# C++常用数据结构简介

<https://www.cnblogs.com/woxinfeixiang2015/p/10373674.html>

简介

一、数组

　　1. 静态数组  
　　　　array

　　2. 动态数组  
　　　　2.1. vector  
　　　　2.2. priority\_queue  
　　　　2.3. deque  
　　　　2.4. stack  
　　　　2.5. queue  
二、单向链表  
　　forward\_list

三、双向链表  
　　list

四、树  
　　1. set  
　　2. multiset  
　　3. map  
　　4. multimap

五、映射  
　　1. unordered\_set  
　　2. unordered\_multiset  
　　3. unordered\_map  
　　4. unordered\_multimap

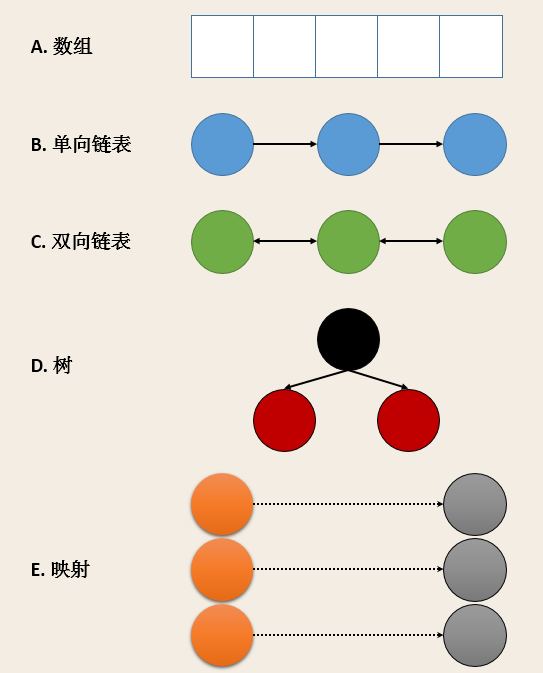
[复制代码](javascript:void(0);)

## 简介

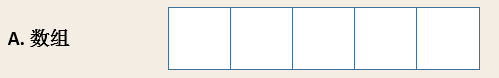
程序员的世界里有一个经典的公式： 数据结构+算法=程序。

所以数据结构及算法的重要性就不用在此赘述了，下面直接进入正题。

在物理层面有以下五种常见的数据结构：



## 一、数组



### 1. 静态数组

在编译期确定数组大小，在运行期无法改变数组大小，所以称之为静态数组。

**C++ 中的 array 由这种结构实现**

[复制代码](javascript:void(0);)

int main ()

{

array<int, 10> a = {0,1,2,3,4,5,6,7,8,9};

a[0] = 22;

cout<<a.at(0)<<endl;

cout<<a.back()<<endl;

return 0;

}

[复制代码](javascript:void(0);)

### 2. 动态数组

在运行期可动态改变数组大小，所以称之为动态数组。C++ 中的动态数组有两个，分别是 vector 和 deque。

#### 2.1. vector

矢量，只能在末尾增删元素

数组大小的增长策略：每次增加的长度为原来的1倍（有些编译器增加0.5倍）。

这样可以保证增加元素的平均时间复杂度为O(1)。

[复制代码](javascript:void(0);)

int main ()

{

vector<int> vv = {1,2,3,4};

vv.push\_back(12);// 在末尾添加元素

vv.pop\_back();// 在末尾删除元素

vv.at(3);// 读取第三个元素

vv[3];// 读写第三个元素

vv.insert(vv.begin()+3, 12);// 将元素插到第三个位置上

return 0;

}

[复制代码](javascript:void(0);)

#### 2.2. priority\_queue

优先队列，默认由 vector 实现，也可由 deque 实现。它保证顶部元素始终是最大值，可用于实现**堆排序。**

[复制代码](javascript:void(0);)

int main ()

{

priority\_queue<int> pp;

pp.push(12);

pp.push(10);

pp.push(11);

pp.top();// 读取顶部元素

pp.pop();// 弹出顶部元素

return 0;

}

[复制代码](javascript:void(0);)

#### 2.3. deque

双端队列，可以在开头或末尾增删元素

[复制代码](javascript:void(0);)

int main ()

{

deque<int> dd;

dd.push\_front(12);// 在开头添加元素

dd.push\_back(10);// 在结尾添加元素

dd.insert(dd.begin()+1, 3);// 在位置1插入元素

dd.front();// 读取开头元素

dd.back();// 读取结尾元素

dd[1]; // 读取第一个元素

dd.pop\_front();// 弹出开头元素

dd.pop\_back();// 弹出末尾元素

return 0;

}

[复制代码](javascript:void(0);)

#### 2.4. stack

栈，默认由 deque 实现，也可由 list 或 vector 实现。是一种**先进后出**的数据结构

[复制代码](javascript:void(0);)

int main ()

{

stack<int> ss;

ss.push(12);// 添加元素

ss.top();// 读取栈顶元素

ss.pop();// 弹出栈顶元素

return 0;

}

[复制代码](javascript:void(0);)

#### 2.5. queue

队列，默认由 deque实现，也可由 list 实现。是一种**先进先出**的数据结构

[复制代码](javascript:void(0);)

int main ()

{

queue<int> qq;

qq.push(12);// 添加元素

qq.front();// 读取队首元素

qq.pop();// 弹出队首元素

return 0;

}

[复制代码](javascript:void(0);)

## 二、单向链表



**forward\_list**

只能从头到尾顺序遍历，不能逆序遍历，即没有 rbegin() 接口

[复制代码](javascript:void(0);)

int main ()

{

forward\_list<int> fl;

fl.push\_front(12);// 在开头添加元素

fl.insert\_after(fl.begin(), 11);

fl.pop\_front();// 在开头删除元素

fl.remove(11);// 删除元素

return 0;

}

[复制代码](javascript:void(0);)

## 三、双向链表



list

[复制代码](javascript:void(0);)

int main ()

{

list<int> ll;

ll.push\_back(12);// 在末尾添加元素

ll.push\_front(10);// 在开头添加元素

ll.back();// 读取末尾元素

ll.front();// 读取开头元素

ll.push\_back(12);

ll.unique();// 删除重复元素

cout<<ll.size()<<endl;

ll.pop\_front();// 在末尾删除元素

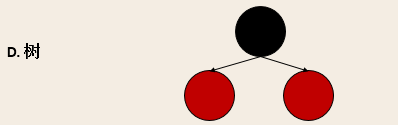
ll.pop\_back();// 在开头删除元素

return 0;

}

[复制代码](javascript:void(0);)

## 四、树



常见的树有二叉树、二叉搜索树、二叉平衡树、红黑树等。

C++ 中的 set multiset map multimap 是用二叉搜索树实现的，这种数据结构支持二分搜索，所以增删改查的复杂度都是O(logn)。

### 1. set

类似数学中的集合，set 中不能包含重复的元素，元素是排好序的，且不能被修改。

[复制代码](javascript:void(0);)

int main ()

{

set<int, less<int>> ss;// 由小到大排序

ss.insert(12);

ss.insert(10);

for(auto itr=ss.cbegin(); itr!=ss.cend(); itr++)cout<<\*itr<<endl;// 输出 10 12

ss.erase(ss.cbegin());// 擦除首元素

ss.count(13);// 元素 13 的个数，0 或 1

ss.find(10);// 查找元素 12，返回迭代器，若没找到返回 ss.end()

return 0;

}

[复制代码](javascript:void(0);)

### 2. multiset

与 set 类似，但可以包含重复元素

[复制代码](javascript:void(0);)

int main ()

{

multiset<int, less<int>> ms;// 由小到大排序

ms.insert(12);

ms.insert(10);

ms.insert(10);

for(auto itr=ms.cbegin(); itr!=ms.cend(); itr++)cout<<\*itr<<endl;// 输出 10 10 12

cout<<""<<endl;

auto pp = ms.equal\_range(10);

for(auto itr=pp.first; itr!=pp.second; itr++)cout<<\*itr<<endl;// 输出 10 10

ms.lower\_bound(10);// = pp.first

ms.upper\_bound(10);// = pp.second

return 0;

}

[复制代码](javascript:void(0);)

### 3. map

元素由 (key,value) 对组成，接口与 set 类似，在插入与遍历元素时有些区别

[复制代码](javascript:void(0);)

int main ()

{

map<int, int> mm;

mm[1] = 1;// 插入元素 (1,1)

mm.insert(make\_pair(2,2));// 插入元素 (2,2)

for(auto itr=mm.cbegin(); itr!=mm.cend(); itr++)

cout<<"("<<itr->first<<","<<itr->second<<")"<<endl;// 输出 (1,1) (2,2)

return 0;

}

[复制代码](javascript:void(0);)

### 4. multimap

[复制代码](javascript:void(0);)

int main ()

{

multimap<int, int> mm;

mm.insert(make\_pair(1,10));// 插入元素 (1,10)

mm.insert(make\_pair(1,11));// 插入元素 (1,11)

mm.insert(make\_pair(2,2));// 插入元素 (2,2)

for(auto itr=mm.cbegin(); itr!=mm.cend(); itr++)// 遍历所有元素

cout<<"("<<itr->first<<","<<itr->second<<")"<<endl;// 输出 (1,10) (1,11) (2,2)

for(auto itr=mm.lower\_bound(1); itr!=mm.upper\_bound(1); itr++)// 遍历 key=1 的元素

cout<<"("<<itr->first<<","<<itr->second<<")"<<endl;// 输出 (1,10) (1,11)

mm.erase(1);// 删除所有 key=1 的元素

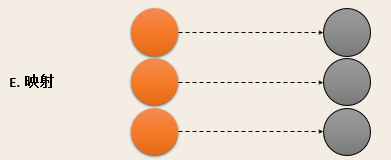
mm.erase(mm.cbegin());// 删除第一个元素

return 0;

}

[复制代码](javascript:void(0);)

## 五、映射



映射类似数学中的函数，每一个 key 对应一个 value，写成函数表达式为：value=f(key)，其中 f 被称为哈希函数。

C++11 中的 unordered\_set unordered\_multiset unordered\_map unordered\_multimap 是用映射实现的，这种数据结构可以在O(1)的时间复杂度下访问单个元素，效率高于二叉搜索树(O(logn))，但是遍历元素的效率比二叉搜索树低。

1. unordered\_set

接口与 set 类似，不在赘述

2. unordered\_multiset

接口与 multiset 类似，不在赘述

3. unordered\_map

接口与 map 类似，不在赘述

4. unordered\_multimap

接口与 multimap 类似，不在赘述