C++程式設計：前置處理

**目錄**

[C++程式設計：前置處理 1](#_Toc205648552)

[1. 引入標頭檔() 1](#_Toc205648553)

[2. #define 巨集定義 1](#_Toc205648554)

[3. 條件編譯（Conditional Compilation） 1](#_Toc205648555)

[3.1 範例：防止重複定義(判斷是否已定義) 2](#_Toc205648556)

[4. 巨集函數（Macro Function） 2](#_Toc205648557)

[4.1 範例：巨集版本的 assert 2](#_Toc205648558)

[5. `\_\_LINE\_\_` 和 `\_\_FILE\_\_` 3](#_Toc205648559)

前置處理（Preprocessing）是在程式正式編譯前，先對原始碼進行「預處理」。

# 引入標頭檔()

#include是常見的基本指令，通常搭配標準函式庫使用。

會將整個檔案的內容「複製貼上」到這一行的位置。

預處理階段就會完成，不是在編譯期才做。

#include <iostream>  // 系統標頭檔（尖括號）

#include "myfile.h"  // 自訂檔案（雙引號）

# #define 巨集定義

用來定義常數、開關旗標（像 DEBUG）、或簡化重複的程式碼片段。

// 在編譯前，會把程式中出現的 N 全部替換成 10

#define N 10

# 條件編譯（Conditional Compilation）

可依據#define的值來決定哪些程式碼會被編譯，控制輸出或除錯等行為。

#if 條件

程式碼

#endif

#include <iostream>

using namespace std;

#define DEBUG 1

int main(){

    #if DEBUG == 1

        cout << "除錯資訊" << endl;

    #endif

    return 0;

}

另外，可以在編譯時動態指定 define 值：

g++ -DDEBUG=1 program.cpp

g++ -DDLEVEL=8 program.cpp

## 範例：防止重複定義(判斷是否已定義)

#ifdef DEBUG

    // DEBUG 有被定義才會執行

#endif

#ifndef DEBUG

    // DEBUG 沒被定義才會執行

#endif

# 巨集函數（Macro Function）

巨集函數是利用#define實作具參數的函式替代樣板。

## 範例：巨集版本的 assert

#include <iostream>

using namespace std;

// 此巨集用於除錯，若條件不符就輸出錯誤並結束程式

#define assert(cond)

    if (!(cond)) {

        cout << "Assertion failed: " << #cond << endl; \ // #cond 是字串化運算子，會將參數變為字串

        exit(0);

    }

int main(void) {

    int i = 0;

    assert(i == 1);  // 輸出：Assertion failed: i == 1

    return 0;

}

# `\_\_LINE\_\_` 和 `\_\_FILE\_\_`

可搭配巨集 assert 使用，印出錯誤所在的「檔名」與「行數」

#include <iostream>

using namespace std;

// 此巨集用於除錯，若條件不符就輸出錯誤並結束程式

#define assert(cond)

    if (!(cond)) {

        cout << "Assertion failed: " << #cond << " "

             << "file " << \_\_FILE\_\_ << " line " << \_\_LINE\_\_ << endl;

        exit(0);

    }

int main(void) {

    int i = 0;

    assert(i == 1);  // 輸出：Assertion failed: i == 1

    return 0;

}