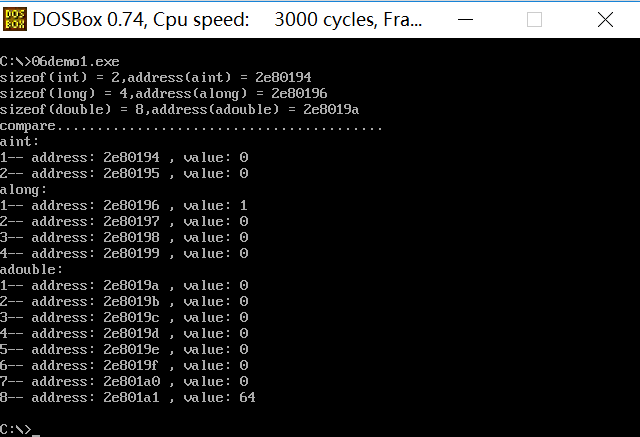
**c语言研究报告六**

1. **研究过程展示**

**1>.程序1部分：**

写一个c程序，打印int、long、double型变量所占的字节数、地址、各个字节的地址和内容。打印地址和内容比较好办，打印地址可以用取址符&，打印内容直接输出就行了，那么怎么打印所占的字节数呢？我们打印一个东西是输出它的值，而变量所占的字节数没有变量或指针来存储，我们可以得到变量所占字节数的方法是用运算符sizeof。sizeof运算符以字节为单位返回其操作数的大小，操作数可以是一个具体的数据对象，如一个变量名，也可以是一个类型，如float，如果它是一个类型，操作数必须被括在圆括号里，如sizeof n和sizeof（float）。sizeof操作符不能用于函数类型，不完全类型或位字段。不完全类型指具有未知存储大小的数据类型，如未知存储大小的数组类型、未知内容的结构或联合类型、void类型等。sizeof操作符的结果类型是size\_t，它在头文件中typedef为unsigned　int类型，size\_t是后者用typedef机制创建的别名。当操作数是指针时，sizeof依赖于编译器。例如Microsoft　C/C++7.0中，near类指针字节数为2，far、huge类指针字节数为4。一般Unix的指针字节数为4。当操作数具有数组类型时，其结果是数组的总字节数。联合类型操作数的sizeof是其最大字节成员的字节数。结构类型操作数的sizeof是这种类型对象的总字节数，包括任何垫补（就是为了内存对齐而补充的内存单元）在内。如果操作数是函数中的数组形参或函数类型的形参，sizeof给出其指针的大小。



06demo1.c整体程序：

**int aint = 0;**

**long along = 1;**

**double adouble = 2.0;**

**int i = 0;**

**char\* pchar1 = (char\*)(&aint);**

**char\* pchar2 = (char\*)(&along);;**

**char\* pchar3 = (char\*)(&adouble);;**

**int main()**

**{**

**printf("sizeof(int) = %d,address(aint) = %lx\n",sizeof(int),(char far\*)(&aint));**

**printf("sizeof(long) = %d,address(along) = %lx\n",sizeof(long),(char far\*)(&along));**

**printf("sizeof(double) = %d,address(adouble) = %lx\n",sizeof(double),(char far\*)(&adouble));**

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

**printf("aint: \n");**

**for(i = 0; i < sizeof(int) ; i++)**

**{**

**printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar1 + i),\*(pchar1 + i));**

**}**

**printf("along: \n");**

**for(i = 0; i < sizeof(long) ; i++)**

**{**

**printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar2 + i),\*(pchar2 + i));**

**}**

**printf("adouble: \n");**

**for(i = 0; i < sizeof(double) ; i++)**

**{**

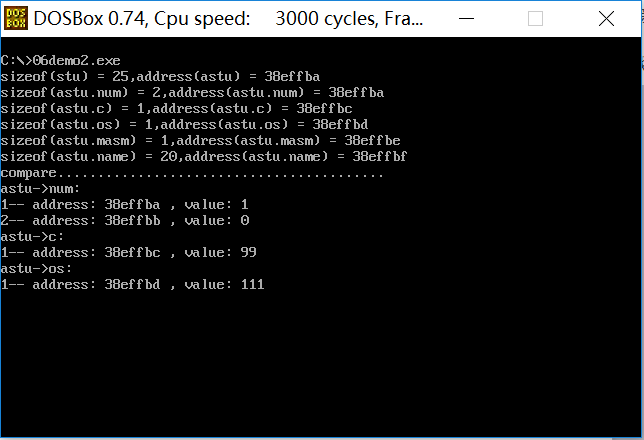
**printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar3 + i),\*(pchar3 + i));**

**}**

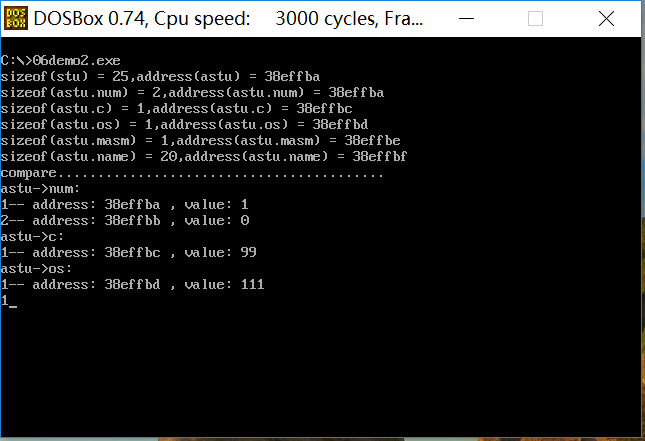
**return 0;**

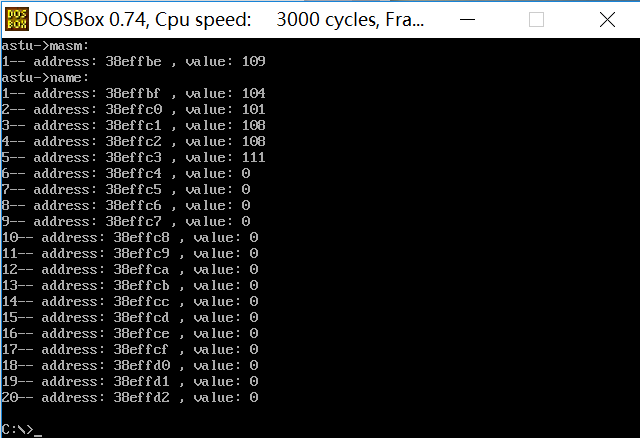
**}**

**2>.程序2部分：**



由于显示效果过长，所以在程序中加了个“scanf("pause",&a);”，随便填入数字“1”，并加回车。





06demo2.c整体程序：

typedef struct

{

int num;

unsigned char c ;

unsigned char os;

unsigned char masm;

char name[20];

}stu;

int main()

{

stu astu = {1,'c','o','m',"hello"};

int i = 0;

int a = 0;

stu\* pstu = &astu;

char\* pchar1 = (char\*)(&(astu.num));

char\* pchar2 = (char\*)(&(astu.c));

char\* pchar3 = (char\*)(&(astu.os));

char\* pchar4 = (char\*)(&(astu.masm));

char\* pchar5 = (char\*)(&(astu.name));

printf("sizeof(stu) = %d,address(astu) = %lx\n",sizeof(stu),(char far\*)(&astu));

printf("sizeof(astu.num) = %d,address(astu.num) = %lx\n",sizeof(astu.num),(char far\*)(&(astu.num)));

printf("sizeof(astu.c) = %d,address(astu.c) = %lx\n",sizeof(astu.c),(char far\*)(&(astu.c)));

printf("sizeof(astu.os) = %d,address(astu.os) = %lx\n",sizeof(astu.os),(char far\*)(&(astu.os)));

printf("sizeof(astu.masm) = %d,address(astu.masm) = %lx\n",sizeof(astu.masm),(char far\*)(&(astu.masm)));

printf("sizeof(astu.name) = %d,address(astu.name) = %lx\n",sizeof(astu.name),(char far\*)(&(astu.name)));

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

printf("astu->num: \n");

for(i = 0; i < sizeof(astu.num) ; i++)

{

printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar1 + i),\*(pchar1 + i));

}

printf("astu->c: \n");

for(i = 0; i < sizeof(astu.c) ; i++)

{

printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar2 + i),\*(pchar2 + i));

}

printf("astu->os: \n");

for(i = 0; i < sizeof(astu.os) ; i++)

{

printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar3 + i),\*(pchar3 + i));

}

scanf("pause",&a);

printf("astu->masm: \n");

for(i = 0; i < sizeof(astu.masm) ; i++)

{

printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar4 + i),\*(pchar4 + i));

}

printf("astu->name: \n");

for(i = 0; i < sizeof(astu.name) ; i++)

{

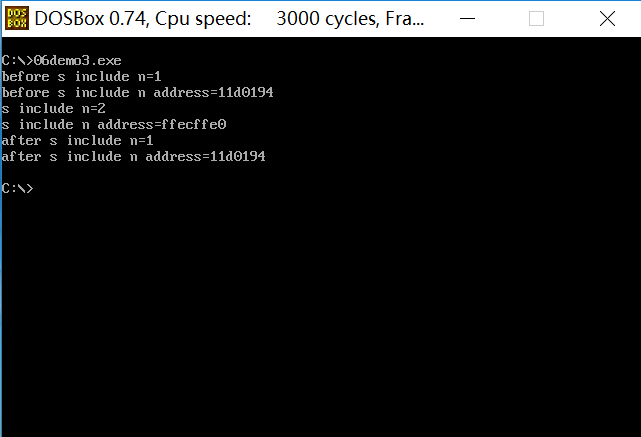
printf("%d-- address: %lx , value: %d\n",(i + 1),(char far\*)(pchar5 + i),\*(pchar5 + i));

}

return 0;

}

**3>.程序3部分：**



06demo3.c整体程序：

int n = 1;

void s(int n)

{

n++;

printf("s include n=%d\n",n);

printf("s include n address=%lx\n",&n);

}

int main()

{

printf("before s include n=%d\n",n);

printf("before s include n address=%lx\n",&n);

s(n);

printf("after s include n=%d\n",n);

printf("after s include n address=%lx\n",&n);

return 0;

}

//如果同一个源文件中，外部变量与局部变量同名，则在局部变量的作用范围内，外部变量被“屏蔽”，即它不起作用

1. **已思考研究并已解决问题汇总**

利用sizeof关键字，进而实现了从c语言对内存中的相应位置的相应数据的查看，并得知了自己想要探知的变量的大小。

自己尝试一下sizeof的未知使用：

1>.sizeof(-):

c程序：

int main()

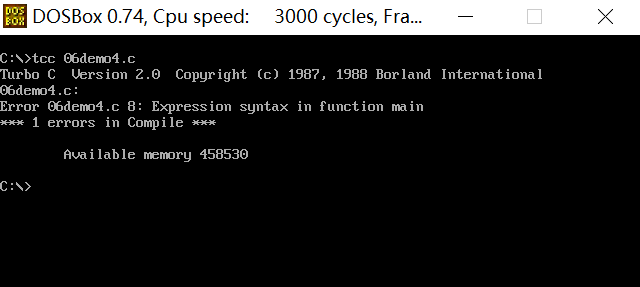
{

printf("sizeof(-)\n",sizeof(-));

return 0;

}

结果：



出现编译错误。

2>.sizeof( a + b)

c程序：

int main()

{

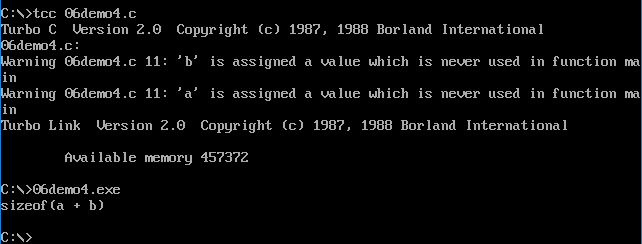
int a = 0,b = 1;

printf("sizeof(a + b)\n",sizeof(a + b));

return 0;

}

结果：



出现了警告，而且最终的返回结果也不符预期。

3>.sizeof(4)

c程序：

int main()

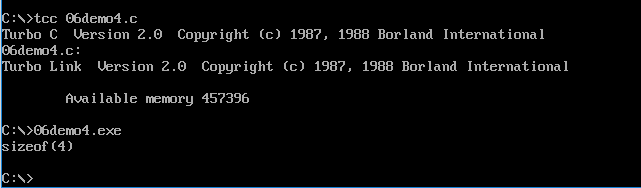
{

printf("sizeof(4)\n",sizeof(4));

return 0;

}

结果：



编译无错误，无警告，但是最终结果不符预期。

4>.sizeof("123")

c程序：

int main()

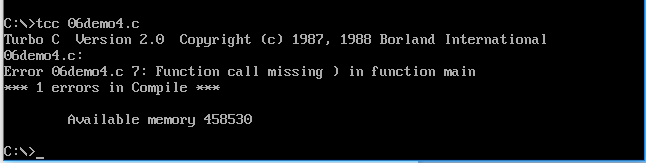
{

printf("sizeof("123")\n",sizeof("123"));

return 0;

}

结果：



出现编译错误。

1. **已思考研究并未解决问题汇总**

是否可以自己编写一个功能与sizeof关键字相似的函数呢？如果打算编写，自己会从什么角度出发？

Sizeof是求字节数，字节也就是内存中一种绝对的单元长度，如果打个比方，我觉得他就像尺子中的厘米单位，当然比他更小的还有毫米，对应的，位可以填充。同样sizeof好像就是一把丈量的尺子一样，他是一把短小精悍的工具，厘米——字节、毫米——位等都存在于内存中，但并不意味着内存更贴近尺子，当然是与sizeof在此处相较而言。内存是个容器，是字节们、位们的容器，是他们的世界，而我们的世界，又是我们的容器。我如果想测量一段字节多长，我会怎么测呢？我想了想我在现实生活中的使用尺子的时候一般是这样做的：

第一种：我会将尺子的0刻度对齐欲测量字节段的首或尾端，然后查看对应的尾或首端读出的“尺子刻度”是多少厘米，也就是多少字节，是多少毫米，也就是多少位。。。。。。

第二种：有时候尺子的0刻度可能断掉了，被墨迹玷污了。。。。。。这时候我会随便找个刻度，当然现实生活中一般选取整厘米处，内存选取整字节处。然后记下这个刻度值，将尺子的这个刻度对齐欲测量字节段的首或尾端，然后查看对应的尾或首端读出的“尺子刻度”，然后两值相减取绝对值。也就是多少厘米，也就是多少字节，是多少毫米，也就是多少位。。。。。。

编写程序的话，会选上面两种。但是还有两个问题。

1>.sizeof是关键字，不是函数；

2>.sizeof能实现对结构体中出现内存对齐情况的正确计算。

1. **研究感想（心得体会）**