1. 地址

```
#include<stdio.h>
void main( )
{
	int test[5]={1,2,3,4,5};
	//越界访问！请避免此类情形
	printf("%d,%d,%d,%d\n",test[-1],test[0],test[4],test[5]);
	//地址是否连续？
	printf("%p,%p,%p,%p\n",&test[-1],&test[0],&test[4],&test[5]);
	//地址相减
	printf("%d,%d\n",&test[0]-&test[1],&test[6]-&test[4]);
	//数组名的本质
	printf("%p,%p,%p,%p\n",test,&test[0],test+1,&test[0]+1);
	printf("%p,%p",&test,&test+1);
}
```

```
0,1,5,0
000000000061FDFC,000000000061FE00,000000000061FE10,000000000061FE14
-1,2
000000000061FE00,000000000061FE00,000000000061FE04,000000000061FE04
000000000061FE00,000000000061FE14
```

启示：

Line 6: 越界得自己避免！而且下标为负数也不会编译不过

Line 8：地址是连续的，间隔是 sizeof(数组元素)

Line 10: 元素地址相减，得到的是隔了多少个元素，而不是多少个 bytes

Line 12：数组名其实就是第一个元素的地址，加 1 就是加一个 sizeof(数组元素)

Line 13: &数组名，其实是数组的首地址，而不是首个元素的地址，因此，加 1 就是加了一个一维数组的结果，即 sizeof(数组)，形象理解，相当于加了一“行”

2. sizeof 运算符

```c
#include<stdio.h>
int main( )
{
    int a[] = {1,2,3,4};
    char b_str[] = "hello";
    char c_str[] = {'h','e','l','l','o'};
    printf("%d\n",sizeof(a));//元素类型 int ， 有4个元素， 所以大小为 4×4 = 16
    printf("%d\n",sizeof(b_str));//元素类型char ， 有6个元素(包含'\0') ， 所以大小为 6
    printf("%d\n\n",sizeof(c_str)); //元素类型char ， 有5个元素， 所以大小为 5

    printf("%d,%p\n",sizeof(a+0),a+0); // 8
    printf("%d,%d\n",sizeof(*a),*a); // 4
    printf("%d,%d\n",sizeof(a[1]),a[1]); // 4
    printf("%d,%p\n",sizeof(&a),&a); //8

    return 0;
}
```

```
16
6
5

8,000000000061FE10
4,1
4,2
8,000000000061FE10
```

a 是数组名，对应于第一个元素的地址。地址占 8 字节。&a 也是一个地址量

3. 变长数组（此名称容易让人误解）

不是说数组的长度会随时变化，变长数组在其生存期内的长度同样是固定的

```c
#include<stdio.h>
int main( )
{
    int n;
    scanf("%d", &n);
    int array1[n];
    printf("array length: %d\n",sizeof(array1)/sizeof(int));
    n=n+10;
    printf("array length: %d\n\n\n",sizeof(array1)/sizeof(int));

    int m=5;
    int array2[m];
    printf("array length: %d\n",sizeof(array2)/sizeof(int));
    m=m+10;
    printf("array length: %d\n",sizeof(array2)/sizeof(int));
    return 0;
}
```

```
8
array length: 8
array length: 8


array length: 5
array length: 5
```