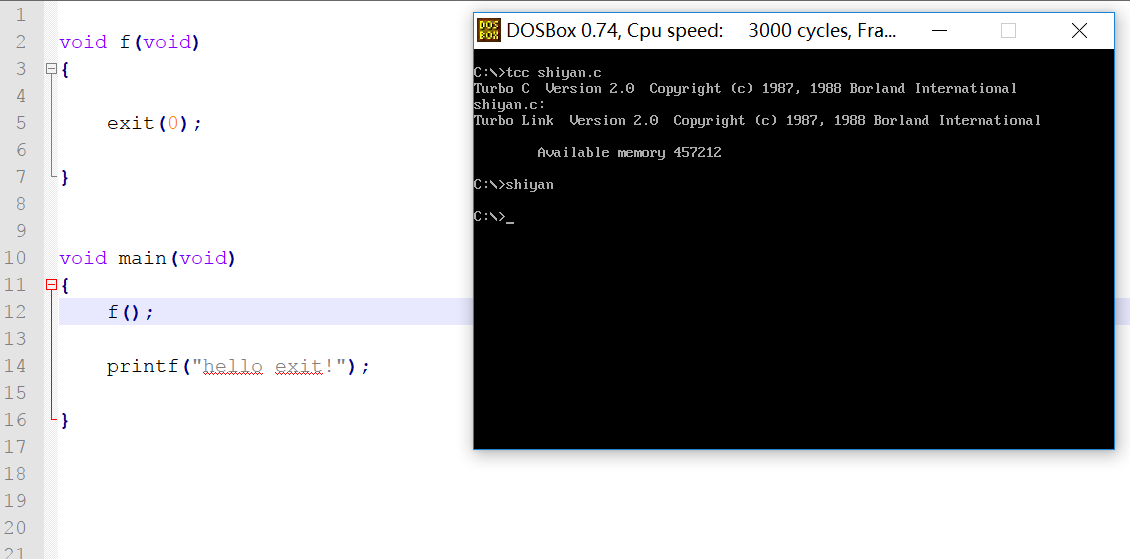
2. exit()问题。

比较一下ruturn和exit（）吧：

在一个f（）中使用return：



在一个f（）中使用exit：



哦呦，本来以为return这个“转世神童”就够厉害了，没想到啊没想到，exit（0）竟然拥有毁灭世界的力量，行吧行吧，厉害死你们了。

粗浅的理解：

Continue<break<return<exit(0).

至于exit（）的参数，用到再说吧。

3. 键盘缓冲区的处理问题。

4. 延时及时响应问题（延时不要用for循环）。

思路：重写中断，使中断可以及时响应；

重写延时函数，使系统随时检测条件（不精确）；

多进程处理（暂时无法实现）。

在51单片机中，一般利用其定时器设置初值，设置模式，数据溢出检测标志位，触发定时器中断处理函数，在里面可以依据51单片机的12MHz晶振或者11.0592MHz晶振进行精确地延时；在stm32单片机中，用于延时的是systick滴答定时器，通过设置模式，初值，也可以达到精确延时。所有精确延时并不难，一种可以像这样操作特殊功能寄存器，根据频率和所要延时的时间进行设置；另一种如果都可以精确测出软件编写的delay（）到底延时了多少，对其进行修正，也可以达到一般情况下的延时目的。

5. 内联汇编的好处与应用。

6. extern与static。

对立。

7. 不通过内联汇编，只使用tcc本身的条件，在c语言中使用汇编代码（参考汇编综合研究四：c0s中调用第一个函数，限制是只能是该函数写在文件开始的位置，要调用其他文件的函数，要用到extrn声明外部符号，实现调用任意位置的函数）。

# 八

1. typedef sturct与struct定义结构体的区别。

2. stu a,b; a=b;含义？使用以下程序赋值：

while(i < 375){

a[i] = \*p;

P++;

i++;

}

3. 如何正确使用头结点与头指针？

int i;

int sum;

head = (struct stu far\*)malloc(sizeof(struct stu));

p = head;

i = 0;

while(i < 375)

{

sum = a[i].c+a[i].os+a[i].java+a[i].masm;

if( (sum > 200)&& (sum < 400) )

{

p->next = &a[i];

p = p->next;

}

i++;

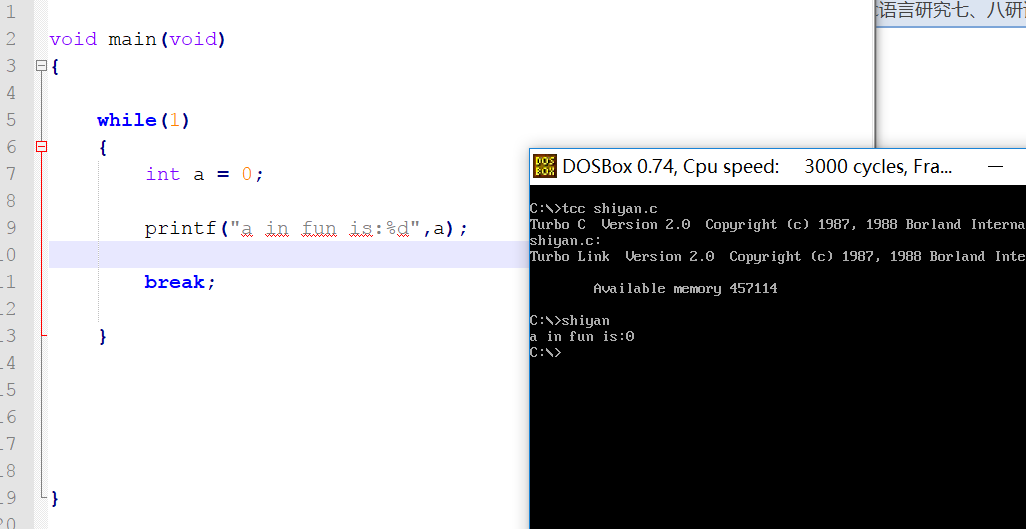
}

p->next = 0;

p = head;

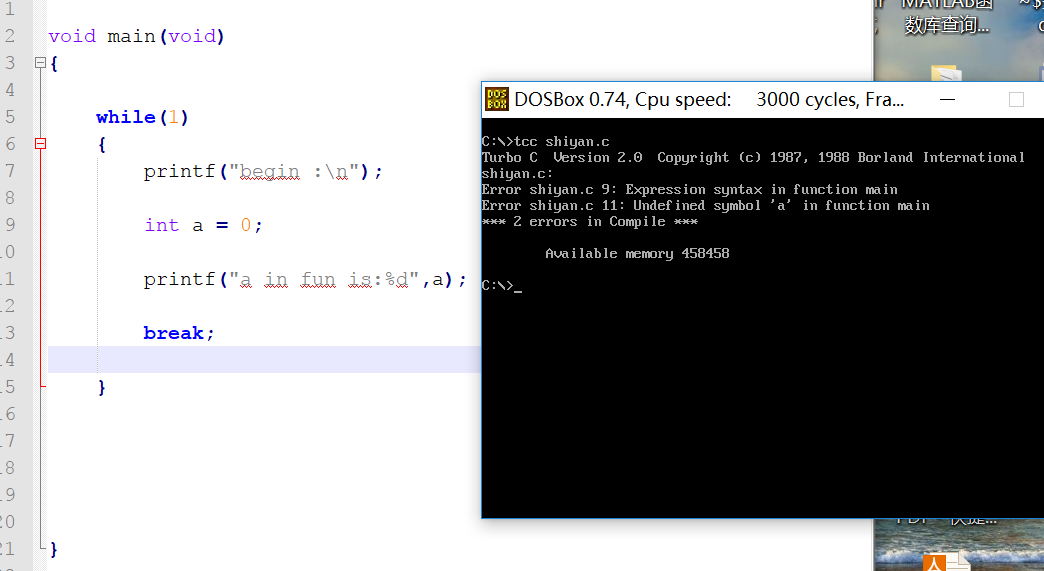
4. 如何理解：复合语句中可以重新定义变量?以及该变量作用域是什么？

在复合语句之内定义并调用:



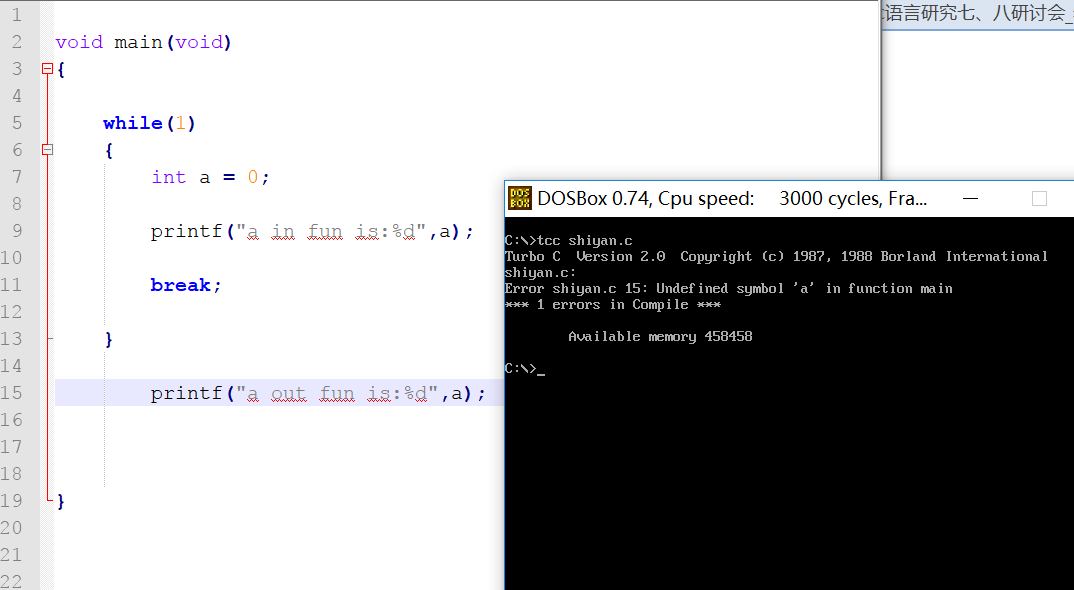
结果正常调用

在“int a = 0”之前加入一条printf（）函数：



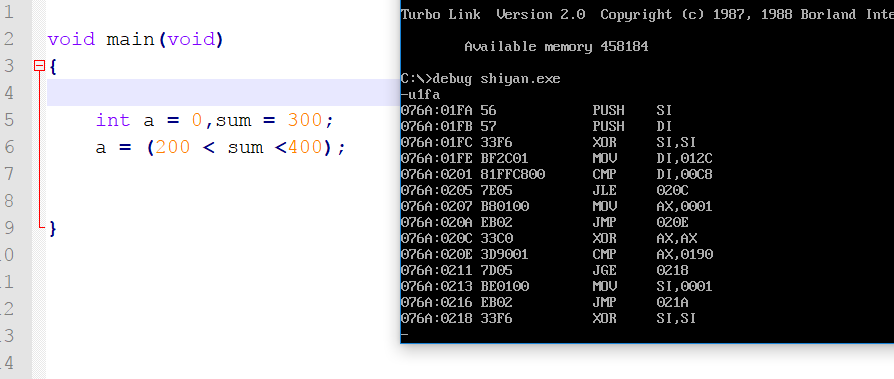
结果编译报错，和平常的现象一致。

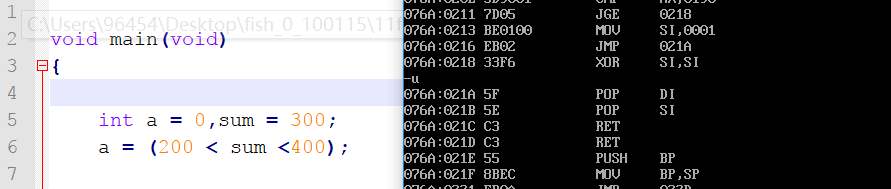
在与这条复合语句并行关系的printf（）函数内调用之前定义的变量 a：



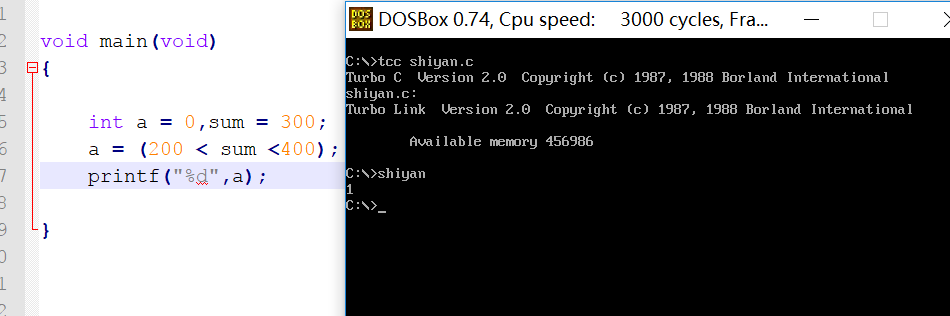
结果编译报错，作用域看来 局限于复合语句之内。

5. if (200 <sum <400)处理过程。



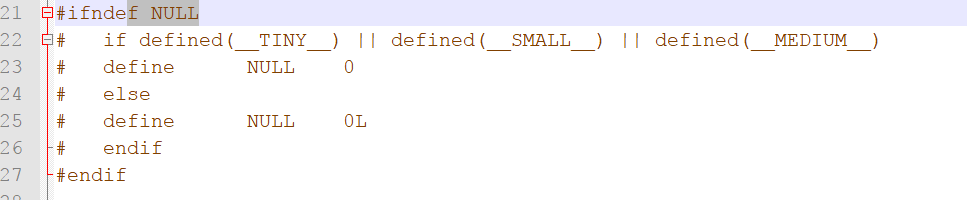


其中，先拿变量sum的值与200进行比较，如果sum>200,顺序执行，否则跳转到076a:020ch。这里初值300，布尔值是true,整形值是1，顺序执行。有一句mov ax,0001h，然后拿ax的1与400进行比较，如果ax<400，顺序执行，这里顺序执行。有一句mov si,0001h，然后跳到076a:021ah。最后变量a的值由si取得，最后结果为1。Printf()验证一下。



6. NULL怎么理解？与0区别是什么？

查看tc中的stdio.h：



可以看到，tc中NULL 与0没区别。

但是，网上的一些忠告也是不容忽视的：

在不同的系统中，

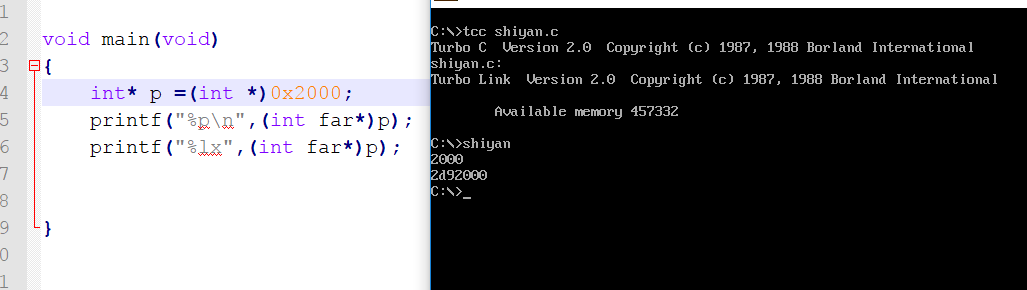
NULL并非总是和0等同，NULL仅仅代表空值，也就是指向一个不被使用的地址，在大多数系统中，都将0作为不被使用的地址，所以就有了类似这样的定义

#define NULL 0

但并非总是如此，也有些系统不将0地址作为NULL，而是用其他的地址，所以说，千万别将NULL和0等价起来，特别是在一些跨平台的代码中，这更是将给你带来灾难。

7. 打印时%p和%lx的区别？

%p只打印出偏移地址，即使加了（far\*）强转：



8. 链表的效率究竟高在哪里？比如，删除一个数据时：

顺序表：直接定位到所删数据，然后整个数据移动。

链表：先进行查找，找到所删数据，改变相邻数据项。

仅考虑这种情况下的效率问题。

这是一个数学题呀。