流程控制

程序由若干条语句组成，程序的执行顺序称为 流程。C++ 提供了三种基本流程结构，以及若干跳转语句来灵活控制程序走向：

- 顺序结构：语句自上而下依次执行。
- 选择结构（分支结构）：根据条件决定执行哪一条分支。
- 循环结构：在满足条件时重复执行某段代码。
- 跳转语句：`break`、`continue`、`goto`、`return`，用于改变流程走向。

一、顺序结构

顺序结构是最基本的流程结构，代码按照书写的先后顺序从上到下逐条执行。

```
#include <iostream>
using namespace std;

int main() {
    int a = 10;
    int b = 20;
    int sum = a + b;          // 先计算
    cout << "sum = " << sum << endl;  // 再输出
    return 0;
}
```

输出：

```
sum = 30
```

二、选择结构

选择结构根据条件表达式的真假，选择性地执行不同的代码块。条件表达式中，**0** 表示假（**false**），非 **0** 表示真（**true**）。

2.1 if 语句

语法:

```
if (条件表达式) {
    // 条件为真时执行的语句
}
```

示例:

```
int score = 85;
if (score >= 60) {
    cout << "及格" << endl;
}
```

2.2 if-else 语句

语法:

```
if (条件表达式) {
    // 条件为真
} else {
    // 条件为假
}
```

示例:

```
int num = 7;
if (num % 2 == 0) {
    cout << num << " 是偶数" << endl;
} else {
    cout << num << " 是奇数" << endl;
}
```

2.3 if-else if-else 多分支

语法:

```
if (条件1) {
    // 条件1成立
} else if (条件2) {
    // 条件2成立
} else if (条件3) {
    // 条件3成立
} else {
    // 以上都不成立
}
```

示例（成绩等级判定）：

```
int score = 85;
if (score >= 90) {
    cout << "优秀" << endl;
} else if (score >= 80) {
    cout << "良好" << endl;
} else if (score >= 60) {
    cout << "及格" << endl;
} else {
    cout << "不及格" << endl;
}
```

> 注意
>
> ：多分支按顺序自上而下判断，一旦某个条件成立，后面的分支将不再判断。因此条件的顺序很重要（例如先判断 `>= 90` 再判断 `>= 80`）。

2.4 嵌套 if

在一个 `if` 内部再写 `if`，即嵌套分支。示例：判断一个数是否为三个数中的最大值。

```
int a = 3, b = 8, c = 5;
if (a >= b) {
    if (a >= c) {
        cout << "最大值是 " << a << endl;
    } else {
        cout << "最大值是 " << c << endl;
    }
```

```
} else {
    if (b >= c) {
        cout << "最大值是 " << b << endl;
    } else {
        cout << "最大值是 " << c << endl;
    }
}
```

> 悬挂 **else** 问题： `else` 总是与它前面最近的、尚未配对的 `if` 配对。建议始终使用花括号 `{}` 明确代码块范围，避免歧义。

2.5 switch 语句

`switch` 适合根据一个整型或枚举表达式的 多个固定取值 进行分支。

语法：

```
switch (表达式) {
    case 常量1:
        // 语句
        break;
    case 常量2:
        // 语句
        break;
    default:
        // 其他情况
        break;
}
```

示例（根据数字输出星期）：

```
int day = 3;
switch (day) {
    case 1: cout << "星期一" << endl; break;
    case 2: cout << "星期二" << endl; break;
    case 3: cout << "星期三" << endl; break;
    case 4: cout << "星期四" << endl; break;
    case 5: cout << "星期五" << endl; break;
```

```
    case 6: cout << "星期六" << endl; break;
    case 7: cout << "星期日" << endl; break;
    default: cout << "非法输入" << endl; break;
}
```

switch 的要点:

- `switch` 的表达式只能是整型（`int`、`char`、`enum` 等）或可隐式转换为整型的类型，不能是浮点数或字符串。
- `case` 后面必须是常量表达式，各 `case` 的值不能重复。
- `break` 的作用是跳出 **switch**。如果省略 `break`，程序会继续执行下一个 `case` 的语句，这称为 **case** 穿透（**fall-through**）。有时可利用穿透实现多个值共享同一段逻辑:

```
char grade = 'B';
switch (grade) {
    case 'A':
    case 'B':
        cout << "良好及以上" << endl;  // A 和 B 都执行这里
        break;
    case 'C':
        cout << "及格" << endl;
        break;
    default:
        cout << "其他" << endl;
}
```

2.6 三目运算符 ?:

三目运算符是一个简洁的条件表达式:

```
结果 = 条件 ? 值1 : 值2;
```

条件为真取 `值1`，否则取 `值2`。示例:

```
int a = 10, b = 20;
int max = (a > b) ? a : b;   // max = 20
cout << "较大值: " << max << endl;
```

if 与 switch 对比

特性	if / else if	switch
判断类型	任意布尔条件、范围判断	整型 /枚举的相等比较
表达式	可以是关系、逻辑表达式	只能是比较具体的值
可读性	范围判断清晰	多固定值分支更清晰
穿透行为	无	有（需 `break` 控制）

三、循环结构

循环结构用于重复执行某段代码，直到条件不再满足。 C++ 有三种循环语句： `for`、`while`、`do-while`。

3.1 for 循环

适合 已知循环次数 的场景。

语法：

```
for (初始化; 循环条件; 迭代更新) {
    // 循环体
}
```

执行流程：初始化 → 判断条件 → 执行循环体 → 迭代更新 → 再判断条件 ……

示例（求 1 到 100 的和）：

```
int sum = 0;
for (int i = 1; i <= 100; i++) {
    sum += i;
}
cout << "1+2+...+100 = " << sum << endl;  // 5050
```

3.2 while 循环

适合 循环次数不确定、依赖条件 的场景。先判断条件，再执行循环体。

语法：

```
while (循环条件) {
    // 循环体
}
```

示例（求一个整数的各位数字之和）：

```
int n = 12345;
int sum = 0;
while (n > 0) {
    sum += n % 10;   // 取个位
    n /= 10;         // 去掉个位
}
cout << "各位之和: " << sum << endl;  // 15
```

3.3 do-while 循环

先执行一次循环体，再判断条件。**无论条件是否成立，循环体至少执行一次。**

语法：

```cpp
do {
    // 循环体
} while (循环条件);   // 注意结尾的分号
```

示例（至少执行一次的输入校验）：

```
int input;
do {
    cout << "请输入一个正整数: ";
    cin >> input;
} while (input <= 0);
cout << "你输入的是: " << input << endl;
```

3.4 范围 for 循环（C++11）

用于遍历数组或容器中的每个元素，语法更简洁。

```
int arr[] = {1, 2, 3, 4, 5};
for (int x : arr) {
```

```
    cout << x << " ";
}
// 输出: 1 2 3 4 5
```

如果需要修改元素，需使用引用 `&`：

```
for (int &x : arr) {
    x *= 2;   // 每个元素翻倍
}
```

3.5 嵌套循环

一个循环内部再包含另一个循环。常用于多维数据处理，如打印图形、矩阵。

示例（打印九九乘法表）：

```
for (int i = 1; i <= 9; i++) {
    for (int j = 1; j <= i; j++) {
        cout << j << "*" << i << "=" << i * j << "\t";
    }
    cout << endl;
}
```

三种循环对比

循环	判断时机	最少执行次数	适用场景
`for`	先判断后执行	0 次	已知循环次数
`while`	先判断后执行	0 次	次数不定、依赖条件
`do-while`	先执行后判断	1 次	至少执行一次

四、跳转语句

4.1 break

- 在 循环 中：立即跳出当前层循环，不再继续。

- 在 **switch** 中: 结束 switch 分支。

示例（查找第一个能被 7 整除的数）：

```
for (int i = 1; i <= 100; i++) {
    if (i % 7 == 0) {
        cout << "第一个能被 7 整除的数: " << i << endl;  // 7
        break;   // 找到后立即跳出循环
    }
}
```

> `break` 只能跳出 当前一层 循环；嵌套循环中需配合其他手段跳出多层。

4.2 continue

跳过本次循环中 `continue` 之后的语句，直接进入下一次循环的条件判断。

示例（输出 1~10 中的奇数）：

```
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0) {
        continue;   // 偶数跳过
    }
    cout << i << " ";
}
// 输出: 1 3 5 7 9
```

4.3 goto

无条件跳转到同一函数内的某个标号处。 现代 **C++** 中一般不推荐使用，会破坏程序结构、降低可读性，仅在某些跳出多层循环的场景偶尔使用。

语法：

```
goto 标号;
...
标号:
    // 语句
```

示例：

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        if (i * j == 12) {
            cout << "找到: i=" << i << ", j=" << j << endl;
            goto end;   // 直接跳出两层循环
        }
    }
}
end:
    cout << "循环结束" << endl;
```

4.4 return

结束当前函数的执行，并将结果（若有）返回给调用者。在 `main` 函数中执行 `return 0;` 表示程序正常结束。

```
int add(int a, int b) {
    return a + b;   // 返回后函数立即结束
}
```

五、综合实例

实例 1：判断是否为素数

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    bool isPrime = (n > 1);
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            isPrime = false;
```

```
            break;
        }
    }
    cout << n << (isPrime ? " 是素数" : " 不是素数") << endl;
    return 0;
}
```

实例 2：简单的菜单选择（switch + 循环）

```
#include <iostream>
using namespace std;

int main() {
    int choice;
    do {
        cout << "\n=== 菜单 ===" << endl;
        cout << "1. 打招呼" << endl;
        cout << "2. 显示版本" << endl;
        cout << "0. 退出" << endl;
        cout << "请选择: ";
        cin >> choice;

        switch (choice) {
            case 1: cout << "你好，欢迎学习 C++！" << endl; break;
            case 2: cout << "版本 1.0" << endl; break;
            case 0: cout << "再见！" << endl; break;
            default: cout << "无效选项，请重新输入" << endl; break;
        }
    } while (choice != 0);
    return 0;
}
```

六、注意事项与最佳实践

1. 条件表达式：C++ 中 `0` 为假，非 `0` 为真；但判断相等时务必用 `==`，不要误写成赋值 `=`。
2. 浮点数比较：不要直接用 `==` 比较两个 `float`/`double`，由于精度问题易出错，应判断差值是否足够小，如 `fabs(a - b) < 1e-6`。

3. 避免死循环：`while`/`for` 循环体中要确保循环条件最终会变为假，否则会造成死循环。
4. 花括号习惯：即使循环体或分支只有一条语句，也建议使用 `{}`，可避免悬挂 `else` 等问题，也便于后期修改。
5. **switch** 记得 **break**：除非有意利用 case 穿透，否则每个 `case` 末尾都要写 `break`。
6. 少用 **goto**：优先使用结构化控制语句（`break`、`continue`、函数封装）替代 `goto`。
7. 循环变量作用域：在 `for (int i = ...)` 中声明的 `i` 只在循环内有效，这是推荐写法。

七、小结

结构	关键字	作用
顺序结构	—	按顺序依次执行
选择结构	`if` / `else` / `switch`	根据条件选择分支
循环结构	`for` / `while` / `do-while`	重复执行代码
跳转语句	`break` / `continue` / `goto` / `return	

版本号 #2
由 admin 创建于 2026-11-12 15:06:16 UTC
由 admin 更新于 2026-11-12 15:16:57 UTC