bit.ly/coachsu-programming00

程式設計

第0章 介紹
Introduction

蘇維宗 (Wei-Tsung Su)
suwt@au.edu.tw
564D

教學大綱

107學年度上學期

課程規範(Class Protocol v1.1)

- 你(妳)可能獲得加分，如果..
  - 在課堂上樂於討論
  - 在網路討論區上樂於討論
  - 其他
- 你(妳)將會獲得扣分，如果..
  - 在禁止飲食的教室吃東西
  - 在課堂上大聲喧嘩
- 你(妳)一定會被當掉，如果..
  - 考試作弊
  - 曠課超過3次

- You MAY get additional points if you ...
  - have any response in on-site class discussion
  - have any response in web-based class discussion board
  - others
- You WILL lose additional points if you ...
  - eat in no-food classroom
  - talk loudly
- You MUST be failed if you
  - cheat in any exams
  - absent more than 3 times

課程目標

本課程目標在於了解程式語言在電腦工程中如何扮演解決問題的角色。藉由學習**C程式語言**培養學生基礎程式設計(如程式架構、邏輯控制、函式、陣列、指標、結構等功能)、除錯、以及進階的程式碼管理與測試等能力，為學生未來大學四年之專業課程打好基礎。

課程助教

研究生:連嘉俊

辦公室:565B

實習課:禮拜五8-9節(15:30 - 17:20) @ 712教室

研修生除非本系必修衝堂，否則一定要參與實習(請找系辦助理登記)

評分方式

平時成績：50%

- 課堂出席：10%
- 實習出席：10%
- 程式作業：20% (約10題)
- 程式檢定：10% (需通過3題)

期中考試：25% (筆試 + 上機考試)

期末考試：25% (上機考試)

題目來源

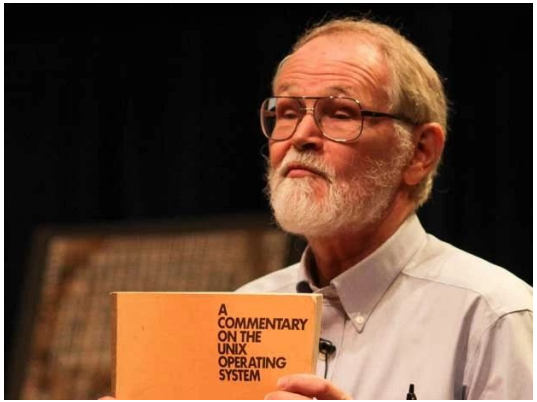
高中生程式解題系統：https://zerojudge.tw

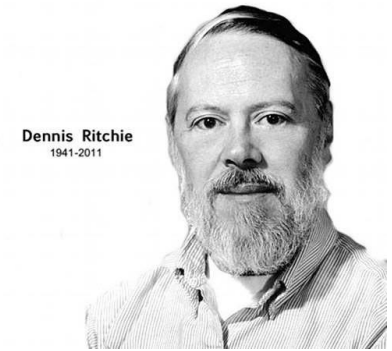
ITSA線上協同學習平台：https://e-tutor.itsa.org.tw/e-Tutor/

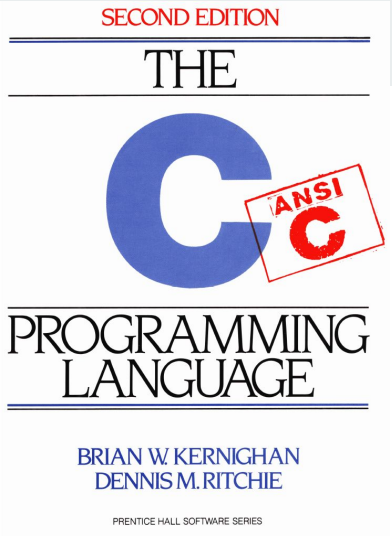
leetcode：https://leetcode.com

...

教科書







The C Programmng Language (2nd Edition), Prentice Hall [Download]

作者:

- Brian Kernighan (參與UNIX開發)
- Dennis Ritchie (參與UNIX開發, C語言之父)
  - 與Ken Thompson於1983年因為開發UNIX系統獲得圖靈獎

參考書(購買)

C程式設計藝術(第7版), 全華圖書

原文作者

- Paul J. Deitel
- Dr. Harvey M. Deitel

main()

目標

瞭解學習程式設計的目的

瞭解程式語言

瞭解程式設計工具

第一支C語言程式

程式設計(Programming)

為何要學程式設計?

程式設計需要學嗎?

程式設計要學什麼?

圖片來源: https://www.themoviedb.org/movie/3050-doctor-dolittle

為什麼要寫程式?

Hour of Code (https://studio.code.org/hoc/5)

程式語言(Programming Language)

程式語言可以用與電腦硬體多接近來分類

機器語言(Machine Language)　*電腦說的電腦話*

組合語言(Assembly Language)　*人說的電腦話*

高階語言(High-level Language)　*人說人話，請**翻譯**翻給電腦聽*

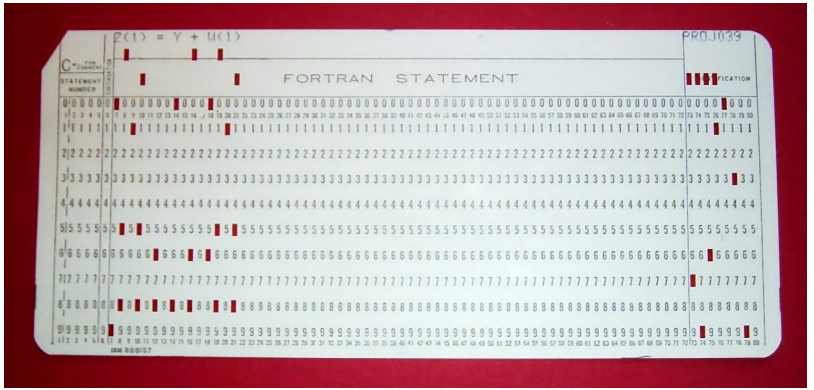
問題:程式語言還可以怎麼分類?

機器語言(Machine Language)

機器語言是電腦唯一會講的程式語言。(電腦說的電腦話)

例如:

```
0001 0000 0100 0000
0001 0001 0100 0001
0011 0010 0000 0001
0010 0100 0010 0010
0000 0000 0000 0000
```



Wiki: Computer programming in the punched card era

組合語言(Assembly Language)

組合語言是用人容易閱讀的文字來撰寫機器語言。(人說的電腦話)

例如:

```
LOAD   0   40          ;0001 0000 0100 0000
LOAD   1   41          ;0001 0001 0100 0001
ADDI   2   0   1       ;0011 0010 0000 0001
STORE  42  2           ;0010 0100 0010 0010
HALT                   ;0000 0000 0000 0000
```

高階語言(High-Level Language)

不管是組合語言或機器語言，都是與機器相依的。

高階語言加強程式的**可讀性**與**可移植性**。(人說人話，請翻譯翻給電腦聽)

例如，Swift, Kotlin, Go, Python, C#, Java, C++, C, ... [Ranking]

例如：

```
C = A + B;
```

想學哪種語言?

A few moments later ...

開發環境

還記得高階語言是『人說人話，請**翻譯**翻給電腦聽』嗎?

那麼是誰來把C語言翻譯給電腦聽呢? **編譯器(compiler)**

目前使用最廣泛的C語言編譯器

GNU Compiler Collection (GCC)

註:後起之秀為Clang / LLVM

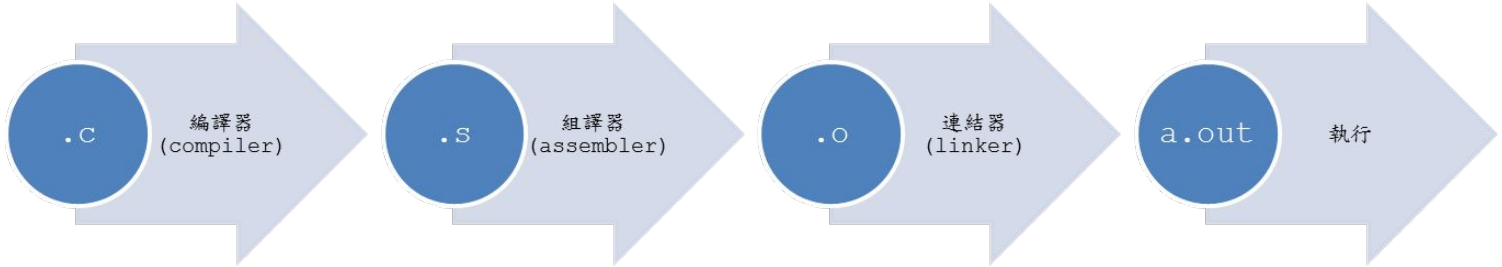
GCC基本使用

`$ gcc prog.c`	預設執行檔名稱為`a.out`
`$ gcc -Wall prog.c`	顯示所有警告訊息(**好習慣!**)
`$ gcc -Wall -o exec prog.c`	`-o`選項可以指定執行檔名稱

註:編譯時如果只有警告訊息(沒有錯誤訊息)還是會產生執行檔，但執行時可能會出現**不可預期的錯誤**。

實際翻譯的過程

編譯器(compiler)	`gcc`	(將C語言翻譯成組合語言)
組譯器(assembler)	`as`	(將組合語言翻譯成機器語言)
連結器(linker)	`ld`	(將多個機器語言合併成一個可執行檔)
載入器(loader)	`ld`	(將可執行檔載入到記憶體中執行)



打造自己的開發環境

A few moments later ...

選項

- IDE
  - Orwell Dev-C++: http://orwelldevcpp.blogspot.tw/
  - CodeLite: https://codelite.org
- 雲端開發環境
  - https://www.onlinegdb.com/online_c_compiler
  - https://www.remoteinterview.io/online-c-compiler
- Linux開發環境
  - 遠端登入
  - 安裝Linux
  - 使用Docker

Let's Coding

第一支C語言程式誕生

編輯原始碼(ch00_01.c)

```
1.  /* This is comment. */
2.  //#include<stdio.h>
3.  //#include<stdlib.h>
4.
5.  int main(void) {
6.      printf("Hello, World!\n");
7.      return EXIT_SUCCESS;
8.  }
```

重點解說

- 原始碼的副檔名為 .c
- 註解(不會被翻譯的內容)
  - //單行註解
  - /* 多行註解 */
- 前置處理器(Preprocessor)
  - 前置處理器會在翻譯前先處理以#開頭的程式碼。

第一支C語言程式誕生(續)

```
$ gcc -Wall ch01_01.c
```

執行結果

???

警告訊息(Warning)

```
ch01_01.c:6:5: warning: implicit declaration of function 'printf'
[-Wimplicit-function-declaration]
  printf("Hello, World!\n");
  ^
ch01_01.c:6:5: warning: incompatible implicit declaration of built-in
function 'printf'
ch01_01.c:6:5: note: include '<stdio.h>' or provide a declaration of
'printf'
```

錯誤訊息(Error)

```
ch01_01.c:7:12: error: 'EXIT_SUCCESS' undeclared (first use in this
function)
  return EXIT_SUCCESS;
         ^
ch01_01.c:7:12: note: each undeclared identifier is reported only
once for each function it appears in
```

Let's Coding Again

真. 第一支C語言程式誕生

編輯原始檔ch00_02.c

```
1.  /* This is comment. */
2.  #include<stdio.h>
3.  #include<stdlib.h>
4.
5.  int main(void) {
6.      printf("Hello, World!\n") ;
7.      return EXIT_SUCCESS ;
8.  }
```

重點解說

- main()函式為預設程式進入點
- int main(void) 為函式的宣告
  - int為回傳值形態(只能一個)
  - void為引數形態(可多個)
- 程式區塊(block)是由{}包圍
  - 透過**縮排(indent)**讓程式碼看起來更整齊

真. 第一支C語言程式誕生(續)

編輯原始檔ch00_02.c

```
1.  /* This is comment. */
2.  #include<stdio.h>
3.  #include<stdlib.h>
4.
5.  int main(void) {
6.      printf("Hello, World!\n");
7.      return EXIT_SUCCESS;
8.  }
```

重點解說

- 每一行程式是以;分隔
- printf()函式將訊息送到標準輸出
  - 此函式宣告在stdio.h
  - \n是換行字元
- return關鍵字用來回傳資料
- EXIT_SUCCESS宣告在stdlib.h

真. 第一支C語言程式誕生(續)

```
$ gcc -Wall ch00_02.c
```

執行結果

```
$ ./a.out
Hello, World!
```

修改想列印的文字並重新編譯

A few moments later …

Q & A

補充資料

使用遠端登入Linux環境

透過SSH遠端登入Linux開發環境

在電腦上安裝終端機(Linux與Mac電腦都有內建終端機)

Windows電腦建議下載cmder可攜式終端機軟體(可放在隨身碟中)

在終端機中執行`ssh`指令遠端登入Linux開發環境

```
> ssh 帳號@主機名稱
```

輸入密碼後即可登入Linux開發環境

常用的Linux指令

`$ clear`	清除終端機畫面
`$ ls`	**l**i**s**t - 顯示目前目錄下的檔案
`$ nano 檔案名稱`	編輯檔案
`$ pico 檔案名稱`	編輯檔案
`$ mv 舊檔名 新檔名`	**m**o**v**e - 變更檔案名稱
`$ cp 原檔名 新檔名`	**c**o**p**y - 複製檔案
`$ rm 檔案名稱`	**r**e**m**ove - 刪除檔案

更多指令可參考鳥哥的Linux私房菜

補充資料

使用Docker建立開發環境

安裝Docker CE

直接到官方網站下載應用程式(Linux, macOS, Windows)

https://www.docker.com/community-edition

但Windows版本必須在支援Hyper-V的專業版/企業版才能使用。否則就需要安裝Docker Toolbox (基於VirtualBox)

https://docs.docker.com/toolbox/toolbox_install_windows

安裝Docker本地端開發環境

第一次執行

```
$ docker pull ellington/ubuntu:cenv
$ docker run -it --name cenv ellington/ubuntu:cenv
```

重新執行時

```
$ docker start cenv
$ docker attach cenv
```