# 数组越界问题：

|  |  |
| --- | --- |
| 现象：该程序会死循环。  原因如下： | 地址  0xA+12  0xA+9  0xA+0  i  arr  内存 |
| （1）arr数组在内存中创建时，内存给arr分配的空间大小是10个元素（4\*10=40个字节）。 |
| （2）printf输出arr[i]时，是从小地址（0xA+0）往大地址（0xA+9）读取的。 |
| （3）创建变量时，变量在内存中存放时，地址是从大到小存放的。即，变量i先被创建，变量arr后被创建，所以i的地址比arr的末尾地址大。 |
| （4）先后被创建的二个变量的地址并不是紧挨着的，具体相差几个字节取决于编译器。Vs编译器环境下，相邻的二个变量的地址之间差2个字节。 |

所以，arr[12]的地址与i的地址是一样的，将arr[12]的内容清零，i的内容也被清零，导致死循环。

其他情况下的运行情况：

（1）访问到arr[10]或arr[11]。

|  |  |
| --- | --- |
| Debug模式下运行，会报错：  报错信息为：运行时检测识别，围绕变量arr的堆栈被损坏。 |  |
| Release模式下，可以正常运行。  会正常打印11个hellow bit。 |  |

（2）更换i和arr定义的顺序。先定义arr，再定义i。这样就可以保证i的地址比arr的地址低。

同样的，在Debug模式下，报错。在Release模式下，正常运行，打印11个结果。

（3）arr可以往大地址方向越界访问，那么能不能往小地址越界访问？ 访问arr[-1]

|  |  |
| --- | --- |
| 可以看到，arr[-1]是可以正常访问的，只是arr初始化并不影响arr[-1]的内容。 |  |
| 可以看到：当程序对arr[-1]进行赋值后，arr[-1]处的内容确实被改变了。 |  |

（4）让arr打印时从高地址往低地址进行赋值操作，直到i的地址，看看是否也会形成死循环。

|  |  |
| --- | --- |
| 由有图可以看出，i的地址比arr[0]低3个字节，即i和arr相距2和字节。现在将程序改为：  #include <stdio.h>  int main()  {  int arr[] = { 1,2,3,4,5,6,7,8,9,10};  int i = 0;  for (i = 9; i >= -3; i--) //64位编译器 i的地址为arr[end]+3  {  arr[i] = 9;  printf("hello bit\n");  }  return 0;  } |  |
| 运行结果是：i地址处的值会被改变，程序陷入死循环。  所以可以确定：（1）在VS编译器下，相邻定义的2个变量的地址也是相近的，地址差为3个字节，中间间隔2个字节。  （2）先被定义的变量具有更高的地址。  （3）对于数组，其首元素的地址是最低的，最后元素的地址是最高的。 |  |