以下为容量器的相关用法，容量器近似为数组，以下具体用法参见事例。

1.容量器（数组）的输入输出及遍历

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
	vector<int> v;
	for(int i=0;i<3;i++){
		int s;cin>>s;
		v.push_back(s);
	}//输入三个整形进入容量器(数组);
	for(int i=0;i<3;i++){
		cout<<v[i]<<" ";
	}//输出容量器里的前三个数;
	int count=v.size();
	for(int i=0;i<count;i++)
	{
		cout<<v[i]<<" ";
	}//遍历输出容量器里的数 1;
	for(vector<int>::const_iterator iter=v.cbegin();iter!=v.cend();iter++)
	{
		cout<<(*iter)<<" ";
	}//遍历输出容量器里的数 2;
	for(auto iter=v.cbegin();iter!=v.cend();iter++)
	{
		cout<<(*iter)<<" ";
	}//遍历输出容量器里的数 3;
	return 0;
}
```

2.容量器（数组）的排列（非常重要）

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main()
{
	vector<int> v;
	for(int i=0;i<3;i++){
		int s;cin>>s;
		v.push_back(s);
	}//输入三个整形进入容量器(数组);
```

```
    reverse(v.begin(),v.end());//反向排列向量的从首到尾的所有元素;
    for(auto iter=v.cbegin();iter!=v.cend();iter++)
    {
        cout<<(*iter)<<" ";
    }//遍历输出容量器里的数 3;
    cout<<endl;
    sort(v.begin(),v.end());//正向排列向量的从首到尾的所有元素;
    for(auto iter=v.cbegin();iter!=v.cend();iter++)
    {
        cout<<(*iter)<<" ";
    }//遍历输出容量器里的数 3;
    cout<<endl;
    return 0;
}
```

3.容量器（数组）成员个数

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main()
{
    vector<int> v(10);
    for(int i=0;i<10;i++){
        v[i]=i;
    }
    cout<<v.size()<<endl;//输出向量大小,即包含多少数;
    cout<<v.empty()<<endl;//输出向量是否为空,非空返回逻辑假为 0,空则返回逻辑真为 1;
    v.clear();
    cout<<v.empty()<<endl;
    return 0;
}
```

4.在容量器（数组）任意地方插入数组成员

这个是容量器超棒的地方，在数组中，如果要在数组中间插入数，让其后的成员后移一位比较麻烦，而用容量器就比较容易。

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> v;
```

```
    for(int i=0;i<3;i++){
        int s;cin>>s;
        v.push_back(s);
    }//输入三个整形进入容量器(数组);
    v.insert(v.cbegin(),8);//在最前面插入新元素,元素值为 8;
    v.insert(v.cbegin()+2,5);//在第二个元素前插入新元素,元素值为 5;
    //(第二个元素与第二个数不一样,在第二个元素前插入新元素,即为插到第三位);
    v.insert(v.cend(),9);//在向量末尾追加新元素,元素值为 5;
    for(auto iter=v.cbegin();iter!=v.cend();iter++)
    {
        cout<<(*iter)<<" ";
    }//遍历输出容量器里的数 3;
    cout<<endl;
    v.erase(v.begin()+2);//删除第二个元素,即为删除第三个数;
    for(auto iter=v.cbegin();iter!=v.cend();iter++)
    {
        cout<<(*iter)<<" ";
    }//遍历输出容量器里的数 3;
    cout<<endl;
    v.erase(v.cbegin()+1,v.cbegin()+3);//从第一个元素删到第三个元素,且左闭右开,即删第二
第三个数;
    for(auto iter=v.cbegin();iter!=v.cend();iter++)
    {
        cout<<(*iter)<<" ";
    }//遍历输出容量器里的数 3;
    cout<<endl;
    v.clear();//清空向量;
    cout<<v.size()<<endl;//输出向量大小;
    return 0;
}
```

5.字符串容量器：分离字符串

分离字符串在字符串的操作中非常困难，用容量器比较容易。

```
#include <iostream>
#include <string>
#include <cstdio>
using namespace std;
int main()
{
    string s1,s2,s3;
    char sa[100],sb[100],sc[100];
    sscanf("abc 123 pc","%s %s %s",sa,sb,sc);//将字符串分离成子串,分隔符为空格;
    s1=sa;s2=sb;s3=sc;
```

```
    cout<<s1<<endl<<s2<<endl<<s3<<endl;
    int a,b,c;
    sscanf("123 45 678","%d %d %d",&a,&b,&c);//将字符串分离成数字,分隔符为空格;
    cout<<a<<endl<<b<<endl<<c<<endl;
    int x,y,z;
    sscanf("923^23$798","%d^%d$%d",&x,&y,&z);//将字符串分离成数字,分隔符为^和$;
    cout<<x<<endl<<y<<endl<<z<<endl;
    return 0;
}
```

6.字符串容量器的输入输出。

```
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>
using namespace std;
int main()
{
    vector<string> v;
    v.push_back("Jack");
    v.push_back("Mike");
    v.push_back("Tom");
    cout<<v[0]<<endl;
    cout<<v[1]<<endl;
    cout<<v[0][0]<<endl;
    cout<<v[1][0]<<endl;
    cout<<v[2].length()<<endl;
    return 0;
}
```

7.输出字符串的长度，并在字符串任意位置添加字符、字符串。
这个难度在字符串操作中可想而知。

```
#include <iostream>
#include <string>
using namespace std;
void couts(string s)
{
    cout<<s.length()<<endl;//输出 s 的长度;
    cout<<s<<endl;//输出字符串 s;
}
int main()
{
```

```
    string s;//创建空字符串 s;
    s="hallo";
    s=s+' ';
    s=s+'d';
    s=s+'y';
    s=s+'h';//直接用加号在末尾添加字符,但注意,不能添加字符串;
    couts(s);
    s=s+"dyh";//添加字符串应该用双引号;
    couts(s);
    s.append("dyh");//或者采用 append 函数;
    couts(s);
    s.insert(s.begin()+1,'0');//将字符插到第一个字符前,注意,字符从 0 开始计数,即将字符插入到第一个位置;
    couts(s);
    s.erase(s.begin()+1);//删除第一个字符(从 0 记位);
    couts(s);
    s.erase(s.begin(),s.begin()+6);//删除从 0-5 的字符;
    couts(s);
    s.replace(6,3,"sst");//从第六个字符开始,将连续三个字符换为“sst”;
    s.replace(3,3,"love");
    couts(s);
    return 0;
}
```

8.字符串查找功能

在已有的长字符串中查找子字符串，难度也是极高。

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s;//创建空字符串 s;
    s="cat dog cat net";
    cout<<s.find('c')<<endl;//查找第一个字符 c,并返回其下标值;
    cout<<s.find("c")<<endl;//查找第一个字符串 c,并返回其下标值;
    cout<<s.find("dog")<<endl;//查找第一个字符串 cat,并返回其下标值;
    cout<<s.find("catd")<<endl;//查找第一个字符串 catd,找不到则返回 4294967295;
    return 0;
}
```

9.字符串赋值、字符串长度、输出

```
#include <iostream>
#include <string>
#include <cstdio>
using namespace std;
int main()
{
	string s;//创建空字符串 s;
	s="hello,C++STL.";//对 s 进行赋值;
	cout<<s.length()<<endl;//输出 s 的长度;
	cout<<s<<endl;//输出字符串 s;
	printf(s.c_str());//用 printf 输出字符数组,需要头文件 cstdio;
	return 0;
}
```

优化：

```
#include <iostream>
#include <string>
#include <cstdio>
using namespace std;
int main()
{
	string s;//创建空字符串 s;
	char ss[1000];
	//scanf 的处理速度比 cin 快的多;//但 scanf 是 c 语言的函数,不支持 string 对象;
	scanf("%s",&ss);
	s=ss;//把整个字符数组赋值给 string 对象;
	cout<<s.length()<<endl;//输出 s 的长度;
	cout<<s<<endl;//输出字符串 s;
	return 0;
}
```

10.字符串大小比较

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
	string s;//创建空字符串 s;
	s="cat dog cat";
	cout<<s.compare("cat")<<endl;//s 比 cat 字符大,返回 1;
	cout<<s.compare("cat dog cat")<<endl;//s 与 cat dog cat 字符相等,返回 0;
	cout<<s.compare("dog")<<endl;//s 比 dog 字符小,返回-1;
```

```
    return 0;
}
```

问：为何第一个输出是 8？

因为它返回是真，大于 0 都代表 s 比 cat 字符大；

11.字符串内部元素排列

内部元素排列，嗯，可以，难倒一片

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main()
{
    string s;//创建空字符串 s;
    s="123456789";
    reverse(s.begin(),s.end());//反向排列;
    cout<<s<<endl;
    return 0;
}
```