第5課：流程控制語句 - 循環結構

第5課：流程控制語句 - 循環結構

📚 課程目標

- 掌握for循環的基本用法和應用場景
- 理解while循環和do-while循環的區別
- 學會使用break和continue控制循環流程
- 掌握嵌套循環的使用
- 理解無限循環的概念及應用

🎯 本課關鍵字

- `for` - 計數循環
- `while` - 條件循環
- `do` - 後測試循環
- `break` - 跳出循環
- `continue` - 跳過本次循環

📝 範例1：for循環基礎

```
/*
* 範例5-1：for循環基礎
* for循環用於已知循環次數的情況
* 語法：for(初始化; 條件; 更新) { 循環體 }
*/
#include <iostream>
int main() {
std::cout << "=== for循環基礎演示 ===" << std::endl;
// 1. 最基本的for循環：從1數到5
std::cout << "1. 從1數到5：" << std::endl;
for (int i = 1; i <= 5; i++) {
std::cout << "i = " << i << std::endl;
}
// 2. 倒數計時：從5到1
std::cout << "\n2. 倒數計時（5到1）：" << std::endl;
for (int count = 5; count >= 1; count--) {
std::cout << count << "..." << std::endl;
```

```
}
std::cout << "發射！" << std::endl;
// 3. 跳著數：每次增加2
std::cout << "\n3. 偶數（0到10）：" << std::endl;
for (int num = 0; num <= 10; num += 2) {
std::cout << num << " ";
}
std::cout << std::endl;
// 4. 計算1到10的總和
std::cout << "\n4. 計算1到10的總和：" << std::endl;
int sum = 0;
for (int i = 1; i <= 10; i++) {
sum += i; // 累加
std::cout << "加 " << i << "，目前總和：" << sum << std::endl;
}
std::cout << "1到10的總和 = " << sum << std::endl;
// 5. 循環變數作用域（C++11以前）
std::cout << "\n5. 循環變數作用域：" << std::endl;
int i; // 需要在外部聲明，才能在循環後使用
for (i = 1; i <= 3; i++) {
std::cout << "循環內 i = " << i << std::endl;
}
std::cout << "循環後 i = " << i << std::endl;
// 6. 循環變數作用域（C++11及以後）
std::cout << "\n6. C++11 for循環作用域：" << std::endl;
for (int j = 1; j <= 3; j++) {
std::cout << "循環內 j = " << j << std::endl;
}
// std::cout << "循環後 j = " << j << std::endl; // 錯誤！j只在循環內有效
// 7. 複合初始化（多個循環變數）
std::cout << "\n7. 多個循環變數：" << std::endl;
for (int a = 1, b = 10; a <= 5; a++, b--) {
std::cout << "a = " << a << ", b = " << b << std::endl;
}
// 8. for循環的靈活性
std::cout << "\n8. 靈活的for循環：" << std::endl;
int x = 0;
for (; x < 5; ) { // 省略初始化和更新
std::cout << "x = " << x << std::endl;
x++; // 在循環體內更新
}
```

```
return 0;
}
```

📝 範例2：while循環

```
/*
* 範例5-2：while循環
* while循環用於循環次數未知，但循環條件明確的情況
* 語法：while(條件) { 循環體 }
*/
#include <iostream>
int main() {
std::cout << "=== while循環演示 ===" << std::endl;
// 1. 基本的while循環
std::cout << "1. 從1數到5：" << std::endl;
int i = 1; // 初始化
while (i <= 5) { // 條件檢查
std::cout << "i = " << i << std::endl;
i++; // 更新
}
// 2. 計算1到100的總和
std::cout << "\n2. 計算1到100的總和：" << std::endl;
int num = 1;
int total = 0;
while (num <= 100) {
total += num;
num++;
}
std::cout << "1到100的總和 = " << total << std::endl;
// 3. 用戶輸入驗證
std::cout << "\n3. 用戶輸入驗證（模擬）：" << std::endl;
int password = 123456;
int userInput = 0;
int attempts = 0;
const int MAX_ATTEMPTS = 3;
// 模擬用戶嘗試輸入密碼
while (userInput != password && attempts < MAX_ATTEMPTS) {
attempts++;
```

```
// 模擬不同輸入
if (attempts == 1) {
userInput = 111111;
} else if (attempts == 2) {
userInput = 654321;
} else {
userInput = 123456; // 正確密碼
}
std::cout << "嘗試 " << attempts << ": 輸入 " << userInput;
if (userInput == password) {
std::cout << " - 登錄成功！" << std::endl;
} else {
std::cout << " - 密碼錯誤！" << std::endl;
}
}
if (userInput != password) {
std::cout << "嘗試次數過多，帳戶已鎖定！" << std::endl;
}
// 4. 計算階乘
std::cout << "\n4. 計算階乘（5!）：" << std::endl;
int n = 5;
int factorial = 1;
int counter = 1;
while (counter <= n) {
factorial *= counter;
std::cout << counter << "! = " << factorial << std::endl;
counter++;
}
std::cout << n << "! = " << factorial << std::endl;
// 5. 斐波那契數列
std::cout << "\n5. 斐波那契數列（前10個）：" << std::endl;
int a = 0, b = 1;
int count = 0;
std::cout << a << " " << b << " ";
count = 2; // 已經輸出了兩個
while (count < 10) {
int next = a + b;
std::cout << next << " ";
// 更新前兩個數
a = b;
```

```
count++;
}
std::cout << std::endl;
// 6. 條件複雜的while循環
std::cout << "\n6. 條件複雜的while循環：" << std::endl;
int value = 100;
int divisor = 3;
std::cout << value << " 除以 " << divisor << " 的過程：" << std::endl;
while (value >= divisor) {
std::cout << value << " / " << divisor << " = " << (value / divisor);
std::cout << "，餘數 " << (value % divisor) << std::endl;
value /= divisor; // value = value / divisor
}
std::cout << "最終值：" << value << std::endl;
// 7. 注意：避免無限循環
std::cout << "\n7. 注意：避免無限循環" << std::endl;
int safeCounter = 0;
bool condition = true;
// 使用安全計數器防止無限循環
while (condition && safeCounter < 5) {
std::cout << "安全循環 " << safeCounter + 1 << std::endl;
safeCounter++;
// 模擬某個條件會改變
if (safeCounter == 3) {
condition = false; // 改變條件，使循環結束
}
}
return 0;
}
```

📝 範例3：do-while循環

```
/*
* 範例5-3：do-while循環
* do-while循環至少執行一次，然後檢查條件
* 語法：do { 循環體 } while(條件);
*/
#include <iostream>
```

```
#include <string>
int main() {
std::cout << "=== do-while循環演示 ===" << std::endl;
// 1. 基本的do-while循環
std::cout << "1. 從1數到5：" << std::endl;
int i = 1;
do {
std::cout << "i = " << i << std::endl;
i++;
} while (i <= 5);
// 2. do-while與while的區別
std::cout << "\n2. do-while與while的區別：" << std::endl;
// while循環：先檢查條件
int a = 10;
std::cout << "while循環（條件一開始就為假）：" << std::endl;
while (a < 5) {
std::cout << "這行不會執行" << std::endl;
a++;
}
std::cout << "while循環結束，a = " << a << std::endl;
// do-while循環：先執行一次
int b = 10;
std::cout << "\ndo-while循環（條件一開始就為假）：" << std::endl;
do {
std::cout << "這行會執行一次，b = " << b << std::endl;
b++;
} while (b < 5);
std::cout << "do-while循環結束，b = " << b << std::endl;
// 3. 用戶菜單系統
std::cout << "\n3. 簡單菜單系統：" << std::endl;
int choice;
do {
std::cout << "\n===== 主菜單 =====" << std::endl;
std::cout << "1. 查看餘額" << std::endl;
std::cout << "2. 存款" << std::endl;
std::cout << "3. 取款" << std::endl;
std::cout << "4. 退出" << std::endl;
std::cout << "請選擇操作（1-4）：";
// 模擬用戶輸入（這裡固定選擇）
static int menuCounter = 0;
menuCounter++;
```

```
if (menuCounter == 1) {
choice = 1; // 查看餘額
} else if (menuCounter == 2) {
choice = 2; // 存款
} else if (menuCounter == 3) {
choice = 4; // 退出
}
std::cout << choice << std::endl;
switch (choice) {
case 1:
std::cout << "您的餘額是：1000元" << std::endl;
break;
case 2:
std::cout << "存款功能（開發中）" << std::endl;
break;
case 3:
std::cout << "取款功能（開發中）" << std::endl;
break;
case 4:
std::cout << "感謝使用，再見！" << std::endl;
break;
default:
std::cout << "無效選擇，請重新輸入！" << std::endl;
}
} while (choice != 4);
// 4. 輸入驗證（確保用戶輸入有效值）
std::cout << "\n4. 輸入驗證（必須輸入正數）：" << std::endl;
int number;
do {
std::cout << "請輸入一個正整數：";
// 模擬用戶輸入
static int inputCounter = 0;
inputCounter++;
if (inputCounter == 1) {
number = -5; // 無效輸入
std::cout << number << std::endl;
} else if (inputCounter == 2) {
number = 0; // 無效輸入（我們要求正數）
std::cout << number << std::endl;
} else {
```

```
std::cout << number << std::endl;
}
if (number <= 0) {
std::cout << "輸入無效！必須是正整數。" << std::endl;
}
} while (number <= 0);
std::cout << "您輸入的是：" << number << std::endl;
// 5. 猜數字遊戲
std::cout << "\n5. 猜數字遊戲：" << std::endl;
int secretNumber = 42;
int guess;
int attempts = 0;
std::cout << "我想到了一個1到100之間的數字..." << std::endl;
do {
attempts++;
std::cout << "第" << attempts << "次猜測：";
// 模擬不同猜測
if (attempts == 1) {
guess = 50;
} else if (attempts == 2) {
guess = 25;
} else if (attempts == 3) {
guess = 42; // 正確答案
}
std::cout << guess << std::endl;
if (guess < secretNumber) {
std::cout << "太小了！" << std::endl;
} else if (guess > secretNumber) {
std::cout << "太大了！" << std::endl;
} else {
std::cout << "恭喜！你猜對了！" << std::endl;
}
} while (guess != secretNumber && attempts < 5);
if (guess != secretNumber) {
std::cout << "遊戲結束！正確數字是：" << secretNumber << std::endl;
}
// 6. 計算平均分數
std::cout << "\n6. 計算多個分數的平均值：" << std::endl;
int score;
int sum = 0;
```

```
char another;
do {
count++;
std::cout << "輸入第" << count << "個分數：";
// 模擬輸入一些分數
if (count == 1) {
score = 85;
} else if (count == 2) {
score = 92;
} else if (count == 3) {
score = 78;
} else {
score = -1; // 結束標誌
}
if (score != -1) {
std::cout << score << std::endl;
sum += score;
// 模擬詢問是否繼續
if (count < 3) {
another = 'y';
} else {
another = 'n';
}
std::cout << "是否繼續輸入分數？(y/n): " << another << std::endl;
} else {
another = 'n';
}
} while (another == 'y' || another == 'Y');
if (count > 0) {
double average = static_cast<double>(sum) / count;
std::cout << "共輸入 " << count << " 個分數" << std::endl;
std::cout << "總分：" << sum << std::endl;
std::cout << "平均分：" << average << std::endl;
} else {
std::cout << "沒有輸入分數" << std::endl;
}
return 0;
}
```

📝 範例4：break與continue

```
/*
* 範例5-4：break與continue
* break：立即跳出當前循環
* continue：跳過本次循環的剩餘部分，進入下一次循環
*/
#include <iostream>
int main() {
std::cout << "=== break與continue演示 ===" << std::endl;
// 1. break基本用法
std::cout << "1. break基本用法：" << std::endl;
std::cout << "尋找第一個能被3和5整除的數（1-100）：" << std::endl;
for (int i = 1; i <= 100; i++) {
if (i % 3 == 0 && i % 5 == 0) {
std::cout << "找到：" << i << std::endl;
break; // 找到後立即跳出循環
}
std::cout << "檢查 " << i << "..." << std::endl;
}
// 2. continue基本用法
std::cout << "\n2. continue基本用法：" << std::endl;
std::cout << "輸出1-10中的奇數：" << std::endl;
for (int i = 1; i <= 10; i++) {
if (i % 2 == 0) { // 如果是偶數
continue; // 跳過偶數，不執行後面的輸出
}
std::cout << i << " ";
}
std::cout << std::endl;
// 3. break在while循環中的使用
std::cout << "\n3. break在while循環中：" << std::endl;
int number = 1;
while (true) { // 看似無限循環
std::cout << "當前數字：" << number << std::endl;
if (number >= 5) {
std::cout << "達到條件，跳出循環" << std::endl;
break; // 跳出循環
}
number++;
```

```
}
// 4. continue在while循環中的使用
std::cout << "\n4. continue在while循環中：" << std::endl;
int n = 0;
while (n < 10) {
n++;
if (n % 3 == 0) {
continue; // 跳過3的倍數
}
std::cout << n << " ";
}
std::cout << std::endl;
// 5. 搜索示例：在數組中查找元素
std::cout << "\n5. 在數組中查找元素：" << std::endl;
int numbers[] = {23, 45, 67, 89, 12, 34, 56};
int target = 89;
bool found = false;
for (int i = 0; i < 7; i++) {
if (numbers[i] == target) {
std::cout << "找到目標 " << target << " 在位置 " << i << std::endl;
found = true;
break; // 找到後立即停止搜索
}
}
if (!found) {
std::cout << "沒有找到目標 " << target << std::endl;
}
// 6. 過濾示例：跳過不符合條件的數據
std::cout << "\n6. 處理數據，跳過無效值：" << std::endl;
int data[] = {12, -5, 8, 0, 15, -3, 20};
int dataCount = 7;
std::cout << "有效的正數：" << std::endl;
for (int i = 0; i < dataCount; i++) {
if (data[i] <= 0) {
continue; // 跳過非正數
}
std::cout << data[i] << " ";
}
std::cout << std::endl;
// 7. 多層循環中的break
std::cout << "\n7. 多層循環中的break：" << std::endl;
```

```
for (int i = 1; i <= 3; i++) {
std::cout << "外層循環 i = " << i << std::endl;
for (int j = 1; j <= 3; j++) {
std::cout << " 內層循環 j = " << j << std::endl;
if (i == 2 && j == 2) {
std::cout << " 條件滿足，跳出內層循環" << std::endl;
break; // 只跳出內層循環
}
}
}
// 8. 標籤與goto（不推薦，但了解其存在）
std::cout << "\n8. 跳出多重循環（不使用goto的方法）：" << std::endl;
bool shouldBreak = false;
for (int i = 1; i <= 3; i++) {
for (int j = 1; j <= 3; j++) {
std::cout << "(" << i << "," << j << ") ";
if (i == 2 && j == 2) {
std::cout << "\n找到(2,2)，跳出所有循環" << std::endl;
shouldBreak = true;
break;
}
}
if (shouldBreak) {
break; // 跳出外層循環
}
}
// 9. 實際應用：質數判斷
std::cout << "\n9. 判斷一個數是否為質數：" << std::endl;
int num = 17;
bool isPrime = true;
if (num <= 1) {
isPrime = false;
} else {
for (int i = 2; i <= num / 2; i++) {
if (num % i == 0) {
isPrime = false;
break; // 找到因數，立即跳出，無需繼續檢查
}
}
}
```

```
std::cout << num << " 是質數" << std::endl;
} else {
std::cout << num << " 不是質數" << std::endl;
}
// 10. 實際應用：計算有效數據的平均值
std::cout << "\n10. 計算有效數據的平均值（跳過-1）：" << std::endl;
int values[] = {10, 20, -1, 30, 40, -1, 50};
int validCount = 0;
int total = 0;
for (int i = 0; i < 7; i++) {
if (values[i] == -1) {
std::cout << "跳過無效值（-1）" << std::endl;
continue; // 跳過無效數據
}
validCount++;
total += values[i];
std::cout << "添加有效值：" << values[i] << std::endl;
}
if (validCount > 0) {
double average = static_cast<double>(total) / validCount;
std::cout << "有效數據個數：" << validCount << std::endl;
std::cout << "總和：" << total << std::endl;
std::cout << "平均值：" << average << std::endl;
} else {
std::cout << "沒有有效數據" << std::endl;
}
return 0;
}
```

📝 範例5：嵌套循環

```
/*
* 範例5-5：嵌套循環
* 循環內部包含另一個循環
* 常用於處理二維數據、圖形輸出等
*/
#include <iostream>
#include <iomanip>
```

```
int main() {
std::cout << "=== 嵌套循環演示 ===" << std::endl;
// 1. 基本的嵌套循環：乘法表
std::cout << "1. 九九乘法表（部分）：" << std::endl;
for (int i = 1; i <= 3; i++) { // 外層循環：控制行
for (int j = 1; j <= 3; j++) { // 內層循環：控制列
std::cout << i << "x" << j << "="
<< std::setw(2) << (i * j) << " ";
}
std::cout << std::endl; // 換行
}
// 2. 矩形圖案
std::cout << "\n2. 矩形圖案（5行10列）：" << std::endl;
int rows = 5;
int cols = 10;
for (int i = 1; i <= rows; i++) {
for (int j = 1; j <= cols; j++) {
std::cout << "*";
}
std::cout << std::endl;
}
// 3. 直角三角形
std::cout << "\n3. 直角三角形：" << std::endl;
for (int i = 1; i <= 5; i++) {
for (int j = 1; j <= i; j++) {
std::cout << "*";
}
std::cout << std::endl;
}
// 4. 倒直角三角形
std::cout << "\n4. 倒直角三角形：" << std::endl;
for (int i = 5; i >= 1; i--) {
for (int j = 1; j <= i; j++) {
std::cout << "*";
}
std::cout << std::endl;
}
// 5. 金字塔
std::cout << "\n5. 金字塔圖案：" << std::endl;
int height = 5;
```

```
// 打印空格
for (int space = 1; space <= height - i; space++) {
std::cout << " ";
}
// 打印星號
for (int star = 1; star <= (2 * i - 1); star++) {
std::cout << "*";
}
std::cout << std::endl;
}
// 6. 二維數組處理
std::cout << "\n6. 二維數組處理：" << std::endl;
int matrix[3][4] = {
{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12}
};
std::cout << "矩陣內容：" << std::endl;
for (int i = 0; i < 3; i++) {
for (int j = 0; j < 4; j++) {
std::cout << std::setw(3) << matrix[i][j];
}
std::cout << std::endl;
}
// 7. 計算矩陣對角線和
std::cout << "\n7. 計算矩陣對角線和：" << std::endl;
int squareMatrix[3][3] = {
{1, 2, 3},
{4, 5, 6},
{7, 8, 9}
};
int diagonalSum = 0;
for (int i = 0; i < 3; i++) {
for (int j = 0; j < 3; j++) {
std::cout << std::setw(3) << squareMatrix[i][j];
if (i == j) { // 主對角線
diagonalSum += squareMatrix[i][j];
}
}
std::cout << std::endl;
}
```

```
std::cout << "主對角線和：" << diagonalSum << std::endl;
// 8. 組合問題：投擲兩個骰子的所有可能
std::cout << "\n8. 投擲兩個骰子的所有可能：" << std::endl;
std::cout << " ";
for (int j = 1; j <= 6; j++) {
std::cout << std::setw(4) << j;
}
std::cout << std::endl;
std::cout << " +----------------------" << std::endl;
for (int die1 = 1; die1 <= 6; die1++) {
std::cout << die1 << " |";
for (int die2 = 1; die2 <= 6; die2++) {
std::cout << std::setw(4) << (die1 + die2);
}
std::cout << std::endl;
}
// 9. 查找二維數組中的元素
std::cout << "\n9. 在二維數組中查找元素：" << std::endl;
int searchMatrix[3][3] = {
{12, 23, 34},
{45, 56, 67},
{78, 89, 90}
};
int targetValue = 56;
bool found = false;
int foundRow = -1, foundCol = -1;
for (int i = 0; i < 3; i++) {
for (int j = 0; j < 3; j++) {
if (searchMatrix[i][j] == targetValue) {
found = true;
foundRow = i;
foundCol = j;
break; // 跳出內層循環
}
}
if (found) {
break; // 跳出外層循環
}
}
if (found) {
```

```
<< foundRow << ", " << foundCol << ")" << std::endl;
} else {
std::cout << "沒有找到 " << targetValue << std::endl;
}
// 10. 不同類型的循環嵌套
std::cout << "\n10. 不同類型的循環嵌套：" << std::endl;
int outer = 1;
while (outer <= 3) {
std::cout << "外層 while 循環，outer = " << outer << std::endl;
for (int inner = 1; inner <= 2; inner++) {
std::cout << " 內層 for 循環，inner = " << inner << std::endl;
}
outer++;
}
return 0;
}
```

📝 範例6：循環的實際應用

```
/*
* 範例6：循環的實際應用
* 綜合應用各種循環解決實際問題
*/
#include <iostream>
#include <iomanip>
#include <cmath>
int main() {
std::cout << "=== 循環的實際應用 ===" << std::endl;
// 1. 統計學生成績
std::cout << "\n1. 統計學生成績：" << std::endl;
int scores[] = {85, 92, 78, 95, 88, 76, 90, 84, 79, 91};
int studentCount = 10;
int highest = scores[0];
int lowest = scores[0];
int total = 0;
// 找出最高分、最低分，計算總分
for (int i = 0; i < studentCount; i++) {
if (scores[i] > highest) {
```

```
highest = scores[i];
}
if (scores[i] < lowest) {
lowest = scores[i];
}
total += scores[i];
}
double average = static_cast<double>(total) / studentCount;
std::cout << "學生人數：" << studentCount << std::endl;
std::cout << "最高分：" << highest << std::endl;
std::cout << "最低分：" << lowest << std::endl;
std::cout << "平均分：" << average << std::endl;
// 計算成績分佈
int gradeA = 0, gradeB = 0, gradeC = 0, gradeD = 0, gradeF = 0;
for (int i = 0; i < studentCount; i++) {
if (scores[i] >= 90) {
gradeA++;
} else if (scores[i] >= 80) {
gradeB++;
} else if (scores[i] >= 70) {
gradeC++;
} else if (scores[i] >= 60) {
gradeD++;
} else {
gradeF++;
}
}
std::cout << "\n成績分佈：" << std::endl;
std::cout << "A等：" << gradeA << " 人" << std::endl;
std::cout << "B等：" << gradeB << " 人" << std::endl;
std::cout << "C等：" << gradeC << " 人" << std::endl;
std::cout << "D等：" << gradeD << " 人" << std::endl;
std::cout << "F等：" << gradeF << " 人" << std::endl;
// 2. 財務計算：複利
std::cout << "\n2. 複利計算：" << std::endl;
double principal = 10000.0; // 本金
double annualRate = 0.05; // 年利率
int years = 10; // 年數
std::cout << std::fixed << std::setprecision(2);
std::cout << "本金：" << principal << " 元" << std::endl;
```

```
std::cout << "投資年限：" << years << " 年" << std::endl;
std::cout << "\n年度增長情況：" << std::endl;
double amount = principal;
for (int year = 1; year <= years; year++) {
amount *= (1 + annualRate);
std::cout << "第" << std::setw(2) << year << "年："
<< std::setw(10) << amount << " 元" << std::endl;
}
double totalInterest = amount - principal;
std::cout << "\n最終金額：" << amount << " 元" << std::endl;
std::cout << "總利息：" << totalInterest << " 元" << std::endl;
// 3. 質數生成器
std::cout << "\n3. 找出1-100之間的所有質數：" << std::endl;
int primeCount = 0;
std::cout << "1-100之間的質數有：";
for (int num = 2; num <= 100; num++) {
bool isPrime = true;
// 檢查是否為質數
for (int i = 2; i <= sqrt(num); i++) {
if (num % i == 0) {
isPrime = false;
break;
}
}
if (isPrime) {
std::cout << num << " ";
primeCount++;
// 每10個換一行
if (primeCount % 10 == 0) {
std::cout << std::endl << " ";
}
}
}
std::cout << std::endl;
std::cout << "總共有 " << primeCount << " 個質數" << std::endl;
// 4. 圖形時鐘（模擬）
std::cout << "\n4. 模擬時鐘（每小時打印一次）：" << std::endl;
std::cout << " ";
for (int hour = 0; hour < 12; hour++) {
std::cout << std::setw(3) << hour;
```

```
std::cout << std::endl;
for (int minute = 0; minute < 60; minute += 5) {
std::cout << std::setw(2) << minute << " |";
for (int hour = 0; hour < 12; hour++) {
// 模擬時鐘位置計算
if (minute == 0) {
std::cout << " X"; // 整點
} else if (minute == 30) {
std::cout << " O"; // 半點
} else {
std::cout << " ."; // 其他時間
}
}
std::cout << std::endl;
}
// 5. 遊戲：猜數字（完整版）
std::cout << "\n5. 猜數字遊戲（完整版）：" << std::endl;
srand(time(0)); // 設置隨機種子
int secret = rand() % 100 + 1; // 1-100的隨機數
int guess;
int attempts = 0;
const int MAX_ATTEMPTS = 7;
std::cout << "我想了一個1到100之間的數字，你有" << MAX_ATTEMPTS << "次機會猜中
它！" << std::endl;
while (attempts < MAX_ATTEMPTS) {
attempts++;
std::cout << "\n第" << attempts << "次猜測，請輸入數字：";
// 模擬不同猜測
if (attempts == 1) {
guess = 50;
std::cout << guess << std::endl;
} else if (attempts == 2) {
guess = (guess < secret) ? 75 : 25;
std::cout << guess << std::endl;
} else if (attempts == 3) {
guess = (guess < secret) ? 88 : 38;
std::cout << guess << std::endl;
} else if (attempts == 4) {
guess = secret; // 這次猜中
std::cout << guess << std::endl;
}
```

```
if (guess < secret) {
std::cout << "太小了！" << std::endl;
} else if (guess > secret) {
std::cout << "太大了！" << std::endl;
} else {
std::cout << "恭喜！你猜對了！" << std::endl;
break;
}
if (attempts == MAX_ATTEMPTS) {
std::cout << "很遺憾，你沒有猜中。正確數字是：" << secret << std::endl;
}
}
// 6. 數據過濾和轉換
std::cout << "\n6. 數據過濾和轉換：" << std::endl;
const int DATA_SIZE = 10;
int originalData[DATA_SIZE] = {15, -3, 8, 0, 22, -7, 19, 4, -1, 11};
int processedData[DATA_SIZE];
int processedCount = 0;
std::cout << "原始數據：";
for (int i = 0; i < DATA_SIZE; i++) {
std::cout << originalData[i] << " ";
}
std::cout << std::endl;
// 過濾負數和零，對正數加倍
for (int i = 0; i < DATA_SIZE; i++) {
if (originalData[i] <= 0) {
continue; // 跳過非正數
}
processedData[processedCount] = originalData[i] * 2;
processedCount++;
}
std::cout << "處理後數據（正數加倍）：";
for (int i = 0; i < processedCount; i++) {
std::cout << processedData[i] << " ";
}
std::cout << std::endl;
std::cout << "有效數據個數：" << processedCount << std::endl;
// 7. 模擬進度條
std::cout << "\n7. 進度條模擬：" << std::endl;
int totalTasks = 50;
std::cout << "處理進度：[";
```

```
for (int task = 1; task <= totalTasks; task++) {
// 模擬處理每個任務
std::cout << "#";
// 每10個任務顯示一次百分比
if (task % (totalTasks / 10) == 0) {
int percent = (task * 100) / totalTasks;
std::cout << " " << percent << "%";
}
}
std::cout << "] 完成！" << std::endl;
return 0;
}
```

🔧 練習題

練習5-1：計算階乘

創建一個程序，計算並輸出1!到10!的值。 要求：使用for循環實現。

練習5-2：猜數字遊戲改進

改進猜數字遊戲，讓程序：

1. 隨機生成1-100之間的數字
2. 記錄玩家猜測的次數
3. 根據玩家猜測給出提示（太大/太小）
4. 玩家有7次機會
5. 遊戲結束後詢問是否再玩一次

練習5-3：打印圖案

使用嵌套循環打印以下圖案：

```
*
***
*****
*******
*********
*******
*****
***
*
```

練習5-4：尋找完全數

完全數是指一個數等於它的真因子之和。例如：6 = 1 + 2 + 3 創建一個程序，找出1-1000之間的所有完全數。

練習5-5：質數篩法

使用埃拉托斯特尼篩法找出1-100之間的所有質數。

📚 重點摘要

1. 三種循環結構比較

循環類型	語法	特點	適用場景
for循環	`for(初始化; 條件; 更新)`	循環次數明確	計數循環、數組遍歷
while循環	`while(條件)`	先判斷後執行	條件循環、輸入驗證
do-while循環	`do {...} while(條件)`	先執行後判斷	至少執行一次的操作

2. 循環控制語句

語句	作用	示例
break	跳出當前循環	`if(條件) break;`
continue	跳過本次循環的剩餘部分	`if(條件) continue;`

3. 嵌套循環的要點

1. 外層循環執行一次，內層循環執行完整一輪
2. 注意循環變數的名稱不要重複
3. 可以使用break跳出內層循環，但要跳出所有循環需要額外處理
4. 嵌套循環的時間複雜度通常為$O(n^2)$或更高

4. 無限循環

```
// 三種無限循環的寫法
for (;;) {
// 循環體
if (條件) break;
}
```

```
while (true) {
// 循環體
if (條件) break;
}
do {
// 循環體
if (條件) break;
} while (true);
```

5. 循環的最佳實踐

1. **明確循環條件**：確保循環能夠正常結束
2. **避免無限循環**：設置合理的終止條件
3. **注意循環變數**：初始化、更新、作用域
4. **使用有意義的變數名**：提高代碼可讀性
5. **考慮性能**：避免不必要的循環操作
6. **處理邊界情況**：考慮空循環、單次循環等情況

🎯 常見問答

Q1: for循環和while循環如何選擇？

A: 當循環次數已知時使用for循環，當循環次數未知但循環條件明確時使用while循環。for循環更適合計數和遍歷，while循環更適合條件控制。

Q2: 如何跳出多重嵌套循環？

A: 有幾種方法：

1. 使用標誌變數
2. 使用goto語句（不推薦）
3. 將嵌套循環封裝到函數中，使用return
4. 使用異常處理

Q3: continue和break有什麼區別？

A: continue是跳過本次循環的剩餘部分，繼續下一次循環；break是立即結束整個循環，跳出循環體。

Q4: 什麼時候使用do-while循環？

A: 當循環體至少需要執行一次時使用do-while循環。常見場景包括：用戶輸入驗證、菜單顯示、遊戲循環等。

Q5: 如何避免無限循環？

A:

1. 確保循環條件最終會變為假
2. 在循環體內修改循環條件
3. 使用break語句提供額外的退出條件

4. 設置安全計數器，防止循環次數過多

🚀 下一步

下一課我們將學習：

- **數組與字符串**（一維數組、多維數組）
- **C風格字符串**
- **C++ string類**
- **字符數組操作**

💻 完整測試程序

```
/*
* 第5課綜合測試
* 綜合應用各種循環結構
*/
#include <iostream>
#include <iomanip>
int main() {
std::cout << "=== 第5課綜合測試：循環結構 ===" << std::endl;
// 1. 各種循環演示
std::cout << "\n1. 各種循環演示：" << std::endl;
// for循環：計算1-10的平方
std::cout << "for循環：1-10的平方" << std::endl;
for (int i = 1; i <= 10; i++) {
std::cout << i << "² = " << (i * i) << std::endl;
}
// while循環：模擬倒計時
std::cout << "\nwhile循環：倒計時" << std::endl;
int count = 5;
while (count > 0) {
std::cout << count << "... ";
count--;
}
std::cout << "發射！" << std::endl;
// do-while循環：確保至少執行一次
std::cout << "\ndo-while循環：至少執行一次" << std::endl;
int value = 10;
do {
std::cout << "即使條件為假，這行也會執行，value = " << value << std::endl;
```

```
value++;
} while (value < 5);
// 2. break和continue
std::cout << "\n2. break和continue演示：" << std::endl;
std::cout << "找到第一個3和5的公倍數：";
for (int i = 1; i <= 50; i++) {
if (i % 3 == 0 && i % 5 == 0) {
std::cout << i << std::endl;
break;
}
}
std::cout << "1-10中的偶數：";
for (int i = 1; i <= 10; i++) {
if (i % 2 != 0) {
continue; // 跳過奇數
}
std::cout << i << " ";
}
std::cout << std::endl;
// 3. 嵌套循環：乘法表
std::cout << "\n3. 嵌套循環：九九乘法表" << std::endl;
for (int i = 1; i <= 9; i++) {
for (int j = 1; j <= i; j++) {
std::cout << j << "×" << i << "="
<< std::setw(2) << (i * j) << " ";
}
std::cout << std::endl;
}
// 4. 循環應用：統計數據
std::cout << "\n4. 統計數據分析：" << std::endl;
int data[] = {23, 45, 67, 12, 89, 34, 56, 78, 90, 11};
int dataSize = 10;
int evenCount = 0, oddCount = 0;
int sum = 0;
for (int i = 0; i < dataSize; i++) {
sum += data[i];
if (data[i] % 2 == 0) {
evenCount++;
} else {
oddCount++;
```

```
}
double average = static_cast<double>(sum) / dataSize;
std::cout << "數據總數：" << dataSize << std::endl;
std::cout << "偶數個數：" << evenCount << std::endl;
std::cout << "奇數個數：" << oddCount << std::endl;
std::cout << "總和：" << sum << std::endl;
std::cout << "平均值：" << average << std::endl;
// 5. 模擬加載進度
std::cout << "\n5. 模擬加載進度：" << std::endl;
std::cout << "加載中：[";
for (int progress = 0; progress <= 100; progress += 5) {
std::cout << "#";
// 模擬一些延遲效果（這裡用循環模擬）
for (int delay = 0; delay < 1000000; delay++) {
// 空循環，模擬延遲
}
}
std::cout << "] 完成！" << std::endl;
// 6. 尋找水仙花數
std::cout << "\n6. 水仙花數（1-999）：" << std::endl;
std::cout << "水仙花數有：";
for (int num = 100; num < 1000; num++) {
int digit1 = num / 100; // 百位
int digit2 = (num / 10) % 10; // 十位
int digit3 = num % 10; // 個位
int sumOfCubes = digit1 * digit1 * digit1 +
digit2 * digit2 * digit2 +
digit3 * digit3 * digit3;
if (num == sumOfCubes) {
std::cout << num << " ";
}
}
std::cout << std::endl;
return 0;
}
```

輸出結果：

```
=== 第5課綜合測試：循環結構 ===
1. 各種循環演示：
for循環：1-10的平方
```

1² = 1
2² = 4
3² = 9
4² = 16
5² = 25
6² = 36
7² = 37
8² = 64
9² = 81
10² = 100
while循環：倒計時
5... 4... 3... 2... 1... 發射！
do-while循環：至少執行一次
即使條件為假，這行也會執行，value = 10
2. break和continue演示：
找到第一個3和5的公倍數：15
1-10中的偶數：2 4 6 8 10
3. 嵌套循環：九九乘法表
1×1= 1
1×2= 2 2×2= 4
1×3= 3 2×3= 6 3×3= 9
1×4= 4 2×4= 8 3×4=12 4×4=16
1×5= 5 2×5=10 3×5=15 4×5=20 5×5=25
1×6= 6 2×6=12 3×6=18 4×6=24 5×6=30 6×6=36
1×7= 7 2×7=14 3×7=21 4×7=28 5×7=35 6×7=42 7×7=49
1×8= 8 2×8=16 3×8=24 4×8=32 5×8=40 6×8=48 7×8=56 8×8=64
1×9= 9 2×9=18 3×9=27 4×9=36 5×9=45 6×9=54 7×9=63 8×9=72 9×9=81
4. 統計數據分析：
數據總數：10
偶數個數：5
奇數個數：5
總和：505
平均值：50.5
5. 模擬加載進度：
加載中：[####################] 完成！
6. 水仙花數（1-999）：
水仙花數有：153 370 371 407

✅ 本課檢查清單

- [ ] 能夠正確使用for、while、do-while循環
- [ ] 理解break和continue的區別及使用場景
- [ ] 能夠編寫嵌套循環解決問題
- [ ] 掌握循環控制變數的初始化、更新和作用域
- [ ] 能夠處理無限循環和循環終止條件
- [ ] 能夠使用循環解決實際問題
- [ ] 了解不同循環結構的適用場景
- [ ] 能夠使用循環進行數據處理和統計

📞 需要幫助？

如果你在練習中遇到問題：

1. 檢查循環條件是否正確，避免無限循環
2. 確保循環變數在循環內部或外部正確初始化
3. 注意break和continue的區別
4. 測試循環的邊界情況（如第一次循環、最後一次循環）
5. 使用調試工具或添加打印語句跟蹤循環執行過程

下一課見！ 🚀