🧮 Expressions, Statements and Operators

🔷 Expressions（運算式）

在 C++ 中，**Expression** 是任何可以被評估為一個值的語法結構。

範例：

```
3 + 5     // 表達式，結果為 8
x * y + 2   // 也是表達式，依賴變數 x 與 y
```

表達式可包含：

- 常數（如 `42`）
- 變數（如 `x`）
- 運算子（如 `+`, `-`, `*`, `/`）
- 函數呼叫（如 `sqrt(x)`）

🔶 Statements and Block Statements（敘述與區塊敘述）

Statement（敘述） 是指 C++ 中的執行單位，通常以分號結尾。

```
int x = 5;     // 宣告敘述
x = x + 1;     // 指派敘述
std::cout << x; // 輸出敘述
```

Block Statement（區塊敘述） 是用 `{}` 括起來的一組敘述，常用於流程控制中：

```
{
  int x = 10;
  std::cout << x;
}
```

❓如何分辨Expression&Statement

🧠 核心觀念：

🔷 Expression（運算式）

✅ **會產生值**。
✅ 可以當作**更大運算的一部分**。

範例：

```
3 + 5       // 是 expression，結果是 8
x = 10      // 是 expression（在 C++ 中會回傳 x 的值 → 10）
x > y       // 是 expression，結果是 true 或 false
sqrt(4)     // 是 expression，結果是 2
```

> ✅ 所以 expression 通常可以「放在其他地方」，比如：

```
int y = (x = 10);   // x = 10 是 expression，會回傳 10，被賦值給 y
```

🔶 Statement（敘述）

✅ 是**一個完整的執行單位**，例如定義變數、執行一行命令、流程控制等。
✅ **不一定產生值**。

範例：

```
int x = 5;     // 敘述：宣告並初始化
x++;           // 敘述：遞增運算
std::cout << x; // 敘述：執行輸出
if (x > 3) {    // if 是敘述，括號裡是 expression
   // 區塊也是一個 compound statement
}
```

⚖️ 差異對照表：

項目	Expression（表達式）	Statement（敘述）
是否會產生值？	✅ 是	❌ 不一定
是否以 `;` 結尾？	❌ 不一定（但可放進敘述）	✅ 通常是
是否能當作另一運算的部分？	✅ 可以嵌套	❌ 不行
舉例	`3 + 4`, `x * y`, `x > 5`	`int x = 5;`, `x++;`, `if (...)`

🧪 小技巧：如何辨認？

❓ 問你自己這句話：

> **「這段語法是否會產生一個值？」**

- 若會 → 是 **expression**
- 若只是執行一件事情，沒回傳值 → 是 **statement**

✅ 實際混合例子拆解：

```
int a = 5 + 3;
```

- `5 + 3` 是 expression（結果是 8）
- `int a = 5 + 3;` 是 statement（完成一個指派）

```
if (x > 0) {
  y = x * 2;
}
```

- `x > 0` 是 expression（產生布林值）
- 整個 `if (...) { ... }` 是 statement
- `y = x * 2;` 也是 statement，內部包含一個 expression：`x * 2`

⚙️ Operators（運算子）

運算子用來進行操作，如加減乘除、邏輯運算等。C++ 提供了許多內建運算子，下面會在 Assignment 部分詳細介紹。

📝 Assignment

➕ Arithmetic Operators（算術運算子）

運算子	功能	範例 (a = 5, b = 2)
+	加法	a + b → 7
-	減法	a - b → 3
*	乘法	a * b → 10
/	除法	a / b → 2
%	取餘數	a % b → 1

⬆️ Increment / Decrement（遞增遞減）

運算子	說明	範例
++	遞增（+1）	++x 或 x++
--	遞減（-1）	--x 或 x--

> 注意：前置與後置寫法會影響運算順序。

```
int x = 5;
int y = ++x; // y = 6, x = 6（先加後用）
int z = x--; // z = 6, x = 5（先用後減）
```

❓ Equality Operators（等值運算子）

運算子	說明	範例
==	是否相等	x == y
!=	是否不相等	x != y

🔍 Relational Operators（關係運算子）

運算子	說明	範例
<	小於	x < y
>	大於	x > y
<=	小於等於	x <= y
>=	大於等於	x >= y

🧠 Logical Operators（邏輯運算子）

運算子	說明	範例
&&	邏輯 AND	x > 0 && y < 10
、		、
!	邏輯 NOT	!(x < 5)

🖇️ Compound Assignment Operators（複合指派運算子）

這些運算子將運算與指派結合，例如 `x += 1` 等同於 `x = x + 1`。

運算子	功能
+=	加後指派
-=	減後指派
*=	乘後指派

運算子	功能
`/=`	除後指派
`%=`	餘後指派

🔢 Precedence（運算子優先順序）

當多個運算子同時出現時，**優先順序** 決定了誰先運算。

常見優先順序從高到低：

1. 括號 `()`
2. 單目運算子 `++`, `--`, `!`
3. 乘除模 `* / %`
4. 加減 `+ -`
5. 關係運算 `< <= > >=`
6. 等值比較 `== !=`
7. 邏輯 AND `&&`
8. 邏輯 OR `||`
9. 指派 `= += -= *= ...`

> **小技巧**：若有疑慮可用括號 `()` 強制運算順序！

Mixed Type Expressions

Conversions

Higher vs. Lower types are based on the size of the values the type can hold

- long double,double,float,unsigned long,long,unsigned int,int
- short and char types are always converted to int

Type Coercion(強制): conversion of one operand to another data type

Promotion: conversion to a higher type

Used in mathematical expressions

Demotion: conversion to a lower type

Used with assignment to lower type
範例:

```
//lower op higher the lower is promoted to a higher
2 * 5.2
//2 is promoted to 2.0
//lower = higher; the higher is demoted to a lower
int num {0};
num = 100.2;
//result num = 100
```

Explicit Type Casting - static_cast <type >

```
int total amount {100};
int total number {8};
double average {0.0};
average total amount / total number; =
cout << average << endl;
// displays 12
average static cast<double>(total amount) / total number; =
cout << average << endl;
// displays 12.5
```

Logical Operators

Precedence

not has higher precedence than **and**
and has higher precedence than **or**
not is a unary operator
and and **or** are binary operators