**c语言研究六研讨会报告**

1. **研究过程展示**
2. 打印char类型，要以无符号形式打印16进制。因为char类型是1个字节。打印%x是压入2个字节，这时要进行扩展，如果是以有符号形式打印的话，则扩展后为ff80，（假设打印80h），这样就不合适，因为前面的bit位都变成了1，而用无符号的话，扩展出来就成了0x0080，这样就很合适。

这样，也就涉及到快速数数的方面。我们对计算机存取数值的灵敏感知，其实没有必要非得看什么原码、反码、补码。想出这些只顾着理论性强的繁琐词汇，并想要学生们非得记住的，一定一定不是一个好主意，而很多学校老师们就很是这样，不仅喜欢教，还喜欢这样考，真的受不了。其实数一直是连续的，至少，是线性排列的。就好像最经典的一维坐标轴：

------------------------------------------------------------------------------------------------------>

-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6

对于无符号数，比如无符号整型，数值范围是（0~255）同样模仿一维坐标轴的形式列出来：

--------------------------------------------------------------------------------------------------> 增大

0 1 2 ...... 127 128 129 ...... 253 254 255

00h 01h 02h ...... 7fh 80h 81h ...... Fdh feh ffh 补码

其中从始至终，箭头方向，就是数值单调递增的方向。

对于有符号数，比如有符号整型，数值范围是（-128~127）同样模仿一维坐标轴的形式列出来：

------------------------------------------>|--------------------------------------------------------->增大

0 1 2 ...... 127 | -128 -127 ...... -3 -2 -1

00h 01h 02h ...... 7fh | 80h 81h ...... Fdh feh ffh 补码

其中从始至终，箭头方向，就是数值递增的方向，但是并不是单调递增，分为了两大部分：“0~127”和“-128~-1”。接着中间这个划分两部分，也就是划分“正负”的界限，就更熟悉了，此处的“127”和“-128”之间，也就是正好二分处。

接着考虑进来之前所说的数是线性排列的概念，思考学长提出的扩展成有符号四字节之后，“-128”又是怎样表示的，数值范围是（-32768~32767）同样模仿一维坐标轴：

------------------------------------------>|------------------------------------------------------>增大

0 1 ...... 32767 | -32768 -32767 ...... -2 -1

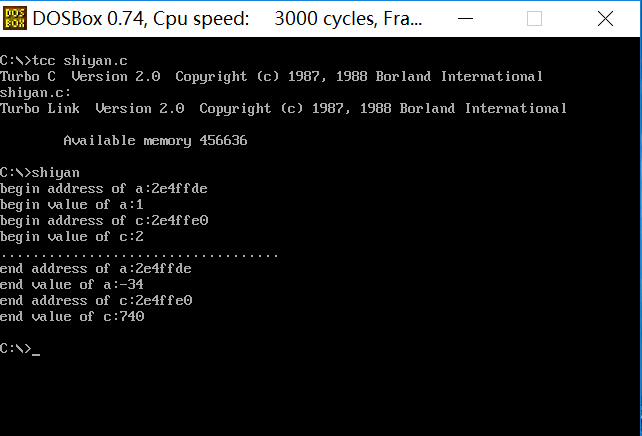
0000h 0001h ...... 7fffh | 8000h 80011h ...... fffeh ffffh 补码

其中，同样-32768 到 -1也是线性增大的，而在之前的有符号整型，-128距离 -1是80h 到ffh 的距离，这样推算下来，-128扩展之后的表示，就是ff80h。

而在这个过程中，补码是一直线性连续的，这样就比较直观，好记忆理解了。

1. 几种验证“参数的存储空间与局部变量的存储空间，在函数运行后收回”方法。

第一种：



程序：

int far \* pa;

int far \* pc;

int f1(int a,int b);

int f1(int a,int b)

{

int c;

c = a+b;

pa = (int far \*)(&a);

pc = (int far \*)(&b);

printf("begin address of a:%lx \n",pa);

printf("begin value of a:%d \n",\*pa);

printf("begin address of c:%lx \n",pc);

printf("begin value of c:%d \n",\*pc);

return c;

}

void main(void)

{

f1(1,2);

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

printf("end address of a:%lx \n",pa);

printf("end value of a:%d \n",\*pa);

printf("end address of c:%lx \n",pc);

printf("end value of c:%d \n",\*pc);

}

第二种：



程序：

int far \* pa;

int far \* pc;

int f1(int a,int b);

int f2(int a,int b);

int f1(int a,int b)

{

int c;

c = a+b;

pa = (int far \*)(&a);

pc = (int far \*)(&b);

printf("f1 address of a:%lx \n",pa);

printf("f1 value of a:%d \n",\*pa);

printf("f1 address of c:%lx \n",pc);

printf("f1 value of c:%d \n",\*pc);

return c;

}

int f2(int a,int b)

{

int c;

c = a+b;

pa = (int far \*)(&a);

pc = (int far \*)(&b);

printf("f2 address of a:%lx \n",pa);

printf("f2 value of a:%d \n",\*pa);

printf("f2 address of c:%lx \n",pc);

printf("f2 value of c:%d \n",\*pc);

return c;

}

void main(void)

{

f1(1,2);

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

f2(3,4);

}