```
//二：initializer_list(初始化列表)
//如果一个函数，它的实参数量不可预知，但是所有参数的类型相同，我们就可以使用这个initializer_list类型的形参来接收；
//initializer_list 是c++11里提供的新类型，也是 一个类模板，vector；
//我们把initializer_list理解成某种类型值的数组。这个类模板里指定的类型模板参数就是这个数组里保存的数据的类型。
//initializer_list<int> myarray; //数组，元素类型是int，空列表(空数组)；
//initializer_list<int> myarray2 = { 12,14,16,20,30 }; //数组，元素是int类型。
//注意initializer_list队形中的元素 永远是常量值，不能被改变；
//(2.1) begin()、end()遍历、size()获取元素个数
printvalue({"aa","bb","cc"},15); //若要往initializer_list形参传递值的一个序列，则必须要把这个序列放到{}里括起来作为一个整体

//其实c++11将使用大括号{}的初始化(初始化列表)作为一种比较通过用的初始化方式，可用于很多的类型，大家可以注意观察和收集；
//函数initializer_list形参的函数也可以包含其他形参；
```

```
void printvalue(initializer_list<string> tmpstr)
{
    //begin()指向数组首元素的指针。end()指向数组尾元素之后；
    /*for (auto beg = tmpstr.begin(); beg != tmpstr.end(); ++beg)
    {
        cout << (*beg).c_str() << endl;
    }*/
    //范围for语句的工作原理
    for (auto &tmpitem : tmpstr)
    {
```

```
//(2.2)拷贝和赋值
//拷贝、赋值一个initializer_list对象 不会拷贝列表中的元素。原来对象和拷贝或者赋值出来的对象共享表中的元素。
initializer_list<string> myarray3 = { "aa","bb","cc" };
initializer_list<string> myarray4(myarray3);
initializer_list<string> myarray5;
myarray5 = myarray4;
```

```
class CT
{
public:
    explicit CT(const initializer_list<int> &tmpvalue)
    {
        //.....
        int test;
        test = 1;

    myarray5 = myarray4;*/

    //(2.3)初始化列表做构造函数参数
    //CT ct1 = { 10,20,30,40 }; //隐式类型转换
    CT ct1{ 10,20,30,40 };
    CT ct2 = CT({ 10,20,30,40 });
```