# 📜 ACM新手村秘籍：STL速查手册

各位新同学大家好！

在C语言课上，如果要实现“排序”或“动态数组”，你可能需要写几十行代码。但在ACM竞赛中，我们有**STL (Standard Template Library)**。它就像你的武器库，只需要调用几行代码，就能实现复杂的数据结构。

这周的任务主要就是熟悉它们。以下是必须要掌握的核心武器，请重点关注它们的**特点**和**复杂度**！

## 1. 变长数组：vector

**特点：** 动态数组。再也不用担心数组开小了。

| **操作** | **描述** | **复杂度** |
| --- | --- | --- |
| push\_back(x) | 把x扔到数组最后面 | O(1) (均摊) |
| pop\_back() | 删掉最后一个元素 | O(1) |
| size() | 当前有多少个元素 | O(1) |
| v[i] | 像普通数组一样访问 | O(1) |
| clear() | 清空所有元素 | O(N) |

* **头文件：** #include <vector>
* **注意：** 尽量避免在头部或中间插入/删除元素，因为这会导致所有后续元素移动，效率很低。

## 2. 栈与队列 (受限线性表)

### 🥞 栈 stack (LIFO: 后进先出)

**特点：** 只能在顶部操作，像洗盘子。

* **头文件：** #include <stack>
* **常用操作：** push(x) (入栈), pop() (出栈), top() (查看栈顶)
* **复杂度：** 所有操作均为 O(1)。

### 🛒 队列 queue (FIFO: 先进先出)

**特点：** 只能在队头和队尾操作，像排队。

* **头文件：** #include <queue>
* **常用操作：** push(x) (入队尾), pop() (队头出队), front() (查看队头)
* **复杂度：** 所有操作均为 O(1)。

### 🔄 双端队列 deque (两头都能插/删)

**特点：** 比 vector 更灵活，可以在头部和尾部都进行快速插入和删除。

* **头文件：** #include <deque>
* **特技：** push\_front(x), pop\_front()
* **复杂度：** 头部和尾部的操作都是 O(1)。

## 3. 自动排序神器：priority\_queue (优先队列)

**特点：** 本质是一个堆 (Heap)，最大的（默认）或最小的元素永远在顶端。

* **头文件：** #include <queue>
* **默认 (大根堆)：** 最大的元素优先被取出。
  + 定义: priority\_queue<int> pq;
* **小根堆：** 最小的元素优先被取出。
  + 定义: priority\_queue<int, vector<int>, greater<int>> pq;
* **常用操作：** push(x), pop(), top()
* **复杂度：** 插入和删除都是 $O(\log N)$。

## 4. 查找与去重专家：set 家族 (集合)

### 🌳 标准版：set

**特点：** **有序、不重复**。自动排序 + 自动去重。

* **底层：** 红黑树（平衡二叉搜索树）。
* **复杂度：** 增删查改均为 $O(\log N)$。
* **常用操作：** insert(x), erase(x), count(x) (查x是否存在)

### 🌳 允许重复版：multiset

**特点：** **有序、可重复**。允许插入相同的值，但依然保持有序。

* **复杂度：** 增删查改均为 $O(\log N)$。

### ⚡ 极速版：unordered\_set

**特点：** **无序、不重复**。

* **底层：** 哈希表 (Hash Table)。
* **复杂度：** 平均 $O(1)$，最坏情况 $O(N)$。
* **场景：** 如果不需要排序，只求查询速度最快，就用它。

## 5. 万能索引：map 家族 (映射/字典)

### 🌳 标准版：map<Key, Value>

**特点：** Key（键）是**有序**的，Key **不能重复**。可以实现“字符串到数字”等复杂映射。

* **底层：** 红黑树。
* **复杂度：** 增删查改均为 $O(\log N)$。
* **用法：** 像数组一样使用 mp[Key] = Value;

### 🌳 允许重复版：multimap

**特点：** 一个 Key 可以对应多个 Value。

### ⚡ 极速版：unordered\_map<Key, Value>

**特点：** Key **无序**。

* **底层：** 哈希表。
* **复杂度：** 平均 $O(1)$，最坏情况 $O(N)$。
* **场景：** 统计元素出现频率、大整数映射等不需要顺序的场景。

## 6. 几个像“作弊”一样的算法函数

* **头文件：** #include <algorithm>

| **函数** | **描述** | **复杂度** |
| --- | --- | --- |
| sort(begin, end) | 对指定范围内的元素进行排序 | $O(N \log N)$ |
| next\_permutation(begin, end) | 将当前排列转换为字典序的下一个排列 | $O(N)$ |
| unique(begin, end) | **去重**（使用前必须先 sort） | $O(N)$ |

给新生的建议：

STL 是最强大的工具，可以帮你把代码量减少到最少。请记住，学好 STL 库函数，有时候就像开了挂一样！

**加油，去 AC 吧！**