

# JAVA程式語言與系統開發

面授老師：林再順

手機：0937-862-065

E-Mail：[921337@gapps.nou.edu.tw](mailto:921337@gapps.nou.edu.tw)

# JAVA程式語言與系統開發

作業繳交日期：

第1次：4月20日(10%)

第2次：6月16日(10%)

作業繳交方式：以電子郵件繳交電子檔

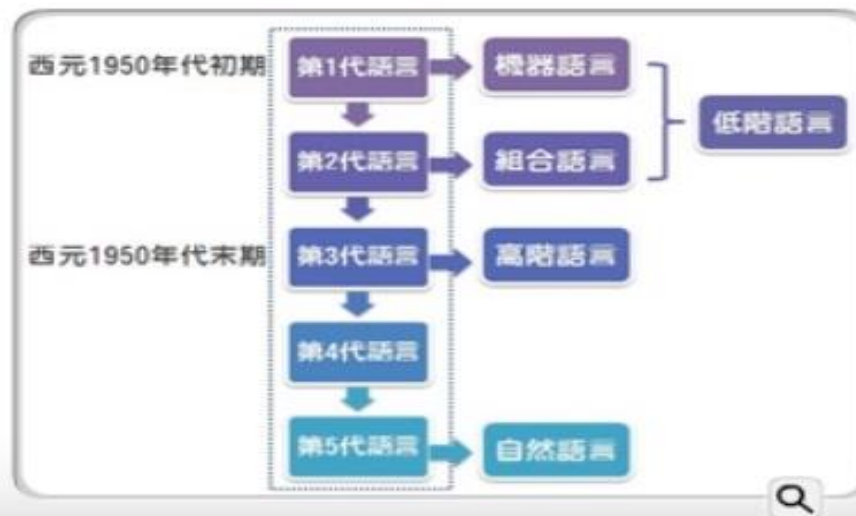
平時出席及互動、自學表現：

數位教材自學(5%)、面授出席率及互動(5%)

# 第1章 程式語言與系統開發入門

## 程式語言發展的5大階段

程式語言(programming language)是我們和電腦溝通的工具，用程式語言寫出來的程式(programs)可以讓人類指揮電腦工作。一般都把程式語言的發展分成5大階段。



# 第1章 程式語言與系統開發入門

## 第1代語言

就是機器語言(machine language)，完全以電腦能夠直接處理的1與0的位元串來表示運算的步驟，不同的電腦使用的機器語言通常也都不一樣。

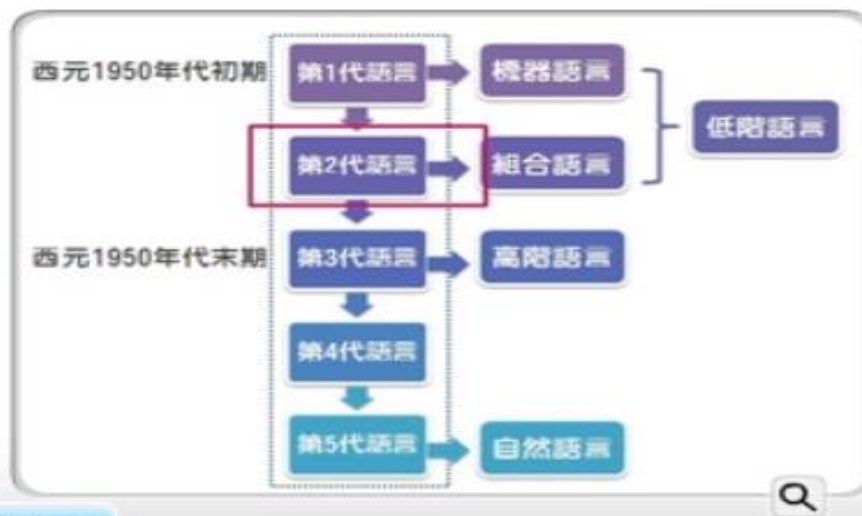
|          |          |     |         |        |  |  |  |
|----------|----------|-----|---------|--------|--|--|--|
| 00110101 | 01111010 |     |         |        |  |  |  |
| 10110000 | 10100100 | ADD | FIELDA  | FIELDB |  |  |  |
| 11101111 | 11100101 |     | FIELD C | TALLY  |  |  |  |
| 01010100 | 10001001 | CPN | TALLY   | MAX    |  |  |  |
| 10001001 | 01011010 | GTR | LOOPA   |        |  |  |  |
|          |          | MVN | PARTNO  | NUMBER |  |  |  |
|          |          | BUN | LABELA  |        |  |  |  |

## 第1代語言

# 第1章 程式語言與系統開發入門

## 第2代語言

以一般人比較好理解的字元來表示機器執行的指令，不過處理時會多所謂的「組譯」(assemble)的動作，講組合語言的程式轉換成機器語言。



## 第2代語言

# 第1章 程式語言與系統開發入門

## 第3代語言

第3代語言也稱為高階語言(high-level languages)，在語法上更容易理解，設計上則具有通用的(general purpose)特性，所以用高階語言寫的程式能在不同的電腦上執行。

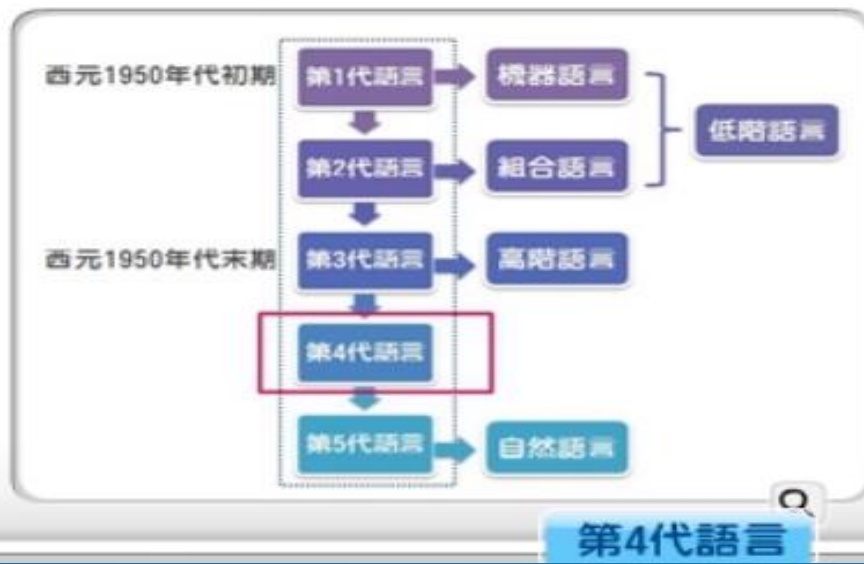




# 第1章 程式語言與系統開發入門

## 第4代語言

第4代語言是以使用者為中心、以解決問題為導向的語言，目的是希望達到簡單易用，提高設計者的生產力。整合性的CASE (computer-aided software engineering) 工具、資料庫系統的SQL語言都算是第4代語言。



# 第1章 程式語言與系統開發入門

## 第5代語言

第5代語言也稱為自然語言(natural languages)，代表語言本身在使用上接近一般的口語，大幅降低使用上的技術門檻。

The diagram illustrates the evolution of programming languages across five generations. It is organized into two time periods: '西元1950年代初期' (Early 1950s) and '西元1950年代末期' (Late 1950s). The languages are listed in a central column, with arrows pointing to their corresponding categories on the right. A bracket groups the first two generations as '低階語言' (Low-level languages). The 5th generation is highlighted with a red box and labeled as '自然語言' (Natural language).

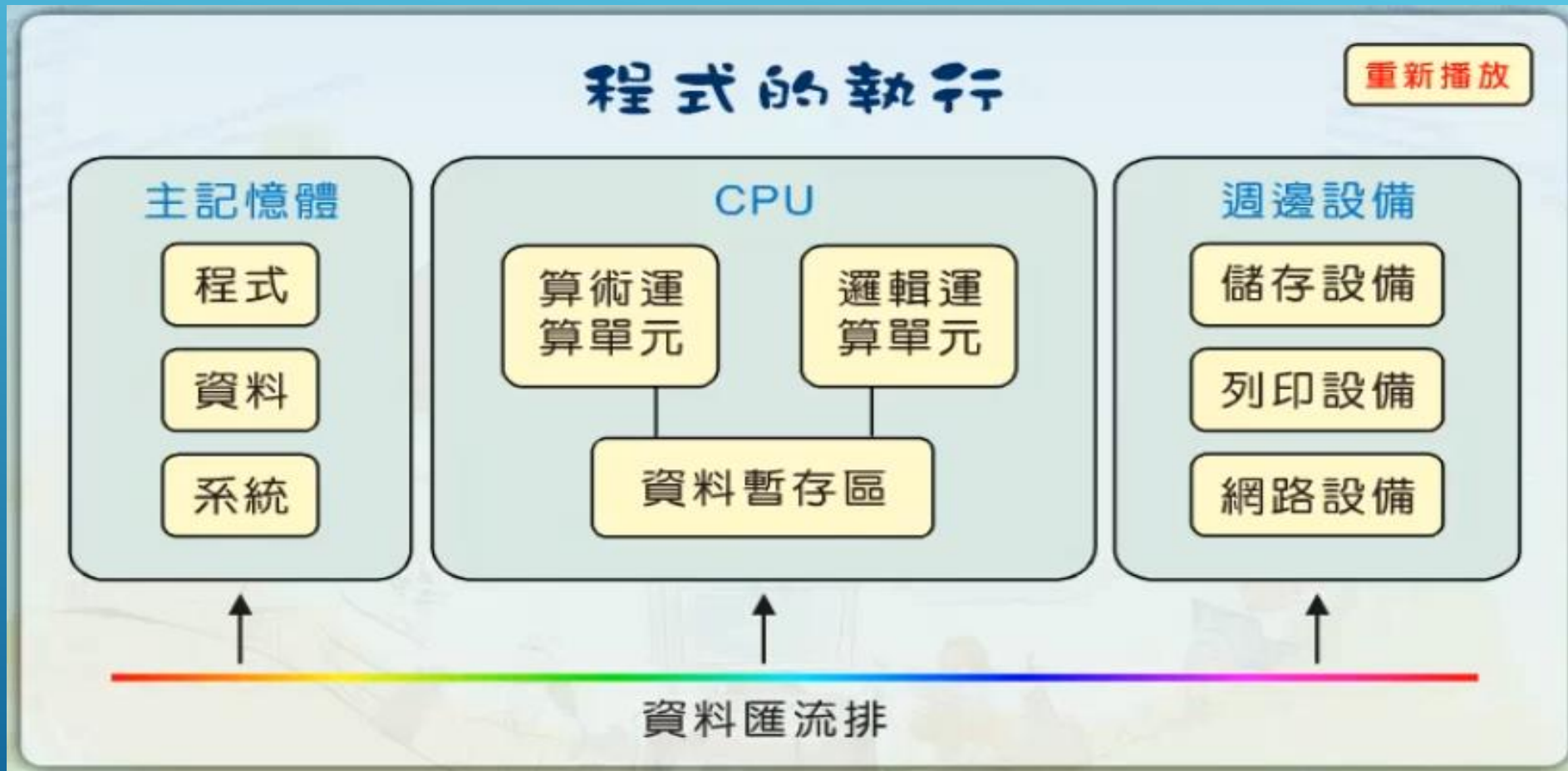
| 時代         | 語言名稱  | 類別   |
|------------|-------|------|
| 西元1950年代初期 | 第1代語言 | 機器語言 |
|            | 第2代語言 | 組合語言 |
| 西元1950年代末期 | 第3代語言 | 高階語言 |
|            | 第4代語言 |      |
|            | 第5代語言 | 自然語言 |

低階語言

第5代語言






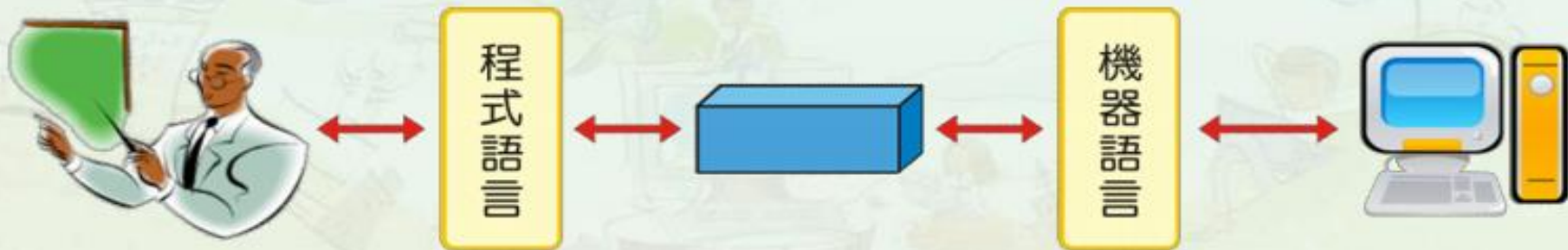
# 第1章 程式語言與系統開發入門



# 第1章 程式語言與系統開發入門

## 語言的抽象化

-  以機器語言層次的程式來處理比較簡單的語言寫出來的程式。
-  用一種比較平易親和的程式語言來撰寫程式。
-  轉換成機器語言，交由 CPU 來執行。



# 第1章 程式語言與系統開發入門

## 程式語言與機器語言之間的轉譯

-  以機器語言存在的系統程式能做程式語言與機器語言之間的轉譯。
-  這一類的程式被稱為「編譯器」(Compiler) 或是「直譯器」(Interpreter)。
-  Fortran、COBOL、Pascal、C、C++、Basic、Java、Ada 等都是有名的程式語言。
-  也有自然語言 (Natural language) 方面的研發，希望能把與電腦之間的溝通更進一步地簡化。



# 第1章 程式語言與系統開發入門

## 程式語言的組成



一個程式語言最主要的組成是其語法 (Syntax) 與語意 (Semantics)。



語法代表程式語言所提供的語言，會決定寫出來的程式的外觀。



語意代表各種語法所描述的功能，以及被執行時產生的作用。

# 第1章 程式語言與系統開發入門

## 程式語言文法主要的成分

-  終結符號 (Terminal symbol)
-  非終結符號 (Non-terminal symbol)
-  產生規則 (Production)
-  目標符號 (Goal symbol)

# 第1章 程式語言與系統開發入門

## 文法規則的觀念





# 第1章 程式語言與系統開發入門

## 貝諾爾格式

文法：

$\langle \text{goal} \rangle := [x]y\{\langle \text{money} \rangle\}$

$\langle \text{money} \rangle := 10 \mid 20 \mid 30$

有效的子句

無效的子句

xy10

$\langle \text{goal} \rangle := [x]y\{\langle \text{money} \rangle\}$

$\langle \text{money} \rangle := 10 \mid 20 \mid 30$

y20x

$\langle \text{goal} \rangle := [x]y\{\langle \text{money} \rangle\}$

$\langle \text{money} \rangle := 10 \mid 20 \mid 30$

# 第1章 程式語言與系統開發入門

$X := 25 - 3 * 7$        $Y := 25 - 3 * / 7$

## 運算式的文法

- |   |  |
|---|--|
| 1 | <運算子句> ::= <運算式> =                             |
| 2 | <運算式> ::= <數值> [<運算元><運算式>]                    |
| 3 | <數值> ::= [<正負號>]<位元組>[<位元組>]                   |
| 4 | <位元組> ::= <位元>[<位元組>]                          |
| 5 | <位元> ::= 0   1   2   3   4   5   6   7   8   9 |
| 6 | <正負號> ::= +   -                                |
| 7 | <運算元> ::= +   -   *   /                        |

# 第1章 程式語言與系統開發入門

## 文法分析的實例

BNF(Backus Naur Form)的表示法:

$\langle \text{goal} \rangle ::= \text{I} - \langle \text{verb} \rangle - \langle \text{person} \rangle - \{ \langle \text{amount} \rangle \} [-\text{dollars}].$

$\langle \text{verb} \rangle ::= \text{owe} \mid \text{lend}$

$\langle \text{person} \rangle ::= \text{you} \mid \text{her} \mid \text{him}$

$\langle \text{amount} \rangle ::= 1 \mid 2 \mid 3 \mid 4 \mid 5$

**I-owe-her-dollars-212318.**  
(不符合上述的文法)

**I-lend-her-255.**  
(符合上述的文法)

# 第1章 程式語言與系統開發入門

命令式的程式語言

函數式的程式語言

邏輯式的程式語言

物件導向式的程式語言

## 程式語言採用的方法論 (Paradigm)

- 通常每一種語言的語法都會有差異，所以從程式語言所採用的方法論 (Paradigm) 來做分類，比較容易找到同一類的語言。
- 常見的程式語言方法論有 4 種：
  1. 命令式的 (Imperative)
  2. 函數式的 (Functional)
  3. 邏輯式 (Logic)
  4. 物件導向的 (Object-oriented)
- 這些方法論決定了程式語言的運算模型 (Model of computation)。



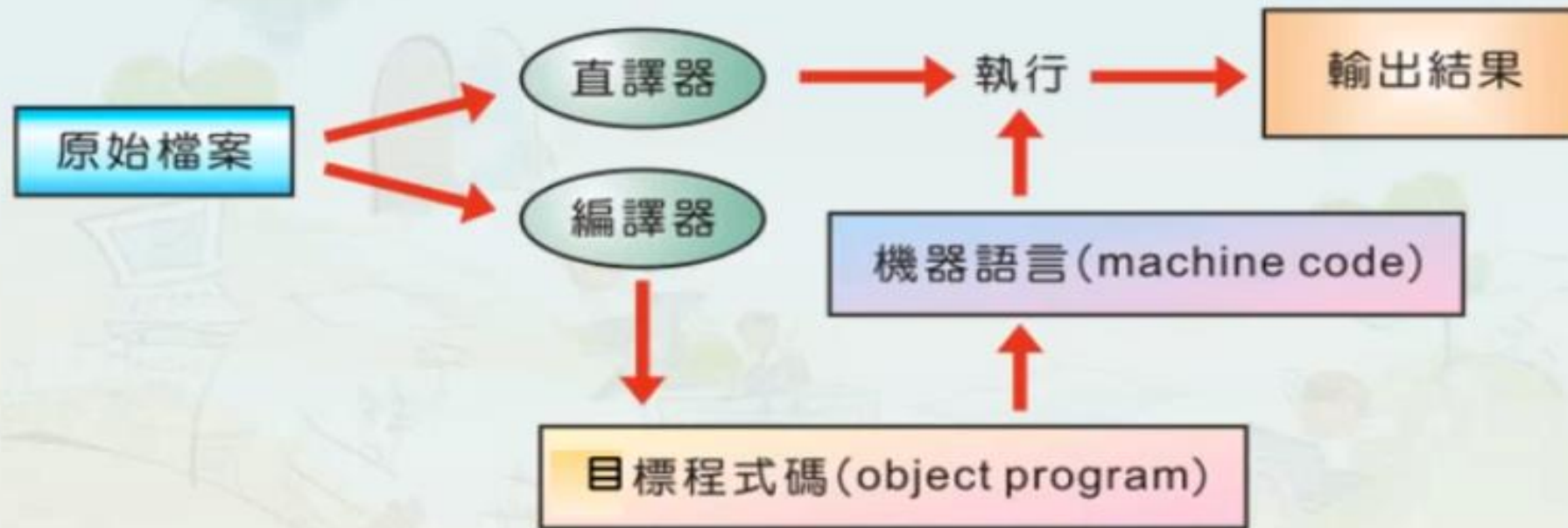
# 第1章 程式語言與系統開發入門

## 物件導向式的程式語言

- 物件導向式的程式語言(object-oriented programming language)具有3種主要的特性
  1. 封裝(encapsulation)
  2. 繼承(inheritance)
  3. 多元性(polymorphism)
- Smalltalk、C++、Java與C#都算是這一類的程式語言。

# 第1章 程式語言與系統開發入門

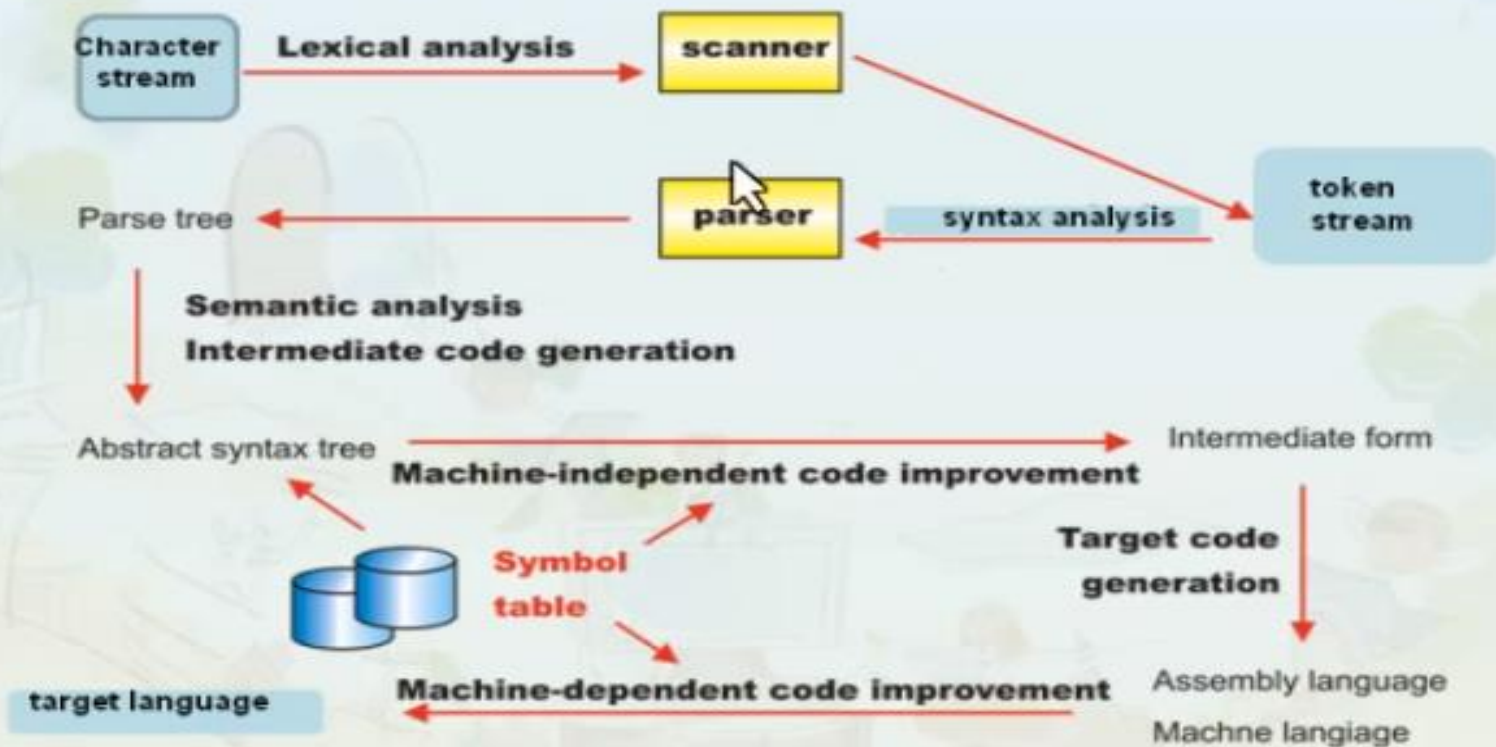
## 一般程式被處理的過程





# 第1章 程式語言與系統開發入門

## 詳細的編譯 (compilation) 流程



# 第1章 程式語言與系統開發入門

## 繫結 (binding) 發生的時機



語言設計的時候



語言製作的時候



寫程式的時候



編譯時期



連結時期





載入時期 (load time)



執行時期 (run time)

# 第1章 程式語言與系統開發入門




## 繫結 (binding) 的種類

-  靜態繫結 (static binding) 表示繫結發生在執行時期之前。
-  假如繫結發生在執行時期則稱為動態繫結(dynamic binding)。
-  繫結的時間 (binding times) 在程式語言的設計上是很重要的問題，提早的繫結時間 (early binding times) 有助於提昇程式效能，延後的繫結時間 (late binding times) 則容許比較大的彈性。



# 第1章 程式語言與系統開發入門

## 程式所描述的資料

-  資訊本身可以分成數種資料型態 (data types)，不同的資料型態有不一樣的格式與儲存方式。
-  寫程式的時候要依照資訊的特性來決定資訊所屬的資料型態。
-  資料型態也可以依需要組合成資料結構 (data structure)，在各種場合中運用。

# 第1章 程式語言與系統開發入門

## 基本資料型態



一般程式語言寫出來的程式處理的最基本的資料常稱為基本值 (Literals)，例如整數值 (Integer literals)、浮點數值 (Floating-Point literals)、布林值 (Boolean literals)、字元值 (Character literals)、字串值 (String literals) 等。



基本值可以指定成變數 (Variable) 的值，該變數就具有基本值所屬的資料型態。






例如變數  $X = 20$ ， $X$  是程式變數，20 是整數值，則  $X$  的資料型態就是整數資料型態 (Integer data type)。



# 第1章 程式語言與系統開發入門

## 認識運算子 (operator)

-  各種資料型態都有一些相關的運算子，用來描述數值的運算。
-  例如常見的加 ( + )、減 ( - )、乘 ( \* )、除 ( / ) 和取餘數的算術運算子。
-  布林值可以使用於布林運算，也就是邏輯運算。



# 第1章 程式語言與系統開發入門

## 認識運算子 (operator)



各種資料型態都有一些相關的運算子，用來描述數值的運算。



例如常見的加 ( + )、減 ( - )、乘 ( \* )、除 ( / ) 和取餘數的算術運算子。



布林值可以使用於布林運算，也就是邏輯運算。

# 第1章 程式語言與系統開發入門

## 程式中資料的使用



### 常數 (constant) :



常數的值在程式執行的過程中不會改變，例如圓周率在數學裡頭是一個固定的數值，在程式裡頭就可以當成是一個常數。



### 變數 (variable) :



變數的值在程式執行的過程中會改變，所以變數也可以看成是一個儲存資料的空間，裡面儲存的值會改變。

# 第1章 程式語言與系統開發入門







## 描述問題



# 第1章 程式語言與系統開發入門

## 程式的邏輯 (program logic)

-  程式的邏輯是程式執行順序的依據。
-  解決任何的問題除了需要完整的資料之外，還要有解決問題的方法與步驟，這就是程式邏輯的由來。
-  假如寫出來的程式沒有按照程式的邏輯來執行，代表所寫的程式沒有正確地描述解決問題的方法。
-  若是寫出來的程式的確按照程式的邏輯來執行，但是結果不正確，就表示原來的解決方法不對。

# 第1章 程式語言與系統開發入門

## 閏年的判定方法以口語的方式描述



我們在判定某一年份是不是閏年時，會先看看能否被4整除，若不能被4整除，則不是閏年；若是能被4整除但不能被100整除，就算是閏年；若是能被4與100整除，但不能被400整除，則不能算閏年；假使可以被400整除，則也算是閏年。