108个CSP-J/S必备关键词

序号	分类	英语单词	中文翻译	数据范围/函数说明	代码中的具体使用
一、数据类型					
1	基础类型（整数）	int	整数	4字节，范围：-2^{31}（-2147483648）～$2^{31}-1$（2147483647）	int a = 10; // 定义整数变量a（取值在int范围内）
2	基础类型（整数）	long long	长整数	8字节，范围：-2^{63} ~ $2^{63}-1$（避免大数溢出）	long long b = 1e18; // 定义长整型变量（10^{18}在long long范围内）
3	基础类型（短整数）	short	短整数	2字节，范围：-32768 ~ 32767	short s = 32767; // 短整型最大值
4	基础类型（无符号整数）	unsigned int	无符号整数	4字节，范围：0 ~ $2^{32}-1$（仅非负数）	unsigned int u = 4294967295; // 无符号int最大值
5	基础类型（无符号短整数）	unsigned short	无符号短整数	2字节，范围：0 ~ 65535	unsigned short us = 65535; // 无符号short最大值
6	基础类型（字符）	char	字符	1字节，范围：-128 ~ 127（有符号）或0 ~ 255（无符号，需unsigned	char c = 'A'; // 定义字符变量c（ASCII码值65）
7	基础类型（布尔）	bool	布尔值	1字节，仅取值：true（真）或false（假）	bool flag = (a > 0); // 条件表达式结果赋值给布尔变量
8	浮点类型（单精度）	float	单精度浮点数	4字节，范围：$\pm3.4\times10^{-45}$ ~ $\pm3.4\times10^{38}$（精度约6-7位有效数字）	float f = 3.14f; // 定义单精度浮点数（末尾加f避免精度丢失）
9	浮点类型（双精度）	double	双精度浮点数	8字节，范围：$\pm1.7\times10^{-308}$ ~ $\pm1.7\times10^{308}$（精度约15-17位有效数字，默认	double d = 3.1415926; // 定义双精度浮点数（更精确）
10	字符串类型	string	字符串	依赖实现（通常动态分配），存储字符序列（需#include <string>）	string s = "abc"; // 定义字符串（长度3）
11	结构体类型	struct	结构体	自定义复合数据类型（可包含多个不同类型成员）	struct Point { int x; int y; }; Point p = {1, 2}; // 定义结构体变量p
二、控制结构					
12	条件判断	if	如果	单分支条件判断	if (a > 0) { cout << "正数"; }
13	条件判断	else	否则	双分支条件判断	if (a % 2 == 0) { ... } else { ... }
14	多分支判断	switch	开关（多分支选择）	多分支选择（需break终止分支）	switch (day) { case 1: cout << "周一"; break; case 2: ... }
15	分支情况	case	情况（switch子句）	switch中定义具体分支值	同上
16	分支默认	default	默认（switch子句）	未匹配case时执行的代码块	同上
17	循环	for	循环（迭代结构）	已知循环次数时使用	for (int i=0; i<5; i++) { sum += i; }
18	循环	while	当...时循环	先判断条件再执行循环体	int i=0; while (i < 5) { cout << i; i++; } // 输出0-4
19	循环	do	执行（先执行后判断）	至少执行一次循环体	do { sum++; } while (sum < 10); // 至少执行一次循环体
20	循环控制	break	中断（退出循环）	终止当前循环	for (int i=0; i<10; i++) { if (i==5) break; } // 循环到i=5时退出
21	循环控制	continue	继续（跳过当前迭代）	跳过当前迭代，进入下一次循环	for (int i=0; i<10; i++) { if (i%2==0) continue; sum++; } // 跳过偶数，只
22	循环控制	goto	跳转（标签跳转）	跳转到指定标签位置（不推荐滥用）	goto end; // 跳转到end标签 end: cout << "结束";
三、变量与常量					
23	变量	variable	变量	存储可变化数据的容器	int count = 0; // 变量count存储计数
24	常量	constant	常量	不可修改的固定值	const int MAX = 100; // 常量MAX不可重新赋值
25	静态变量	static	静态变量	作用域内持久化存储（仅在首次调用时初始化）	void func() { static int cnt = 0; cnt++; } // cnt在多次调用中保留值
26	外部变量	extern	外部变量	声明其他文件中定义的变量（需配合#include）	extern int global; // 声明全局变量global（定义在其他文件）
27	寄存器变量	register	寄存器变量	建议编译器将变量存储在寄存器中（加速访问，现代编译器自动优化）	register int i; // 建议i存储在寄存器中
四、输入输出					
28	输入流对象	cin	标准输入流	C++风格输入（自动类型推导，需#include <iostream>）	cin >> a >> b; // 读取两个整数到a和b

29	输出流对象	cout	标准输出流	C++风格输出（支持<<运算符拼接）	cout << "Hello" << " World"; // 输出"Hello World"
30	错误流对象	cerr	标准错误流	输出错误信息（无缓冲，直接显示）	cerr << "输入错误！"; // 输出错误信息
31	日志流对象	clog	标准日志流	输出日志信息（有缓冲，适合调试）	clog << "程序启动"; // 输出日志
32	格式化输入函数	scanf	格式化输入	从标准输入读取数据（需#include <cstdio>，需取地址符&）	int a; scanf("%d", &a); // 读取整数到a
33	格式化输出函数	printf	格式化输出	向标准输出写入数据（支持格式符如%d、%f、%s）	printf("Sum: %d", sum);// 输出整数sum并换行
34	输出宽度设置	setw	设置输出宽度	控制输出字段宽度（需#include <iomanip>）	cout << setw(5) << a; // 输出a占5个字符宽度（右对齐）
35	浮点精度设置	setprecision	设置浮点数精度	控制浮点数输出的有效位数（需#include <iomanip>）	cout << setprecision(3) << 3.1415; // 输出3.14（保留3位有效数字）
36	固定小数位输出	fixed	固定小数位	强制浮点数以固定小数位输出（如3.14而非3.14159）	cout << fixed << setprecision(2) << 3.1415; // 输出3.14(2位小数)
五、运算符					
37	算术运算符	+	加	整数/浮点数加法	sum = a + b;
38	算术运算符	-	减	整数/浮点数减法	diff = a - b;
39	算术运算符	*	乘	整数/浮点数乘法	product = a * b;
40	算术运算符	/	除	整数除法取整（int/long long），浮点数精确除法（float/double）	quotient = a / b; // 整数除法（如5/2=2）
41	算术运算符	%	取模（求余）	仅适用于整数（int/long long），结果符号与被除数一致	remainder = a % b; // 计算a除以b的余数（b≠0）
42	自增/自减	++	自增1	i++（后置，先使用后自增）；++i（前置，先自增后使用）	i++; // 等价于i = i + 1
43	自增/自减	--	自减1	j--（后置）；--j（前置）	j--; // 等价于j = j - 1
44	赋值运算符	=	赋值	将右侧值赋给左侧变量	a = 10; // 将10赋值给变量a
45	复合赋值	+=	加等于	等价于a = a + b	sum += i; // 等价于sum = sum + i
46	复合赋值	-=	减等于	等价于a = a - b	diff -= i; // 等价于diff = diff - i
47	复合赋值	*=	乘等于	等价于a = a * b	product *= i; // 等价于product = product * i
48	复合赋值	/=	除等于	等价于a = a / b	quotient /= i; // 等价于quotient = quotient / i
49	逻辑运算符	&&	逻辑与	两个条件均成立时为真	if (a > 0 && b < 0) { ... }
50	逻辑运算符	||	逻辑或	两个条件均不成立时为假	if (a > 0 || b < 0) { ... }
51	逻辑运算符	!	逻辑非	条件不成立时为真	if (!flag) { ... } // flag为false时成立
52	关系运算符	==	等于	两个值相等时为真	if (a == b) { ... }
53	关系运算符	!=	不等于	两个值不等时为真	if (a != b) { ... }
54	关系运算符	>	大于	左侧值大于右侧时为真	if (a > b) { ... }
55	关系运算符	<	小于	左侧值小于右侧时为真	if (a < b) { ... }
56	关系运算符	>=	大于等于	左侧值大于等于右侧时为真	if (a >= b) { ... }
57	关系运算符	<=	小于等于	左侧值小于等于右侧时为真	if (a <= b) { ... }
六、函数					
58	函数	function	函数	可复用的代码块（需定义返回值、参数和函数体）	int add(int x, int y) { return x + y; } // 定义加法函数
59	返回值	return	返回	终止函数并返回结果值	同上
60	形参	parameter	参数（函数定义时）	函数定义中声明的参数（仅声明，不分配内存）	int add(int x, int y) // x和y为形参
61	实参	argument	参数（函数调用时）	函数调用时传递的实际值（分配内存并参与计算）	add(3, 5) // 3和5为实参

62	局部变量	local variable	局部变量	函数内部定义的变量（仅在函数内有效）	int func() { int a = 10; } // a为局部变量（仅在func内有效）
63	全局变量	global variable	全局变量	程序全局作用域定义的变量（所有函数均可访问）	int global = 5; // 全局变量（作用域为整个程序）
64	递归函数	recursive	递归（函数调用自身）	函数在执行过程中调用自身（需终止条件避免无限递归）	int factorial(int n) { if (n==0) return 1; else return n * factorial(n-1); }
七、数学函数（cmath库）				需#include <cmath>，用于数值计算	
65	绝对值函数	abs	绝对值	整数：返回x的绝对值（int/long long）；浮点数：fabs(x)	int x = -5; cout << abs(x); // 输出5
66	平方根函数	sqrt	平方根	输入非负浮点数，返回算术平方根（double类型）	double y = sqrt(16.0); // 输出4.0
67	幂函数	pow	幂运算	计算x的y次幂（double pow(double x, double y)）	double z = pow(2, 3); // 输出8.0（2^3）
68	向上取整函数	ceil	向上取整	返回不小于x的最小整数（double类型，结果转为整数需强制转换）	double a = 3.2; cout << (int)ceil(a); // 输出4
69	向下取整函数	floor	向下取整	返回不大于x的最大整数（double类型）	double b = 3.8; cout << (int)floor(b); // 输出3
70	四舍五入函数	round	四舍五入	返回最接近x的整数（double类型）	double c = 3.49; cout << round(c); // 输出3
71	浮点取余函数	fmod	浮点取余	计算x除以y的余数（double fmod(double x, double y)）	double d = fmod(7.5, 2.5); // 输出0.0（7.5=3×2.5+0）
72	直角三角形斜边函数	hypot	斜边长度	计算直角三角形的斜边长度（sqrt(x^2+y^2)）	double hyp = hypot(3, 4); // 输出5.0（$\sqrt{3^2+4^2}=5$）
73	自然对数函数	log	自然对数	返回ln(x)（自然对数，x>0）	double e = log(10); // 输出2.302...（ln(10)）
74	常用对数函数	log10	常用对数	返回$\log_{10}(x)$（以10为底的对数，x>0）	double f = log10(100); // 输出2.0（$\log_{10}(100)=2$）
75	指数函数	exp	自然指数	计算e的x次幂（double exp(double x)）	double g = exp(1); // 输出2.718...（e^1）
76	三角函数	sin/cos/tan	正弦/余弦/正切	输入弧度值，返回对应三角函数值（double类型）	double angle = 30; double rad = angle * M_PI / 180; // 弧度转换 cout
77	反三角函数	asin/acos/atan	反正弦/反余弦/反正切	返回对应弧度的角度值（double类型，范围：asin/acos[-π/2, π/2]，atan(-π	double rad = asin(0.5); // 输出π/6（约0.523弧度）
八、算法函数（algorithm库）				需#include <algorithm>，用于数组/容器操作	
78	排序函数	sort	排序	对区间[first, last)内的元素升序排序（默认operator<，可自定义比较函数）	int arr[5] = {3,1,4,2,5}; sort(arr, arr+5); // 排序后arr={1,2,3,4,5}
79	排序函数	sort(自定义)	自定义排序	传入比较函数（如降序）	sort(arr, arr+5, greater<int>()); // 降序排序（需#include <functional>）
80	反转函数	reverse	反转	反转区间[first, last)内的元素顺序	vector<int> v = {1,2,3}; reverse(v.begin(), v.end()); // v={3,2,1}
81	最大值函数	max/min	最大值/最小值	返回两个元素的较大/较小值（支持自定义比较）	int m = max(3, 5); // 输出5
82	交换函数	swap	交换	交换两个变量的值	int a=1, b=2; swap(a,b); // a=2, b=1
83	查找函数	find	查找元素	在区间[first, last)内查找值为val的元素，返回迭代器（未找到返回last）	vector<int> v = {1,3,5}; auto it = find(v.begin(), v.end(), 3); // it指向3
84	统计函数	count	统计次数	统计区间[first, last)内等于val的元素个数	int cnt = count(arr, arr+5, 2); // 统计arr中2的出现次数（假设
85	下界函数	lower_bound	下界（首个≥val的位置）	在有序区间[first, last)内返回首个≥val的元素的迭代器	int arr[5] = {1,2,3,4,5}; auto it = lower_bound(arr, arr+5, 3); // it指向3
86	上界函数	upper_bound	上界（首个>val的位置）	在有序区间[first, last)内返回首个>val的元素的迭代器	auto it = upper_bound(arr, arr+5, 3); // it指向4（首个>3的元素）
87	二分查找	binary_search	二分查找	在有序区间[first, last)内查找val是否存在，返回布尔值	bool exist = binary_search(arr, arr+5, 3); // 存在则返回true
88	最小最大值对	minmax	最小最大值对	返回区间内最小值和最大值的pair（C++11新增）	auto mm = minmax(arr, arr+5); // mm.first=1, mm.second=5
89	打乱顺序	random_shuffle	随机打乱	随机打乱区间[first, last)内的元素顺序（需#include <random>）	random_shuffle(v.begin(), v.end()); // 随机打乱vector v
90	聚合函数	accumulate	累加/聚合	计算区间[first, last)内元素的和（可自定义聚合操作）	int sum = accumulate(arr, arr+5, 0); // 求和（初始值0）
九、预处理与宏				需#开头，编译前处理的指令	
91	预处理指令	#include	包含头文件	引入其他文件的内容（如#include <iostream>）	#include <iostream> // 包含输入输出流头文件
92	预处理指令	#define	宏定义	定义常量或代码片段（文本替换）	#define PI 3.14159 // 定义圆周率常量
93	预处理指令	#ifdef/#ifndef	条件编译	根据宏是否定义选择编译代码	#ifdef DEBUG cout << "调试模式"; #endif // DEBUG定义时输出

94	预处理指令	#pragma	编译器指令	向编译器发送特定指令（如优化、警告控制）	#pragma GCC optimize("O2") // 开启O2优化
十、指针与引用				C++核心概念（CSP-J初赛要求基础理解）	
95	指针	pointer	指针	存储变量内存地址的变量	int a = 10; int* p = &a; // p是指向a的指针（&取地址符）
96	解引用	dereference	解引用	获取指针指向的变量值	cout << *p; // 输出10（*解引用符）
97	引用	reference	引用	变量的别名（必须初始化，不可重新绑定）	int a = 10; int& ref = a; // ref是a的引用（修改ref即修改a）
98	空指针	nullptr	空指针	表示不指向任何对象的指针（C++11替代NULL）	int* p = nullptr; // p不指向任何对象
十一、文件操作				需#include <fstream>，用于读写文件	
99	文件流对象（输入）	ifstream	文件输入流	打开文件进行读取	ifstream fin("input.txt"); // 打开input.txt文件用于读取
100	文件流对象（输出）	ofstream	文件输出流	打开文件进行写入	ofstream fout("output.txt"); // 打开output.txt文件用于写入
101	文件打开模式	ios::in	读模式	以读取方式打开文件（默认）	fin.open("input.txt", ios::in); // 显式指定读模式
102	文件打开模式	ios::out	写模式	以写入方式打开文件（覆盖原有内容）	fout.open("output.txt", ios::out); // 显式指定写模式
103	文件状态检查	is_open	检查文件是否打开成功	返回布尔值表示文件是否成功打开	if (!fin.is_open()) { cerr << "文件打开失败"; }
十二、其他高频概念				初赛常考但易遗漏的细节	
104	类型转换	static_cast	静态类型转换	显式转换类型（安全，用于相关类型转换）	double d = 3.14; int a = static_cast<int>(d); // 截断为3
105	断言	assert	断言（调试辅助）	检查条件是否为真（调试模式下触发错误）	assert(n > 0); // 若n≤0则程序终止并报错
106	命名空间	namespace	命名空间	避免命名冲突（如std命名空间）	using namespace std; // 使用标准命名空间（避免重复写std::）
107	模板	template	模板	定义通用类或函数（CSP-J初赛要求理解概念）	template <typename T> T add(T a, T b) { return a + b; } // 泛型加法函数
108	异常处理	try/catch	异常处理	捕获并处理程序中的错误	try { if (x < 0) throw "负数"; } catch (const char* msg) { cout << msg; }