第2課：基本數據類型與變數

第2課：基本數據類型與變數

📚 課程目標

- 理解C++的基本數據類型
- 學會變數的聲明、定義和初始化
- 掌握不同數據類型的使用場景
- 了解數據類型的範圍和限制

🎯 本課關鍵字

- `int` - 整數類型
- `double` - 雙精度浮點數
- `char` - 字符類型
- `bool` - 布林類型
- `float` - 單精度浮點數
- `short` - 短整數
- `long` - 長整數
- `unsigned` - 無符號類型

📝 範例1：整數類型的基本使用

```
/*
* 範例2-1：整數類型 (int)
* 整數是最基本的數據類型，用於存儲沒有小數部分的數字
*/
#include <iostream>
#include <climits> // 包含整數類型的限制資訊
int main() {
std::cout << "=== 整數類型演示 ===" << std::endl;
// 1. 變數聲明與初始化
int age = 25; // 聲明並初始化一個整數變數
int year; // 只聲明，未初始化（值不確定）
year = 2024; // 賦值
std::cout << "年齡：" << age << " 歲" << std::endl;
std::cout << "年份：" << year << " 年" << std::endl;
// 2. 同時聲明多個變數
```

```
int width = 1920, height = 1080;
std::cout << "解析度：" << width << "x" << height << std::endl;
// 3. 整數的範圍（使用 climits 中的常數）
std::cout << "\n整數類型的範圍：" << std::endl;
std::cout << "int 的最小值：" << INT_MIN << std::endl;
std::cout << "int 的最大值：" << INT_MAX << std::endl;
// 4. 整數運算
int a = 10, b = 3;
std::cout << "\n整數運算：" << std::endl;
std::cout << a << " + " << b << " = " << (a + b) << std::endl;
std::cout << a << " - " << b << " = " << (a - b) << std::endl;
std::cout << a << " * " << b << " = " << (a * b) << std::endl;
std::cout << a << " / " << b << " = " << (a / b) << std::endl;
std::cout << a << " % " << b << " = " << (a % b) << std::endl;
// 注意：整數除法會捨棄小數部分
std::cout << "整數除法 10/3 = " << (10/3) << " (不是 3.333)" << std::endl;
return 0;
}
```

📝 範例2：浮點數類型

```
/*
* 範例2-2：浮點數類型 (float, double)
* 浮點數用於存儲帶有小數部分的數字
* double 比 float 精度更高，更常用
*/
#include <iostream>
#include <iomanip> // 用於控制輸出格式
int main() {
std::cout << "=== 浮點數類型演示 ===" << std::endl;
// 1. float 類型（單精度浮點數）
float pi_float = 3.1415926535f; // 注意後綴 f 表示 float
float radius = 5.0f;
float area_float = pi_float * radius * radius;
std::cout << "float 計算圓面積：" << std::endl;
std::cout << "半徑：" << radius << " 公尺" << std::endl;
std::cout << "面積：" << area_float << " 平方公尺" << std::endl;
// 2. double 類型（雙精度浮點數，更精確）
```

```
double pi_double = 3.141592653589793;
double area_double = pi_double * radius * radius;
std::cout << "\ndouble 計算圓面積：" << std::endl;
std::cout << "面積：" << area_double << " 平方公尺" << std::endl;
// 3. 浮點數運算
double x = 10.5, y = 3.2;
std::cout << "\n浮點數運算：" << std::endl;
std::cout << std::fixed << std::setprecision(2); // 固定小數點，顯示2位小數
std::cout << x << " + " << y << " = " << (x + y) << std::endl;
std::cout << x << " - " << y << " = " << (x - y) << std::endl;
std::cout << x << " * " << y << " = " << (x * y) << std::endl;
std::cout << x << " / " << y << " = " << (x / y) << std::endl;
// 4. 精度比較
std::cout << "\n精度比較（圓周率）：" << std::endl;
std::cout << std::setprecision(10); // 顯示10位小數
std::cout << "float : " << pi_float << std::endl;
std::cout << "double : " << pi_double << std::endl;
return 0;
}
```

📝 範例3：字符類型

```
/*
* 範例2-3：字符類型 (char)
* 用於存儲單個字符（字母、數字、符號等）
* 使用單引號 ' ' 包圍
*/
#include <iostream>
int main() {
std::cout << "=== 字符類型演示 ===" << std::endl;
// 1. 基本字符
char letter = 'A';
char digit = '7';
char symbol = '$';
char space = ' ';
char newline = '\n'; // 特殊字符：換行
std::cout << "字母：" << letter << std::endl;
std::cout << "數字：" << digit << std::endl;
```

```
std::cout << "符號：" << symbol << std::endl;
std::cout << "空格：" << "開始" << space << "結束" << std::endl;
std::cout << "換行：" << "第一行" << newline << "第二行" << std::endl;
// 2. 字符的本質是數字（ASCII碼）
std::cout << "\n字符的ASCII碼：" << std::endl;
std::cout << "'A' 的ASCII碼：" << (int)letter << std::endl;
std::cout << "'7' 的ASCII碼：" << (int)digit << std::endl;
std::cout << "'$' 的ASCII碼：" << (int)symbol << std::endl;
// 3. 字符運算（基於ASCII碼）
char small_a = 'a';
char capital_a = 'A';
std::cout << "\n字符運算：" << std::endl;
std::cout << "小寫 a：" << small_a << " (ASCII: " << (int)small_a << ")" <<
std::endl;
std::cout << "大寫 A：" << capital_a << " (ASCII: " << (int)capital_a << ")" <<
std::endl;
std::cout << "大小寫轉換：" << std::endl;
std::cout << "小寫轉大寫：" << (char)(small_a - 32) << std::endl; // 'a' 轉 'A'
std::cout << "大寫轉小寫：" << (char)(capital_a + 32) << std::endl; // 'A' 轉
'a'
// 4. 特殊字符（跳脫序列）
std::cout << "\n特殊字符（跳脫序列）：" << std::endl;
std::cout << "換行：第一行\\n第二行" << std::endl;
std::cout << "製表符：姓名\\t年齡\\t成績" << std::endl;
std::cout << "雙引號：他說：\\\"你好！\\\"" << std::endl;
std::cout << "反斜線：路徑 C:\\\\Program Files\\\\" << std::endl;
return 0;
}
```

📝 範例4：布林類型

```
/*
* 範例2-4：布林類型 (bool)
* 用於表示邏輯值：真(true)或假(false)
* 在C++中，true是1，false是0
*/
#include <iostream>
int main() {
```

```
std::cout << "=== 布林類型演示 ===" << std::endl;
// 1. 布林變數的聲明與初始化
bool isSunny = true;
bool isRaining = false;
bool isWeekend = true;
std::cout << "今天晴天：" << isSunny << std::endl;
std::cout << "今天下雨：" << isRaining << std::endl;
std::cout << "今天是週末：" << isWeekend << std::endl;
// 2. 布林值在輸出時的顯示
std::cout << "\n布林值顯示（使用boolalpha）：" << std::endl;
std::cout << std::boolalpha; // 以 true/false 形式顯示布林值
std::cout << "今天晴天：" << isSunny << std::endl;
std::cout << "今天下雨：" << isRaining << std::endl;
// 3. 關係運算返回布林值
int a = 10, b = 5;
std::cout << "\n關係運算：" << std::endl;
std::cout << a << " > " << b << " : " << (a > b) << std::endl;
std::cout << a << " < " << b << " : " << (a < b) << std::endl;
std::cout << a << " == " << b << " : " << (a == b) << std::endl;
std::cout << a << " != " << b << " : " << (a != b) << std::endl;
// 4. 邏輯運算
bool condition1 = true;
bool condition2 = false;
std::cout << "\n邏輯運算：" << std::endl;
std::cout << "條件1：" << condition1 << std::endl;
std::cout << "條件2：" << condition2 << std::endl;
std::cout << "條件1 AND 條件2：" << (condition1 && condition2) << std::endl;
std::cout << "條件1 OR 條件2：" << (condition1 || condition2) << std::endl;
std::cout << "NOT 條件1：" << (!condition1) << std::endl;
// 5. 實際應用：條件判斷
int score = 85;
bool isPass = (score >= 60); // 分數大於等於60分為及格
std::cout << "\n實際應用：考試成績" << std::endl;
std::cout << "分數：" << score << std::endl;
std::cout << "是否及格：" << isPass << std::endl;
return 0;
}
```

📝 範例5：不同整數類型的區別

```
/*
* 範例2-5：不同整數類型
* short, int, long, long long
* signed（有符號，預設）和 unsigned（無符號）
*/
#include <iostream>
#include <climits>
int main() {
std::cout << "=== 不同整數類型的區別 ===" << std::endl;
// 1. 不同大小的整數類型
short smallNumber = 32767; // 通常 16 位元
int normalNumber = 2147483647; // 通常 32 位元
long long bigNumber = 9223372036854775807LL; // 通常 64 位元
std::cout << "short 變數：" << smallNumber << std::endl;
std::cout << "int 變數：" << normalNumber << std::endl;
std::cout << "long long 變數：" << bigNumber << std::endl;
// 2. 有符號和無符號類型
signed int negativeNumber = -100; // 有符號，可以為負數
unsigned int positiveOnly = 100; // 無符號，只能為非負數
std::cout << "\n有符號和無符號：" << std::endl;
std::cout << "有符號整數：" << negativeNumber << std::endl;
std::cout << "無符號整數：" << positiveOnly << std::endl;
// 警告：無符號整數的陷阱
unsigned int zeroMinusOne = 0;
zeroMinusOne = zeroMinusOne - 1; // 會變成很大的正數
std::cout << "\n無符號整數的陷阱：" << std::endl;
std::cout << "0 - 1 = " << zeroMinusOne << " (在無符號整數中)" << std::endl;
// 3. 類型的大小（位元組數）
std::cout << "\n各類型的大小（位元組）：" << std::endl;
std::cout << "short: " << sizeof(short) << " 位元組" << std::endl;
std::cout << "int: " << sizeof(int) << " 位元組" << std::endl;
std::cout << "long: " << sizeof(long) << " 位元組" << std::endl;
std::cout << "long long: " << sizeof(long long) << " 位元組" << std::endl;
// 4. 類型的最小值和最大值
std::cout << "\n各類型的範圍：" << std::endl;
std::cout << "short: [" << SHRT_MIN << ", " << SHRT_MAX << "]" << std::endl;
std::cout << "int: [" << INT_MIN << ", " << INT_MAX << "]" << std::endl;
std::cout << "unsigned int: [0, " << UINT_MAX << "]" << std::endl;
```

```
std::cout << "long long: [" << LLONG_MIN << ", " << LLONG_MAX << "]" <<
std::endl;
return 0;
}
```

📝 範例6：類型轉換與自動類型推導

```
/*
* 範例2-6：類型轉換與自動類型推導 (C++11)
* 1. 隱式類型轉換（自動轉換）
* 2. 顯式類型轉換（強制轉換）
* 3. auto 關鍵字（C++11）
*/
#include <iostream>
int main() {
std::cout << "=== 類型轉換與自動類型推導 ===" << std::endl;
// 1. 隱式類型轉換（編譯器自動進行）
int intNum = 10;
double doubleNum = 3.14;
// 整數轉浮點數
double result1 = intNum + doubleNum; // intNum 被轉換為 double
std::cout << "隱式轉換：" << intNum << " + " << doubleNum << " = " << result1
<< std::endl;
// 浮點數轉整數（小數部分被捨棄）
int result2 = intNum + doubleNum; // doubleNum 被轉換為 int
std::cout << "隱式轉換（小數丟失）：" << intNum << " + " << doubleNum << " = " <<
result2 << std::endl;
// 2. 顯式類型轉換（C風格）
double pi = 3.14159;
int intPi = (int)pi; // C風格類型轉換
std::cout << "\n顯式轉換（C風格）：" << std::endl;
std::cout << "pi = " << pi << std::endl;
std::cout << "(int)pi = " << intPi << std::endl;
// 3. 顯式類型轉換（C++風格，推薦）
double price = 19.99;
int intPrice = static_cast<int>(price); // C++風格類型轉換
std::cout << "\n顯式轉換（C++風格）：" << std::endl;
std::cout << "price = " << price << std::endl;
```

```
std::cout << "static_cast<int>(price) = " << intPrice << std::endl;
// 4. 自動類型推導（C++11 auto）
auto x = 10; // x 被推導為 int
auto y = 3.14; // y 被推導為 double
auto z = 'A'; // z 被推導為 char
auto flag = true; // flag 被推導為 bool
std::cout << "\n自動類型推導（auto）：" << std::endl;
std::cout << "auto x = 10; // x 是 " << typeid(x).name() << std::endl;
std::cout << "auto y = 3.14; // y 是 " << typeid(y).name() << std::endl;
std::cout << "auto z = 'A'; // z 是 " << typeid(z).name() << std::endl;
std::cout << "auto flag = true; // flag 是 " << typeid(flag).name() <<
std::endl;
// 5. auto 的實際應用
auto sum = x + y; // sum 被推導為 double
std::cout << "\nauto 實際應用：" << std::endl;
std::cout << "x + y = " << sum << " (類型: " << typeid(sum).name() << ")" <<
std::endl;
return 0;
}
```

🔧 練習題

練習2-1：溫度轉換

創建一個程序，將攝氏溫度轉換為華氏溫度。 公式：華氏 = 攝氏 × 9/5 + 32

練習2-2：計算圓的周長和面積

根據半徑計算圓的周長和面積。 公式：

- 周長 = 2 × π × 半徑
- 面積 = π × 半徑²

練習2-3：字符處理

創建一個程序，讓用戶輸入一個小寫字母，將其轉換為大寫字母並顯示。 提示：小寫字母的ASCII碼比大寫字母大32。

練習2-4：布林邏輯

創建一個程序，判斷一個年份是否為閏年。 閏年規則：

1. 能被4整除但不能被100整除，或者
2. 能被400整除

練習2-5：類型轉換練習

創建一個程序，演示以下類型轉換：

1. float 轉 int（注意小數丟失）

2. char 轉 int（顯示ASCII碼）
3. int 轉 char（根據ASCII碼顯示字符）
4. bool 轉 int（true和false轉換為數字）

📚 重點摘要

1. 基本數據類型總結

類型	大小（通常）	範圍（通常）	用途
`int`	4位元組	-2,147,483,648 到 2,147,483,647	整數
`short`	2位元組	-32,768 到 32,767	小整數
`long`	4或8位元組	與int相同或更大	較大整數
`long long`	8位元組	-9.22×10^{18} 到 9.22×10^{18}	很大整數
`float`	4位元組	約6-7位有效數字	單精度浮點數
`double`	8位元組	約15-16位有效數字	雙精度浮點數
`char`	1位元組	-128 到 127 或 0 到 255	字符
`bool`	1位元組	true 或 false	布林值

2. 變數命名規則

- 只能包含字母、數字、底線（_）
- 不能以數字開頭
- 區分大小寫
- 不能使用C++關鍵字
- 建議使用有意義的名稱

3. 變數聲明與初始化

```
int x; // 聲明，未初始化（值不確定）
int y = 10; // 聲明並初始化
int z(20); // 直接初始化（C++風格）
int w{30}; // 列表初始化（C++11，推薦）
```

4. 類型轉換

- **隱式轉換**：編譯器自動進行，可能丟失精度
- **顯式轉換**：程式設計師明確指定
  - C風格：`(目標類型)值`
  - C++風格：`static_cast<目標類型>(值)`（推薦）

5. auto 關鍵字（C++11）

- 讓編譯器自動推導變數類型
- 必須在聲明時初始化
- 不能用于函數參數

🎯 常見問答

Q1: 應該使用 float 還是 double？

A: 在大多數情況下，建議使用 `double`，因為它精度更高。`float` 只在需要節省記憶體時使用。

Q2: unsigned 和 signed 有什麼區別？

A: `signed`（有符號）可以表示正數和負數，`unsigned`（無符號）只能表示非負數。`unsigned` 的正數範圍是 `signed` 的兩倍。

Q3: 為什麼要使用 auto？

A: `auto` 可以簡化代碼，特別是在類型名稱很長時。但要注意，過度使用可能降低代碼可讀性。

Q4: 變數未初始化會怎樣？

A: 未初始化的變數值是不確定的，可能是任意值。使用未初始化的變數可能導致不可預測的結果。

Q5: 如何選擇合適的整數類型？

A: 根據需要的數值範圍選擇：

- 小數值：`short` 或 `int`
- 一般整數：`int`
- 大數值：`long` 或 `long long`
- 只需要非負數：`unsigned` 類型

🚀 下一步

下一課我們將學習：

- **運算符與表達式**（算術、關係、邏輯運算符）
- **運算符優先級**
- **複合賦值運算符**
- **遞增遞減運算符**

💻 完整測試程序

```
/*
* 第2課綜合測試
* 練習使用所有基本數據類型
*/
#include <iostream>
#include <iomanip>
#include <climits>
int main() {
std::cout << "=== 第2課綜合測試 ===" << std::endl;
// 1. 學生資訊
std::string name = "張小明";
int age = 18;
double height = 172.5;
char gender = 'M';
bool isGraduated = false;
std::cout << "\n1. 學生資訊：" << std::endl;
std::cout << "姓名：" << name << std::endl;
std::cout << "年齡：" << age << " 歲" << std::endl;
std::cout << "身高：" << height << " 公分" << std::endl;
std::cout << "性別：" << gender << " (M:男, F:女)" << std::endl;
std::cout << std::boolalpha << "是否畢業：" << isGraduated << std::endl;
// 2. 數學計算
std::cout << "\n2. 數學計算：" << std::endl;
double radius = 7.5;
const double PI = 3.1415926535;
double circleArea = PI * radius * radius;
double circleCircumference = 2 * PI * radius;
std::cout << std::fixed << std::setprecision(2);
std::cout << "圓的半徑：" << radius << " 公分" << std::endl;
std::cout << "圓的面積：" << circleArea << " 平方公分" << std::endl;
std::cout << "圓的周長：" << circleCircumference << " 公分" << std::endl;
// 3. 類型轉換練習
std::cout << "\n3. 類型轉換練習：" << std::endl;
double price = 99.99;
int intPrice = static_cast<int>(price);
std::cout << "商品價格：" << price << " 元" << std::endl;
std::cout << "整數價格（去掉小數）：" << intPrice << " 元" << std::endl;
```

```
// 4. 字符處理
std::cout << "\n4. 字符處理：" << std::endl;
char lowercase = 'g';
char uppercase = static_cast<char>(lowercase - 32);
std::cout << "小寫字母：" << lowercase << std::endl;
std::cout << "轉換為大寫：" << uppercase << std::endl;
// 5. 布林邏輯
std::cout << "\n5. 布林邏輯：" << std::endl;
int score = 85;
bool isExcellent = (score >= 90);
bool isGood = (score >= 80 && score < 90);
bool isPass = (score >= 60);
std::cout << "分數：" << score << std::endl;
std::cout << "是否優秀：" << isExcellent << std::endl;
std::cout << "是否良好：" << isGood << std::endl;
std::cout << "是否及格：" << isPass << std::endl;
// 6. 使用 auto
std::cout << "\n6. 使用 auto 關鍵字：" << std::endl;
auto studentCount = 30; // int
auto averageScore = 78.5; // double
auto firstLetter = 'A'; // char
auto allPassed = true; // bool
std::cout << "學生人數：" << studentCount << " 人" << std::endl;
std::cout << "平均分數：" << averageScore << " 分" << std::endl;
std::cout << "第一個字母：" << firstLetter << std::endl;
std::cout << "全部及格：" << allPassed << std::endl;
return 0;
}
```

輸出結果：

```
=== 第2課綜合測試 ===
1. 學生資訊：
姓名：張小明
年齡：18 歲
身高：172.5 公分
性別：M (M:男, F:女)
是否畢業：false
2. 數學計算：
圓的半徑：7.50 公分
圓的面積：176.71 平方公分
```

```
圓的周長：47.12 公分
3. 類型轉換練習：
商品價格：99.99 元
整數價格（去掉小數）：99 元
4. 字符處理：
小寫字母：g
轉換為大寫：G
5. 布林邏輯：
分數：85
是否優秀：false
是否良好：true
是否及格：true
6. 使用 auto 關鍵字：
學生人數：30 人
平均分數：78.5 分
第一個字母：A
全部及格：true
```

✅ 本課檢查清單

- [ ] 理解 int、double、char、bool 等基本類型
- [ ] 能夠正確聲明和初始化變數
- [ ] 了解不同數據類型的範圍和限制
- [ ] 能夠進行隱式和顯式類型轉換
- [ ] 了解 auto 關鍵字的基本用法
- [ ] 能夠選擇合適的數據類型解決問題

📞 需要幫助？

如果你在練習中遇到問題：

1. 檢查變數類型是否正確
2. 確保使用了正確的格式（字符用單引號，字符串用雙引號）
3. 注意整數除法和浮點數除法的區別
4. 檢查類型轉換是否導致精度丟失

下一課見！ 🚀