1. **宏定义的使用。**

指定对齐值: 预编译命令#pragma pack (value)指定的对齐值value。

宏定义 无参 可以代表数据 和一段程序 常量

有参 参数需要括起来。可以代表变量和具有变量的一段程序。

注意：#define f(x) （x\*x）

f(2+2) 就为 2+2\*2+2=8 而不是4\*4=16

1. **拷贝函数memcpy()函数的使用。**

void \*memcpy(void \*destin,void \*source,unsigned n)

{

char \*xdes= destin;//将目的首地址给指针变量xdes

char \*xsrc= source; //将源的首地址给指针变量xsrc

while(n--)\*xdes++=\*xsrc++; // 将n个数据copy过去

// 先\*xdes=\*xsrc 然后xdes++ xsrc++

}

如果想实现段间的copy 可以这样写

void \*memcpy\_1(void far \*destin,void far\*source,unsigned n)

{

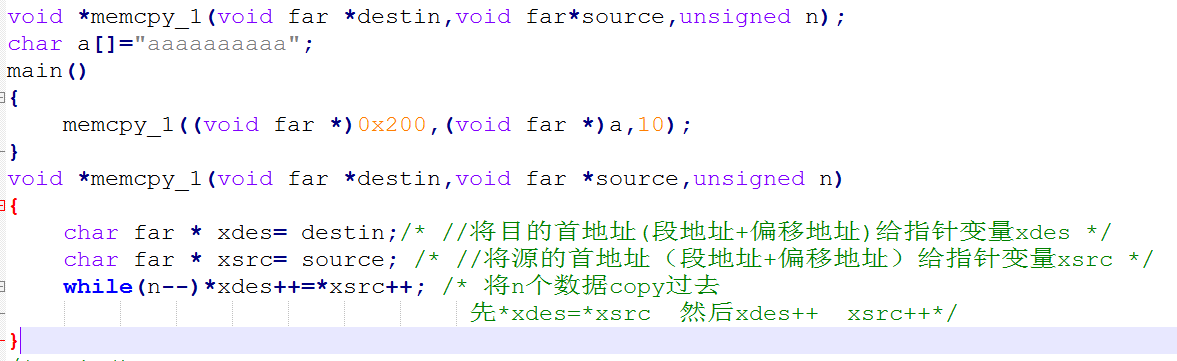
char far\*xdes= destin;//将目的首地址(段地址+偏移地址)给指针变量xdes

char far\*xsrc= source; //将源的首地址（段地址+偏移地址）给指针变量xsrc

while(n--)\*xdes++=\*xsrc++; // 将n个数据copy过去

// 先\*xdes=\*xsrc 然后xdes++ xsrc++

}



结果显示：



注意：

char \*p=”hello world”

指针变量指向字符串常量 字符串放在全局空间处

这个时候全局指针变量与局部指针变量都是指向这个字符串的内存空间

1. **结构体类型定义的多个变量作为全局变量与局部变量分别在16位与32位中如何存储。**

答：

16位是指数据线为16位，一次性操作可以取到两个字节，这样的话用两个字节对齐的方式会提高内存读取速度，比如char int 放在一块 如果不对齐，我们看看16位单片机怎么读取，首先读到了char型和int型的低字节 然后再读一次我们读到了int型的高字节和一个未知的字节数据 最后还需要再把int型的低字节和高字节拼到一块组成本来的int型数据，从上看出 读出需要三步。

如果对齐，我们再看看16位单片机怎么读取，对齐： char型其实分配两个空间，int型两个空间，那么读取会读到char型数据和一个字节的未知数据，再读就读到了完整的int型数据。这样我们的读出只有两步。

所以对齐方式提高内存读取效率。

32位那么就是按四个字节对齐喽。

但是我们的8086就没有采用对齐方式。

## 学长补充：

1. **数据结构的中的链表有什么特点？结构体指针在链表这种数据结构的实现中起到了什么样的作用？**

答：

链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域。

链表结构可以充分利用计算机内存空间，实现灵活的内存动态管理，当我们使用malloc函数申请的内存空间有的时候是不连续的，我们可以通过链表实现逻辑上的连续。同时链表由于增加了结点的指针域，空间开销比较大。

线性表的链式存储表示，有一个缺点就是要找一个数，必须要从头开始找起，十分麻烦。

结构体指针的链表这种数据结构的实现的用处，比如对于（每个人的各科成绩和排名）在内存中分配如果不连续，那我们就可以定义结构体链表，数据域存放一个人的各科成绩和排名，指针域存储下个人的数据域的地址。这样就可以把每个人的各科成绩和排名连在了一块。

1. **结构体成员变量中指向结构体类型的指针有哪些用途？结构体成员变量中指向结构体类型的指针和普通指针的区别是什么？**

答：结构体成员变量中指向结构体类型的指针用来得到结构体链表

普通指针 可以指向int，char,long 等数据类型的内存空间，也可以指向一个数组或者是字符串。

结构体类型指针 指向结构体的内存空间。

**学长补充：**

1. **对于参数进行系统的学习。**

函数分有参和无参

无参:为void

有参:分为不定参数和确定参数

不定参数： fun( ) 里面可以放参数，放的个数也不限制，也可以不放

fun(char\*,…) …代表不定参数，可以放类型不定，个数不定的参数

不定参数的压栈方式是根据什么？

得出结论不定参数是根据实参的类型进行压栈的。如果实参是常量没有类型且小于int型最大则会按int型存放进去。

确定参数：

1. **对字符串指针进行研究。**

答：对于字符串指针，我们发现字符串在编译时候就为其分配空间，而我们的字符串指针变量其实是的到字符串其空间的首地址。对于数组比如 a[]={1,2,3}或者a[]=”sdahsd”这里的1，2，3还有sdahsd字符都是编译时就指定好内存的。

1. **a[4]={‘a’,’a’,’a’,’a’};printf(“%s\n”,a);会显示什么？为什么？**

答：会显示四个a 还有一些乱码，因为%s是以0为结尾，这个数组没有以0结尾，所以把后面几个空间的数据也以字符的形式打印出来知道遇到0就会结束。

1. **用char \*\*型指针实现综合研究9的showstr函数。**

用二维数组的思想去理解即可。

void showstr(char \*buffer,…)

{

char \*\*p;

p=&buffer;

printf(“%s”，\*p);

}

buffer放字符串的首地址 则p应该放buffer的地址

1. **用多种方法实现实验9。**

**（三个函数stdarg.h中 void va\_start（va\_list ap,last）**

**type va\_arg（va\_list ap,type）**

**void va\_end(va\_list ap)）**

答：

typedef void \*va\_list; /\* 自己定义一个void型的指针类型 \*/

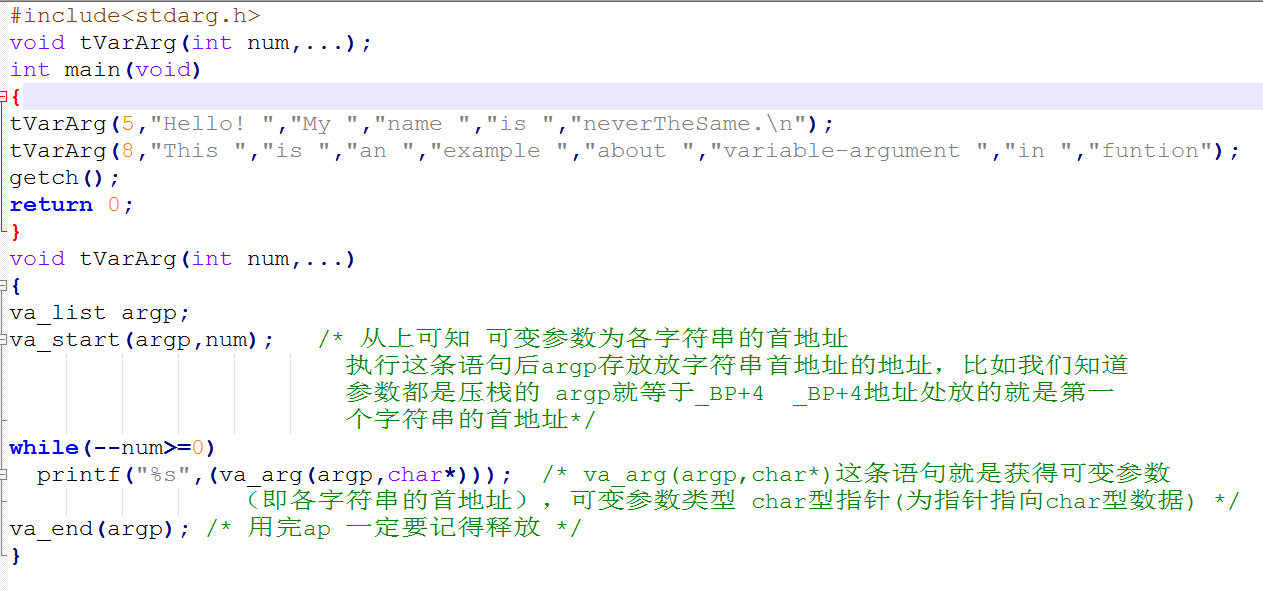
#define va\_start(ap, parmN) (ap = ...)

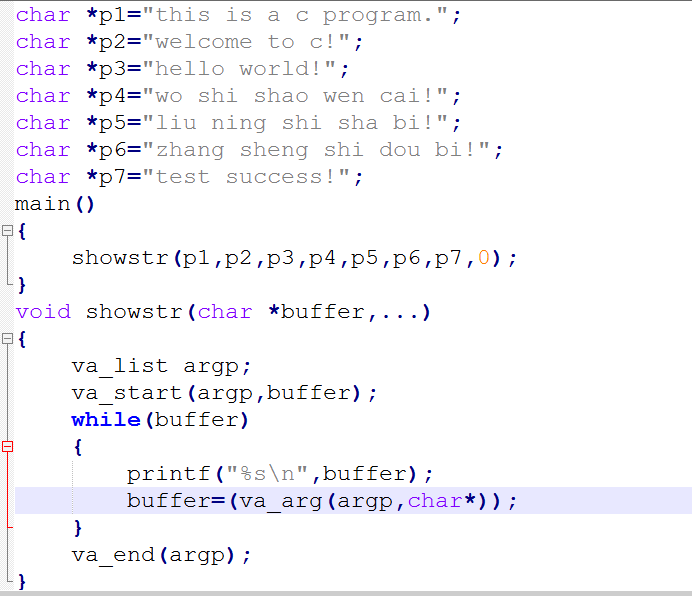
/\* 得ap指向可变参数的第一个参数.ap的类型为va\_list。parmN一般为字符串的首地址，三个点代表可变参数 \*/

#define va\_arg(ap, type) (\*((type \*)(ap))++)

/\* ap存放可变参数地址，type为需要获得的参数的类型,然后\*ap获得当前ap所指向的参数 之后ap++使ap指向可变参数的下一个参数 \*/

#define va\_end(ap) /\* 释放指针ap 使ap=0 \*/

****

****

1. **实现向函数传递不定参数的广泛用处。**
2. 当你要传递的参数类型不确定的时候就可以采用传递不定参数。
3. 当你传递的参数个数不确定时也可以采用传递不定参数

/\* scanf函数要注意格式匹配问题:

对于%c的输入，%c%c%c 输入必须是 abc 不能是 a b c也不能是 a,b,c （这个特殊）

%c,%c,%c 输入必须是 a,b,c 不能是 a b c也不能是 abc

%c %c %c 输入 a b c和abc都行， 不能是 a,b,c 空格，回车，tab 效果一样

对于%d的输入， %d%d%d 输入必须是1 2 3 不能是 123也不能是 1,2,3

%d,%d,%d 输入必须是1,2,3 不能是 123也不能是 1 2 3

%d %d %d 输入必须是1 2 3 不能是 123也不能是 1,2,3

对于%s的输入， %s%s 输入必须是 shao wen 空格，回车，tab 效果一样

%s %s 输入必须是 shao wen

%s,%s 输入必须是 shao wen

\*/

/\* scanf getchar getc(stdin) gets

printf putchar putc(stdout) puts 从键盘缓冲区读取后经过数据流缓冲区

scanf %s只要是空格 TAB 回车 都会觉得字符串结束 %c会接受回车

gets 只能是回车才会觉得字符串结束

getch 从键盘缓冲区直接读后赋值给al 然后传给变量

getche 从键盘缓冲区直接读 一边显示，一边赋值给al然后传给变量 \*/

/\*long \*(\*a)[3][4];\*/

/\*很重要

文件：怎么样生成一个文件

往文件里写数据 \*/

/\* 查资料知道 在路径中'\'可能是转译字符

所以路径在程序中可能是 D://MASM5.0//c//read.txt

也可能是 D:\\MASM5.0\\c\\read.txt

fopen("路径下的文件名","文件使用方式")

"r" 会从文件开始把全部读出来 只能读。

"w"和"a"都是用来写的，文件不存在都会自动建立文件。"w"会把文件删空重新写,"a"会在文件末尾开始继续写，保留原有的内容。

r+ w+ a+ 都是可读可写 r+必须保证文件已存在

w+会重新建立文件

a+会在已有文件进行读写。

\*/

/\* 在这一章我真心遇到好大的问题 \*/

/\*

其实tcc 把所有的.obj文件连接起来，就可以随意调用各函数，但是同一文件下的各函数相会调用

必须明确其正确的作用域，不然编译器会自己创建一个隐函数，返回值为int型，即声明放在前面。把其他函数放在main函数前 在main函数调用就不会

有任何警告。

1)h文件作用:

1.方便开发:包含一些文件需要的共同的常量,结构,类型定义,函数,变量申明；

　　2.使函数的作用域从函数声明的位置开始，而不是函数定义的位置(实践总结)

　　3.提供接口:对一个软件包来说可以提供一个给外界的接口(例如: stdio.h)。

2)h文件里应该有什么:常量,结构,类型定义,函数,变量申明。

3)h文件不应该有什么:变量定义, 函数定义。

4)extern问题:

　1.对于变量需要extern；

　2.对于函数不需要因为函数的缺省状态是extern的.如果一个函数要改变为只在文件内可见,加static \*/