國立屏東大學 資訊工程學系 物件導向程式設計

2. Hello World! 您的第一隻C++語言程式

由於C承襲了C語言的高效率,與廣大的C語言開發人員,自其誕生以來[]C++語言一直都是最受歡迎與最 受重視的程式語言之一,同時也是幾乎所有大專校院資訊相關系所的必修程式語言。本章後續將針對其程 式設計的開發流程加以說明,並透過一個簡單的程式(Hello World)來示範在Linux與Mac OS系統上的詳細 開發過程。=====-程式開發流程====== < imgcaption Fig CPPDEV center C程式開發流程>



on>

使用C++語言開發程式的流程十分簡單,可概分為<mark>撰寫原始程式碼(source code)[[編譯(compile)原始程式</mark> <mark>與執行(run/execute)</mark>,請參考figure ##,其步驟包含:

- 1. 以文字檔編輯軟體撰寫原始程式。
- 2. 完成後,產生副檔名.cpp的原始程式檔。
- 3. 將原始程式交由編譯器(Compiler)編譯。
- 4. 编譯成功後,會產生目的檔(Objective File)]
- 5. 符合作業平台的可執行檔(例如Windows平台下的EXE檔)。
- 6. 若是編譯時發生錯誤,則重新回到步驟1進行除錯。
- 7. 最後就可在作業平台上執行程式。

具體來說,首先我們必須使用程式碼編輯器[Code Editor]在figure ##中標示為A之處)來撰寫程式,完成 後會產生C語言的原始程式[Source Code]在<imgref Fig_CPPDEV>中標示為B之處)檔案,為方便辨識起 見,其副檔名通常會命名為[]cpp][]我們通常將撰寫原始程式的過程稱為「寫程式[]]]Coding[]或「編程」。 值得注意的是,由於C++語言的原始程式檔案格式為純文字格式,所以你可以選擇使用任意的文字編輯器 [Text Editor] 來進行原始程式的撰寫,不過程式碼編輯器通常提供了許多便利程式開發的功能,例如程式 語言的語法突顯[]Syntax Highlighting[]]自動縮排[Auto Indenting]]]自動完成[Auto Completion]等功能, 對於程式設計師而言是更方便的選擇[] <note> ++++ 程式碼編輯器 我們可使用任何一套文字檔編輯軟 體(text editor)來進行程式的撰寫,再進行後續的編譯與執行。或者您也可以使用如Dev-C++[]Code::Blocks或是Microsoft Visual Studio這一類的IDE(integrated development environment 整合 式開發環境),來進行程式的撰寫、編譯、執行甚至除錯等工作。您可以自行尋找適合或喜好的開發方式。 </note> 當原始程式撰寫完成後,還必須交由編譯器[]Compiler[]在<imgref Fig_CPPDEV>中標示為C之處) 來將其轉換為可執行檔[]Executable File[]的格式,才能夠在作業系統內加以執行。在這個轉換的過程中, 編譯器會先確認原始程式碼是否符合C++語言的語法規則,然後才會將其轉換為目的檔[]Object Code[] 在<imgref Fig_CPPDEV>中標示為D之處),其中包含有可以在CPU上執行的機器碼[]Machine Code[]指令 以及其執行所需的資料。如果在編譯時發現錯誤,則必須重新審視原始程式碼的正確性,將錯誤加以修正

後再重新進行編譯,直到成功為止才會產生目的檔。我們將這種找出程式碼的錯誤並加以修正的過程,稱 之為除錯[]Debug[][]對於程式設計師而言,除錯是非常重要的能力之一。一旦成功地產生目的檔後[]C++ 語言的編譯器就會自動啟動聯結程式 []Linker[]在<imgref Fig_CPPDEV>中標示為E之處) , 來將目的檔與 相關的函式庫「Lbrary「在<imgref Fig CPPDEV>中標示為F之處)進行結合,以產生符合作業系統要求的 可執行檔□Executable File□在<imgref Fig CPPDEV>中標示為G之處),例如Microsoft Windows平台下 的EXE檔,或是Unix/Linux系統中的ELF檔;當然,如果在進行聯結時發生錯誤,一樣必須進行原始程式碼 的除錯直到聯結成功為止。最後,所產生的可執行檔就可以透過作業系統加以執行。 ===== - 編寫程 式(coding) ===== 由於C語言的原始程式檔案格式為純文字,所以您可以使用任一套文字檔編輯軟 體(text editor)來編寫程式。請使用您偏好的文字檔編輯軟體來編寫一個名為hello.c的檔案,並鍵入以下內 容: <code c++ hello.cpp [enable line numbers="true"]> /* Hello, C++! */ // This is my first C++ program #include <iostream> using namespace std; int main() { cout << "Hello, C++!" << endl; return 0; } </code> \\ ===== - 編譯與執行程式(Compile and Run) ===== < nowiki>C++</nowiki> 語言的原始程式檔案只是一般的純文字檔(text file)□而非可執行檔(executable file)□如果沒有經過<fc red>編譯(compile)</fc>是無法執行的。簡單來說,文字檔是使用人類看的懂的編碼方式;執行檔則必須 是電腦認識的機器指令,通常是<fc red>二進位檔(binary file)</fc>[]所以還需要有一個編譯程 式(Compiler)將原始程式檔轉換成電腦認識的二進位檔。要如何編譯<nowiki>C++</nowiki>語言的原始 程式碼呢?我們可以使用下面的指令來進行編譯[] < code console> [1:18 user@ws example] g++ hello.cpp </code> 若沒有任何的錯誤訊息產生, 那麼您已經順利地完成了編譯的動作。如果您看到錯誤 訊息,那麼請再仔細檢查一下是否原始程式有輸入錯誤的地方,修改後請再次編譯,直到沒有問題為 止[]<nowiki>g++</nowiki>預設會將所欲compile的程式,經編譯無誤後產生一個名為a.out的檔案。檢 查您的目錄下是否多了一個名為a.out的檔案?請用以下指令執行這個可執行檔[] < code console> [1:18] user@ws example] ./a.out </code> 您應該可以看到以下的輸出的結果 <code console> [1:18 user@ws example] ./a.out Hello, C++! [1:18 user@ws example] </code> <nowiki>g++</nowiki>是內建於許 多Linux/Unix系統上的<nowiki>C</nowiki>語言編輯器,由GNU開發。您還可以使用編譯器-o參數,來指 定所輸出的可執行檔檔名,例如以下的例子,將hello.cpp編譯成為hello並加以執行□ < code console> [1:18 user@ws example] g++ hello.cpp -o hello [1:18 user@ws example] ./hello Hello, C++! [1:18 user@ws example] </code> ===== - 程式碼說明 ===== 本節說明前述的hello.cpp程式碼的內容□ ==== - Console Framework與程式進入點 ==== 首先這種在console模式下執行的程式,大多具備以下的 架構,我們將其稱為Console Framework□ <code c h Console Framework> int main() {

程式碼

return 0;

} </code> Console模式的程式執行時,電腦系統會將其載入到主記憶體,並從程式當中特定的位置開始 一行一行、逐行地依序執行程式碼。這個所謂的特定位置,也就是指程式首先被執行的地方,我們稱之 為Entry Point(程式進入點)。事實上,絕大多數的程式語言都有類似的機制以啟動程式的執行,其 中Console模式的C++語言程式,其entry point是程式中的main函式(function]以後會詳細說明)。在前面 的Console Framework當中[]int main()就是在定義這個entry point]在int main()的後面緊接著一對大括 號{},電腦會從左大括號開始,一行一行地加以執行,直到遇到右大括號為止。因此,一個簡單的C++語 言程式,就是將欲由電腦執行的功能,以符合C++語言語法規則的方式寫成程式碼,依序寫在int main() 的大括號內。由於進入點的原型為int main()[]這就表示main函式在結束時必須傳回一個整數(用以讓作業 系統知道程式執行的狀況,通常以0表示程式正常結束。),因此我們在函式結束前傳回一個整數0[] <note> Console終端機模式所謂的console指的是作業系統中一個用以操作的管道,它只支援標準輸入與 輸出,也就是只接受來自鍵盤的輸入以及回應在螢幕上的文字輸出;我們可以透過console來操作作業系 統,包含下達指令與執行程式等操作。在Windows系統中的命令列提示字元[Putty]以及 在Linux/Unix/MacOS中的terminal都是這一類型的操作環境。++ </note>

前述的hello.cpp程式,就是一個Console模式下的程式,也具備如同Console Framework一樣的程式碼架 構。在後續的小節,我們將繼續說明hello.cpp這個程式的細節。

2.0.1 註解(comment)

在hello.cpp的開頭處,有以下這兩行:

/* Hello, C++! */
// This is my first C++ program

其實這兩行對於程式的執行沒有任何作用,只是所謂的註解(comment)而已。在程式語言中,凡是註解的 部份,編譯器在進行編譯時都會略過此一部份,因此註解並不會對程式的執行造成任何的影響,其目的只 是為您所撰寫的程式做一些說明。通常註解的內容不外乎版本、版權宣告、作者資訊、撰寫日期及程式碼 的說明等。在C++語言中註解使用的方式有以下兩種方式:

• 以//開頭到行尾的部份,皆視為註解。例如:

 //註解的使用可以放在每一行的開頭,也可以放在行中其它位置。特別要說明的是,註解有時 不一定只是做程式碼的說明,還可以暫時把可能有問題的程式碼加以註解,以便進行程式的 偵錯,這在程式設計上是常用的一種除錯的方式。

// 這是C++的註解
int i; // 宣告變數i
// int maybe wrong here;

- 以/*開頭到*/結尾的部份,皆視為註解。這種方式可將多行的內容都視為註解。例如:
 ○註解可用於單行的註解,如本例中第1行;也可以用於多行的註解。
- /* 這也是C的註解,可用於單行的註解*/ /* 這種方式也可以應用在多行的註解 對於需要較多說明的時候 可以使用這種方式*/

2.0.2 印出字串

最後要說明的是在hello.cpp的main函式內,唯一的一行程式碼(code)[]

cout << "Hello, C++!" << endl;</pre>

像這樣的程式碼,我們將其稱為敘述(statement),也就是要告訴電腦要幫我們執行的工作為何。在main function中,可以有一行以上的statement[]每一行都必須以分號(;)結尾[]

□cout « "Hello, C++!" « endl;□ 是一個標準的statement□其目的為輸出"Hello, C++!字串到螢幕上,並將 游標移到下一行。

其中cout是用以輸出到stdout(標準輸出)的方法,透過《運算子,我們將一個字串"Hello, C++!"交由cout 來加以處理,其後的« endl則將endl也交由cout一併處理(此處的endl是指end of line的意思,其作用為換

行)。

標準輸入與輸出(standardinput/output)]]大概是許多學過C語言的同學,在學習C++時最不能適應的地方。 其實[]C++的做法讓輸出與輸出變得更為簡單,只要用習慣就會喜歡上這種寫法。在使用C語言時,由 於printf在輸出資料時必須自行指定型態(透過format specifier)]]可能因程式設計師的指定錯誤,而導致錯 誤的輸出結果。但在C++中的cout]]會自動判斷變數型態,因此較為安全與方便。要注意的是原本在C語言 中,常使用#include <stdio.h>載入的標準輸出入相關內容的標頭檔,而在C++中改成使用#include <iostream>]]來載入iostream(其中包含有cout的定義)。

在iostream當中,除了cout可以用來輸出外,還可以使用[]cin[]來取得使用者的輸入,其做法是利用「»」 運算子,來將使用者所輸入的數值存放於特定的變數裡。請參考以下範例:

```
#include <iostream>
using namespace std;
int main()
{
   int a;
   char s [100];
   cout << "This is a sample program." << endl;</pre>
   cout << endl:
   cout << "Type your age : ";</pre>
   cin >> a;
   cout << "Type your name: ";</pre>
   cin >> s;
   cout << endl;</pre>
   cout << "Hello " << s << " you're " << a << " old." << endl;
   cout << endl << endl << "Bye!" << endl;</pre>
   return 0;
}
```

最後,要提醒同學,原本在C語言中的printf()與scanf()函式,在C++中仍可以使用。不過在我們的例子中,將儘量使用新的cout與cin來代替。

2.0.3 命名空間與函式標頭檔

C++語言使用命名空間(namespace)來管理在其函式/類別庫中眾多的識別字名稱,其中std是我們已經使 用過的一個namespace□你可以試著將程式中using namespace std;這行移除,再編譯程式看看,是不是 會發現許多編譯的錯誤?其原因是包含endl, cout等,皆命名於std這個命名空間中,如果不先行載入此命 名空間,則每次使用時都必須以std::endl, std::cout等方式清楚說明欲使用的識別字位於哪個命名空間中。

除此之外,我們也可以宣告自己的namespace:

```
#include <iostream>
#include <cmath>
using namespace std;
namespace first
{
   int a;
   int b;
}
namespace second
{
   double a;
   double b;
}
int main ()
{
   first::a = 2;
   first::b = 5;
   second::a = 6.453;
   second::b = 4.1e4;
   cout << first::a + second::a << endl;</pre>
   cout << first::b + second::b << endl;</pre>
   return 0;
}
```

與C語言一樣,在C++語言中,已預先定義好許多有用的函式(function),我們可以視需要在程式中選擇使 用這些function[]來完成特定的工作。這些function的標頭檔,依功能或性質被分類存放在不同的檔案中, 我們必須使用[]#include[]加以載入。在程式中的[]#include <iostream>[]就是用以載入與輸入輸出相關的 標頭定義檔。

若是要使用原本C語言的各種函式,其對應的標頭檔案當然也必須載入,不過要注意以下兩點:

1. 其它需要載入的標頭檔,仍使用#include載入,但.h的副檔名已不再使用

2. 傳統的C語言函式標頭檔,則以字元c開頭

```
#include <iostream> // 載入與輸出輸入相關
#include <cmath> // 載入傳統的C語言函式庫標頭檔
using namespace std;
int main ()
{
    double a;
```

```
a = 1.2;
a = sin (a);
cout << a << endl;
return 0;
}
```

From: https://junwu.nptu.edu.tw/dokuwiki/ - Jun Wu的教學網頁 國立屏東大學資訊工程學系 CSIE, NPTU Total: 205994

Permanent link: https://junwu.nptu.edu.tw/dokuwiki/doku.php?id=cpp:helloworld



