解決問題 Problem Solving

1.釐清問題	Problem Definition
2.問題分析	Problem Analysis
3.設計算法	Design of Algorithm
4.發展解決方案	Implementation 實施
5.除錯、測試	Debugging & Testing
6.文件編製	Documentation

1.釐清問題 Problem Definition

- ☑ 最重要的步驟 (問題定義)
- ☑ 確保對問題有清晰的了解，
  並使用清楚精確文字描述定義問題：
  - 要解決的是什麼問題
  - 解決的方案應提供些什麼功能？
  - 要考慮什麼限制 constraints 或特殊情況？
- 對問題的了解有多清晰，
  決定於對問題的認識程度
- 例: 計算薪俸稅salary tax ...

1.釐清問題 (續)

- 參考網頁: http://www.ird.gov.hk
- (a) 累進稅率 Progressive Tax rate

NET Income	Rate	Tax
First 40,000	2%	800
Next 40,000	7%	2,800
Next 40,000	12%	4,800
Remainder	17%	

1.釐清問題 (續)

- (a) 累進稅率 Progressive Tax rate
- (b) 標準稅率 Standard Tax rate 15%
  標準稅 Tax = Income × 15%
- 應繳稅Tax payable = min (a , b)

- Net income = Income – allowances
- 應課稅入息 = 總收入 – 免稅額

1.釐清問題 (續)

- 免稅額 **allowances**：

(1) 個人免稅額	**$120,000**
(2) 父母免稅額	**$38,000/**每人
(3) 子女免稅額	**$70,000/**每人

假設此程式的限制 **constraint** 是
只計算單身人仕的薪俸稅

1.釐清問題 (續)

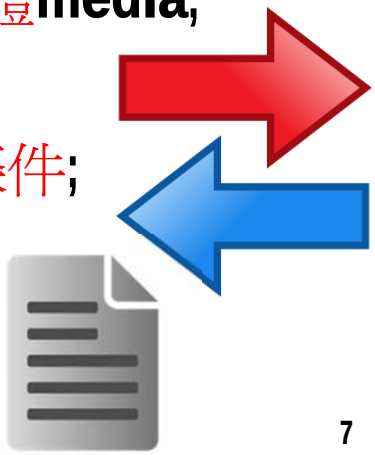
- 收集用家要求
  requirement specification:

1. 使用者必須爲單身人仕
2. 使用者需要輸入：收入及受供養父母人數
3. 計算：個人及父母免稅額
4. 計算：累進稅及標準稅
5. 計算：薪俸稅(最少)
6. 輸出結果：**3,4,5**

2.分析問題 Problem Analysis

辨識 **identify** 以下各項:

1. 輸入**inputs**、格式**form**、媒體**media**;
2. 輸出**outputs**、格式、媒體;
3. 限制**constraints** 、情況、條件;
4. 公式**Formulas or equations.**



2.分析問題 (續)

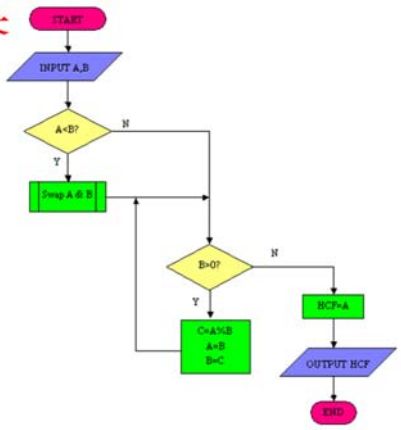
以計算薪俸稅爲例:

1. 輸入 – 個人收入、受供養父母人數
   (鍵盤輸入)
2. 輸出 – 應課稅入息、個人、父母免稅額
   、累進稅、標準稅、薪俸稅
3. 限制 – 個人收入 **$0 – $1,000,000**、
   受供養父母人數 **0–4**
4. 公式 – 見**p.3-5**

3.設計算法 Algorithm Design

算法是一連串(有限)的步驟

- 當被執行時，
  可以提供**解決問題**的方案
  e.g. sorting, searching
- 把輸入值轉化爲輸出值



3.設計算法 (續)

算法必須符合以下條件:

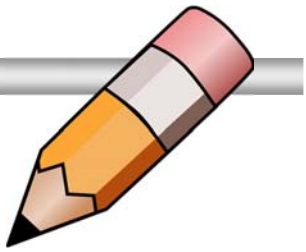
1. **不含糊 Unambiguousness**
2. **一般性 Generality**

   1吋 = 2.54 cm, n吋= ? Cm

   ```
   cm = 2.54*inch;
   ```

3. **正確 Correctness**
4. 在**有限時間**內 **Finiteness** 輸出**結果**

3.設計算法 (續)



為指定的問題，設計算法，必須

1. 可以**證明**它是**正確**的
2. 可以把設計寫在**紙上** (不單在腦中)
3. 可以用合適的**工具**去代表及記載

a. **人類語言natural language**
   有許多含糊的地方

b. 比較合適的工具：
   偽代碼Pseudo-code、**流程圖Flow-chart**

3.1 算法的**表示方式**

偽代碼、流程圖的**用途**：

1. 有助**溝通**工具 — **Communication**
2. 發展解決方案 — **Implementation**
3. 除錯、測試 — **Debugging & Testing**
4. 文件編製 — **Documentation**
5. 系統維修及擴展可作**參考**

3.2算法：偽代碼 Pseudo-code

- 使用：中/英文+電腦語言(**如果、只要…**)
- 三種**控制**結構：**basic control constructs**
- **(1) 順序**控制結構 **sequence**
  e.g. a=10; b=20; c=a+b; c++;
- **(2) 選擇**控制結構 **selection**
  e.g. if(a>=10) …
- **(3) 迭代**控制結構 **repetition / iteration**
  e.g. while(n>0) …

3.2.1 順序控制結構 Sequence

一連串順序的步驟，例如：

1. **Read_Annual_Income** 輸入全年收入
2. **Read_No_of_Parents** 受供養父母人數
3. **Calculate_Net_Income** 計算應課稅入息
4. **Calculate_Progressive_Tax** 計算累進稅
5. **Calculate_Standard_Tax** 計算標準稅
6. **Print_Chargeable_Tax** 列印薪俸稅

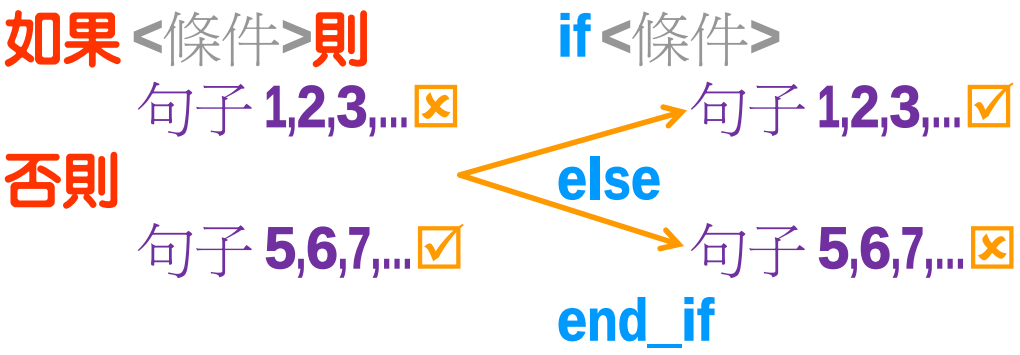
3.2 .1 順序控制結構 Sequence

- 'begin' & 'end' 表示區段 開始、結束
- 區段內句子必須有系統地縮排 readability

```
Calculate_Net_Income 計算應課稅入息
{
    Parent_Allowance = No_of_Parents×38000
    Net_Income = Annual_Income - Parent_Allowance
}
```


3.2.2 選擇控制結構 Selection

- 如果，則，否則 'if', 'else' & 'end_if' :



3.2.2 選擇控制結構

// 累進稅≤標準稅

如果 Progressive_Tax <= Standard_Tax 則

①Tax_Payable = Progressive_Tax

否則

②Tax _Payable = Standard_Tax

輸出 Tax _Payable

3.2.2 巢狀選擇控制結構

- 簡單的選擇 Y/N
  2-way decision-making

```
if(m>=50) grade='P';
else grade='F';
```

- 複雜的選擇nested if
  multi-way

```
if(m>=90) grade='A';
else if(m>=80) grade='B';
else if(m>=70) grade='C';
...
```

- 例如: 累進稅率
  (首4萬、次4萬．．．)

3.2.2 巢狀選擇控制結構

如果 Net_income <= 40000 則
    Progressive_Tax = Net_income ×2%
否則如果Net_income <= 80000 則
    Progressive_Tax = 800 + (Net_income - 40000) ×7%
否則如果Net_income <= 120000 則
    Progressive_Tax = 3600 + (Net_income - 80000) ×12%
否則
    Progressive_Tax = 8400 + (Net_income - 120000) ×17%

3.2.2 巢狀選擇控制結構

- 多層巢狀選擇控制結構會造成渾亂
- 可轉爲簡單(2-way)選擇控制結構(沒否則)

如果[Net_income <= 40000]
    則 Progressive_Tax = Net_income ×2%
如果(Net_income > 40000) 及 (Net_income <= 80000)
    則 Progressive_Tax = 800 + (Net_income - 30000) ×7%
如果(Net_income > 80000) 及 (Net_income <= 120000)
    則 Progressive_Tax = 3600 + (Net_income - 60000) ×12%
如果(Net_income > 120000)
    則 Progressive_Tax = 8400 + (Net_income - 90000) ×17%

3.2.3 迭代控制結構 Iteration

- **重複執行**某程式段，直至乎合某條件
- **while, do-while**: 只要…進行…停止進行

只要 <條件>進行

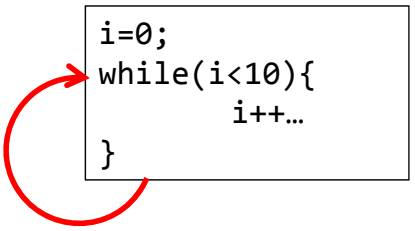
執行句子

停止進行

while <condition>

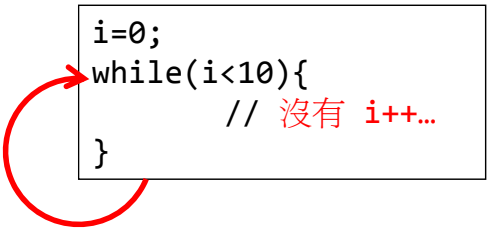
loop-statements

end_while



3.2.3 迭代控制結構 Repetition

- 只要條件爲『真Y』便**重複**執行某一區段程序
- 直至條件爲『假N』便**停止**執行
- 該區段程序可以改變條件(i<10)的值
- 否則會進入**無窮**迴路 **infinite loop**



3.2.3 迭代控制結構 Repetition

輸出 "Enter positive annual income:"

輸入 Annual_Income 全年收入

只要 Annual_Income <= 0進行

輸出"Error! Please enter again."

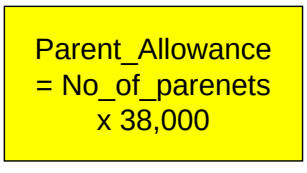
輸入 Annual_Income

停止進行

```
printf("Enter…");
scanf("%i",&income);
while(income<=0){
        printf("Error…");
        scanf("%i",&income);
}
```


3.3算法：流程圖 Flow Chart

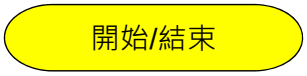
http://www.rff.com/flowchart_samples.htm



處理Process

決定Decision

輸入輸出Input or Output

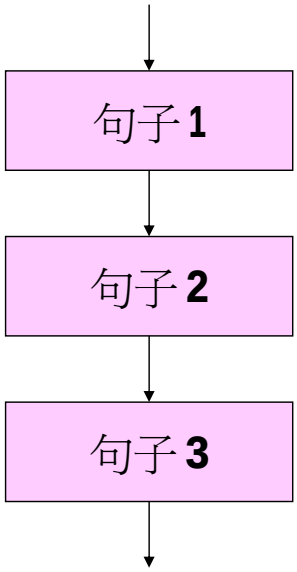


終端Terminal

常用圖形符號

3.3 流程圖 (1)

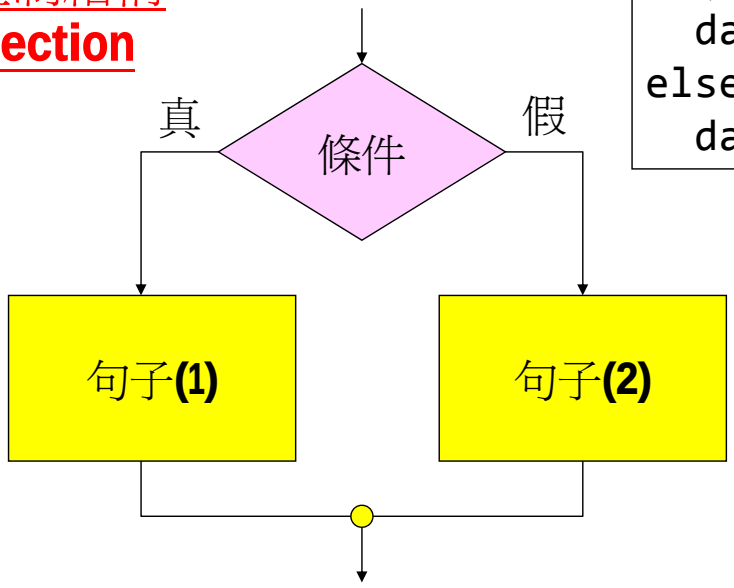
順序控制結構
①Sequence




```
a=3;
b=4;
c=a+b;
printf…
scanf…
```


3.3 流程圖 (2)

選擇控制結構
②Selection



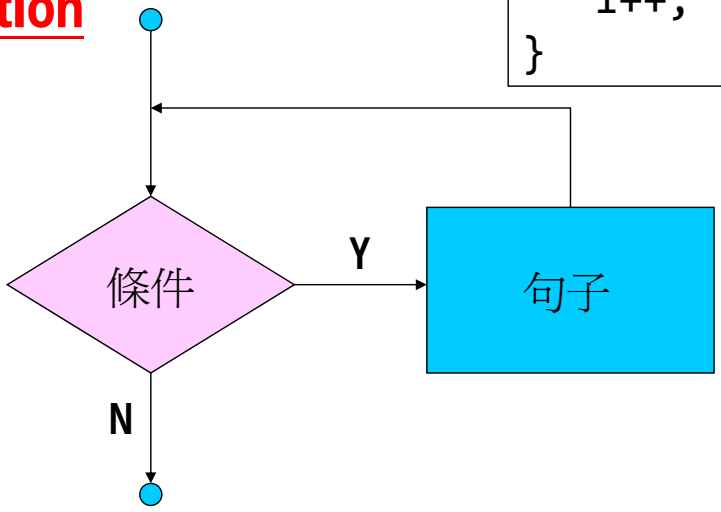

```
if(mm==2)
  days = 28;
else
  days = 31;
```

3.3 流程圖 (3)

```
i = ?
do{
    i++;
}while(i<10);
```

```
i = ?
while(i<10){
    i++;
}
```

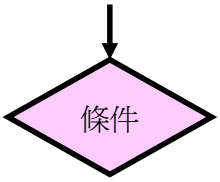
迭代控制結構
③Repetition



3.4 偽代碼、流程圖比較

	偽代碼	流程圖
性質	文本	圖像
文字描述	無限	有限
篇幅大小	一般	佔較大空間
預備功夫	一般	較多
注重	行動Operation	流程Control Flow

如果、則、否則
只要
輸入、輸出



4.發展解決方案 Implementation

- 把算法**Algorithm**
  翻譯爲電腦程序**Program**;
- 電腦程序是一連串
  有特定次序的句子**statements**，
  當被執行時，
  會產生解決方案**solution**。

4.1 程式錯誤 Programming Error

- 設計錯誤:
  - **(1) 設計錯誤Design errors**
    - **(a)** 設計算法
    - **(b)** 算法→程式
    - **(c)** 測試後發現錯誤
  - ☑ 需要重新再檢視
    問題分析→算法→ 程式設計
    →編譯→測試數據

4.2程式錯誤 (2)

- 程式錯誤:
  - **(2) 編譯錯誤 Compilation errors**
    - ☑ 語法錯誤 Syntax errors
    - ☑ 由編譯器compiler發現
    - ☑ 大都提供診斷訊息diagnostic
  - **(a) 警告 Warning messages**
  - **(b) 錯誤 Error messages (; 未宣告...)**

4.3程式錯誤 (3)

- **(3) 執行錯誤Run-time errors**
  - ☑ 程式被執行時，由電腦發現
  - ☑ 程式試圖執行不合法動作
    例如：不合法檔案存取、÷0
- **(4) 邏輯錯誤Logical errors**
  - ☑ 最難發現 i<10, i>10, i<=10
    i=i+1; i=1+1;

5.除錯及測試Debug & Testing

- **程式測試 Program Testing**
- 使用大量的**測試數據 test data sets**
- 走遍**每一段**程式碼 **logical path**
- 嘗試去發現錯誤
- 但**不能証明**它沒有錯誤
- 例如：**Net_Income = 0, 40000, 40001, 80000, 80001, 120000, 120001.**

6.文件編寫 Documentation

有了一個可行**workable**，已全面測試過的程式，仍需文件編製，因爲

1. 軟件再用 **software re-use;**
2. 參考用 **for reference;**
3. 方便日後維修 **maintenance**
   (修改、除錯、增新擴展**)**
4. 這不是最後階段，
   應在每個階段中進行文件編制

6.文件編製 (續)

內容包括:-

- 問題定義 **problem definition**
- 精確的使用者要求 **user requirements**
- 輸入輸出**I/O**、條件限制、公式、等
- 算法 **Algorithm (**僞代碼、流程圖**)**
- 測試樣本**/**結果 **sample test run**
- 源程式碼 **Source program listing**