一、STL 模板

什么是 STL 模板（Standard Template Library）？

STL（标准模板库）是 C++里的一套预先写好的工具箱（库），里面包含很多常用函数和算法，帮你快速解决问题，不用从头写代码。

STL 就像玩积木的工具箱，里面有不同形状和颜色的积木，随时拿来用，快速搭建你想要的东西。

（一）常用函数

1. 输入输出优化（竞赛必备）

(1) 关闭同步流（提高 cin/cout 速度）

```
ios::sync_with_stdio(false);
cin.tie(nullptr);
cout.tie(nullptr);
```

- 作用：关闭 C++与 C 标准库的同步，加快输入输出速度。
- 注意：关闭同步后禁止混用 cin/cout 与 scanf/printf。
- 例子：在数据量超过 1e5 时使用。，大数组（>1e6）必须用 scanf/printf 或关闭同步的 cin/cout 。

(2) 文件重定向

```
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
```

- 注意：竞赛中常用，但需确保文件路径正确。

程序示例

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

```
    // 关闭同步流（速度提升 3-5 倍）
    ios::sync_with_stdio(false);
    cin.tie(0);  // 解除 cin 与 cout 的绑定
    cout.tie(0);

    // 手动设置缓冲区（可选）
    char buf[1<<20];
    cin.rdbuf()->pubsetbuf(buf, sizeof buf);
    cout.rdbuf()->pubsetbuf(buf, sizeof buf);

    int n;
    cin >> n;  // 高速读取
    cout << n; // 高速输出
}
```

2. 数学函数（<cmath>）

函数	功能	示例	易错点
pow(base, exp)	幂运算	pow(2, 3) = 8.0	返回 double 类型
sqrt(x)	平方根	sqrt(9) = 3.0	负数返回 NaN
round(x)	四舍五入	round(2.5) = 3.0	C++11 标准
__gcd(a,b)	最大公约数	__gcd(12,18)=6	非标准函数（GCC）
abs(x)	绝对值	abs(-5)=5	整型用 abs，浮点用 fabs

3. 字符串处理（<string>）

```
string s = "CSP-J Contest";
// 查找子串（返回位置索引）
size_t pos = s.find("J");   // pos=4
// 截取子串（从索引 4 截取 2 个字符）
string sub = s.substr(4, 2);   // "J"
// 替换（从索引 0 替换 3 个字符）
s.replace(0, 3, "NOI");   // "NOI-J Contest"
// 字符串转数字
int num = stoi("123");          // 123
double d = stod("3.14");       // 3.14
// 数字转字符串
string num_str = to_string(2023);
```

4. 算法（<algorithm> ）

函数原型	功能说明	示例
min(a, b)	返回两个值中的较小值	min(3, 5) → 3
max(a, b)	返回两个值中的较大值	max(3, 5) → 5
swap(a, b)	交换两个变量的值	swap(x, y)（x=1,y=2 → x=2,y=1）
sort(begin, end)	对区间升序排序	sort(v.begin(), v.end())
binary_search(begin, end, val)	检查有序区间是否包含某个值	binary_search(v.begin(), v.end(), 42) → true/false
lower_bound(begin, end, val)	返回第一个 ≥ val 的元素的迭代器	lower_bound(v.begin(), v.end(), 5)
upper_bound(begin, end, val)	返回第一个 > val 的元素的迭代器	upper_bound(v.begin(), v.end(), 5)

reverse(begin, end)	反转区间元素顺序	reverse(s.begin(), s.end()) ("abc" → "cba")
unique(begin, end)	移除相邻重复元素 (返回去重后的尾迭代器)	unique(v.begin(), v.end()) (需先排序)
accumulate(begin, end, init)	累加区间值 (需 <numeric>)	

(二) STL 容器 (STL Containers)

1. 什么是容器 (Container) ?

容器是 C++标准模板库(STL)中用来存储一组数据的“盒子”或“仓库”。不同容器管理数据的方式不同，适合不同需求。

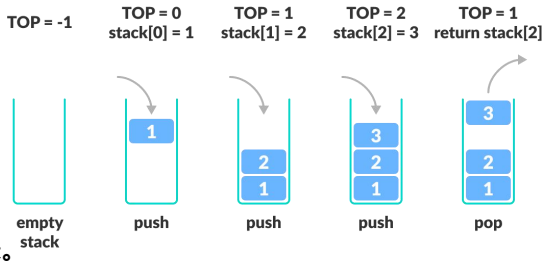
想象你有很多玩具，不同的箱子可以帮你装玩具、拿玩具、整理玩具，箱子设计不同，使用方法也不同。

2. 常用容器讲解

(1) 栈 (Stack)

栈是一种“后进先出” (Last-In, First-Out, LIFO) 的容器。

- 你往栈里放东西叫压栈 (push)。
- 你从栈里拿东西叫弹栈 (pop)。
- 最后放进去的第一个拿出来。
- 像一叠叠书，最后放上去的书先被拿下来。



C++ 示例:

```
#include <iostream>
#include <stack>
using namespace std;
int main() {
    stack<int> s;
    s.push(10);   // 压入 10
```

```
    s.push(20);  // 压入 20
    cout << "栈顶元素是：" << s.top() << endl; // 输出 20
    s.pop(); // 弹出 20
    cout << "弹出后栈顶是：" << s.top() << endl; // 输出 10
}
```

考试注意事项 & 易错点：

易错点	正确说明
忘了 #include <stack>	使用前必须包含头文件
pop() 之后忘记 top()	弹栈后获取栈顶元素
弹空栈导致程序崩溃	弹栈前要用 empty() 判断是否为空

（2）队列（Queue）

队列是“先进先出”（First-In, First-Out, FIFO）的容器。

- 进队叫入队（push）。
- 出队叫出队（pop）。
- 先进去的先出来。
- 像排队买糖果，先排的人先买到。

C++ 示例：

```
#include <iostream>
#include <queue>
using namespace std;
int main() {
    queue<int> q;
    q.push(10); // 入队 10
    q.push(20); // 入队 20
```

```
    cout << "队头是：" << q.front() << endl; // 输出 10
    q.pop();  // 出队 10
    cout << "出队后队头是：" << q.front() << endl; // 输出 20
}
```

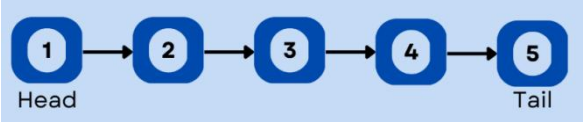
考试注意事项 & 易错点：

易错点	正确说明
忘写头文件	需包含 <queue>
错把 front() 写成 top()	队列用 front() 访问队头元素
出队前不检查空队列	用 empty() 判断是否为空

（3）链表（List）

链表是一串用“指针”链接起来的节点，每个节点有数据和指向下一个节点的地址。

- 支持快速插入和删除。
- STL 中用 list 容器表示双向链表。
- 像一串火车车厢，每节车厢都知道下一个车厢在哪。



C++ 示例：

```
#include <iostream>
#include <list>
using namespace std;
int main() {
    list<int> l = {10, 20, 30};
    l.push_back(40);  // 尾部添加
    l.push_front(5);  // 头部添加
    for (int x : l) {
        cout << x << " ";  // 输出：5 10 20 30 40
```

```
    }
}
```

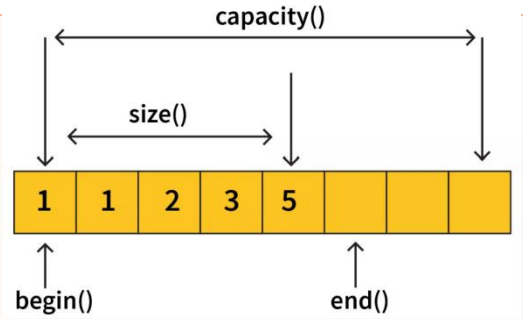
考试注意事项 & 易错点：

易错点	正确说明
忘包含头文件	需要 #include <list>
插入方法混淆	头部插入用 push_front，尾部用 push_back
链表访问慢	链表不能用下标访问，只能遍历

（4）向量（动态数组）（Vector）

- 向量（Vector）是动态大小的数组，可以自动调整大小，支持快速随机访问。
- 像一个能自动变大变小的盒子，能装更多玩具。

C++ 示例：




```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> v = {10, 20, 30};
    v.push_back(40);  // 添加元素
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";  // 输出：10 20 30 40
    }
}
```

考试注意事项 & 易错点：

易错点	正确说明

忘写头文件	需包含 <vector>
用 [] 访问越界	访问时要保证 i < v.size()
不懂大小动态变化	push_back 会自动扩容

动态数组的高频操作

```
vector<int> v = {5, 3, 7};
// 排序（升序）
sort(v.begin(), v.end());   // {3,5,7}
// 二分查找（要求有序）
auto it = lower_bound(v.begin(), v.end(), 5);   // 首个>=5 的位置
// 删除重复元素（先排序）
sort(v.begin(), v.end());
v.erase(unique(v.begin(), v.end()), v.end());
v.push_back(x);          // 末尾添加
v.pop_back();            // 末尾删除
v.size();                // 大小
v.resize(n);             // 调整大小
```

（5）集合（set）

```
set<int> s = {3, 1, 4};
// 查找元素（O(log n)）
if(s.find(4) != s.end()) { /* 存在 */ }
// 获取最小/最大值
int min_val = *s.begin();
int max_val = *s.rbegin();
s.insert(x);             // 插入
```

```
s.erase(x);              // 删除
```

(6) 映射（map）

```
map<string, int> score;
score["Alice"] = 95;
// 安全访问（避免自动插入）
if(score.count("Bob")) {
    cout << score.at("Bob");
}
// 遍历（按 key 排序）
for(auto &[name, sc] : score) {
    cout << name << ": " << sc << endl;
}
```