S T L

标准库 standard library

常用函数

8.0-1-a

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{       cout<<fixed<<setprecision(12)<<30.8/7.1<<endl;
        return 0;
}
```

fixed<<setprecision 格式函数

这段代码作用是把计算结果准确到小数点后 12 位。

8.0-1-b

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
        cout<<fixed<<setprecision(5)<<30.8/7.1<<endl;
        return 0;
}
```

改了一个数字，看结果如何？

8.0-1-c

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
        cout<<fixed<<setprecision(18)<<30.8/7.1<<endl;
        return 0;
}
```

改了一个数字，看结果如何？

8.0-2-a　　abs 函数，求整数的绝对值

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
	float a,a2;
	cin>>a;
	a2=abs(a);
	cout<<a2;
	return 0;
}
```

输入：6
输出：6

输入：-6
输出：6

8.0-2-b　　fabs 函数，求浮点数绝对值

```
#include<iostream>// 浮点数绝对值
#include<cmath>
using namespace std;
int main()
{
	float a;
	cin>>a;
	cout<<fabs(a);
	return 0;
}
```

输入 -3.16
输出 3.16

8.0-3　　swap 交换函数

```
#include<iostream>
using namespace std;
int main()
{
	int a,b;
	cin>>a>>b;
	swap(a,b);
	cout<<a<<" "<<b;
	return 0;
}
```

输入 3　6
输出 6　3

8.0-4

max,min 函数，选择大小

```
#include<iostream>
using namespace std;
int main()
{
	int a,b;
	cin>>a>>b;
	cout<<" 最大数："<<max(a,b)<<endl;
	cout<<" 最小数："<<min(a,b);
	return 0;
}
```

输入：3　　6

输出：最大数：6

　　最小数：3

8.0-5-a

四舍五入函数

```
#include<iostream>// 四舍五入函数
#include<cmath>
using namespace std;
int main()
{
	double a;
	cin>>a;
	cout<<round(a);
	return 0;
}
```

输入：7.33

输出：7

输入：7.66

输出：8

8.0-5-b

floor 函数，向下取整

```
#include<iostream>// 向下取整函数
#include<cmath>
using namespace std;
int main()
{
	double a;
	cin>>a;
	cout<<floor(a);
	return 0;
}
```

输入：6.77

输出：6

8.0-5-c

ceil 函数，向上取整

```
#include<iostream>// 向上取整函数
#include<cmath>
using namespace std;
int main()
{
	double a;
	cin>>a;
	cout<<ceil(a);
	return 0;
}
```

输入： 6.33

输出： 7

8.0-6-a

sqrt 函数。平方根函数

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
	cout<<"4 的平方根 "<<sqrt(4)<<endl;
	cout<<"5 的平方根 "<<sqrt(5);
	return 0;
}
```

8.0-6-b

使用 sqrt 函数求三角形面积（海伦公式）

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
	float n,a,b,c;
	cin>>a>>b>>c;
	n=(a+b+c)/2;
	cout<<sqrt(n*(n-a)*(n-b)*(n-c));
	return 0;
}
```

输入： 3 4 5

输出： 6

8.0-7-a

```
#include<iostream>//pow 函数，a 的 b 次幂
#include<cmath>
using namespace std;
int main()
{
	double a,b;
	cin>>a>>b;
	cout<<pow(a,b);
	return 0;
}
```

POW，是 C、C++ 中的数学函数，功能为计算 a 的 b 次幂，返回幂指数的结果

8.0-7-b

```
#include<iostream>//exp 函数
#include<cmath>
using namespace std;
int main()
{
	double x;
	cin>>x;
	cout<<exp(x);
	return 0;
}
```

exp，高等数学里以自然常数 e 为底的指数函数，e 是一个常数为 2.71828。

8.0-8-a

```
#include<iostream>// 正弦值函数
#include<cmath>
using namespace std;
int main()
{
	double a;
	cin>>a;
	cout<<sin(a);
	return 0;
}
```

正弦值是在直角三角形中，对边的长比上斜边的长的值。任意锐角的正弦值等于它的余角的余弦值，任意锐角的余弦值等于它的余角的正弦值。

8.0-8-b

```
#include<iostream>// 余弦值函数
#include<cmath>
using namespace std;
int main()
{
	double a;
	cin>>a;
	cout<<cos(a);
	return 0;
}
```

余弦值是一个三角函数，用于计算两个向量之间的夹角，在数学中，余弦值是一个三角函数，表示为 cos(x)，其中 x 是一个角度，余弦值的定义是三角形的邻边与斜边之比，在三角形中，邻边是指与角度相邻的边，斜边是指与角度相对的边，余弦值可以用来计算两个向量之间的夹角。

余弦值的取值范围是 -1 到 1 之间。当角度为 0 度时，余弦值为 1，表示两个向量的方向相同；当角度为 90 度时，余弦值为 0，表示两个向量垂直；当角度为 180 度时，余弦值为 -1，表示两个向量的方向相反。

8.0-9-a

```
#include<iostream>// 对数函数
#include<cmath>
using namespace std;
int main()
{
	double a;
	cin>>a;
	cout<<log(a);
	return 0;
}
```

对数，是一个数学名词，也是一种数学方法！

如果 a 的 n 次方等于 b（a 大于 0，且 a 不等于 1），那么数 n 叫做以 a 为底 b 的对数，记做 n=loga 的 b 次方，也可以说 log（a）b=n。其中，a 叫做“底数”，b 叫做“真数”，n 叫做“以 a 为底 b 的对数”。

8.0-9-b

```
#include<iostream>// 以 2 为底对数函数
#include<cmath>
using namespace std;
int main()
{
	double a;
	cin>>a;
	cout<<log2(a);
	return 0;
}
```

8.0-10-a 反转数组函数

输出：8 7 6 9 10

```
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
	int a[5]={6,7,8,9,10};
	reverse(a,a+3);
	for(int i=0;i<5;i++) cout<<a[i]<<" ";
	return 0;
}
```

8.0-10-b 反转数组函数

输出：10 9 8 7 6

```
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
	int a[5]={6,7,8,9,10};
	reverse(a,a+5);
	for(int i=0;i<5;i++) cout<<a[i]<<" ";
	return 0;
}
```

8.0-10-c 反转字符数组函数

输出：dcbaef

```
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
	string a="abcdef";
	reverse(a.begin(),a.begin()+4);
	cout<<a;
	return 0;
}
```

8.0-10-d

反转字符数组函数

输出：fedcba

```
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
	string a="abcdef";
	reverse(a.begin(),a.begin()+6);
	cout<<a;
	return 0;
}
```

8.0-11

```
#include<iostream>// 随机数函数
#include<cstdlib>
using namespace std;
int main()
{
    cout<<rand()<<" ";
    cout<<rand()<<" ";
    cout<<rand()<<" ";
    cout<<rand();
    return 0;
}
```

8.0-12

```
#include<iostream>
#include<iomanip>//setw 场宽函数，用于设置输出字段宽度
using namespace std;
int main()
{
	int a[5]={6,7,8,9,10};
	for(int i=0;i<5;i++)	cout<<setw(3)<<a[i];
	return 0;
}
```

```
  6  7  8  9 10
```

8.0-13

```
#include<bits/stdc++.h>//typedef 用法，把很长难写的类型名称变得短小易写
using namespace std;
int main()
{
	typedef long long ll;// 将 long long 修改为 ll

	ll a;// 定义 a 为 long long 数据类型
	a=123456789012345;

	ll b;// 定义 b 为 long long 数据类型
	b=2345678934325435345;

	cout<<a<<" "<<b;

	return 0;
}
```

```
123456789012345

2345678934325435345
```

sort

8.1-1-a

```
5
5 3 12 4 6
3 4 5 6 12
12 6 5 4 3
```

```
#include<iostream>//sort 排序 从下标 0 开始
#include <algorithm>
using namespace std;
int main()
{
	int a[100],n,i;
	cin>>n;
	for(i=0;i<n;i++)		cin>>a[i];
	sort(a,a+n);// 从小到大完整写法：sort(a,a+len,less<int>());
	for(i=0;i<n;i++)		cout<<a[i]<<" ";
	cout<<endl;

	sort(a,a+n,greater<int>());// 从大到小
	for(i=0;i<n;i++)		cout<<a[i]<<" ";
	return 0;
}
```

8.1-1-b//sort 排序 从下标 1 开始

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
	int a[100],n,i;
	cin>>n;
	for(i=1;i<=n;i++) cin>>a[i];

	sort(a+1,a+n+1);
	for(i=1;i<=n;i++)		cout<<a[i]<<" ";
	return 0;
}
```

8.1-2

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	int a[]= {-1,9,-34,100,45,2,98,32};
	int len=sizeof(a)/sizeof(int);//sizeof 数组长度，
	// 也可以这样写：int len=sizeof(a)/4;, 一个整数占 4 个字节

	sort(a,a+len);// 由小到大排序
	for(int i=0; i<len; i++)
		cout<<a[i]<<" ";
	cout<<endl;

	sort(a,a+len,greater<int>());// 由大到小排序
	for(int i=0; i<len; i++)
		cout<<a[i]<<" ";
	return 0;
}
```

8.1-3

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
	int a[100],n,i;
	cin>>n;
	for(i=0;i<n;i++)		cin>>a[i];

	sort(a+3,a+6);// 从下标 3 开始，到第 6 个数字排序
	for(i=0;i<n;i++)		cout<<a[i]<<" ";

	return 0;
}
```

从下标 3 开始，到第 6 个数，参与排序，其他数字不排序。

输入：

8

8 7 6 5 4 3 2 1

输出：

8 7 6 3 4 5 2 1

8.1-4-a

```
#include<bits/stdc++.h>
using namespace std;
int a[100],n,i,k;
bool cmp(int x,int y)
{
	return x<y;
}
int main()
{
	cin>>n;
	for(i=1;i<=n;i++)	cin>>a[i];

	sort(a+1,a+n+1,cmp);// 从小到大排序

	for(i=1;i<=n;i++)	cout<<a[i]<<" ";
	return 0;
}
```

```
5
5 3 1 2 4
1 2 3 4 5
```

bool cmp 函数不同写法

```
bool cmp(int x,int y)
{
	if(x<y) return true;
	else return false;
}
```

```
bool cmp(int x,int y)
{
	return x<y?true:false;
}
```

8.1-4-b

```
#include<bits/stdc++.h>
using namespace std;
int a[100],n,i,k;
bool cmp(int x,int y) {
	return x>y;
}
int main()
{
	cin>>n;
	for(i=1;i<=n;i++)	cin>>a[i];

	sort(a+1,a+n+1,cmp);// 从大到小排序

	for(i=1;i<=n;i++)	cout<<a[i]<<" ";
	return 0;
}
```

8.1-5

```
#include <bits/stdc++.h>//sort 对字符排序
using namespace std;

int main()
{
	char a[11]="asdfghjklk";
	for(int i=0; i<10; i++)
		cout<<a[i];

	cout<<endl;

	sort(a,a+10,greater<char>());
	for(int i=0; i<10; i++)
		cout<<a[i];

	return 0;
}
```

```
asdfghjklk
slkkjhgfda
```

8.1-6

```
5
2 90
7 85
10 80
3 90
4 100


4 100
2 90
3 90
7 85
10 80
```

```
#include <bits/stdc++.h>//sort 对结构体排序
using namespace std;

struct node
{
	int x;// 学号
	int y;// 成绩
} p[1001];

int cmp(node a,node b)
{
	if (a.y != b.y)
		return a.y > b.y;  // 如果 a.y 不等于 b.y，就按 y（成绩）从大到小排
	else
		return a.x < b.x;  // 如果 y 相等按 x（学号）从小到大排
}

int main()
{
	int n;
	scanf("%d",&n);
	for (int i = 1; i <= n; i++)
		cin>>p[i].x>>p[i].y;

	sort(p+1,p+n+1,cmp);// 排序，比较函数为 cmp
	for (int i = 1; i <=n; i++)
		 cout<<p[i].x<<" "<<p[i].y<<endl;

	return 0;
}
```

二分查找函数 lower-bound/upper-bound

```
8.2-1// 升序数组使用 lower_bound/upper_bound 示例
#include<bits/stdc++.h>
using namespace std;

int main()
{
  int a[] = {1, 1, 2, 2, 3, 3, 3, 4, 4, 4};
  cout<< lower_bound(a, a+10, 0)-a <<endl; // 输出下标 0
  cout<< lower_bound(a, a+10, 1)-a <<endl; // 输出下标 0
  cout<< lower_bound(a, a+10, 3)-a <<endl; // 输出下标 4
  cout<< lower_bound(a, a+10, 4)-a <<endl; // 输出下标 7
  cout<< lower_bound(a, a+10, 5)-a <<endl; // 输出下标 10

  cout<< upper_bound(a, a+10, 0)-a <<endl; // 输出下标 0
  cout<< upper_bound(a, a+10, 1)-a <<endl; // 输出下标 2
  cout<< upper_bound(a, a+10, 3)-a <<endl; // 输出下标 7
  cout<< upper_bound(a, a+10, 4)-a <<endl; // 输出下标 10
  cout<< upper_bound(a, a+10, 5)-a <<endl; // 输出下标 10
  return 0;
}
```

```
8.2-2// 降序数组使用 lower_bound/upper_bound 的正确示例
#include<bits/stdc++.h>
using namespace std;

int main()
{
  int a[] = {4, 4, 3, 3,  2, 2,  1, 1};
  cout<< lower_bound(a, a+8, 0,greater<int>())-a <<endl; // 输出 8
  cout<< lower_bound(a, a+8, 4,greater<int>())-a <<endl; // 输出 0
  cout<< lower_bound(a, a+8, 1,greater<int>())-a <<endl; // 输出 6
  cout<< lower_bound(a, a+8, 3,greater<int>())-a <<endl; // 输出 2
  cout<< lower_bound(a, a+8, 5,greater<int>())-a <<endl; // 输出 2

  cout<< upper_bound(a, a+8, 0,greater<int>())-a <<endl; // 输出 8
  cout<< upper_bound(a, a+8, 4,greater<int>())-a <<endl; // 输出 2
  cout<< upper_bound(a, a+8, 1,greater<int>())-a <<endl; // 输出 8
  cout<< upper_bound(a, a+8, 3,greater<int>())-a <<endl; // 输出 4
  cout<< upper_bound(a, a+8, 5,greater<int>())-a <<endl; // 输出 4
  cout<<binary_search(a,a+8,2,greater<int>())<<endl;
  return 0;
}
```

vector 向量容器

8.3-1

```
#include <iostream>//vecter 创建（本章节选自东方博宜）
#include <vector>
using namespace std;
int main()
{
	vector<int> a;// 创建一个整数 vector

	//a[0]=10;// 错误添加元素方式
	a.push_back(10);// 正确增加元素方式
	a.push_back(20);
	a.push_back(30);
	for(int i=0;i<a.size();i++)		cout<<a[i]<<" ";
	return 0;
}
```

```
10 20 30
```

8.3-2

```
#include<bits/stdc++.h> // 从尾部添加数据
using namespace std;
int main()
{
	vector<int> a(10);// 初始化元素个数为 10

	a.push_back(10);
	a.push_back(20);
	a.push_back(300);
	for(int i=0;i<a.size();i++) // 元素添加到后面
		cout<<a[i]<<" ";
	return 0;
}
```

```
0 0 0 0 0 0 0 0 10 20
300
```

8.3-3

```
90 90 90 90 90
```

```
#include <iostream>//vecter 初始化
#include <vector>
using namespace std;
int main()
{
	vector<int> a(5,90);// 初始化为 5 个元素都是 90

	for(int i=0;i<a.size();i++)
	{
		cout<<a[i]<<" ";
	}
	return 0;
}
```

8.3-4

```
1 2 3 4 5
```

```
#include <iostream>//vecter 初始化
#include <vector>
using namespace std;
int main()
{

	int b[]={1,2,3,4,5};
	vector<int> a(b,b+5);// 复制 b 数组

	for(int i=0;i<a.size();i++)
	{
		cout<<a[i]<<" ";
	}
	return 0;
}
```

8.3-5

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

	int b[]={1,2,3,4,5};
	vector<int> a(b,b+5);

	vector<int> c(a);// 复制 a    vector

	for(int i=0;i<c.size();i++)
	{
		cout<<c[i]<<" ";
	}
	return 0;
}
```

```
1 2 3 4 5
```

8.3-6

```
#include<bits/stdc++.h>//vecter 插入、删除
using namespace std;
int main()
{
	int b[]={1,2,3,4,5};
	vector<int> a(b,b+5);// 复制 b 数组

	a.insert(a.begin(),100);// 在头部插入

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";
	return 0;
}
```

```
100 1 2 3 4 5
```

8.3-7

```
1 100 2 3
```

```
#include<bits/stdc++.h>//vecter 插入、删除
using namespace std;
int main()
{
	vector<int> a;
	a.push_back(1);
	a.push_back(2);
	a.push_back(3);

	a.insert(a.begin()+1,100);// 在下标 1 位置插入

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";
	return 0;
}
```

8.3-8

```
1 100 100 100 100 100 2 3
```

```
#include<bits/stdc++.h>//vecter 插入、删除
using namespace std;
int main()
{
	vector<int> a;
	a.push_back(1);
	a.push_back(2);
	a.push_back(3);

	a.insert(a.begin()+1,5,100);// 下标 1 位置插入 5 个 100

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";

	return 0;
}
```

8.3-9

```
1 4 5 6 2 3
```

```
#include<bits/stdc++.h>// vecter 插入、删除
using namespace std;
int main(){
	vector<int> a;
	a.push_back(1);
	a.push_back(2);
	a.push_back(3);

	vector<int> b;
	b.push_back(4);
	b.push_back(5);
	b.push_back(6);

	a.insert(a.begin()+1,b.begin(),b.end());// 插入 vecror
	for(int i=0;i<a.size();i++)	cout<<a[i]<<" ";
	return 0;
}
```

8.3-10

```
5 6 1 2 3
```

```
#include<bits/stdc++.h>// vecter 插入、删除
using namespace std;
int main(){
	vector<int> a;
	a.push_back(1);
	a.push_back(2);
	a.push_back(3);

	vector<int> b;
	b.push_back(4);
	b.push_back(5);
	b.push_back(6);
	a.insert(a.begin(),b.begin()+1,b.end());// 插入 vector
	for(int i=0;i<a.size();i++)	cout<<a[i]<<" ";
	return 0;
}
```

8.3-11

```
#include<bits/stdc++.h>//vecter 插入、删除
using namespace std;
int main()
{
	vector<int> a;
	a.push_back(1);
	a.push_back(2);
	a.push_back(3);

	a.erase(a.begin()+1);// 删除下标 1 位置数据

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";
	return 0;
}
```

```
1 3
```

8.3-12

```
#include<bits/stdc++.h>//vecter 批量删除
using namespace std;
int main()
{
	vector<int> a;
	a.push_back(1);
	a.push_back(2);
	a.push_back(3);

	a.erase(a.begin()+1,a.end());// 删除下标 1 位置及后面所有数据

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";
	return 0;
}
```

```
1
```

8.3-13

```
#include<bits/stdc++.h>//vecter 第一个、最后一个元素
using namespace std;
int main()
{
	vector<int> a;
	a.push_back(1);
	a.push_back(2);
	a.push_back(3);
	cout<<" 第一个元素 "<<a.front()<<endl;
	cout<<" 最后一个元素 "<<a.back()<<endl;
	cout<<" 第一个元素 "<<*a.begin()<<endl;
	cout<<" 最后一个元素 "<<*(a.end()-1)<<endl;
	for(int i=0;i<a.size();i++)	cout<<a[i]<<" ";
	return 0;
}
```

```
第一个元素 1
最后一个元素 3
第一个元素 1
最后一个元素 3
1 2 3
```

8.3-14

```
#include<bits/stdc++.h>//vecter 交换
using namespace std;
int main()
{
	int c[]={1,2,3};
	vector<int> a(c,c+3);

	int d[]={4,5,6,7};
	vector<int> b(d,d+4);

	swap(a,b);

	for(int i=0;i<a.size();i++)	cout<<a[i]<<" ";
	cout<<endl;
	for(int i=0;i<a.size();i++) // 元素数量不同也可以交换，缺少的元素为 0
		cout<<b[i]<<" ";
	return 0;
}
```

```
4 5 6 7
1 2 3 0
```

8.3-15-a

```
#include<bits/stdc++.h>//vector 排序
using namespace std;
int main()
{
	int c[]={5,3,2,1,4};
	vector<int> a(c,c+5);

	sort(a.begin(),a.end());// 排序，从小到大

	for(int i=0;i<a.size();i++)    cout<<a[i]<<" ";
	cout<<endl;

	reverse(a.begin(),a.end());// 反转

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";
	return 0;
}
```

```
1 2 3 4 5
5 4 3 2 1
```

8.3-15-b

```
#include<bits/stdc++.h>//vector 反转
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	vector<int> a(c,c+5);

	reverse(a.begin(),a.begin()+3);// 部分反转

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";
	return 0;
}
```

```
3 2 1 4 5
```

8.3-16

```
#include<bits/stdc++.h>//vector 重设大小
using namespace std;
int main()
{
	int c[]={5,3,2,1,4};
	vector<int> a(c,c+5);

	a.resize(10);// 重设 vector 大小

	for(int i=0;i<a.size();i++)
		cout<<a[i]<<" ";
	return 0;
}
```

```
5 3 2 1 4 0 0 0 0 0
```

8.3-17

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

	v.push_back(20);
	v.push_back(26);
	v.push_back(12);
	v.push_back(52);

	v.reserve(30);		// 调整数据空间大小

	for(int i=0; i<v.size(); ++i)		cout<<v[i]<<" ";
	return 0;
}
```

```
20 26 12 52
```

8.3-18

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
	vector<int> v;
	v.push_back(20);

	v.clear(); // 全部清空元素

	for(int i=0; i<v.size(); ++i)		cout<<v[i]<<" ";
	return 0;
}
```

8.3-19

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
	vector<int> v;
	v.push_back(20);
	cout<<v.empty();// 判断是否为空
	return 0;
}
```

```
0
```

8.3-20

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
	vector<int> v;
	cout<<v.empty();// 判断是否为空
	return 0;
}
```

```
1
```

8.3-21

```
#include<bits/stdc++.h>//vecter 长度
using namespace std;
int main()
{

	int b[]={1,2,3,4,5};
	vector<int> a(b,b+5);// 复制 b 数组

	cout<<sizeof(a)<<endl;// 计算变量占用字节数
	cout<<sizeof(a)/4<<endl;// 一个整数占用 4 个字节
	cout<<sizeof(a)/sizeof(int)<<endl;//sizeof(int)  整数占用字节数

	cout<<sizeof(b)/sizeof(int)<<endl;// 可以观察到 vector 比数组多一个数据
	cout<<a.size() <<endl;// 获取 vector 元素个数（最常用的获取元素个数方法）
	return 0;
}
```

```
24
6
6
5
5
```

8.3-22

```
#include<bits/stdc++.h>// 观察 capacity 与 size 区别
using namespace std;
int main()
{
	vector<int> v;
	for(int i=0;i<100;i++)
	{
		v.push_back(i+1);
		cout<<v[i]<<" "<<v.size()<<" "<<v.capacity()<<endl;
	}
	return 0;
}
```

```
1 1
2 2 2
3 3 4
4 4 4
5 5 8
6 6 8
7 7 8
8 8 8
9 9 16
10 10 16
11 11 16
……
16 16 16
17 17 32
18 18 32
……
31 31 32
32 32 32
33 33 64
34 34 64
……
64 64 64
65 65 128
66 66 128
……
98 98 128
99 99 128
100 100 128
```

8.3-23

```
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
```

```
#include<bits/stdc++.h>// 二维 vector
using namespace std;
int main()
{
	vector<vector <int> > a(20);

	for(int i=0;i<5;i++)
	{
		for(int j=0;j<5;j++)
			a[i].push_back(i+j);
	}

	for(int i=0;i<5;i++)
	{
		for(int j=0;j<5;j++)
			cout<<a[i][j]<<" ";
		cout<<endl;
	}
	return 0;
}
```

8.3-24// 结构体容器的 vector

```
#include <bits/stdc++.h>
using namespace std;

struct stu
{
  int x,y;
};

int main()
{
      int j;
      vector<stu> v1;                          // 结构体容器
      vector<stu> v2;
      struct stu a= {1,2};
      struct stu b= {2,3};
      struct stu c= {4,5};
      v1.push_back(a);
      v1.push_back(b);
      v1.push_back(c);
      v2.push_back(c);
      v2.push_back(b);
      v2.push_back(a);
      swap(v1,v2);                             // 两结构体元素交换
      for(int i=0; i<v1.size(); i++)           // 输出 v1 所有元素
            cout<<v1[i].x<<" "<<v1[i].y<<endl;
      cout<<"\n";
      for(int i=0; i<v2.size(); i++)           // 输出 v2 所有元素
            cout<<v2[i].x<<" "<<v2[i].y<<endl;
      return 0;
}
```

```
4 5
2 3
1 2

1 2
2 3
4 5
```

8.3-25// 数组方式访问 vector

```
26 12 52
 首元素为 :26
末元素为 :52
52 12 26
 v 是否为空 :1
```

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
	vector<int> v;		// 定义了一个存放 int 类型的 vector 容器
	v.reserve(30);		// 调整数据空间大小
	v.push_back(20);	// 尾端插入新元素
	v.push_back(26);
	v.push_back(12);
	v.push_back(52);
	v.insert(v.begin(),2);//begin() 为 vector 头部，在此处插入 2
	v.insert(v.end(),43); //end() 为 vector 尾部，在此处插入 43
	v.insert(v.begin()+2,15);			// 在第 2 个元素前插入 15
	v.erase(v.begin()+1);				// 删除第 2 个元素
	v.erase(v.begin(),v.begin()+2);		// 删除前三个元素
	v.pop_back();// 删除末尾的一个元素
	for(int i=0; i<v.size(); ++i)		//size() 为 v 中元素个数
		cout<<v[i]<<' ';
	cout<<"\n 首元素为 :"<<v.front()<<'\n'; // 首元素引用
	cout<<" 末元素为 :"<<v.back()<<'\n';	 // 末元素引用
	reverse(v.begin(),v.end());			// 反转整个 vector 元素
	for(int i=0; i<v.size(); ++i)
		cout<<v[i]<<' ';
	v.clear();							// 全部清空元素
	cout<<"\n v 是否为空 :"<<v.empty()<<'\n';// 判断是否为空
	return 0;
}
```

iterator 迭代器

8.4-1

```
#include<bits/stdc++.h>// 迭代器
using namespace std;
int main()
{
	int b[]={1,3,5,7,9};
	vector<int> a(b,b+5);

	vector<int>::iterator it;// 定义迭代器，命名为 it
	it=a.begin();// 迭代器指向 vector 首元素

	cout<<*it<<" "<<a[0]<<endl;// 观察：迭代器指向到 a[0]

	for(int i=0;i<5;i++)	cout<<a[i]<<" ";
	return 0;
}
```

```
1 1
1 3 5 7 9
```

8.4-2

```
#include<bits/stdc++.h>// 迭代器
using namespace std;
int main(){
	int b[]={1,3,5,7,9};
	vector<int> a(b,b+5);

	vector<int>::iterator it;// 定义迭代器，命名为 it
	it=a.begin();// 迭代器指向 vector 首元素
	it++;
	cout<<*it<<endl;// 观察：迭代器指向到 a[1]

	for(int i=0;i<5;i++)	cout<<a[i]<<" ";
	return 0;
}
```

```
3
1 3 5 7 9
```

8.4-3

```
#include<bits/stdc++.h>// 迭代器
using namespace std;
int main()
{
	int b[]={1,3,5,7,9};
	vector<int> a(b,b+5);

	vector<int>::iterator it;// 定义迭代器，命名为 it
	it=a.begin();// 迭代器指向 vector 首元素
	(*it)++;
	cout<<*it<<" "<<a[0]<<endl;// 观察 :a[0] 的数据被修改

	for(int i=0;i<5;i++)	cout<<a[i]<<" ";

	return 0;
}
```

```
2 2
2 3 5 7 9
```

8.4-4

```
#include<bits/stdc++.h>// 迭代器遍历
using namespace std;
int main()
{
	int b[]={1,3,5,7,9};
	vector<int> a(b,b+5);

	vector<int>::iterator it;// 定义迭代器，命名为 it
	it=a.begin();// 迭代器指向 vector 首元素

	for(it=a.begin();it!=a.end();it++) // 遍历 vector
		cout<<*it<<" ";

	return 0;
}
```

```
1 3 5 7 9
```

8.4-5

```
#include<bits/stdc++.h>// 迭代器反向遍历
using namespace std;
int main()
{
	int b[]={1,3,5,7,9};
	vector<int> a(b,b+5);

	vector<int>::reverse_iterator rit;// 定义迭代器，命名为 rit

	for(rit=a.rbegin();rit!=a.rend();rit++) // 遍历 vector
		cout<<*rit<<" ";
	return 0;
}
```

```
9 7 5 3 1
```

8.4-6// 迭代器访问 vector

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
	int j;
	vector<int> v;
	v.reserve(30);                          // 调整数据空间大小
	for(int i=0; i<10; i++)
		v.push_back(i);                         // 尾端插入新元素
	vector<int>::iterator i;                // 定义 vector 的迭代器 i
	for(i=v.begin(); i!=v.end(); i++)       // 迭代器遍历
		cout<<*i<<" ";
	cout<<"\n v 中的元素个数 :"<<v.size()<<'\n';// 元素实际个数
	reverse(v.begin(),v.end());             // 反转
	for(i=v.begin(); i!=v.end(); i++)       // 迭代器遍历
		cout<<*i<<" ";
	v.clear();                              // 全部清空元素
	cout<<"\n v 是否为空 :"<<v.empty()<<'\n'; // 判断是否为空
	return 0;
}
```

```
0 1 2 3 4 5 6 7 8 9
 v 中的元素个数 :10
9 8 7 6 5 4 3 2 1 0
 v 是否为空 :1
```

deque 双向队列

8.5-1

```
#include<bits/stdc++.h>// 空队列，添加数据
using namespace std;
int main()
{
	deque<int> d;// 创建双向队列
	d.push_front(1);
	d.push_front(2);
	d.push_front(3);// 观察：在队头添加数据
	for(int i=0;i<d.size();i++)
		cout<<d[i]<<" ";
	return 0;
}
```

```
3 2 1
```

8.5-2

```
#include<bits/stdc++.h>// 空队列，添加数据
using namespace std;
int main()
{
	deque<int> d;// 创建双向队列
	d.push_back(1);
	d.push_back(2);
	d.push_back(3);// 观察：在队尾添加数据
	for(int i=0;i<d.size();i++)
		cout<<d[i]<<" ";
	return 0;
}
```

```
1 2 3
```

8.5-3

```
#include<bits/stdc++.h>// 双向队列
using namespace std;
int main(){
	deque<int> d(5);// 创建包含 5 个数据的双向队列
	for(int i=0;i<d.size();i++)
		cout<<d[i]<<" ";
	return 0;
}
```

```
0 0 0 0 0
```

8.5-4

```
#include<bits/stdc++.h>// 队列，5 个数据都是 10
using namespace std;
int main(){
	deque<int> d(5,10);
	for(int i=0;i<d.size();i++)
		cout<<d[i]<<" ";
	return 0;
}
```

```
10 10 10 10 10
```

8.5-5

```
#include<bits/stdc++.h>// 队列，复制数组
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	deque<int> d(c,c+5);
	for(int i=0;i<d.size();i++)
		cout<<d[i]<<" ";
	return 0;
}
```

```
1 2 3 4 5
```

8.5-6

```
#include<bits/stdc++.h>// 双向队列输出队头、队尾
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	deque<int> d(c,c+5);
	cout<<d.front()<<" "<<d.back()<<endl;// 双向队列输出对头队尾
	return 0;
}
```

```
1 5
```

8.5-7

```
#include<bits/stdc++.h>// 队列，删除
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	deque<int> d(c,c+5);
	d.pop_front();// 删除队头
	d.pop_back();// 删除队尾
	for(int i=0;i<d.size();i++)
		cout<<d[i]<<" ";
	return 0;
}
```

```
2 3 4
```

8.5-8

```
#include<bits/stdc++.h>// 双向队列迭代器遍历
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	deque<int> d(c,c+5);
	deque<int>::iterator it;
	for(it=d.begin();it<d.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
1 2 3 4 5
```

8.5-9

```
#include<bits/stdc++.h>// 双向队列判断是否为空
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	deque<int> d;
	cout<<d.empty();// 双向队列判断是否为空

	return 0;
}
```

```
1
```

8.5-10

```
#include<bits/stdc++.h>// 双向队列判断是否为空
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	deque<int> d(c,c+5);
	cout<<d.empty();// 双向队列判断是否为空

	return 0;
}
```

```
0
```

list 链表

8.6-1

```
1 2 3
100 1 2 3 200
```

```
#include<iostream>// 链表
#include<list>
using namespace std;
int main()
{
	list<int> l;// 创建链表

	l.push_back(1);// 尾部添加元素
	l.push_back(2);
	l.push_back(3);

	list<int>::iterator it;
	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	cout<<endl;

	l.push_front(100);// 头部添加元素

	l.push_back(200);// 尾部添加元素

	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	cout<<endl;
	return 0;
}
```

8.6-2

```
#include<bits/stdc++.h>// 链表
using namespace std;
int main()
{
	list<int> l(5);// 链表初始化为 5 个数据
	list<int>::iterator it;

	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
0 0 0 0 0
```

8.6-3

```
#include<bits/stdc++.h>// 链表，5 个 100
using namespace std;
int main(){
	list<int> l(5,100);

	list<int>::iterator it;
	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
100 100 100 100 100
```

8.6-4

```
#include<bits/stdc++.h>// 链表，复制数组
using namespace std;
int main(){
	int c[]={1,2,3,4,5};
	list<int> l(c,c+5);
	list<int>::iterator it;

	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
1 2 3 4 5
```

8.6-5

```
#include<iostream>// 链表
#include<list>
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	list<int> l(c,c+5);

	l.insert(l.begin(),10);// 在头部插入
	l.insert(l.end(),20);// 在尾部插入

	list<int>::iterator it;
	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
10 1 2 3 4 5 20
```

8.6-6

```
#include<bits/stdc++.h>// 链表
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	list<int> l(c,c+5);

	list<int>::iterator it=l.begin();
	for(int i=1;i<3;i++) // 在第三个位置插入
		it++;
	l.insert(it,100);

	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
1 2 100 3 4 5
```

8.6-7

```
#include<bits/stdc++.h>// 链表
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	list<int> l(c,c+5);

	list<int>::iterator it=l.begin();
	for(int i=1;i<3;i++) // 在第三个位置插入
		it++;
	l.insert(it,3,100);// 插入 3 个 100

	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
1 2 100 100 100 3 4 5
```

8.6-8

```
#include<bits/stdc++.h>// 链表, 移除
using namespace std;
int main()
{
	int c[]={1,2,3,4,5};
	list<int> l(c,c+5);
	l.remove(4);// 移除数字 4

	list<int>::iterator it;
	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
1 2 3 5
```

8.6-9

```
#include<bits/stdc++.h>// 链表排序
using namespace std;
int main()
{
	int c[]={5,1,3,4,2};
	list<int> l(c,c+5);
	l.sort();// 从小到大排序

	list<int>::iterator it;
	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
1 2 3 4 5
```

8.6-10

```
#include<bits/stdc++.h>// 链表排序
using namespace std;
bool cmp(int a,int b)
{
	return a>b;
}
int main()
{
	int c[]={5,1,3,4,2};
	list<int> l(c,c+5);
	l.sort(cmp);// 从大到小排序

	list<int>::iterator it;
	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
5 4 3 2 1
```

8.6-11

```
#include<bits/stdc++.h>// 链表反转
using namespace std;
int main()
{
	int c[]={5,1,3,4,2};
	list<int> l(c,c+5);
	l.sort();// 从小到大排序

	l.reverse();// 反转

	list<int>::iterator it;
	for(it=l.begin();it!=l.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
5 4 3 2 1
```

set 集合 关联容器

8.7-1

```
#include <iostream>
#include<set>
using namespace std;
int main()
{
	int a[]={5,1,3,1,2,4};
	set<int> s(a,a+6);

	set<int>::iterator it;
	for(it=s.begin();it!=s.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
1 2 3 4 5
```

8.7-2

```
#include <iostream>//set 排序
#include<set>
using namespace std;
int main()
{
	int a[]={5,1,3,1,2,4};
	set<int,less<int> > s(a,a+6);// 从小到大

	set<int>::iterator it;
	for(it=s.begin();it!=s.end();it++)
	{
		cout<<*it<<" ";
	}
	return 0;
}
```

```
1 2 3 4 5
```

8.7-3

```
#include <iostream>//set 排序
#include<set>
using namespace std;
int main()
{
	int a[]={5,1,3,1,2,4};
	set<int,greater<int> > s(a,a+6);// 从大到小

	set<int>::iterator it;
	for(it=s.begin();it!=s.end();it++)
	{
		cout<<*it<<" ";
	}
	return 0;
}
```

```
5 4 3 2 1
```

8.7-4

```
#include <iostream>//set 关联容器  插入
#include<set>
using namespace std;
int main()
{

	int a[]={5,1,3,1,2,4};
	set<int> s(a,a+6);

	s.insert(10);// 插入，不要指定位置，或者插入重复数据

	set<int>::iterator it;
	for(it=s.begin();it!=s.end();it++)
	{
		cout<<*it<<" ";
	}
	return 0;
}
```

```
1 2 3 4 5 10
```

8.7-5

```
#include <iostream>//set 关联容器 删除
#include<set>
using namespace std;
int main()
{
	int a[]={5,1,3,1,2,4};
	set<int> s(a,a+6);

	s.erase(s.begin());// 删除

	set<int>::iterator it;
	for(it=s.begin();it!=s.end();it++)
	{
		cout<<*it<<" ";
	}
	return 0;
}
```

```
2 3 4 5
```

8.7-6

```
#include <iostream>//set 关联容器 删除
#include<set>
using namespace std;
int main()
{
	int a[]={50,10,30,10,20,40};
	set<int> s(a,a+6);

	s.erase(30);// 删除 指定值

	set<int>::iterator it;
	for(it=s.begin();it!=s.end();it++)
		cout<<*it<<" ";
	return 0;
}
```

```
10 20 40 50
```

8.7-7

```
#include <iostream>//set 关联容器 查找
#include<set>
using namespace std;
int main()
{
	int a[]={50,10,30,10,20,40};
	set<int> s(a,a+6);

	set<int>::iterator it;

	it=s.find(30);// 查找，如果找不到，显示最后一个元素
		cout<<*it;

	return 0;
}
```

```
30
```

8.7-8

```
1 wang 98
3 li 98
2 zhang 100
```

```
#include <iostream>
#include<set>
using namespace std;
struct Student{
	int num;
	string name;
	int score;
	// 重载 operator 运算符 <
	bool operator<(const Student &s) const
	{
		// 按照分数升序，分数相同按照学号升序
		if(score<s.score||score==s.score&&num<s.num)
			return true;
		else return false;
	}
};
int main()
{
	set<Student> s; //set<Student,less<Student> > s;
	Student s1={2,"zhang",100};
	Student s2={1,"wang",98};
	Student s3={3,"li",98};
	Student s4={1,"wang",98};

	s.insert(s1);
	s.insert(s2);
	s.insert(s3);
	s.insert(s4);

	set<Student>::iterator it;
	for(it=s.begin();it!=s.end();it++)
		cout<<it->num<<" "<<it->name<<" "<<it->score<<endl;
	return 0;
}
```

8.7-9

```
2 zhang 100
3 li 98
1 wang 98
```

```
#include <iostream>
#include<set>
using namespace std;
struct Student{
	int num;
	string name;
	int score;
	// 重载 operator 运算符 <
	bool operator>(const Student &s) const
	{
		// 按照分数升序，分数相同按照学号升序
		if(score>s.score||score==s.score&&num>s.num)
			return true;
		else return false;
	}
};
int main()
{
	set<Student,greater<Student> > s;
	Student s1={2,"zhang",100};
	Student s2={1,"wang",98};
	Student s3={3,"li",98};
	Student s4={1,"wang",98};

	s.insert(s1);
	s.insert(s2);
	s.insert(s3);
	s.insert(s4);

	set<Student>::iterator it;
	for(it=s.begin();it!=s.end();it++)
		cout<<it->num<<" "<<it->name<<" "<<it->score<<endl;
	return 0;
}
```

map 映射 关联容器

8.8-1

```
#include<bits/stdc++.h>// 创建
using namespace std;
int main()
{
	map<int,string> m;
	m[1000]="zhang";
	m[1001]="wang";
	cout<<m[1000]<<" "<<m[1001];
	return 0;
}
```

```
zhang wang
```

8.8-2

```
#include<bits/stdc++.h>//map 创建
using namespace std;
int main()
{
	map<int,string> m;
	m[1000]="zhang";
	m[1001]="wang";
	m[1000]="li";// 后面的数据会覆盖前面的数据
	cout<<m[1000]<<" "<<m[1001];
	return 0;
}
```

```
li wang
```

8.8-3

```
#include<bits/stdc++.h>//map 创建
using namespace std;
int main()
{
	map<int,string> m;//map 排序默认从小到大

	m[1002]="zhao";
	m[1003]="sun";
	m[1001]="wang";
	m[1000]="li";
	map<int,string>::iterator it;
	for(it=m.begin();it!=m.end();it++)
		cout<<it->first<<" "<<it->second<<endl;
	return 0;
}
```

```
1000 li
1001 wang
1002 zhao
1003 sun
```

8.8-4

```
#include<bits/stdc++.h>//map 创建
using namespace std;
int main()
{
	map<int,string,greater<int> > m;//map 排序从大到小

	m[1002]="zhao";
	m[1003]="sun";
	m[1001]="wang";
	m[1000]="li";
	map<int,string>::iterator it;
	for(it=m.begin();it!=m.end();it++)
	{
		cout<<it->first<<" "<<it->second<<endl;
	}
	return 0;
}
```

```
1003 sun
1002 zhao
1001 wang
1000 li
```

8.8-5

```
#include<bits/stdc++.h>//map 插入
using namespace std;
int main()
{
	map<int,string> m;
	m[1002]="zhao";
	m[1003]="sun";
	m[1001]="wang";
	m[1000]="li";

	pair<int,string> p(1004,"zhou");
	m.insert(p);// 插入

	map<int,string>::iterator it;
	for(it=m.begin();it!=m.end();it++)
		cout<<it->first<<" "<<it->second<<endl;
	return 0;
}
```

```
1000 li
1001 wang
1002 zhao
1003 sun
1004 zhou
```

8.8-6

```
#include<bits/stdc++.h>//map 插入
using namespace std;
int main()
{
	map<int,string> m;
	pair<int,string> p1(1004,"zhou");
	pair<int,string> p2(1004,"zhouzhou");
	m.insert(p1);// 插入
	m.insert(p2);// 重复插入键值相同数据 ， 不成功

	map<int,string>::iterator it;
	for(it=m.begin();it!=m.end();it++)
		cout<<it->first<<" "<<it->second<<endl;
	return 0;
}
```

```
1004 zhou
```

8.8-7

```
#include<bits/stdc++.h>//map 删除
using namespace std;
int main()
{
	map<int,string> m;
	m[1002]="zhao";
	m[1003]="sun";
	m[1001]="wang";
	m[1000]="li";

	m.erase(1003);// 删除

	map<int,string>::iterator it;
	for(it=m.begin();it!=m.end();it++)
		cout<<it->first<<" "<<it->second<<endl;
	return 0;
}
```

```
1000 li
1001 wang
1002 zhao
```

8.8-8

```
#include<bits/stdc++.h>//map 删除
using namespace std;
int main()
{
	map<int,string> m;
	m[1002]="zhao";
	m[1003]="sun";
	m[1001]="wang";
	m[1000]="li";
	m.erase(m.begin());// 删除

	map<int,string>::iterator it;
	for(it=m.begin();it!=m.end();it++)
		cout<<it->first<<" "<<it->second<<endl;
	return 0;
}
```

```
1001 wang
1002 zhao
1003 sun
```

8.8-9

```
#include<bits/stdc++.h>//map 查找
using namespace std;
int main(){
	map<int,string> m;
	m[1002]="zhao";
	m[1003]="sun";

	if(m.find(1003)!=m.end())	cout<<"1003 找到了 "<<endl;
	return 0;
}
```

```
1003 找到了
```

8.8-10

```
#include<bits/stdc++.h>//map 查找
using namespace std;
int main(){
	map<int,string> m;
	m[1002]="zhao";
	m[1003]="sun";

	if(m.count(1003))	cout<<"1003 找到了 "<<endl;
	return 0;
}
```

```
1003 找到了
```

8.8-11

```
#include<bits/stdc++.h>//map 查找
using namespace std;
int main(){
	map<int,string> m;
	m[1002]="zhao";
	m[1003]="sun";

	map<int,string>::iterator it;
	it=m.find(1003);
	if(it!=m.end())	cout<<"1003 找到了，值为 "<<it->second;
	return 0;
}
```

```
1003 找到了，值为 sun
```

pair 关联容器

8.9-1

```
#include<bits/stdc++.h>// 创建 pair
using namespace std;
int main()
{
	pair<int,string> p;
	p.first=1000;
	p.second="zhang";
	cout<<p.first<<" "<<p.second;
	return 0;
}
```

```
1000 zhang
```

8.9-2

```
#include<bits/stdc++.h>// 创建 pair
using namespace std;
int main()
{
	pair<int,string> p(1001,"wang");

	cout<<p.first<<" "<<p.second;
	return 0;
}
```

```
1001 wang
```

8.9-3

```
#include<bits/stdc++.h>// 创建 pair
using namespace std;
int main()
{
	pair<int,string> p;
	p=make_pair(1003,"li");
	cout<<p.first<<" "<<p.second;
	return 0;
}
```

```
1003 li
```

8.9-4

```
#include<bits/stdc++.h>// 创建 pair
using namespace std;
int main()
{
	typedef pair<int,string> Stu;

	Stu s1(1000,"zhang");

	Stu s2;
	s2.first=1002;
	s2.second="wang";

	Stu s3=make_pair(1003,"li");

	cout<<s1.first<<" "<<s1.second<<endl;
	cout<<s2.first<<" "<<s2.second<<endl;
	cout<<s3.first<<" "<<s3.second<<endl;
	return 0;
}
```

```
1000 zhang
1002 wang
1003 li
```

8.9-5

```
#include<bits/stdc++.h>//pair 连续输入，排序，输出
using namespace std;
pair<int,int> a[100] ;
int n,i;
int main()
{
	cin>>n;
	int x,y;
	for(i=1;i<=n;i++)// 输入 pair
	{
		cin>>x>>y;
		a[i]=make_pair(x,y);
	}
	sort(a+1,a+1+n); // 按照字典码排序

	for(i=1;i<=n;i++)
	{
		cout<<a[i].first<<" " <<a[i].second<<endl;
	}

	return 0;
}
```

```
3
1 100
2 200
3 300


1 100
2 200
3 300
```

stack

```cpp
#include <bits/stdc++.h> //8.10-1 栈的添加、弹出、输出
using namespace std;
int main(){
	stack<int> s;// 定义空栈
	s.push(100);// 添加元素 入栈
	s.push(20);
	s.push(30);
	cout<<s.top()<<endl;// 输出栈顶元素

	s.pop();// 出栈  弹出栈顶元素
	cout<<s.top()<<endl;// 输出栈顶元素
	return 0;
}
```

```
30
20
```

```cpp
#include <bits/stdc++.h> //8.10-2 栈的连续添加、弹出、输出
using namespace std;
int main(){
	stack<int> s;// 定义空栈
	int x;
	while(1)
	{
		cin>>x;
		if(x==-1) break;// 输入 -1 结束
		s.push(x);// 入栈
	}
	while(s.empty()==false) // 连续弹出栈顶元素
	{
		cout<<s.top()<<endl;
		s.pop() ;
	}
	return 0;
}
```

```
1 2 3 -1
3
2
1
```

queue

8.11-1

```
10
30
```

```
#include<bits/stdc++.h>// 队列的入队、出队
using namespace std;
int main(){
	queue<int> q;
	q.push(10);
	q.push(20);
	q.push(30);
	cout<<q.front()<<" "<<q.back();// 输出队头、队尾
	return 0;
}
```

8.11-2

```
1 2 3 -1
1
2
3
```

```
#include <bits/stdc++.h> // 队列的连续入队、出队
using namespace std;
int main()
{
	queue<int> q;// 定义空栈
	int x;
	while(1)
	{
		cin>>x;
		if(x==-1) break;// 输入 -1 结束
		q.push(x);// 入栈
	}
	while(q.empty()==false) // 连续弹出栈顶元素
	{
		cout<<q.front()<<endl;
		q.pop() ;
	}
	return 0;
}
```

qriority_queue

8.12-1

```
#include<bits/stdc++.h>// 优先队列
using namespace std;// 大顶堆（大根堆）
int main()
{

	//priority_queue<int,vector<int>,less<int> > q;
	priority_queue<int> q;// 本句等同于上一句
	q.push(3);
	q.push(5);
	q.push(1);// 入队，添加元素
	q.push(4);
	q.push(2);

	while(q.size()!=0)
	{
		cout<<q.top()<<" ";
		q.pop();
	}
	return 0;
}
```

```
5 4 3 2 1
```

8.12-2

```
1 2 3 4 5
```

```
#include<bits/stdc++.h>// 优先队列
using namespace std;// 小顶堆（小根堆）
int main()
{

	priority_queue<int,vector<int>,greater<int> > q;

	q.push(3);
	q.push(5);
	q.push(1);// 入队，添加元素
	q.push(4);
	q.push(2);

	while(q.size()!=0)
	{
		cout<<q.top()<<" ";
		q.pop();
	}
	return 0;
}
```

```
4 sun 99
3 wang 98
1 li 98
2 zhang 97
```

```
#include<bits/stdc++.h>//8.12-3 优先队列 结构体
using namespace std;// 大顶堆（大根堆）
struct Student{
	int num;// 学号
	string name;// 姓名
	int score;// 成绩
	bool operator<(const Student &s) const    // 重写运算符 operator<
	{
		//          成绩相同，按照学号降序
		if(score<s.score||score==s.score&&num<s.num)    return true;
		else return false;
	}
};
int main()
{
	priority_queue<Student,vector<Student>,less<Student> > q;// 从大到小排序

	Student s1={2,"zhang",97};
	Student s2={3,"wang",98};
	Student s3={1,"li",98};
	Student s4={4,"sun",99};

	q.push(s1);
	q.push(s2);
	q.push(s3);
	q.push(s4);

	while(q.size()!=0)
	{
		cout<<(q.top()).num<<"  "<<(q.top()).name<<"  "<<(q.top()).
score<<endl;
		q.pop();
	}
	return 0;
}
```