第 1 章
C++簡介

本章提要

- 1-1 甚麼是程式語言？
- 1-2 C++ 程式語言簡介

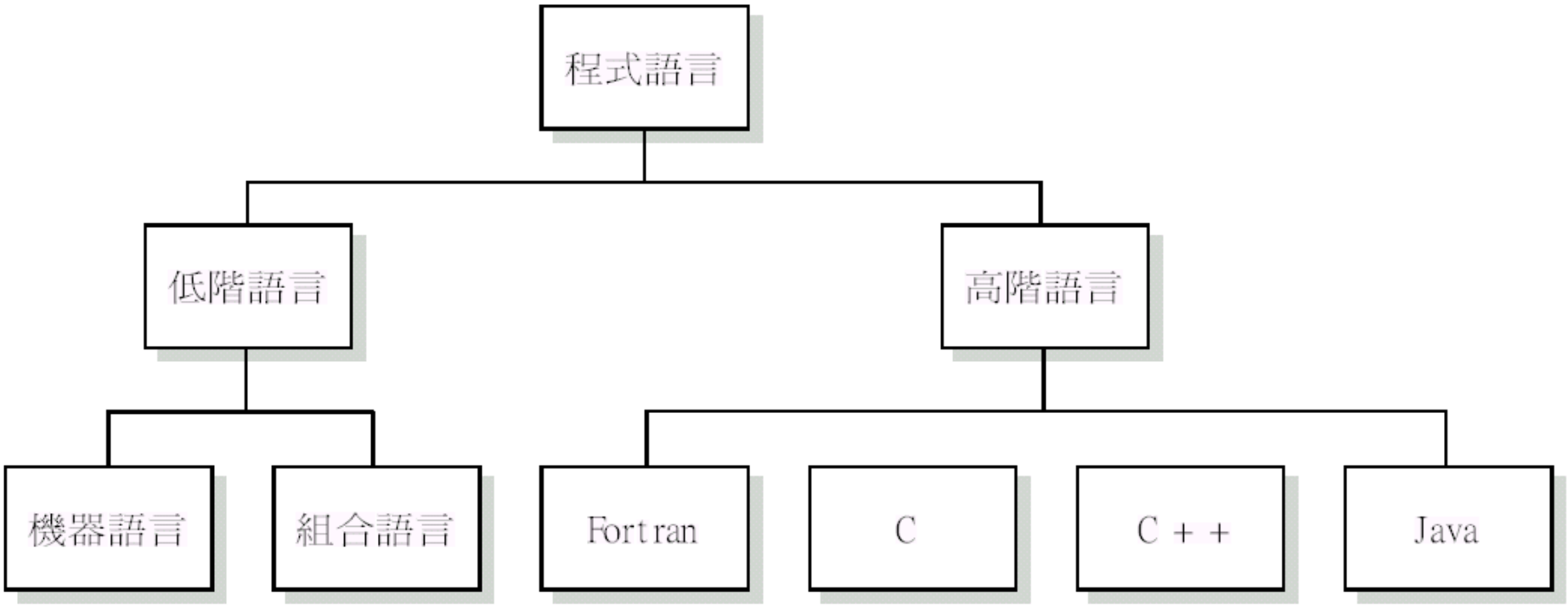
1-1 甚麼是程式語言？

- 身為電腦使用者, 我們常會說：**用 xx 程式來做 xx 事情**。那這麼多能做不同事情、發揮不同功效的應用程式是如何產生的呢？簡單的說, 是程式設計人員 (programmer) 寫出來的。而程式設計人員**寫程式**所用的語言, 就稱為電腦程式語言, 或簡稱**程式語言 (Programming Language)**。

程式類語言的演進與分類

- 如果不考慮一百多年前的差分機(Difference Engine), 第一個程式語言的出現至今才不過半個世紀多, 但在短短的 50 年, 就已發展出數量多到令人眼花撩亂的程式語言種類。程式語言大略可依如下的方式分類：

程式類語言的演進與分

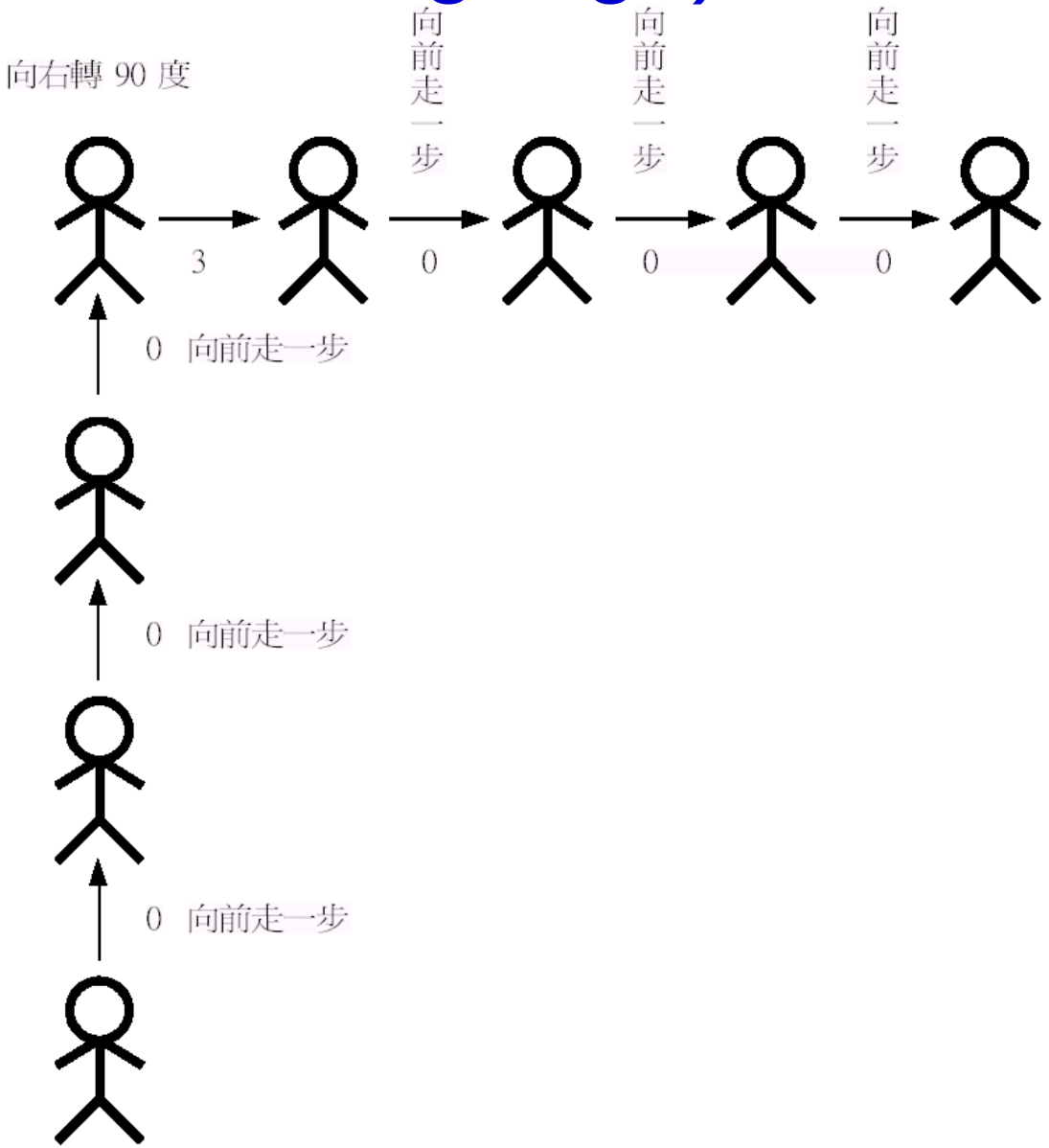


機器語言 (Machine Language)

- 對於電腦來說, 它真正所懂得的語言只有一種, 就是**機器語言**。所謂的機器語言, 其實是以特定的數字來表示電腦所能進行的各個動作, 我們稱這些數字為**機器碼 (Machine Code)**、或機器語言。舉例來說, 如果把電腦比喻為一個人, 而 0 代表向前走一步、1 代表向後退一步、2 代表往左轉 90 度、而 3 代表往右轉 90 度。那麼當我們要命令這個人往前走三步、再往右走三步時, 就必須下達 **0003000** 的指令,

機器語言 (Machine Language)

- 當電腦看到這一串數字後，就會依照每個數字所代表的意義做出指定的動作：



機器語言 (Machine Language)

- 任何一個人都可以看得出來, 這種以數字表達的語言並不適合人閱讀。因此大家很快就發現要用這種方式撰寫程式, 實在太難、太辛苦了, 所以人們就開始思考如何能以更友善的方式來撰寫程式。

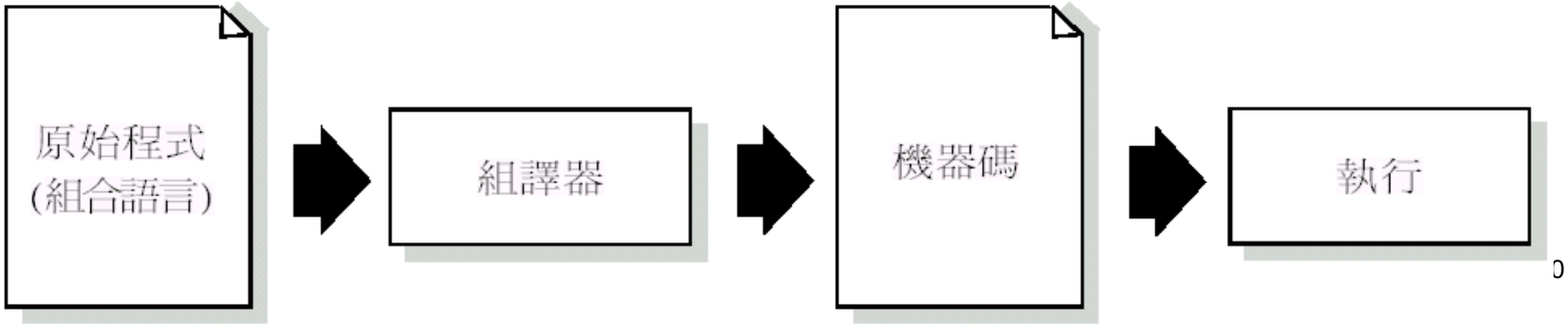
組合語言 (Assembly)

- 為了解決機器語言的難題, 有人就想到了用一些符號來替代數字, 以方便人們辨識各個指令。以前面的例子來說, 如果以 forward 代替原本的 0、backward 代替 1、left 代替 2、而用 right 替代 3。那麼同樣要往前走三步、再往右走三步的程式, 寫起來就變成這樣：

```
forward
forward
forward
right
forward
forward
forward
```

組合語言 (Assembly)

- 像是這樣使用文字符號替代機器碼撰寫程式的語言，就稱為組合語言。這種寫法絕對要比原本的 0003000 要容易懂得多，不過這樣一來，雖然人們看得懂，但是電腦卻看不懂，而必須透過一個翻譯的動作，將這個用組合語言寫成的程式轉換成電腦看得懂的機器語言，負責這個翻譯動作的就是**組譯**



組合語言 (Assembly)

- 不管是機器語言或組合語言都有個缺點, 就是每種電腦的中央處理器 (CPU) 其機器語言並不相同。例如一般個人電腦用的 Pentium 處理器, 其機器語言就和昇陽工作站所用的 UltraSPARC 處理器不同。以前面的行走例子來說, 在甲電腦上 0 代表的是向前走；但換了乙電腦, 0 可能代表的是向後走, 一樣下達 0003000, 到達的地方就天差地遠了。

高階語言

- 不論是機器語言還是組合語言，對於程式的描述都是以電腦所能進行的最基本動作為步驟，因此這兩種語言被稱為**低階語言 (Low-Level Language)**。換言之，本來是希望這個人向前走三步，然後向右走三步，但是因為電腦所能進行的基本動作的限制，使得我們所寫出來的程式必須以**向前走一步、向前走一步、向前走一步、向右轉 90 度、向前走一步、向前走一步、向前走一步**這樣繁瑣的方式一步步描述實際進行的動作，寫起程式來其實並不便利。

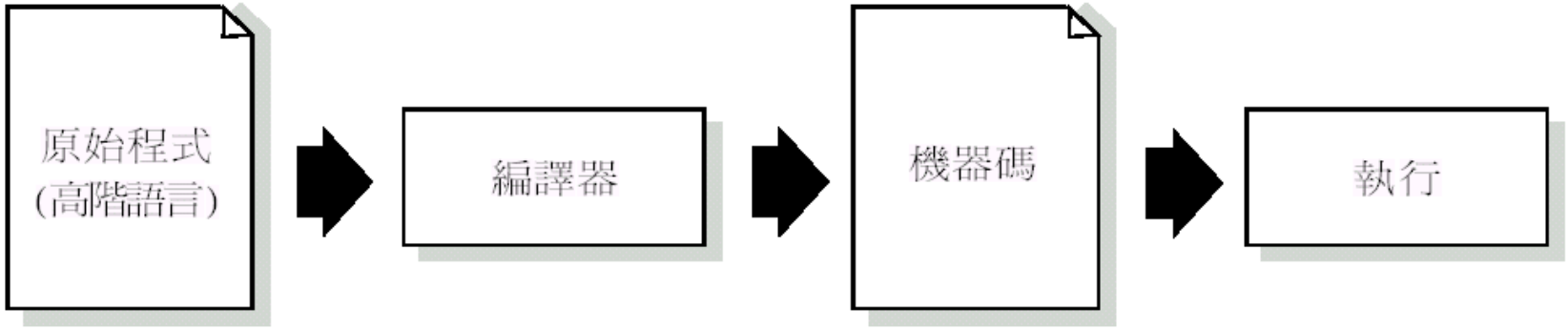
高階語言

- 為了解決這樣的問題, 因此就有人設計新的語言, 用比較接近人類思考的方式來撰寫程式。這種新的程式語言就稱為**高階語言 (High-Level Language)**, 從第一個廣被使用的 Fortran 語言開始, 至今曾流行過的還有 C、Pascal、Basic 等等。使用這種語言, 寫出來的程式就會像是：

```
向前走三步
向右走三步
```

高階語言

- 這樣不但更容易閱讀和理解, 也比使用低階語言所寫出來的程式精簡多了。不過電腦並無法看懂這樣的程式, 和組合語言一樣需要一個轉譯的動作, 將使用高階語言所撰寫的程式轉換成電腦所能看懂的機器語言, 然後才能依此執行。這個轉換的動作是由各程式語言的**編譯器 (Compiler)** 或是**解譯器 (Interpreter)** 來進行。



物件導向程式語言

- 隨著軟體技術的不斷演進, 高階語言也仍不斷在改良、演進。約在 1960 年代興起了**物件導向 (Object-Oriented)** 的觀念, 簡單的說就是將資料和程式結合成**物件**, 在設計程式時以物件的方式來思考及設計程式的內容。
- 目前比較著名的物件導向程式語言, 除了 C++ 外, 還有 Ada、Java 、 Smalltalk 等。

編譯式與直譯式的程式語言

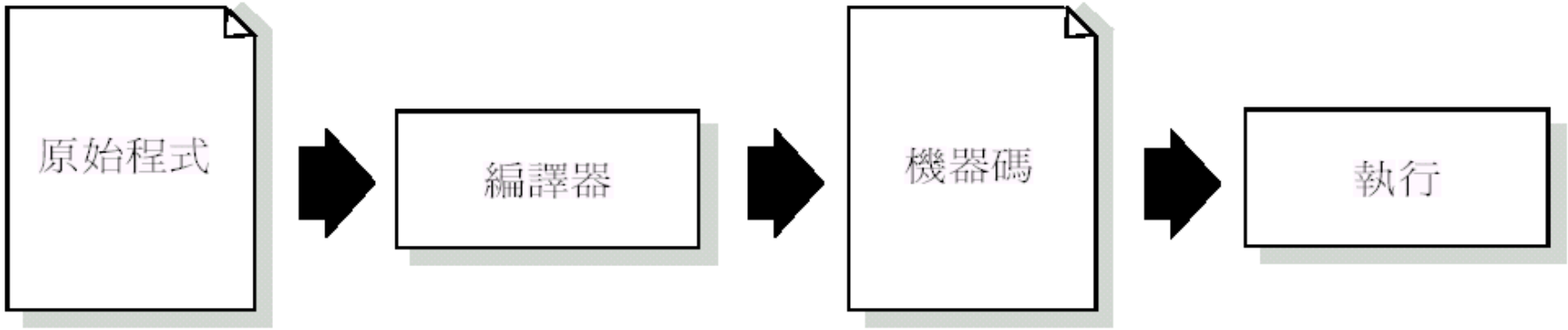
- 除了以程式語言本身的特質來分類以外，也可以依據程式執行的方式來區別，依此種方式區分，可將程式語言分成**編譯式語言**與**直譯式語言**兩大類。

編譯式的程式語言

- 所謂編譯式的程式語言，其執行的方式是先將整個程式從頭到尾讀完，然後電腦才開始依據程式中所描述的動作一一執行。這就好像是一本翻譯書，是先將原文書整個從頭到尾先翻譯好，然後才拿給讀者閱讀。
- 這種作法的好處是，只要書已經翻譯好，那麼不論往後甚麼時候需要閱讀，都不需要再重新翻譯，就可以直接閱讀翻譯好的版本。不過相對來說，在第一次想要閱讀之前，就必須先花時間等待譯者翻譯完畢才行。

編譯式的程式語言

- 對應到電腦實際的運作方式, 編譯式語言必須先用所謂的**編譯器 (Compiler)** 將撰寫好的原始程式轉譯成為機器碼, 然後再執行這份機器碼。以後每次要執行同一個程式時, 就只需要執行這份轉譯好的機器碼, 而不需要花費時間重複再做轉譯的動作。



編譯式的程式語言

- 不過由於每種電腦的機器語言並不相同，比如說我們常用的個人電腦和昇陽工作站的機器語言就不相同，因此，同一個程式如果要在不同的機器上執行，就必須使用專為該種電腦所設計的編譯器，轉譯出符合該電腦的機器碼才行。

直譯式的程式語言

- 直譯式的程式語言剛好和編譯式的程式語言相反，並不先將整個程式讀完，而是每讀取程式中所描述的一個動作，電腦隨即執行相對應的動作，這樣邊讀程式、邊做動作。這就好像是參加國際會議時的現場口譯一樣，主講者邊講，口譯者便即時翻譯給大家聽。

直譯式的程式語言

- 這種作法的好處是，只要主講者講完第一句話，口譯者便開始翻譯，而不需要等待主講者整篇發言完畢。但相對的，由於口譯者即時翻譯完就結束了，因此如果需要重新聽一遍，就只好從頭再請口譯者翻譯了。

直譯式的程式語言

- 對應到電腦實際的運作方式, 直譯式的程式語言就是透過所謂的**解譯器 (Interpreter)**, 每閱讀完程式中所描述的一部份, 便立即要求電腦進行對應的動作, 一直到整個程式執行完畢為止。由於沒有儲存轉譯的結果, 因此往後每次要再執行相同的程式時, 都必須要再花費時間透過解譯器重新轉譯；相對的, 直譯式的好處是程式寫完立即就可以開始執行, 而不需等待轉譯。

程式語言種類與執行方式的關係

- 大部分的程式語言，都有其固有的執行方式。比如說，C/C++ 一般都是編譯式；而 Basic 一般則是直譯式。不過這並非絕對，也有人為 C++ 語言設計出直譯式的解譯器；相同的，也有軟體公司為 Basic 語言做出編譯器。不論是編譯式或是直譯式，兩種方法各有巧妙，而以目前較為流行的程式語言來說，兩種方式都各有擅場。

1-2 C++ 程式語言簡介

- C++ 語言簡史
- C++ 的未來
- C++ 程式開發工具

C++ 語言簡史

- C 語言：C++ 的前身
- C++ 語言的發展
- C++ 的標準化

C 語言：C++ 的前身

- C 語言是 1972 年在 AT&T 貝爾實驗室 (Bell Laboratory) 中發展出來的, 原創者 Dennis Ritchie 當時是為了要發展 UNIX 作業系統, 所以需要一種具有類似組合語言般的高效率, 以及能很方便移植到各機型 (高可攜性) 之程式語言, 於是創造了 C 語言。後來, 由於 C 語言所具備的效率、彈性、可攜性等各項優點, 逐漸成為一種廣受歡迎的程式語言。

C 語言：C++ 的前身

- C 語言的優點包括：
  1. 程式碼精簡，產生的程式執行效率佳。
  2. 具有很高的可攜性。
  3. 完全支援模組化的程式設計。
  4. 彈性大而擴充性強。

C 語言：C++ 的前身

- C 語言在 1980 年代時愈來愈流行, 但在商業領域及政府單位對各種事物要求標準化的需求下, 在 1983 年時, 美國國家標準學會 (ANSI) 開始著手制定 C 語言的標準, 並在 1989 年時推出第一版的 C 語言標準。目前最新版的標準則是 1999 年定案的, 稱為 C99。

C++ 語言的發展

- C++ 和 C 語言一樣, 也是在貝爾實驗室中發展出來的。其原創者 Bjarne Stroustrup 創造 C++ 的目的, 是希望把寫程式變成一種相當愉快的事情, 而且讓設計者可以很輕鬆地寫出好的程式來。他從 1979 年開始, 以 C 語言為基本架構, 再加上物件導向程式設計相關功能, 發展出一個名為 **C with Classes** 的新語言, 也就是最初的 C++ 語言。

C++ 語言的發展

- 此後 **C with Classes** 不斷被改良, 並在 1983 年時正式命名為 C++。在 C 語言中, **++** 是一個遞增運算子 (Increment operator), 如果將 C 當成是變數, 則 C++ 就代表 C=C+1 之意；由此可看出 C++ 的目標就是要創造一個更好的 C, 而且仍可與 C 相容並保有 C 原來的各項優點。

C++ 語言的發展

- 由於 C++ 可與 C 相容, 所以大部份在 C 語言上發展出來的軟體仍可以在 C++ 中繼續使用, 再加上其本身具有許多比 C 更強的特性, 使得 C++ 在短短幾年之間, 搖身一變成為目前最熱門的語言之一。
- 最初的 C++ 只是在 UNIX 上的一個轉譯器 (名為 Cfront), 可將 C++ 程式先轉成 C 語言後再加以編譯; 其後, 隨著使用者不斷增加, 而逐漸有了在各作業系統上的編譯器, 使 C++ 應用程式的產生更加方便。

C++ 的標準化

- C++ 的標準化是由 ANSI 和國際標準組織 (ISO) 共同進行的, 兩個組織均成立工作小組專責制定 C++ 的標準, 同時也針對使用者的需求, 修正及強化 C++ 的功能。
- ANSI/ISO 的 C++ 標準經過漫長的討論與修正, 在 1998 年正式發布了編號為 ISO/IEC 14882：1998 的 C++ 標準, 簡稱為 ISO C++98 或 C++98。在此之後, 各主要 C++ 編譯器開發者大多也都遵循 ISO 的標準來設計他們的 C++ 語法。

C++ 的未來

- C++ 標準的下一代：C++0X
  - 在 ISO 完成 C++98 標準後, 也隨即展開下一代 C++ 標準的制定工作, 這個下一版的 C++ 標準預計在 200X 年能正式推出, 所以被稱為 C++0X。
  - 雖然目前難以預測下一代的 C++ 標準會有什麼樣的變化, 但可以預見的是：在業界的要求下, 新 C++0X 仍會與目前的 C++ 相容, 並增加一些讓程式設計人員更方便的功能。

C++ 的未來

- 因此我們現在學好 C++, 不必擔心過 35 年後會出現一個令人陌生的新 C++ 語言, 又要重學很多東西。我們現在所學、所寫 的 C++ 程式, 可能不必做任何修改、或是只需做局部修改, 就能符合下一代的 C++ 標準, 也能用在下一代的 C++ 編譯器。

C++ 的未來

- C++ 與 Java、C# 的競爭
  - 對程式語言陌生的讀者，雖然沒寫過什麼程式，可能也聽說過被視為 C++ 對手的 Java、C# 程式語言。這兩個語言也是目前流行的物件導向程式語言，使用的軟體公司 / 程式設計人員也不少，很多人也將這幾個語言視為彼此互相競爭的對手。

C++ 的未來

- 其實這幾個語言雖然有些相似之處，但也各有特色、各有適合其發揮之處，以下簡單列出其間的差異：
  - C++：發展最久，應用也最廣泛，幾乎所有的作業系統 / 平台上都可找得到開發 C++ 應用程式的工具，而且編譯出來的程式，其執行效能最佳。
  - Java：目前主要應用在開發網路及行動裝置的應用程式，其特色是跨平台，換言之編譯好的 Java 程式，可以在任何已安裝 Java 執行環境 (JRE, Java Runtime Enviroment) 的作業系統上執行。

C++ 的未來

- C#：微軟所開發的程式語言，也已被 ISO 及歐洲標準組織 (ECMA) 認可為一項程式語言標準。C# 和 Java 有個相似的特性：C# 的程式在編譯後，需在 .NET 環境下執行，換句話說，只要有 .NET 環境的平台，都能執行 C# 應用程式，達到跨平台執行的效果，但由於在 Windows 以外的作業系統很難看到 .NET 的蹤影，所以實際上主要的應用仍侷限在微軟 Windows 作業系統。和 C++ 相較，用 C# 開發 Windows 圖形介面的程式較為方便。

C++ 的未來

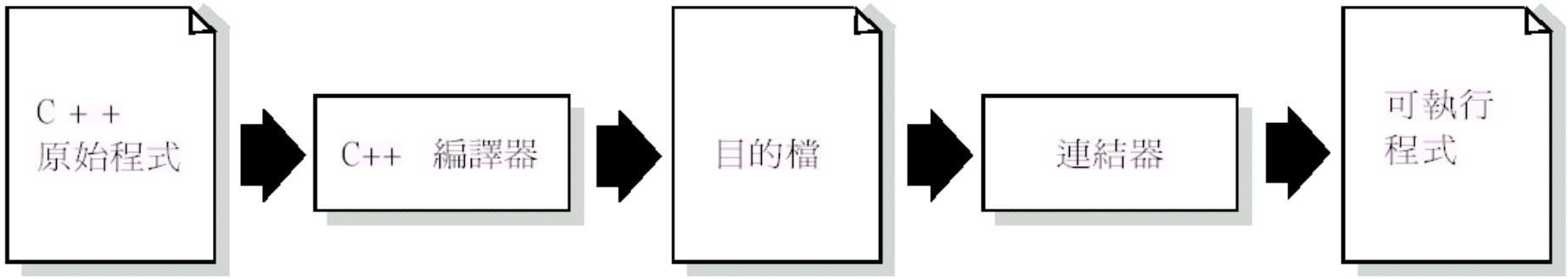
- 由於 C++ 目前的應用仍相當廣泛, 在較重視執行效能或與硬體相關的場合 (例如撰寫作業系統), 幾乎都仍需使用 C/C++ 來撰寫程式, 所以學習 C++ 並不需擔心被淘汰。而且學會 C++ 後, 要再學習 Java 或 C# 都相當容易上手, 因此學習 C++ 語言仍是當前的不二選擇。

C++ 程式開發工具

- 工欲善其事必先利其器，要學習 C++ 語言，當然要先準備好一套實用的開發工具來進行開發。
  - 命令列工具與整合開發環境
  - Visual C++ 系列
  - Borland C++Builder/C++BuilderX 系列
  - GCC
  - Dev C++

命令列工具與整合開發環境

- 前面說過, C++ 是編譯式的程式語言, 細分來看, C++ 原始程式要經過編譯、連結的動作, 才能產生可執行的程式檔。因此一個 C++ 程式的產生過程大致如下：



命令列工具與整合開發環境

- 在上列的開發過程中，我們需用到三個工具程式：編譯器、連結器、及用來輸入 C++ 原始程式的文字編輯器。其中最傳統的編譯器、連結器都是在文字模式的命令提示下執行的，所以可稱之為命令列工具。
- 而在程式的開發過程中，我們通常需要來回多次修改程式、測試程式多次。換言之，我們來回切換執行文字編輯器、編譯器、連結器好幾次，相當不便。

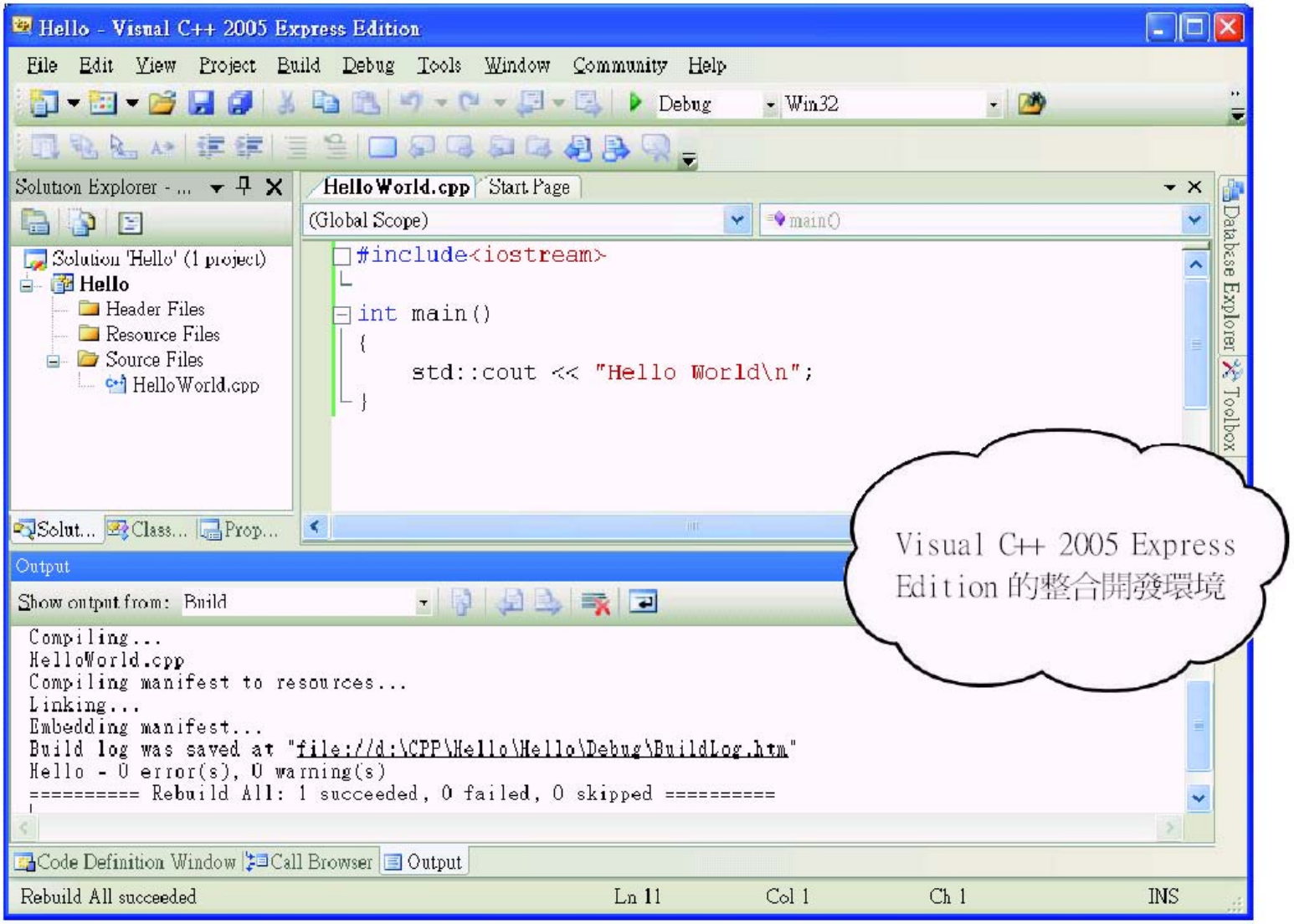
命令列工具與整合開發環境

- 因此就有人發展出**整合開發環境 (IDE, Integrated Development Environment)**，簡單的說, IDE 提供了一個編寫程式專用的編輯器介面, 從這個介面即可啟動編譯器、連結器來進行編譯與連結的動作。此外大多的 IDE 也結合了實用的除錯、圖形介面設計、專案管理等多種輔助功能與工具, 讓程式 / 專案開發的過程更輕鬆、方便。

Visual C++ 系列

- 微軟公司的 Visual C++ 是 Windows 平台上主流的 C++ 整合開發環境之一。Visual C++ 的 IDE 和微軟公司其它程式語言產品 (包括 Visual Basic、C#、J# 等) 其實是共用一個名為 Visual Studio 的 IDE。對於想同時用微軟其它語言或技術的開發人員而言, 由於只需熟悉一套操作介面即可, 所以使用起來較為方便。

Visual C++ 系列


Hello - Visual C++ 2005 Express Edition
File Edit View Project Build Debug Tools Window Community Help
Debug
Win32
Solution Explorer - ...
Solution 'Hello' (1 project)
Hello
Header Files
Resource Files
Source Files
HelloWorld.cpp
HelloWorld.cpp
Start Page
(Global Scope)
main()
#include<iostream>
int main()
{
std::cout << "Hello World\n";
}
Database Explorer
Toolbox
Solut...
Class...
Prop...
Output
Show output from: Build
Compiling...
HelloWorld.cpp
Compiling manifest to resources...
Linking...
Embedding manifest...
Build log was saved at "file://d:\CPP\Hello\Hello\Debug\BuildLog.htm"
Hello - 0 error(s), 0 warning(s)
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
Code Definition Window
Call Browser
Output
Rebuild All succeeded
Ln 11
Col 1
Ch 1
INS
Visual C++ 2005 Express Edition 的整合開發環境

Visual C++ 系列

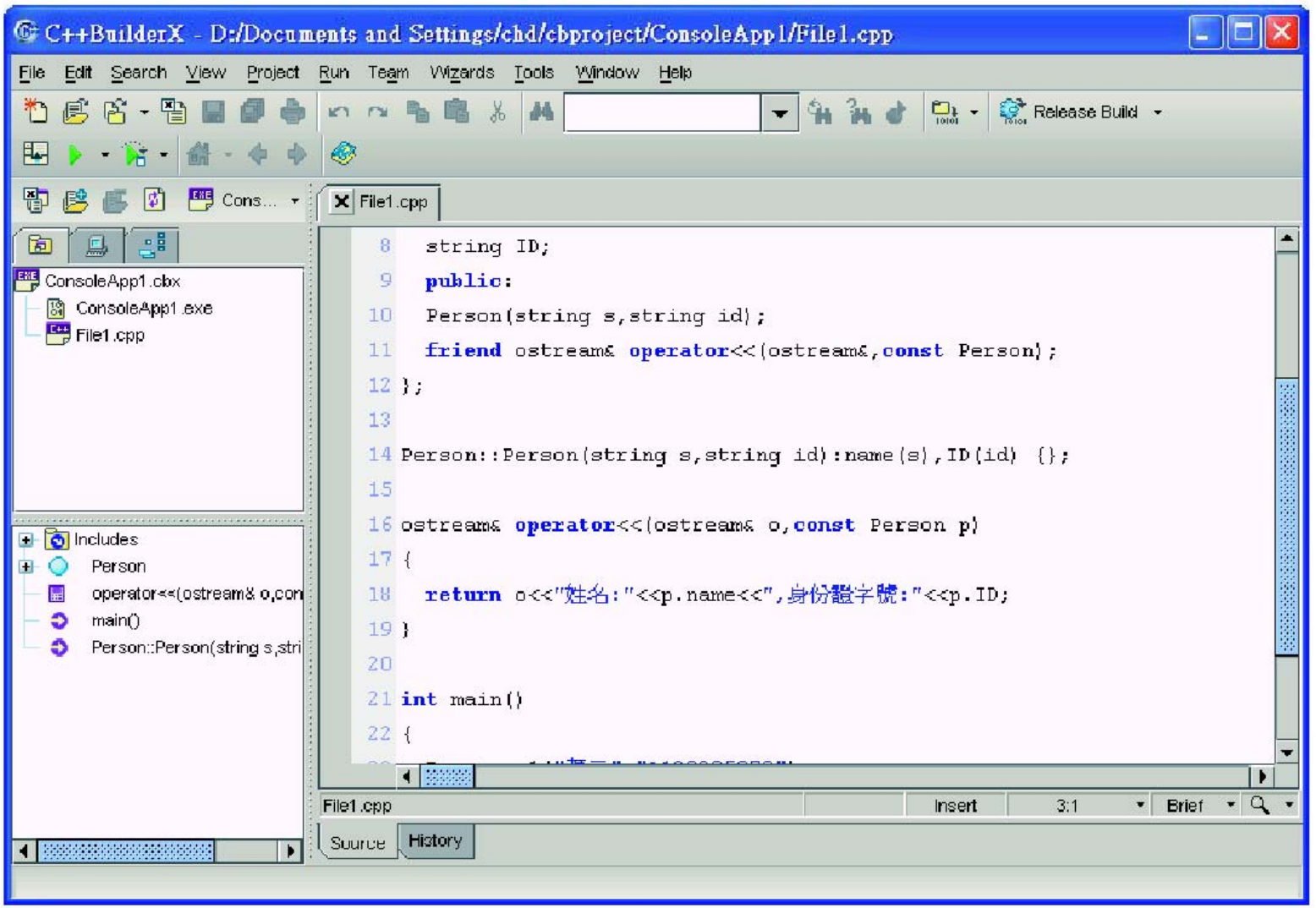
- 近三年來, 微軟為推展其 .NET 技術, 也將相關技術加到 Visual C++ 中, 其特色之一就是能以微軟自家的 **Managed C++** 語法, 撰寫專門用在 .NET 環境下執行的 C++ 程式。可參考微軟 .NET 網站 (http://www.microsoft.com/Net/) Visual C++ 網站 (http://msdn.microsoft.com/visualc/)、或其它相關書籍。

Borland C++Builder /C++BuilderX 系列

- Borland 公司的 C++Builder/C++BuilderX 整合開發環境, 由於具備類似 Visual Basic 的設計介面, 可以很容易設計圖形使用者介面, 因此廣受使用者歡迎。而且 C++BuilderX 同時支援 Windows、Linux、Solaris 等三種作業系統。
- 從官方網站 http://www.borland.com/downloads/ 除了可下載個人版及企業版的試用版外, 也可找到只能在文字模式下使用的純編譯器版本。

Borland C++Builder /C++BuilderX 系列

C++BuilderX 的環境



GCC

- GCC 最初是 C 語言的編譯器, 當時它代表 **GNU C Compiler** 這 3 個字的縮寫。不過後來 GCC 支援的程式語言擴及 C++、Java、Fortran、Ada 等等, 這時它已變成是多種程式語言編譯器的集合, 因此也正名為 **GNU Compiler Collection**。GCC 屬於 GNU 自由軟體計劃的一部份, 因此在 Unix、Linux、FreeBSD 等平台上都能看到 GCC 的芳蹤, 所以若您有使用這些作業系統, 就可直接用 GCC 來編譯包括 C++ 在內的多種程式。

GCC

此外也有人將 GCC 移植到 Windows 平台上, 所以在 Windows 中也能使用 GCC 來編譯程式。

- GCC 中的 C++ 編譯器名稱為 **g++**, 要用它來編譯程式, 只要在 **g++** 後面加上 C++ 原始程式的名稱當參數, 即可進行編譯：

GCC

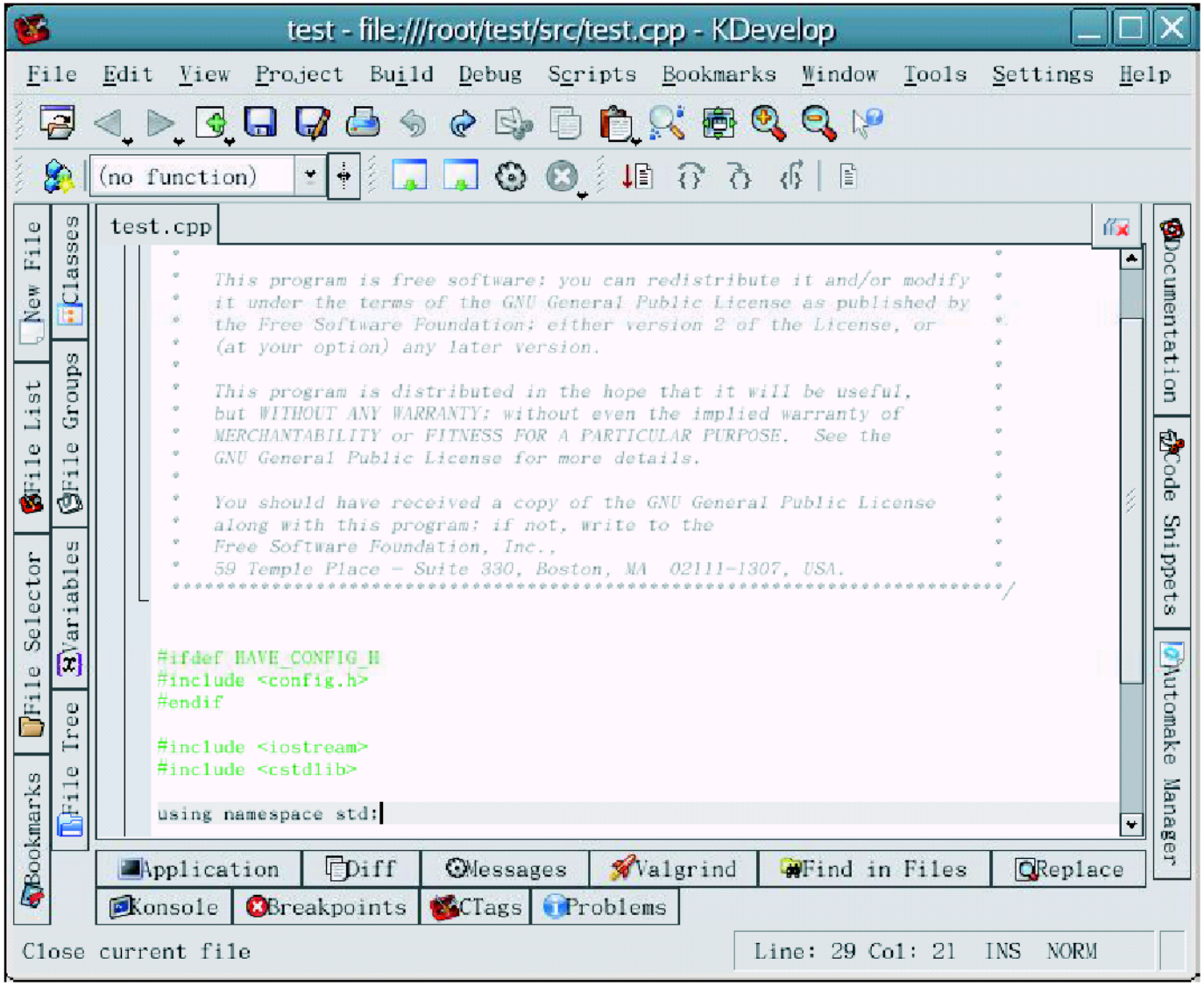
在 Linux 下用 g++ 編譯 C++ 程式

- 有關 GCC 的詳細資訊、檔案下載、說明文件, 都可到官方網站 http://gcc.gnu.org/ 取得。

GCC

- 在 Unix/Linux 平台中當然也有整合式的開發環境, 例如 KDE 這個知名的桌面環境就有一套 KDevelop 整合開發環境, KDevelop 可藉由安裝不同的 Plugin 而支援 C/C++、Fortran、Java 等多種語言：

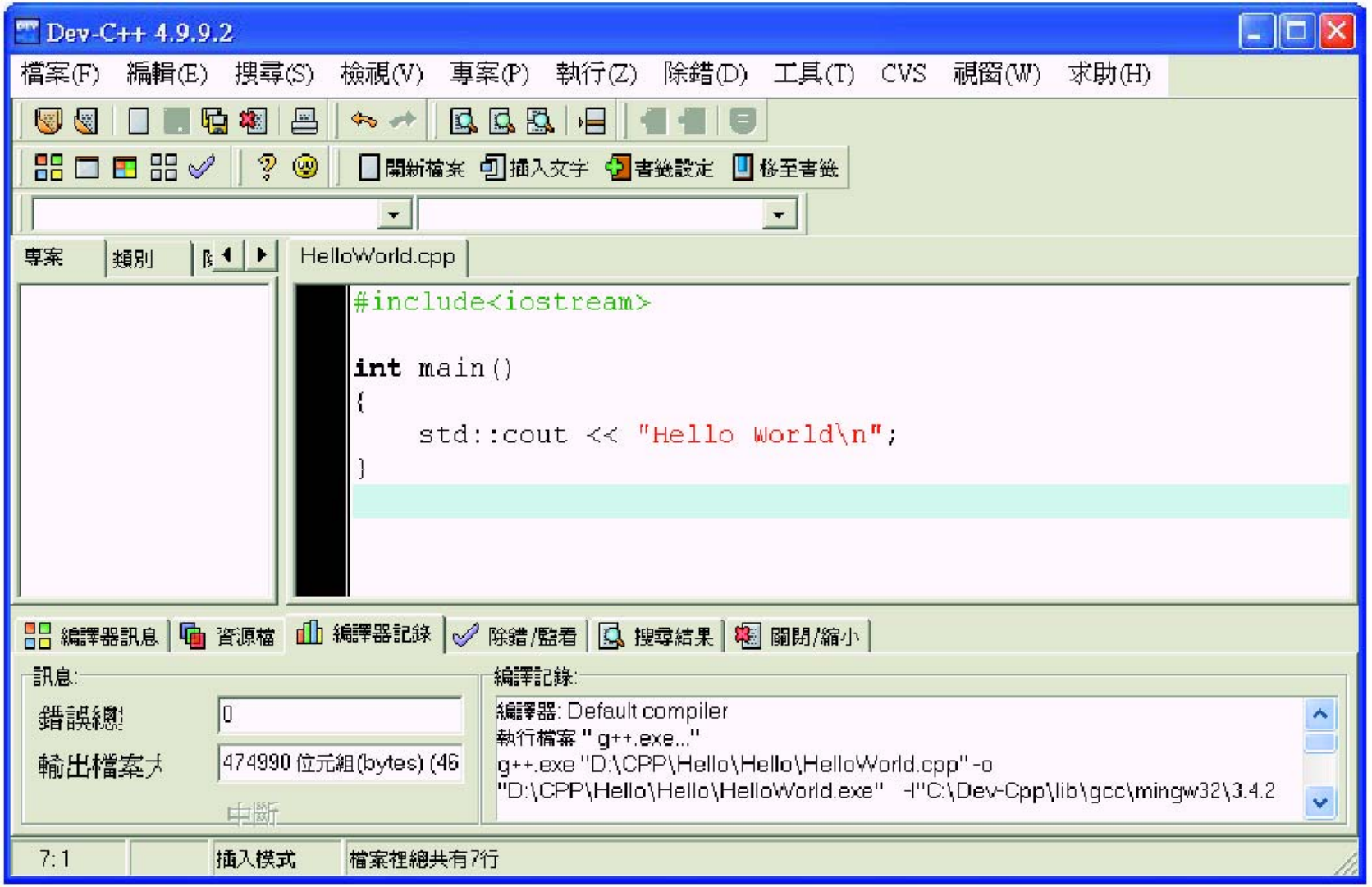
GCC


test - file:///root/test/src/test.cpp - KDevelop
File Edit View Project Build Debug Scripts Bookmarks Window Tools Settings Help
(no function)
test.cpp
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program; if not, write to the
Free Software Foundation, Inc.,
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif
#include <iostream>
#include <cstdlib>
using namespace std;
New File
Classes
File List
File Groups
File Selector
Variables
File Tree
Bookmarks
Documentation
Code Snippets
Automake Manager
Application
Diff
Messages
Valgrind
Find in Files
Replace
Konsole
Breakpoints
CTags
Problems
Close current file
Line: 29 Col: 21 INS NORM

Dev C++

- Dev C++ 是 Windows 平台上的 GCC 整合開發環境, 也就是說, Dev C++ 本身僅提供 IDE 這個介面環境, 程式的編譯、連結都是交給GCC 負責。Dev C++ 的特色是它是自由軟體 (Free Software), 所以我們可以從網路上直接下載它來使用。
- 關於 Dev C++ 的進一步資訊請至其官方網站：http://www.bloodshed.net/devcpp.html。

Dev C++


Dev-C++ 4.9.9.2
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(Z) 除錯(D) 工具(T) CVS 視窗(W) 求助(H)
開新檔案 插入文字 書籤設定 移至書籤
專案 類別
HelloWorld.cpp
#include<iostream>
int main()
{
std::cout << "Hello World\n";
}
編譯器訊息 資源檔 編譯器記錄 除錯/監看 搜尋結果 關閉/縮小
訊息:
錯誤總
0
輸出檔案大
474990 位元組(bytes) (46
中斷
編譯記錄:
編譯器: Default compiler
執行檔案 " g++.exe..."
g++.exe "D:\CPP\Hello\Hello\HelloWorld.cpp" -o
"D:\CPP\Hello\Hello\HelloWorld.exe" -I"C:\Dev-Cpp\lib\gcc\mingw32\3.4.2
7: 1
插入模式
檔案裡總共有7行