第6課：數組與字符串

第6課：數組與字符串

📚 課程目標

- 理解一維和多維數組的概念
- 掌握數組的聲明、初始化和訪問
- 學習C風格字符串的操作
- 掌握C++ string類的使用
- 理解字符數組與字符串的關係

🎯 本課關鍵字

- 數組（Array）
- 字符串（String）
- 字符數組（Character Array）
- `std::string` 類
- 多維數組（Multidimensional Array）

📝 範例1：一維數組基礎

```
/*
* 範例6-1：一維數組基礎
* 數組是相同類型元素的集合，連續存儲在內存中
*/
#include <iostream>
int main() {
std::cout << "=== 一維數組基礎演示 ===" << std::endl;
// 1. 數組的聲明和初始化
std::cout << "1. 數組的聲明和初始化：" << std::endl;
// 方法1：聲明後單獨賦值
int arr1[5]; // 聲明一個包含5個整數的數組
arr1[0] = 10;
arr1[1] = 20;
arr1[2] = 30;
arr1[3] = 40;
arr1[4] = 50;
std::cout << "arr1[2] = " << arr1[2] << std::endl;
```

```
// 方法2：聲明時初始化
int arr2[5] = {1, 2, 3, 4, 5}; // 完全初始化
int arr3[5] = {1, 2, 3}; // 部分初始化，剩餘元素為0
int arr4[] = {10, 20, 30, 40, 50}; // 自動推導數組大小
std::cout << "arr2[3] = " << arr2[3] << std::endl;
std::cout << "arr3[4] = " << arr3[4] << " (部分初始化，未指定值為0)" <<
std::endl;
std::cout << "arr4 的大小自動推導為 " << sizeof(arr4)/sizeof(arr4[0]) <<
std::endl;
// 2. 訪問數組元素
std::cout << "\n2. 訪問數組元素：" << std::endl;
int numbers[] = {10, 20, 30, 40, 50};
int size = sizeof(numbers) / sizeof(numbers[0]); // 計算數組元素個數
std::cout << "數組元素：" << std::endl;
for (int i = 0; i < size; i++) {
std::cout << " numbers[" << i << "] = " << numbers[i] << std::endl;
}
// 3. 數組元素的修改
std::cout << "\n3. 數組元素的修改：" << std::endl;
numbers[2] = 300; // 修改第三個元素
std::cout << "修改後 numbers[2] = " << numbers[2] << std::endl;
// 4. 數組的大小和內存
std::cout << "\n4. 數組的大小和內存：" << std::endl;
std::cout << "數組總大小：sizeof(numbers) = " << sizeof(numbers) << " 位元組" <<
std::endl;
std::cout << "每個元素大小：sizeof(numbers[0]) = " << sizeof(numbers[0]) << " 位
元組" << std::endl;
std::cout << "元素個數：sizeof(numbers)/sizeof(numbers[0]) = "
<< sizeof(numbers)/sizeof(numbers[0]) << std::endl;
// 5. 數組的地址
std::cout << "\n5. 數組的地址：" << std::endl;
for (int i = 0; i < size; i++) {
std::cout << " numbers[" << i << "] 的地址："
<< &numbers[i] << "，值：" << numbers[i] << std::endl;
}
// 6. 數組的注意事項
std::cout << "\n6. 數組的注意事項：" << std::endl;
// 訪問越界是未定義行為
// std::cout << "numbers[10] = " << numbers[10] << std::endl; // 錯誤！
// 數組不能直接賦值
```

```
int b[3];
// b = a; // 錯誤！數組不能直接賦值
// 需要逐元素複製
for (int i = 0; i < 3; i++) {
b[i] = a[i];
}
std::cout << "數組複製成功" << std::endl;
return 0;
}
```

📝 範例2：多維數組

```
/*
* 範例6-2：多維數組
* 多維數組是數組的數組，常用的是二維數組（矩陣）
*/
#include <iostream>
#include <iomanip>
int main() {
std::cout << "=== 多維數組演示 ===" << std::endl;
// 1. 二維數組的聲明和初始化
std::cout << "1. 二維數組的聲明和初始化：" << std::endl;
// 方法1：逐行初始化
int matrix1[3][4] = {
{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12}
};
// 方法2：連續初始化
int matrix2[2][3] = {1, 2, 3, 4, 5, 6};
// 方法3：部分初始化
int matrix3[3][3] = {
{1}, // 第一行：1, 0, 0
{4, 5}, // 第二行：4, 5, 0
{7, 8, 9} // 第三行：7, 8, 9
};
// 2. 訪問二維數組元素
std::cout << "\n2. 訪問二維數組元素：" << std::endl;
```

```
std::cout << "matrix1[2][3] = " << matrix1[2][3] << std::endl;
std::cout << "matrix2[1][2] = " << matrix2[1][2] << std::endl;
std::cout << "matrix3[0][2] = " << matrix3[0][2] << " (部分初始化，未指定值為0)"
<< std::endl;
// 3. 遍歷二維數組
std::cout << "\n3. 遍歷二維數組（矩陣形式）：" << std::endl;
std::cout << "matrix1 的內容：" << std::endl;
for (int i = 0; i < 3; i++) { // 行
for (int j = 0; j < 4; j++) { // 列
std::cout << std::setw(4) << matrix1[i][j];
}
std::cout << std::endl;
}
// 4. 三維數組
std::cout << "\n4. 三維數組：" << std::endl;
int cube[2][3][4] = {
{ // 第一層
{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12}
},
{ // 第二層
{13, 14, 15, 16},
{17, 18, 19, 20},
{21, 22, 23, 24}
}
};
std::cout << "cube[1][2][3] = " << cube[1][2][3] << std::endl;
// 5. 多維數組的內存佈局
std::cout << "\n5. 多維數組的內存佈局：" << std::endl;
int arr2D[2][3] = {{1, 2, 3}, {4, 5, 6}};
std::cout << "二維數組的內存地址：" << std::endl;
for (int i = 0; i < 2; i++) {
for (int j = 0; j < 3; j++) {
std::cout << "arr2D[" << i << "][" << j << "] 地址: "
<< &arr2D[i][j] << " 值: " << arr2D[i][j] << std::endl;
}
}
// 6. 多維數組的應用：矩陣運算
std::cout << "\n6. 多維數組的應用：矩陣運算（矩陣相加）" << std::endl;
```

```
int B[2][3] = {{6, 5, 4}, {3, 2, 1}};
int C[2][3]; // 存儲結果
// 矩陣相加
for (int i = 0; i < 2; i++) {
for (int j = 0; j < 3; j++) {
C[i][j] = A[i][j] + B[i][j];
}
}
// 顯示結果
std::cout << "矩陣 A：" << std::endl;
for (int i = 0; i < 2; i++) {
for (int j = 0; j < 3; j++) {
std::cout << std::setw(3) << A[i][j];
}
std::cout << std::endl;
}
std::cout << "矩陣 B：" << std::endl;
for (int i = 0; i < 2; i++) {
for (int j = 0; j < 3; j++) {
std::cout << std::setw(3) << B[i][j];
}
std::cout << std::endl;
}
std::cout << "矩陣 C = A + B：" << std::endl;
for (int i = 0; i < 2; i++) {
for (int j = 0; j < 3; j++) {
std::cout << std::setw(3) << C[i][j];
}
std::cout << std::endl;
}
// 7. 多維數組作為函數參數
std::cout << "\n7. 多維數組作為函數參數：" << std::endl;
// 注意：傳遞多維數組時，除了第一維，其他維度必須指定大小
auto printMatrix = [](int mat[][3], int rows) {
for (int i = 0; i < rows; i++) {
for (int j = 0; j < 3; j++) {
std::cout << std::setw(3) << mat[i][j];
}
std::cout << std::endl;
}
};
```

```
std::cout << "通過函數打印矩陣：" << std::endl;
printMatrix(A, 2);
return 0;
}
```

📝 範例3：C風格字符串

```
/*
* 範例6-3：C風格字符串
* C風格字符串是以空字符('\0')結尾的字符數組
*/
#include <iostream>
#include <cstring> // C風格字符串函數頭文件
int main() {
std::cout << "=== C風格字符串演示 ===" << std::endl;
// 1. C風格字符串的創建
std::cout << "1. C風格字符串的創建：" << std::endl;
// 方法1：字符數組（需要手動添加'\0'）
char str1[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
// 方法2：字符串字面量（自動添加'\0'）
char str2[] = "Hello";
// 方法3：指針形式
const char* str3 = "Hello"; // 字符串字面量是常量
std::cout << "str1: " << str1 << std::endl;
std::cout << "str2: " << str2 << std::endl;
std::cout << "str3: " << str3 << std::endl;
// 2. 字符串長度
std::cout << "\n2. 字符串長度：" << std::endl;
std::cout << "strlen(str1): " << strlen(str1) << std::endl;
std::cout << "sizeof(str1): " << sizeof(str1) << " (包含'\\0')" << std::endl;
// 3. 字符串複製
std::cout << "\n3. 字符串複製：" << std::endl;
char source[] = "Hello World";
char destination[20];
strcpy(destination, source); // 複製字符串
std::cout << "複製後 destination: " << destination << std::endl;
// 安全版本：strncpy
char dest2[10];
```

```
strncpy(dest2, source, 9); // 最多複製9個字符
dest2[9] = '\0'; // 確保以'\0'結尾
std::cout << "安全複製後 dest2: " << dest2 << std::endl;
// 4. 字符串連接
std::cout << "\n4. 字符串連接：" << std::endl;
char hello[] = "Hello";
char world[] = " World";
char greeting[20];
strcpy(greeting, hello);
strcat(greeting, world); // 連接字符串
std::cout << "連接後: " << greeting << std::endl;
// 5. 字符串比較
std::cout << "\n5. 字符串比較：" << std::endl;
char strA[] = "apple";
char strB[] = "banana";
char strC[] = "apple";
int result1 = strcmp(strA, strB); // 比較字符串
int result2 = strcmp(strA, strC);
std::cout << "strcmp(\"apple\", \"banana\"): " << result1 << std::endl;
std::cout << "strcmp(\"apple\", \"apple\"): " << result2 << std::endl;
if (result1 < 0) {
std::cout << "\"apple\" 小於 \"banana\"" << std::endl;
}
if (result2 == 0) {
std::cout << "\"apple\" 等於 \"apple\"" << std::endl;
}
// 6. 字符串查找
std::cout << "\n6. 字符串查找：" << std::endl;
char text[] = "Hello World, welcome to C++";
char* found;
// 查找字符
found = strchr(text, 'W');
if (found != nullptr) {
std::cout << "找到 'W' 在位置: " << (found - text) << std::endl;
std::cout << "從該位置開始的字符串: " << found << std::endl;
}
// 查找子字符串
found = strstr(text, "welcome");
if (found != nullptr) {
std::cout << "找到子字符串 \"welcome\" 在位置: " << (found - text) << std::endl;
}
```

```
// 7. 字符串轉換
std::cout << "\n7. 字符串轉換：" << std::endl;
// 字符串轉數字
char numStr[] = "12345";
int number = atoi(numStr); // ASCII to integer
std::cout << "atoi(\"12345\"): " << number << std::endl;
char floatStr[] = "3.14159";
double pi = atof(floatStr); // ASCII to float
std::cout << "atof(\"3.14159\"): " << pi << std::endl;
// 數字轉字符串
char buffer[20];
int value = 42;
sprintf(buffer, "數字是: %d", value); // 格式化輸出到字符串
std::cout << "sprintf 結果: " << buffer << std::endl;
// 8. 字符串分割
std::cout << "\n8. 字符串分割：" << std::endl;
char csv[] = "apple,banana,orange,grape";
char* token;
std::cout << "分割 \"" << csv << "\"：" << std::endl;
token = strtok(csv, ","); // 第一次調用
while (token != nullptr) {
std::cout << " " << token << std::endl;
token = strtok(nullptr, ","); // 後續調用
}
return 0;
}
```

📝 範例4：C++ string類

```
/*
* 範例6-4：C++ string類
* std::string 是C++標準庫中的字符串類，比C風格字符串更安全易用
*/
#include <iostream>
#include <string> // string類頭文件
#include <algorithm>
int main() {
std::cout << "=== C++ string類演示 ===" << std::endl;
```

```
// 1. string對象的創建和初始化
std::cout << "1. string對象的創建和初始化：" << std::endl;
// 方法1：默認構造函數
std::string str1;
std::cout << "默認構造: \"" << str1 << "\"" << std::endl;
// 方法2：使用C風格字符串初始化
std::string str2 = "Hello";
std::cout << "從C字符串初始化: \"" << str2 << "\"" << std::endl;
// 方法3：使用另一個string對象初始化
std::string str3(str2);
std::cout << "從string對象初始化: \"" << str3 << "\"" << std::endl;
// 方法4：使用多個相同字符
std::string str4(5, '*');
std::cout << "5個'*': \"" << str4 << "\"" << std::endl;
// 方法5：使用子字符串
std::string str5 = "Hello World";
std::string str6(str5, 6, 5); // 從位置6開始，長度5
std::cout << "子字符串: \"" << str6 << "\"" << std::endl;
// 2. 字符串操作
std::cout << "\n2. 字符串操作：" << std::endl;
std::string s = "Hello";
// 獲取長度
std::cout << "長度: " << s.length() << " (或 " << s.size() << ")" << std::endl;
// 檢查是否為空
std::cout << "是否為空: " << (s.empty() ? "是" : "否") << std::endl;
// 訪問字符
std::cout << "第一個字符: " << s[0] << std::endl;
std::cout << "第二個字符: " << s.at(1) << std::endl; // 帶範圍檢查
// 修改字符
s[1] = 'a';
std::cout << "修改後: \"" << s << "\"" << std::endl;
// 3. 字符串連接
std::cout << "\n3. 字符串連接：" << std::endl;
std::string s1 = "Hello";
std::string s2 = " World";
// 方法1：使用+運算符
std::string s3 = s1 + s2;
std::cout << "使用+: \"" << s3 << "\"" << std::endl;
// 方法2：使用+=運算符
s1 += s2;
std::cout << "使用+=: \"" << s1 << "\"" << std::endl;
```

```
// 方法3：使用append函數
std::string s4 = "Hello";
s4.append(" C++");
std::cout << "使用append: \"" << s4 << "\"" << std::endl;
// 4. 字符串比較
std::cout << "\n4. 字符串比較：" << std::endl;
std::string a = "apple";
std::string b = "banana";
std::string c = "apple";
// 使用比較運算符
std::cout << "\"apple\" == \"banana\": " << (a == b) << std::endl;
std::cout << "\"apple\" != \"banana\": " << (a != b) << std::endl;
std::cout << "\"apple\" < \"banana\": " << (a < b) << std::endl;
std::cout << "\"apple\" == \"apple\": " << (a == c) << std::endl;
// 使用compare函數
int result = a.compare(b);
if (result < 0) {
std::cout << "\"apple\" 小於 \"banana\"" << std::endl;
}
// 5. 子字符串操作
std::cout << "\n5. 子字符串操作：" << std::endl;
std::string text = "Hello World, welcome to C++ programming";
// 提取子字符串
std::string sub1 = text.substr(6, 5); // 從位置6開始，長度5
std::cout << "substr(6, 5): \"" << sub1 << "\"" << std::endl;
// 查找子字符串
size_t pos = text.find("welcome");
if (pos != std::string::npos) {
std::cout << "找到 \"welcome\" 在位置: " << pos << std::endl;
}
// 查找字符
pos = text.find('W');
if (pos != std::string::npos) {
std::cout << "找到 'W' 在位置: " << pos << std::endl;
}
// 從末尾查找
pos = text.rfind('o');
if (pos != std::string::npos) {
std::cout << "從末尾找到 'o' 在位置: " << pos << std::endl;
}
```

```
std::cout << "\n6. 字符串修改：" << std::endl;
std::string str = "I like C programming";
// 插入
str.insert(7, "++ ");
std::cout << "插入後: \"" << str << "\"" << std::endl;
// 替換
str.replace(2, 4, "love"); // 從位置2開始，替換4個字符
std::cout << "替換後: \"" << str << "\"" << std::endl;
// 刪除
str.erase(15, 12); // 從位置15開始，刪除12個字符
std::cout << "刪除後: \"" << str << "\"" << std::endl;
// 7. 字符串轉換
std::cout << "\n7. 字符串轉換：" << std::endl;
// 數字轉字符串
int num = 42;
std::string numStr = std::to_string(num);
std::cout << "整數轉字符串: \"" << numStr << "\"" << std::endl;
double pi = 3.14159;
std::string piStr = std::to_string(pi);
std::cout << "浮點數轉字符串: \"" << piStr << "\"" << std::endl;
// 字符串轉數字
std::string intStr = "123";
int intVal = std::stoi(intStr); // string to int
std::cout << "字符串轉整數: " << intVal << std::endl;
std::string doubleStr = "3.14";
double doubleVal = std::stod(doubleStr); // string to double
std::cout << "字符串轉浮點數: " << doubleVal << std::endl;
// 8. 字符串遍歷
std::cout << "\n8. 字符串遍歷：" << std::endl;
std::string word = "Hello";
std::cout << "使用下標遍歷: ";
for (size_t i = 0; i < word.length(); i++) {
std::cout << word[i] << " ";
}
std::cout << std::endl;
std::cout << "使用範圍for循環: ";
for (char ch : word) {
std::cout << ch << " ";
}
std::cout << std::endl;
```

```
std::cout << "\n9. 字符串算法：" << std::endl;
std::string demo = "Hello World";
// 轉換大小寫（需要#include <algorithm>和<cctype>）
std::transform(demo.begin(), demo.end(), demo.begin(), ::toupper);
std::cout << "轉換為大寫: \"" << demo << "\"" << std::endl;
std::transform(demo.begin(), demo.end(), demo.begin(), ::tolower);
std::cout << "轉換為小寫: \"" << demo << "\"" << std::endl;
// 反轉字符串
std::reverse(demo.begin(), demo.end());
std::cout << "反轉後: \"" << demo << "\"" << std::endl;
return 0;
}
```

📝 範例5：字符數組與字符串綜合應用

```
/*
* 範例6-5：字符數組與字符串綜合應用
* 綜合應用C風格字符串和C++ string類
*/
#include <iostream>
#include <string>
#include <cstring>
#include <algorithm>
#include <vector>
int main() {
std::cout << "=== 字符數組與字符串綜合應用 ===" << std::endl;
// 1. 密碼驗證系統
std::cout << "\n1. 密碼驗證系統：" << std::endl;
const char* correctPassword = "secret123";
char userPassword[50];
// 模擬用戶輸入（實際應用中會從輸入流讀取）
strcpy(userPassword, "secret123");
if (strcmp(userPassword, correctPassword) == 0) {
std::cout << "密碼正確！登入成功。" << std::endl;
} else {
std::cout << "密碼錯誤！" << std::endl;
}
// 2. 字符串統計
```

```
std::cout << "\n2. 字符串統計：" << std::endl;
std::string text = "The quick brown fox jumps over the lazy dog";
// 統計單詞數
int wordCount = 0;
bool inWord = false;
for (char ch : text) {
if (std::isspace(ch)) {
inWord = false;
} else if (!inWord) {
inWord = true;
wordCount++;
}
}
std::cout << "文本: \"" << text << "\"" << std::endl;
std::cout << "單詞數: " << wordCount << std::endl;
// 統計字母頻率
int letterCount[26] = {0};
for (char ch : text) {
if (std::isalpha(ch)) {
char lower = std::tolower(ch);
letterCount[lower - 'a']++;
}
}
std::cout << "字母頻率:" << std::endl;
for (int i = 0; i < 26; i++) {
if (letterCount[i] > 0) {
std::cout << " " << char('a' + i) << ": " << letterCount[i] << std::endl;
}
}
// 3. 字符串反轉
std::cout << "\n3. 字符串反轉：" << std::endl;
std::string original = "Hello World";
std::string reversed = original;
std::reverse(reversed.begin(), reversed.end());
std::cout << "原始: \"" << original << "\"" << std::endl;
std::cout << "反轉: \"" << reversed << "\"" << std::endl;
// 4. 回文判斷
std::cout << "\n4. 回文判斷：" << std::endl;
std::string palindrome1 = "racecar";
std::string palindrome2 = "hello";
```

```
std::string cleanStr;
// 移除空格和標點，轉換為小寫
for (char ch : str) {
if (std::isalnum(ch)) {
cleanStr += std::tolower(ch);
}
}
// 判斷是否回文
std::string reversed = cleanStr;
std::reverse(reversed.begin(), reversed.end());
return cleanStr == reversed;
};
std::cout << "\"" << palindrome1 << "\" 是回文嗎？"
<< (isPalindrome(palindrome1) ? "是" : "否") << std::endl;
std::cout << "\"" << palindrome2 << "\" 是回文嗎？"
<< (isPalindrome(palindrome2) ? "是" : "否") << std::endl;
// 5. CSV數據解析
std::cout << "\n5. CSV數據解析：" << std::endl;
std::string csvData = "John,Doe,30,New York\nJane,Smith,25,Los
Angeles\nBob,Johnson,35,Chicago";
std::vector<std::vector<std::string>> records;
std::string line;
size_t start = 0, end = 0;
// 按行分割
while ((end = csvData.find('\n', start)) != std::string::npos) {
line = csvData.substr(start, end - start);
start = end + 1;
// 按逗號分割每行
std::vector<std::string> fields;
size_t fieldStart = 0, fieldEnd = 0;
while ((fieldEnd = line.find(',', fieldStart)) != std::string::npos) {
fields.push_back(line.substr(fieldStart, fieldEnd - fieldStart));
fieldStart = fieldEnd + 1;
}
fields.push_back(line.substr(fieldStart));
records.push_back(fields);
}
// 處理最後一行
if (start < csvData.length()) {
line = csvData.substr(start);
```

```
size_t fieldStart = 0, fieldEnd = 0;
while ((fieldEnd = line.find(',', fieldStart)) != std::string::npos) {
fields.push_back(line.substr(fieldStart, fieldEnd - fieldStart));
fieldStart = fieldEnd + 1;
}
fields.push_back(line.substr(fieldStart));
records.push_back(fields);
}
// 顯示解析結果
std::cout << "解析結果:" << std::endl;
for (const auto& record : records) {
std::cout << " 姓名: " << record[0] << " " << record[1]
<< ", 年齡: " << record[2]
<< ", 城市: " << record[3] << std::endl;
}
// 6. 簡單文本加密
std::cout << "\n6. 簡單文本加密（凱撒密碼）:" << std::endl;
std::string plainText = "HELLO WORLD";
int shift = 3;
std::string encrypted;
for (char ch : plainText) {
if (std::isalpha(ch)) {
char base = std::isupper(ch) ? 'A' : 'a';
encrypted += static_cast<char>((ch - base + shift) % 26 + base);
} else {
encrypted += ch;
}
}
std::cout << "原始文本: \"" << plainText << "\"" << std::endl;
std::cout << "加密文本: \"" << encrypted << "\"" << std::endl;
// 解密
std::string decrypted;
for (char ch : encrypted) {
if (std::isalpha(ch)) {
char base = std::isupper(ch) ? 'A' : 'a';
decrypted += static_cast<char>((ch - base - shift + 26) % 26 + base);
} else {
decrypted += ch;
}
}
std::cout << "解密文本: \"" << decrypted << "\"" << std::endl;
```

```
// 7. 字符串排序
std::cout << "\n7. 字符串排序:" << std::endl;
std::vector<std::string> words = {"banana", "apple", "cherry", "date",
"elderberry"};
std::cout << "排序前:" << std::endl;
for (const auto& word : words) {
std::cout << " " << word << std::endl;
}
std::sort(words.begin(), words.end());
std::cout << "排序後:" << std::endl;
for (const auto& word : words) {
std::cout << " " << word << std::endl;
}
// 按長度排序
std::sort(words.begin(), words.end(),
[](const std::string& a, const std::string& b) {
return a.length() < b.length();
});
std::cout << "按長度排序後:" << std::endl;
for (const auto& word : words) {
std::cout << " " << word << " (長度: " << word.length() << ")" << std::endl;
}
return 0;
}
```

🔧 練習題

練習6-1：數組統計

創建一個程序，實現以下功能：

1. 從用戶輸入讀取10個整數到數組中
2. 計算並輸出數組的平均值
3. 找出數組中的最大值和最小值
4. 統計正數、負數和零的個數

練習6-2：矩陣運算

創建一個程序，實現以下矩陣運算：

1. 輸入兩個3x3矩陣
2. 計算矩陣的和
3. 計算矩陣的乘積

4. 計算矩陣的轉置
5. 檢查矩陣是否對稱

練習6-3：字符串處理

創建一個字符串處理程序，實現以下功能：

1. 統計字符串中單詞的個數
2. 反轉字符串中的單詞順序（如"Hello World"變為"World Hello"）
3. 檢查字符串是否為回文
4. 將字符串中的每個單詞首字母大寫

練習6-4：簡單數據庫

創建一個簡單的學生信息管理系統：

1. 使用結構體數組存儲學生信息（姓名、學號、成績）
2. 實現添加、刪除、修改、查詢功能
3. 按成績排序學生
4. 計算全班平均分和最高分

練習6-5：文本分析器

創建一個文本分析器，實現以下功能：

1. 讀取一段文本
2. 統計字符數、單詞數、行數
3. 找出最長的單詞
4. 統計每個單詞的出現頻率
5. 生成單詞頻率報告

📚 重點摘要

1. 數組要點

- **定義**：相同類型元素的集合，連續存儲
- **聲明**：`type name[size];`
- **訪問**：使用下標 `array[index]` ，下標從0開始
- **大小**：`sizeof(array) / sizeof(array[0])` 計算元素個數

2. 多維數組

- **本質**：數組的數組
- **聲明**：`type name[rows][cols];`
- **內存佈局**：按行優先順序存儲

3. C風格字符串

- **特點**：以 `'\0'` 結尾的字符數組
- **頭文件**：`<cstring>`
- **常用函數**：`strlen`, `strcpy`, `strcat`, `strcmp`, `strstr`

4. C++ string類

- **優點**：更安全、更易用、自動管理內存
- **頭文件**：`<string>`
- **常用操作**：`length()`, `append()`, `find()`, `substr()`, `compare()`

5. 選擇指南

- **數組**：固定大小、性能關鍵、簡單數據
- **vector**（下一課）：動態大小、需要自動擴展
- **C風格字符串**：與C代碼交互、性能關鍵
- **string類**：日常使用、安全性要求高

🎯 常見問答

Q1: 數組和vector有什麼區別？

A: 數組大小固定，vector可以動態擴展。數組性能稍好，vector更靈活安全。現代C++推薦使用vector。

Q2: C風格字符串和string類如何轉換？

A:

- C風格字符串轉string：`std::string str = cstr;`
- string轉C風格字符串：`str.c_str()`

Q3: 如何避免數組越界？

A:

1. 使用範圍檢查（如 `at()` 方法）
2. 使用標準庫容器（如vector）
3. 始終檢查索引是否有效

Q4: 多維數組在內存中如何存儲？

A: 多維數組在內存中是連續存儲的，按行優先順序（C/C++）或列優先順序（某些語言）排列。

Q5: 什麼時候使用字符數組，什麼時候使用string？

A: 需要與C代碼交互或極致性能時使用字符數組，其他情況使用string類更安全方便。

🚀 下一步

下一課我們將學習：

- **函數**（定義、聲明、參數傳遞、返回值）
- **函數重載**
- **遞歸函數**
- **函數模板**

💻 完整測試程序

```
/*
* 第6課綜合測試
* 綜合應用數組與字符串
*/
#include <iostream>
#include <string>
#include <cstring>
#include <iomanip>
int main() {
std::cout << "=== 第6課綜合測試：數組與字符串 ===" << std::endl;
// 1. 一維數組測試
std::cout << "\n1. 一維數組測試：" << std::endl;
int scores[5] = {85, 92, 78, 95, 88};
int sum = 0;
std::cout << "學生成績：" << std::endl;
for (int i = 0; i < 5; i++) {
std::cout << " 學生" << (i + 1) << ": " << scores[i] << " 分" << std::endl;
sum += scores[i];
}
double average = static_cast<double>(sum) / 5;
std::cout << "平均分：" << average << " 分" << std::endl;
// 2. 二維數組測試
std::cout << "\n2. 二維數組測試（矩陣）：" << std::endl;
int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
std::cout << "3x3 矩陣：" << std::endl;
for (int i = 0; i < 3; i++) {
for (int j = 0; j < 3; j++) {
std::cout << std::setw(4) << matrix[i][j];
}
std::cout << std::endl;
}
// 3. C風格字符串測試
std::cout << "\n3. C風格字符串測試：" << std::endl;
char firstName[20] = "John";
char lastName[20] = "Doe";
char fullName[40];
strcpy(fullName, firstName);
strcat(fullName, " ");
strcat(fullName, lastName);
std::cout << "全名：" << fullName << std::endl;
```

```
std::cout << "姓名長度：" << strlen(fullName) << std::endl;
// 4. C++ string類測試
std::cout << "\n4. C++ string類測試：" << std::endl;
std::string city = "Taipei";
std::string country = "Taiwan";
std::string location = city + ", " + country;
std::cout << "地點：" << location << std::endl;
std::cout << "地點長度：" << location.length() << std::endl;
std::cout << "第一個字符：" << location[0] << std::endl;
// 5. 字符串比較
std::cout << "\n5. 字符串比較：" << std::endl;
std::string fruit1 = "apple";
std::string fruit2 = "banana";
if (fruit1 < fruit2) {
std::cout << "\"" << fruit1 << "\" 在字典序中小於 \"" << fruit2 << "\"" <<
std::endl;
}
// 6. 字符串查找和提取
std::cout << "\n6. 字符串查找和提取：" << std::endl;
std::string sentence = "The quick brown fox jumps over the lazy dog";
size_t foxPos = sentence.find("fox");
if (foxPos != std::string::npos) {
std::cout << "找到 \"fox\" 在位置：" << foxPos << std::endl;
std::string afterFox = sentence.substr(foxPos);
std::cout << "從該位置開始的字符串：" << afterFox << std::endl;
}
// 7. 字符數組與字符串轉換
std::cout << "\n7. 字符數組與字符串轉換：" << std::endl;
// C風格字符串轉string
char cstr[] = "Hello from C";
std::string cppstr = cstr;
std::cout << "C風格字符串轉string：" << cppstr << std::endl;
// string轉C風格字符串
const char* backToCstr = cppstr.c_str();
std::cout << "string轉C風格字符串：" << backToCstr << std::endl;
// 8. 簡單字符串加密
std::cout << "\n8. 簡單字符串加密（凱撒密碼，偏移量3）：" << std::endl;
std::string plain = "HELLO";
std::string encrypted;
for (char ch : plain) {
```

```
char base = std::isupper(ch) ? 'A' : 'a';
encrypted += static_cast<char>((ch - base + 3) % 26 + base);
} else {
encrypted += ch;
}
}
std::cout << "明文：" << plain << std::endl;
std::cout << "密文：" << encrypted << std::endl;
return 0;
}
```

輸出結果：

```
=== 第6課綜合測試：數組與字符串 ===
1. 一維數組測試：
學生成績：
學生1: 85 分
學生2: 92 分
學生3: 78 分
學生4: 95 分
學生5: 88 分
平均分：87.6 分
2. 二維數組測試（矩陣）：
3x3 矩陣：
1 2 3
4 5 6
7 8 9
3. C風格字符串測試：
全名：John Doe
姓名長度：8
4. C++ string類測試：
地點：Taipei, Taiwan
地點長度：13
第一個字符：T
5. 字符串比較：
"apple" 在字典序中小於 "banana"
6. 字符串查找和提取：
找到 "fox" 在位置：16
從該位置開始的字符串：fox jumps over the lazy dog
7. 字符數組與字符串轉換：
C風格字符串轉string：Hello from C
```

```
string轉C風格字符串：Hello from C
8. 簡單字符串加密（凱撒密碼，偏移量3）：
明文：HELLO
密文：KHOOR
```

✅ 本課檢查清單

- [ ] 理解一維數組的聲明、初始化和訪問
- [ ] 能夠使用多維數組（特別是二維數組）
- [ ] 掌握C風格字符串的基本操作
- [ ] 能夠使用C++ string類進行字符串處理
- [ ] 理解字符數組與字符串的關係和轉換
- [ ] 能夠解決簡單的字符串處理問題
- [ ] 了解數組和字符串的常見應用場景

📞 需要幫助？

如果你在練習中遇到問題：

1. 確保數組下標從0開始，不要越界
2. 對於C風格字符串，記得以'\0'結尾
3. 使用string類時，注意方法的返回值類型
4. 多維數組作為函數參數時，需要指定除第一維外的大小
5. 處理字符串時考慮邊界情況（空字符串、超長字符串等）

下一課見！ 🚀