# STL

STL 是“Standard Template Library”的缩写，中文译为“**标准模板库**”。STL 是 C++标准库的一部分。该库提供了常用的**数据结构和算法**。STL 由于是 c++给写好的数据结构，有时候它的时间复杂度会略高于自己实现的数据结构,容器可以理解成能够实现很多功能的系统函数，或者说用来存放数据的对象，开发者可以根据接口规范（调用格式）直接调用，而不用关心其内部实现的原理和具体代码，十分方便快捷。

**本讲内容主要讲解四种基本容器（pair、栈、队列和 vector）**

# 1.pair

pair 实质上是一个二元结构体，其主要的两个成员变量是 first 和 second，这两个变量可以直接使用。

（1）头文件

```
#include<iostream>
using namespace std;
```

（2）pair <Typename1, Typename2> p;

Typename1,Typename2 分别是为 p 的 first 和 second 的元素类型（可以是 int/double/float 等）

作用：是将两个元素（可以是不同类型的元素）整合为一种

```
pair <Typename1, Typedname2> p;
//两者效果相同
struct point{
    Typename1 first;
    Typedname2 second;
}p;
```

（3）pair 的使用

①可直接对 first 和 second 直接赋值

```
pair <int ,double> p;
p.first=2;
p.second=3.14;
//一种赋值方法
p=make_pair(2,3.14);
//另一种赋值方法
p={2,3.14};
//又一种赋值方法
```

②可定义成类似于数组结构体(可借助 sort 排序)

```
pair <int ,double> p[n];
sort(p,p+n);//以 first 为第一关键字，second 为第二关键字进行排序（first 相同时对 second 进行排序，否则对 first 进行排序）
```

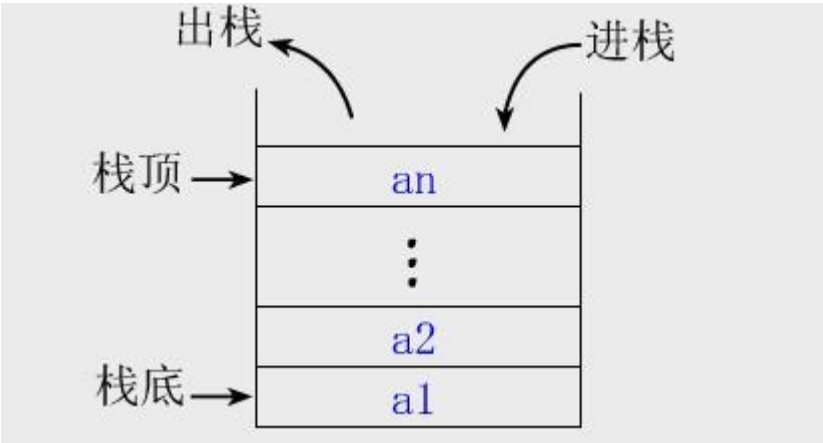
③两 pair 之间的比较（先比较 first，如果 first 相等，再比较 second）

```
pair <int ,int> p1,p2;
p1={1,2},p2={2,2};//p1<p2
p1={1,3},p2={1,2};//p1>p2
p1={2,4},p2={2,4};//p1==p2
```

## 2. 栈

栈的特点：　　1)先进后出(FILO)——first in last out

　　2)在栈顶删除元素，在栈顶加入元素




（1）头文件

```
#include<stack>
using namespace std;
```

（2）stack 的定义

```
stack <typename> name;//typename 可以任何基本类型或者容器,name 是栈的名字
```

（3）stack 的使用

```
stack <int> sta;//stack 的定义
```

①sta.push();

//向栈顶添加元素，并压入栈底，时间复杂度为 0(1)

②sta.top();

//取栈顶元素，返回定义的栈中元素的类型,时间复杂度为 0(1)

③sta.pop();

//将栈顶元素出栈,时间复杂度为 0(1),通常②③步一起使用

④sta.size();

//获取栈中剩余的元素个数，判断栈是否为空,时间复杂度为 0(1)

//如果栈空，等价于 sta.size()==0,否则栈不为空

⑤sta.empty();

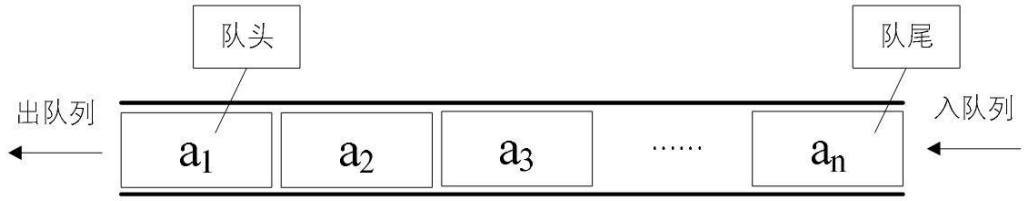
//判断栈是否为空

//sta.empty()的返回值为 1 表示栈无元素，返回值为 0 表示栈还剩有元素,时间复杂度为 0(1)

## 3. 队列

队列的特点：　　1)先进先出(FIFO)——first in first out

2)在队头删除元素，在对尾加入元素




（1）头文件

```
#include<queue>
using namespace std;
```

（2）queue 的定义

```
queue <typename> name;//typename 可以任何基本类型或者容器,name 是队列的名字
```

（3）queue 的使用

```
queue <int> q;//queue 的定义
```

①q.push();

//向队列尾添加一个元素,时间复杂度为 O(1)

②q.pop();

//将队列中的头元素删除,时间复杂度为 O(1)

//通常②③步一起使用

③q.front();

//获取队列的头元素,时间复杂度为 O(1)

④q.back();

//获取队列尾的元素,时间复杂度为 O(1)

⑤q.empty();

//判断一个队列是否为空,时间复杂度为 O(1)

//q.empty()的返回值为 1 表示队列无元素，返回值为 0 表示队列还剩有元素

⑥q.size();

//返回队列中所剩下的元素,时间复杂度为 O(1)

## 4. vector

vector 直接翻译为“向量”，一般说成“变长（动态）数组”，即“长度根据需要而自动改变的数组”；

（1）头文件

```
#include <vector>
using namespace std;
```

（2）vector 的定义

```
    vector <typename> name;
    //定义一个一维数组 name[size],size 不确定,其长度可以根据需要而变化
    //typename 可以是任何基本类型,例如 int ,double ,char ,结构体,也可以是 vector
    vector <int> a;//相当于一个长度动态变化的 int 数组
    vector <double> b[233];//相当于第一维长 233,第二维长度动态变化的 double 数组
    struct rec{...};
    vector <struct rec> c;//自定义的结构体类型保存在 vector 中
    vector <vector<char> > d;//相当于一个一维和二维长度都是动态变化的 char 数组
```

（3）vector 的访问—— 一般通过下标访问或迭代器访问（迭代器访问不讲）

对于容器 vector<int> v，可以使用 v[index]来访问它的第 index 个元素。其中，0≤index≤v.size()-1，v.size()表示 vector 中元素的个数

（4）vector 的使用

```
vector <int> a;//vector 的定义
```

a.size()/a.empty()

*size*函数返回*vector*的实际长度(包含的元素个数),*empty*函数返回一个*bool*类型,表明*vector*是否为空(为空,返回*True*,反之返回*False*),二者的复杂度都是$O(1)$

a.clear()　　*clear*函数把*vector*清空

a.begin()/a.end()

*begin*函数返回指向*vector*中第一个元素的迭代器,例如$a$是一个非空的*vector*,则$*a.begin()$与$a[0]$的作用相同。
所有容器都可以视作一个“前闭后开”的结构,*end*函数返回*vector*的尾部,即第$n$个元素再往后的“边界”。$*a.end()$与$a[n]$都是越界访问,其中$n=a.size()$。

a.front()/a.back()

*front*函数返回*vector*的第一个元素,等价于$*a.begin()$和$a[0]$
*back*函数返回*vector*的最后一个元素,等价于$*--a.end()$和$a[asize()-1]$

a.push_back()/a.pop_back()

$a.push_back(x)$表示把元素$x$插到*vector* $a$的尾部
$a.pop_back()$删除*vector* $a$的最后一个元素