标准库函数

```
#include <bits/stdc++.h>
```

1. 输入输出函数

- <iostream>：核心输入输出流库
  - cin >> var：读取输入数据
  - cout << var：输出数据
  - endl：输出换行并刷新缓冲区
- <iomanip>：格式化输出
  - fixed：固定小数位数显示
  - setprecision(n)：设置小数精度

```
double pi = 3.1415926;
cout << fixed << setprecision(2) << pi;  // 输出 3.14
```

2. 字符串处理

- <string>：字符串操作
  - string::substr(pos, len)：提取子串
  - string::find(str)：查找子串位置
  - stoi(str) / stod(str)：字符串转数值
- 字符处理函数（需 <cctype>）
  - islower(c) / isupper(c)：判断小写/大写字母
  - tolower(c) / toupper(c)：大小写转换

```
char c = 'A';
cout << (char)tolower(c);  // 输出 'a'
```

3. 容器操作

- `<vector>` / `<deque>` / `<list>`：动态数组与链表
  - `push_back(val)`：尾部插入元素
  - `pop_back()`：删除尾部元素
- `<map>` / `<set>`：关联容器
  - `insert({key, val})`：插入键值对
  - `find(key)`：查找元素
- `<queue>` / `<stack>`：队列与栈
  - `push(val)` / `pop()`：入队（栈）与出队（栈）

4. 算法函数

- `<algorithm>`：核心算法库
  - `sort(begin, end)`：排序（默认升序）

    ```
    vector<int> v = {5, 3, 1};
    sort(v.begin(), v.end());  // 结果 {1, 3, 5}
    ```

  - `reverse(begin, end)`：翻转序列
  - `max(a, b)` / `min(a, b)`：返回极值
  - `swap(a, b)`：交换值
  - `fill(begin, end, val)`：填充值
  - `unique(begin, end)`：去重相邻重复元素
- 二分查找
  - `binary_search(begin, end, val)`：判断是否存在
  - `lower_bound(begin, end, val)`：返回首个不小于值的迭代器

5. 数学函数

- `<cmath>`：数学运算
  - `sqrt(x)`：平方根
  - `pow(x, y)`：x的y次方
  - `sin(x)` / `cos(x)` / `tan(x)`：三角函数

- 数值处理
  - `abs(x)`：绝对值
  - `ceil(x)` / `floor(x)`：向上/向下取整

6. 其他实用函数

- 内存操作（`<cstring>`）
  - `memset(ptr, val, size)`：填充内存块

```
int arr[5];
memset(arr, 0, sizeof(arr));  // 全初始化为0
```

- 全排列（需 `<algorithm>`）
  - `next_permutation(begin, end)`：生成下一个排列

```
string s = "123";
do { cout << s << " "; } while (next_permutation(s.begin(), s.end()));
// 输出 "123 132 213 231 312 321"
```

注意事项

1. 优点：快速开发，减少头文件包含；
2. 缺点：编译时间增加、非标准（仅支持 GCC 等编译器）；
3. 适用场景：竞赛编程或快速原型开发，生产环境建议明确包含具体头文件。

示例代码（综合使用多个函数）

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

int main() {
    vector<int> v = {5, 2, 8, 1};
    sort(v.begin(), v.end());                    // 排序
    reverse(v.begin(), v.end());                 // 翻转
    cout << "Max: " << *max_element(v.begin(), v.end()) << endl; // 最大值

    string s = "Hello";
    transform(s.begin(), s.end(), s.begin(), ::toupper); // 转大写
    cout << s << endl;                           // 输出 "HELLO"
    return 0;
}
```