



澳門城市大學  
Universidade da Cidade de Macau  
City University of Macau

# 計算機科學導論



主講人 |

姓名 張琪

Name Zhang Qi

澳門城市大學

City University of Macau



# 第四章 計算機軟件工程簡介 和程序設計語言基礎

本章學習要點：

- 1 程序設計語言概述
  - 2 程序設計方法
  - 3 程序設計基礎知識
  - 4 軟件工程的概念
  - 5 軟件開發模型與方法
  - 6 統一建模語言（UML）
- 

## 4.1 程序設計語言概述

---

### 4.1.1 程序的基本概念

- 程序就是能够實現特定功能的一組指令序列的集合
  - 其中，指令可以是機器指令、匯編語言指令，也可以是高級語言的語句命令，甚至還可以用自然語言描述的運算、操作命令等
  - 程序設計是程序員根據程序設計語言的語法規定，編寫指令指示計算機完成某些工作的過程
  - 程序員根據程序設計語言規則編寫程序，這些指令序列也被稱為代碼（碼農是什麼意思？）
  - 這些編寫的指令代碼集合稱為源代碼或者源程序
- 打開一個你電腦上的軟件，想一想它們是怎麼編寫出來的？

## 4.1 程序設計語言概述

---

### 4.1.2 計算機程序設計語言

- 程序設計語言使得人們能夠與計算機進行交流，其種類非常多，總來說可以分為低級語言和高級語言兩大類



➤ 圖片顯示的這幾種程序語言你們熟悉嗎？

## 4.1 程序設計語言概述

---

### 4.1.2 計算機程序設計語言

- 低級語言

- 低級語言包括兩種類型：機器語言和匯編語言

- (1) 機器語言

- 機器語言面向機器，可以由CPU直接識別和執行
- 不同的機器能夠識別的機器語言是不相同的
- 機器語言指令都是用一串0、1構成的二進制位串來表示的
- 指令系統是機器提供的機器指令的集合
- 用二進制編碼表示的指令，稱為機器指令，或稱為機器碼
- 用機器指令編寫的程序稱為機器語言程序，或稱為目標程序，這是計算機能夠直接執行的程序
- 機器語言難以閱讀和理解，編寫和修改都比較困難，而且通用性較差

➤ 機器語言舉個栗子？

## 4.1 程序設計語言概述

### 4.1.2 計算機程序設計語言

- 低級語言

- 低級語言包括兩種類型：機器語言和匯編語言

- (2) 匯編語言

- 匯編語言也稱符號語言
- 指令助記符是指令英文名稱的縮寫，容易記憶
- 所謂匯編語言，就是采用字母、數字和符號來代替由一個個0和1構成的指令操作碼、寄存器、數據和存儲地址等，并在程序中用它們代替二進制編碼數，這樣編寫出來的程序就稱為符號語言程序或匯編語言程序
- 大多數情況下，一條彙編指令直接對應一條機器指令，少數對應幾條機器指令
- 匯編語言具有一個本質上與機器語言一一對應的指令系統。匯編語言的實質和機器語言是相同的

➤ 匯編語言舉個栗子？

## 4.1 程序設計語言概述

---

### 4.1.2 計算機程序設計語言

#### ● 低級語言

#### ● 低級語言的特點

- 機器語言和匯編語言都是低級語言。它們具有許多相同的特徵
- 都與特定的計算機硬件系統緊密相關，來自于特定系統的指令系統，可移植性差
- 對程序員專業知識要求高，要求對計算機硬件的結構和工作原理非常熟悉
- 每條指令的功能比較單一，程序員編寫源程序時指令非常繁瑣
- 由于直接針對特定硬件編程，所以最終的可執行代碼非常精煉，并且執行效率高
- 兩者主要的區別在于：機器語言編寫的程序無需翻譯或編譯，CPU能夠直接識別和執行。而匯編語言源程序必須經過彙編才能得到目標程序

➤ 匯編語言相對於機器語言有什麼優勢？

## 4.1 程序設計語言概述

---

### 4.1.2 計算機程序設計語言

- 高級語言

- 高級語言的產生

- 一個問題：如何解決程序的可移植性，即：程序員編寫的源程序如何可以從一台計算機很容易地轉到另一台計算機上工作。爲了解決這些問題，人們引入了高級語言來編寫程序
- 所謂高級語言是一種由表達各種意義的“詞”和“公式”，按照一定的“語法規則”來編寫程序的語言，又稱爲程序設計語言或算法語言
- 高級語言之所以“高級”，就是因爲它使程序員可以完全不用與計算機的硬件打交道，可以不必瞭解機器的指令系統

➤ 高級語言程序樣本舉例說明？





## 4.1 程序設計語言概述

---

### 4.1.2 計算機程序設計語言

- 高級語言

- 高級語言的常見類型

- BASIC 語言
- FORTRAN 語言
- COBOL 語言
- PASCAL 語言
- C 語言
- C++ 和 C# 語言
- 其他高級語言，Python 等

➤ 高級語言種類舉例簡介？





## 4.1 程序設計語言概述

---

### 4.1.2 計算機程序設計語言

- 高級語言

- 高級語言的優點：

- 語句的功能強，程序員編寫的源程序比較短，容易學習，使用方便，可移植性較好，便于推廣和交流

- 高級語言的缺點：

- 編譯程序比匯編程序複雜，而且編譯出來的目標程序往往效率不高，目標程序的長度比有經驗的程序員所編寫的同樣功能的匯編語言程序要長一半以上，運行時間也要長一些
- 因此，在很多對時間要求比較高的系統，如某些實時控制系統或者大型計算機控制系統中，低級語言，主要是匯編語言，仍然得到了一定的應用





## 4.2 程序設計方法

---

### 4.2.1 結構化程序設計

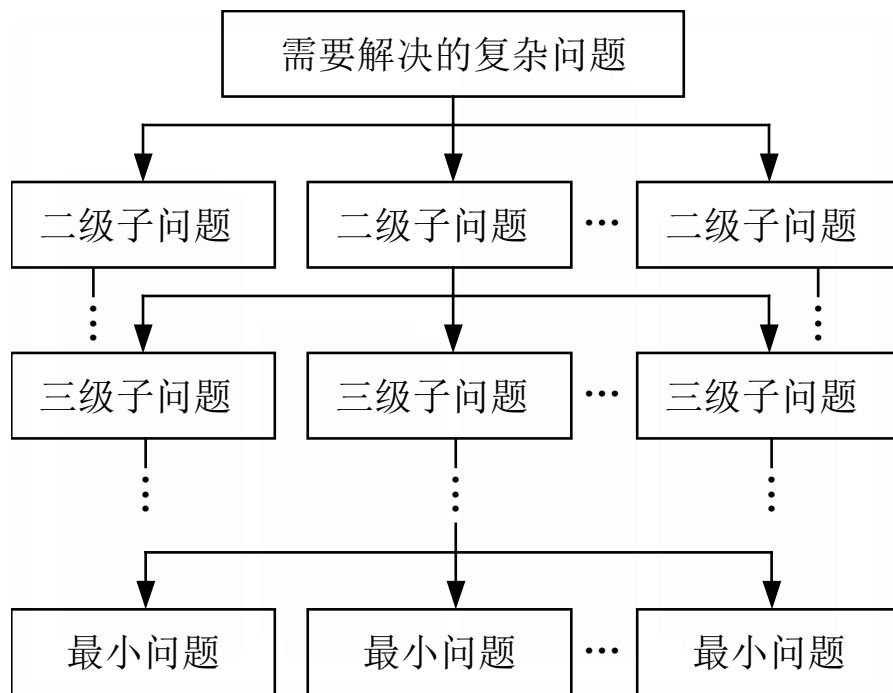
- 1965年，艾茲格·W·迪科斯徹（Edsger Wybe Dijkstra，1930~2002）提出了結構化的概念，它是軟件發展的一個重要的里程碑
- 結構化程序設計是以模塊功能和處理過程設計為主的詳細設計的基本原則，採用自頂向下、逐步求精及模塊化的程序設計方法
- 使用三種基本控制結構構造程序，任何程序都可由順序、選擇、循環三種基本控制結構構造
- 結構化程序設計思想
  - 採用自頂向下、逐步求精的設計方法和單入口單出口的控制結構



## 4.2 程序設計方法

### 4.2.1 結構化程序設計

- 采用自上而下解決問題的思路如圖：



- 如何理解結構化程序設計，舉個栗子
- “要把大象裝冰箱，總共分幾步？三步，第一步，把冰箱門打開，第二步，把大象裝進去，第三部，把冰箱門蓋上”



## 4.2 程序設計方法

---

### 4.2.1 結構化程序設計

- 20世紀60年代末到70年代初，當時採用結構程序設計方法的兩個最著名項目是：
  - ①紐約時報信息庫管理系統，含8.3萬行源代碼，只花了11人·年，第一年使用過程中，只發生過一次使系統失效的軟件故障；
  - ②美國宇航局空間實驗室的模擬系統，含40萬行源代碼，只用兩年時間就全部完成。





## 4.2 程序設計方法

---

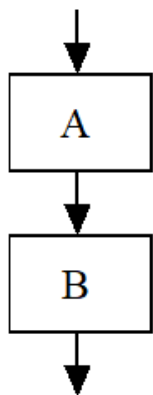
### 4.2.1 結構化程序設計

- 結構化方法
  - 結構化方法有助于在正式編寫程序之前充分理解問題的實質和實現方法，并且可以在具體編碼過程中提供指導。
- 結構化程序設計方法
- 結構化程序設計的原則是：
  - (1) 使用順序、選擇、循環3種基本控制結構表示程序邏輯
  - (2) 程序語句組織成容易識別的語句模塊，每個模塊都是單入口、單出口
  - (3) 嚴格控制GOTO語句的使用

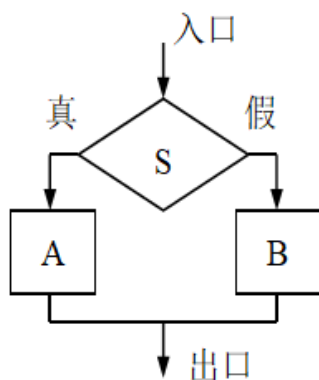


## 4.2 程序設計方法

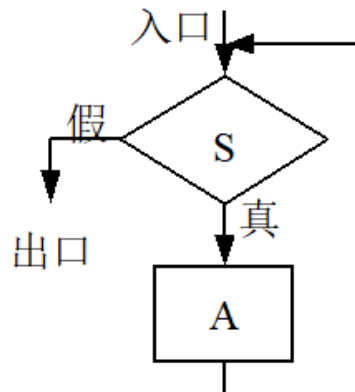
### 4.2.1 結構化程序設計



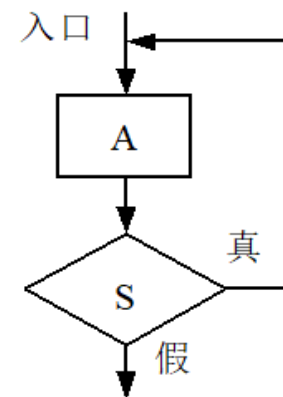
(a) 顺序结构



(b) 选择结构



(c) while循环



(d) do-while循环

➤ 如何理解上述的結構，舉個栗子



## 4.2 程序設計方法

---

### 4.2.2 面向對象程序設計

- 面向對象的思想：
- OO(Object Oriented，面向對象)的程序設計把客觀事物看作具有屬性和行爲的對象，通過抽象找出同一類對象的共同屬性(靜態特徵)和行爲(動態特徵)，形成類
- 阿倫(Alan Kay，1940~)設計出了名震業界的Smalltalk語言，被業界公認為“面向對象編程系列語言”的代表作品

➤ 如何理解面向對象的思想，舉個栗子





## 4.2 程序設計方法

### 4.2.2 面向對象程序設計

- 例如，對一批書籍進行歸納、抽象。讀者可能關心的是書籍的學科種類、價格、出版社信息，而銷售這批書籍的書店經銷商希望開發一個書籍的進貨、銷售情況的管理軟件，則他所關心的屬性除了以上這些信息以外，更關心可能是書籍的銷售量和銷售速度
- 由此可以看出，同一個研究對象，由于所研究問題的側重點不同，就可能產生不同的抽象結果；即使對於同一個問題，解決問題的要求不同，也可能產生不同的抽象結果，這些結果的不同就表現在不同的數據和方法上
- 面向對象程序設計方法通過封裝將功能分散到不同的類中去，這樣做的好處是降低了代碼的複雜程度，使類可以重用。但這種做法增加了代碼的重複性
- 日志功能就是一個典型案例。在一個系統中，日志功能往往需要橫跨多個業務模塊，那麼就需要在這些模塊中分別加入日志相關代碼



## 4.2 程序設計方法

---

### 4.2.4 事件驅動程序設計

- 事件驅動程序設計是一種程序設計模型，這種模型的程序執行流程是由使用者的動作，如鼠標或鍵盤的按鍵動作或者是由其他程序的信息來決定的
- 相對於批程序設計而言，程序執行的流程是由程序設計員來決定的。事件驅動程序設計這種設計模型是在互動程序的情況下孕育而生的



## 4.2 程序設計方法

---

### 4.2.5 程序設計風格

- 隨著計算機技術的發展，軟件規模擴大了，軟件的複雜性也增強了。爲了提高程序的可閱讀性，就要建立良好的編程風格
- 程序設計風格是指一個人編制程序時所表現出來的習慣、邏輯思路等特點
- 在程序設計中要使程序結構合理、清晰，形成良好的編程習慣，對程序的要求不僅是可以在機器上執行，給出正確的結果，而且還要便于程序的調試和維護
- 這就要求編寫的程序不僅自己能看得懂，而且也要讓別人能看得懂

➤ *如何理解要寫的自己能看得懂，而且也要讓別人能看得懂？*



## 4.2 程序設計方法

---

### 4.2.6 編譯程序

- 20世紀50年代，IBM的約翰·巴克斯帶領一個研究小組對FORTRAN語言及其編譯器進行開發。受限于人們對編譯理論知識的缺乏，開發工作變得既複雜又艱苦
- 與此同時，諾姆·喬姆斯基(Noam Chomsky，1928~)開始對自然語言結構的研究。他的發現最終使得編譯器的結構異常簡單，甚至還有一些自動化形式

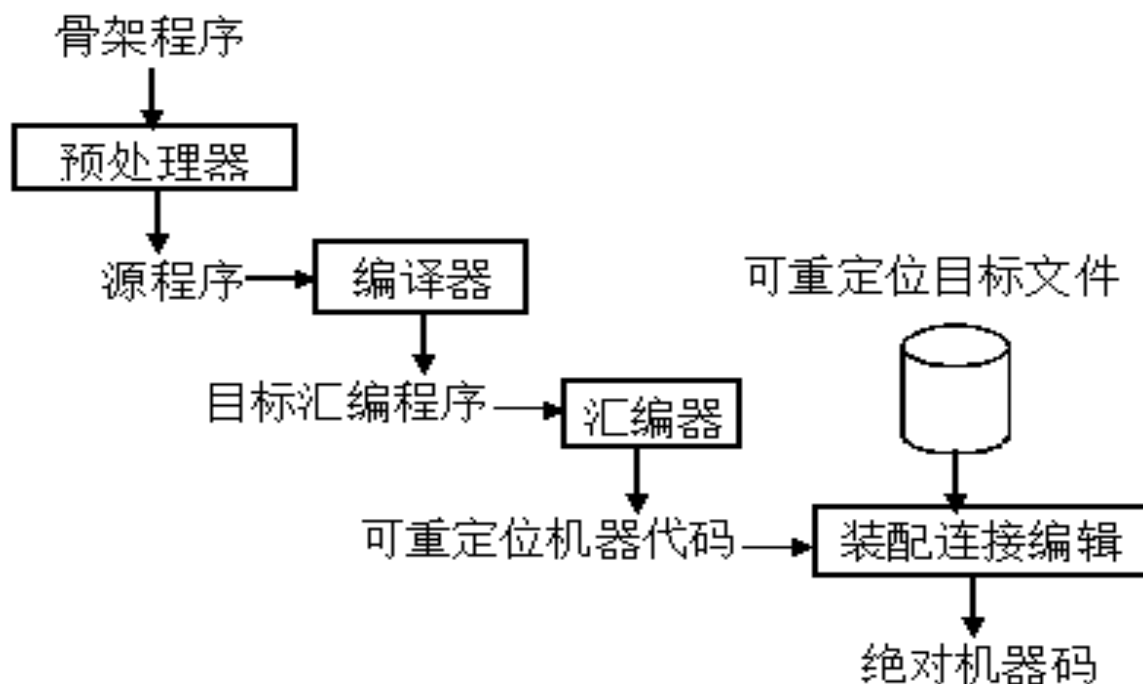
➤ 為什麼需要編譯程序？



## 4.2 程序設計方法

### 4.2.6 編譯程序

- 高級語言編寫的源程序需要“翻譯”成計算機能夠識別的機器語言，機器才能執行，這種“翻譯”程序被稱為語言處理程序
- 語言處理的過程：



➤ 為什麼需要編譯程序？

## 4.2 程序設計方法

### 4.2.6 編譯程序

- 一個翻譯程序能够把諸如FORTRAN、Pascal、C、Ada、Smalltalk或Java這樣的“高級語言”編寫的源程序轉換成邏輯上等價的諸如匯編語言之類的“低級語言”的源程序，這樣的翻譯程序則稱之為編譯程序
- 編譯程序的功能如圖所示：



# 思考題

---

- 常見的高級語言有哪些，請列舉幾個。
- 什麼是匯編語言？請簡要說明。
- 常見的程序設計方法有哪些？請列舉幾個。
- 為什麼需要編譯程序？請簡要說明。

---

休息一下

**Take a break**



## 4.3 程序設計基礎知識

---

### 4.3.1 基本概念

- 程序設計是指用計算機語言對所要解決的問題中的數據以及處理問題的方法和步驟所做的完整而準確的描述的過程。程序設計步驟如下：
    - (1) 確定要解決的問題
    - (2) 分析問題。在著手解決問題之前，應該通過分析，充分理解問題，明確原始數據、解題要求、需要輸出的數據及形式等
    - (3) 選擇計算方法
    - (4) 確定數據結構和算法。算法是解題的過程。首先集中精力于算法的總體規劃，然後逐層降低問題的抽象性，逐步充實細節，直到最終把抽象的問題具體化成可用程序語句表達的算法。這是一個自上而下、逐步細化的過程
- 以把大象裝進冰箱為例



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 用高級語言編寫的源程序能提高程序員的開發效率，高級語言程序設計依賴于各自特定的語句和語法
- 在高級語言中，語句是構成源程序的基本單位
- *舉個高級語言編寫的栗子*



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 高級語言的共同特性
- 高級語言的基本符號
  - 高級語言的語法成分都是由基本符號組成的，基本符號可以分爲單字符和多字符兩種。單字符基本符號由單個字符組成，在高級語言中通常包括下列幾種單字符基本符號
- 字母：
  - 大寫英文字母A ~ Z，小寫英文字母a ~ z，共52個符號。
- 數字：
  - 0 ~ 9，共10個數字符號
- 特殊字符：
  - + (加)、- (減)、\* (乘)、/ (除)、^ (乘方)、= (等號)、( (左括號)、) (右括號)、> (大于)、< (小于)、, (逗號)、(空格)等



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 高級語言的共同特性
- 高級語言的基本元素
  - 基本元素由基本符號組成，可分為數、邏輯值、名字、標號和字符串等5大類
- 數
  - 它由0 ~ 9共10個基本數字和其他一些符號(如小數點 “.”、正負號 “+、-” 及指數符號 “E” 等所構成
- 邏輯值
  - 由真(True)和假(False)兩個值構成
- 舉個高級語言編寫的栗子



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 高級語言的共同特性
- 名字
  - 由字符組成，一般約定名字的開頭是字母或者下劃綫，其後可為字母或數字，如XYZ、A123、\_C等。名字可用來定義常量、變量、函數、過程或子程序的，也被用來定義成某些東西，故也稱為標識符。在高級語言中，一般還規定了組成名字的字符的長度，即字符個數
- 標號
  - 是在高級語言中的程序語句前所加的一個名字，主要用來指示程序可能的轉移方向
- 字符串
  - 由一串字符所組成。在不同的高級語言中，字符串中的多個字符放在一對單引號或雙引號中
- 舉個高級語言編寫的栗子



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 高級語言的共同特性
- 基本的數據類型
  - 任何一個計算機程序都不可能沒有數據，數據是程序操作的對象。通常，一種高級語言都會定義一些基本的數據類型，通常包括整數類型、實數類型和字符類型等
  - 高級語言中，在使用變量前，必須為每個變量分配所需大小的內存單元空間。因此，幾乎任何一種高級語言都要求變量必須先定義後使用標號
- 數組類型
  - 數組是若干個相同類型數據的集合
- 此外還有集合，隊列等更複雜的數據類型



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 高級語言的共同特性
- 運算符與表達式
- 高級語言的表達式由基本符號、基本元素和各種數據通過運算符連接而成，運算符大致包括以下幾類：
  - 邏輯運算：與、或、非、异或等
  - 算術運算：加、減、乘、除、取模等
  - 數據比較：大于、小于、等于、不等于等
  - 數據傳送：輸入、輸出、賦值等
- 通過各種運算符連接而得到的表達式有以下幾種類型：
  - 算術表達式：表達式的運算結果是數值，非常近似于日常的數學計算公式
  - 關係運算表達式：表達式的運算結果是邏輯值
  - 字符串表達式：表達式的運算結果是字符串

## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 高級語言的共同特性
- 語句
  - 語句是構成高級語言源程序的基本單位，是由基本元素、運算符、表達式等組成。任何一種高級語言往往都支持賦值、條件判斷、循環、輸入輸出等語句。程序員利用這些語句的結合，能够很方便地編制出功能強大的程序
- 庫函數和用戶自定義的函數
  - 爲了支持用戶編寫出功能強大的源程序，幾乎所有的高級語言都爲用戶提供了豐富的庫函數，這些庫函數能够實現某些特定的功能，比如計算一個比較複雜的數學函數
  - 在源程序中，用戶也可以自己定義自己的函數（子程序或過程），以便以後可以反復調用這些代碼集合





## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 高級語言的共同特性
- 注釋
  - 任何一種程序設計語言都強調注釋的重要性。源程序所包含的代碼往往比較冗長，添加必要的注釋不僅有助於閱讀程序，更重要的是，在需要對程序功能進行擴充時，注釋可以極大地幫助程序員對原始程序的理解
  - 經常會出現這樣一種情況，程序員自己編寫的程序，經過一段時間後，可能就是半年或者幾個月以後，程序員自己也讀不懂自己的程序了。況且，程序不僅要自己看得懂，更重要的是也要讓別人能夠看懂



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 變量和基本數據類型

- 1 常量

- 常量也稱常數，是一種恒定的、不隨時間改變的數值或數據項

- 2 變量

- 變量是指在程序的運行過程中可以發生改變的量，是程序中數據的臨時存放場所。

- 3 數據類型

- 用來約束數據的解釋

- 4 表達式

- 表達式是操作符、操作數和標點符號組成的序列，其目的是用來說明一個計算過程



## 4.3 程序設計基礎知識

---

### 4.3.2 高級語言程序設計的基本內容

- 數據結構

- 數據結構是指數據元素之間的相互關係的集合，包括了數據的邏輯結構、物理結構以及數據的運算



## 4.3 程序設計基礎知識

---

### 4.3.3 高級語言程序設計的基本內容

- 現代編程環境
  - 集成開發環境
  - 集成開發環境(Integrated Development Environment, IDE)是一種輔助程序開發人員開發軟件的應用軟件，在開發工具內部就可以輔助編寫源代碼文本、并編譯打包成爲可用的程序，有些甚至可以設計圖形接口
  - 代碼搜索
  - 隨著開源軟件的快速發展，互聯網上聚集了大量的代碼，尤其是一些有影響力的代碼倉庫，例如 Github 等，其中包含了大量可以被程序員複用的資源

➤ 舉例說明一下



## 4.4 軟件工程的概念

---

### 4.4.1 軟件工程產生的背景

- 軟件危機的含義

- 隨著計算機應用的日益普及和深入，軟件在計算機系統中所占比重不斷增加。用傳統方法開發出來的許多大型軟件甚至根本無法投入使用，結果造成了大量人力、物力和財力的浪費
- 人們通常將大型軟件開發和維護過程中遇到的一系列嚴重問題稱為“軟件危機”

➤ 舉例說明一下

➤ 一個操作系統的代碼量






## 4.4 軟件工程的概念

---

### 4.4.1 軟件工程產生的背景

- 軟件危機的具體表現
    - 軟件開發進度難以預測
    - 軟件開發成本難以控制
    - 用戶對產品功能難以滿意
    - 軟件產品質量無法保證
    - 軟件產品難以維護
    - 軟件缺少適當的文檔資料
  - 軟件危機的出現表明：必須尋找新的技術和方法來指導大型軟件的開發，即用工程化方法來管理軟件開發過程。這種思想導致了軟件工程學科的產生
- 



## 4.4 軟件工程的概念

---

### 4.4.2 軟件工程的基本概念

- 根據 IEEE 給出的定義，軟件工程的基本概念如下：
  - 將系統化、嚴格約束的、可量化的方法應用于軟件的開發運行和維護，即將工程化應用于軟件軟件開發成本難以控制
  - 將工程化應用于軟件的方法的研究





## 4.4 軟件工程的概念

---

### 4.4.3 軟件生命週期

- 爲了用工程化方式有效地管理軟件產品，逐步形成了“軟件生命週期”的概念，軟件生命週期就是軟件產品從考慮其概念開始，到該軟件產品消亡爲止的整個時期
- 軟件生命週期一般包含：概念階段、需求分析階段、設計階段、實現階段、測試階段、交付使用階段，以及維護階段
- 從經濟學意義上講，軟件的維護費用通常遠高于軟件開發的費用，因而開發軟件應考慮整個軟件生命週期的全部費用。因此，軟件生命週期的概念就顯得尤爲重要







## 4.5 軟件開發模型與方法

---

軟件開發模型可定義為：“軟件開發全部過程、活動和任務的結構框架”。軟件開發模型能清晰、直觀地表達軟件開發的全過程，它明確規定了要完成的主要活動和任務，是軟件項目開發工作的基礎

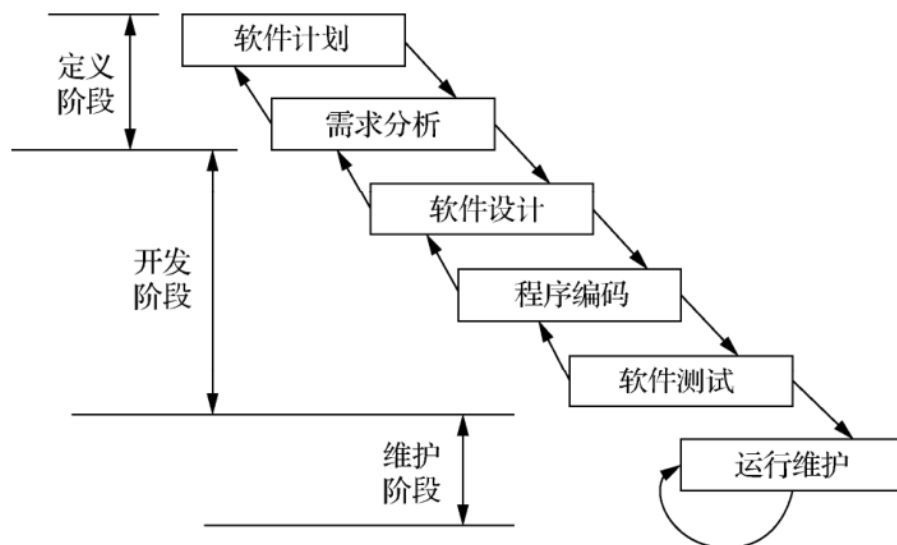
### 4.5.1 瀑布模型

- 瀑布模型，是最早出現的軟件開發模型。它將軟件開發過程中的各項活動規定為依固定順序連接的若干階段工作，最終得到軟件系統或軟件產品
- 瀑布模型的全過程歸結為 3 個階段（即定義階段、開發階段、維護階段）、6 個活動（即軟件計劃、需求分析、軟件設計、程序編碼、軟件測試和運行維護）



## 4.5 軟件開發模型與方法

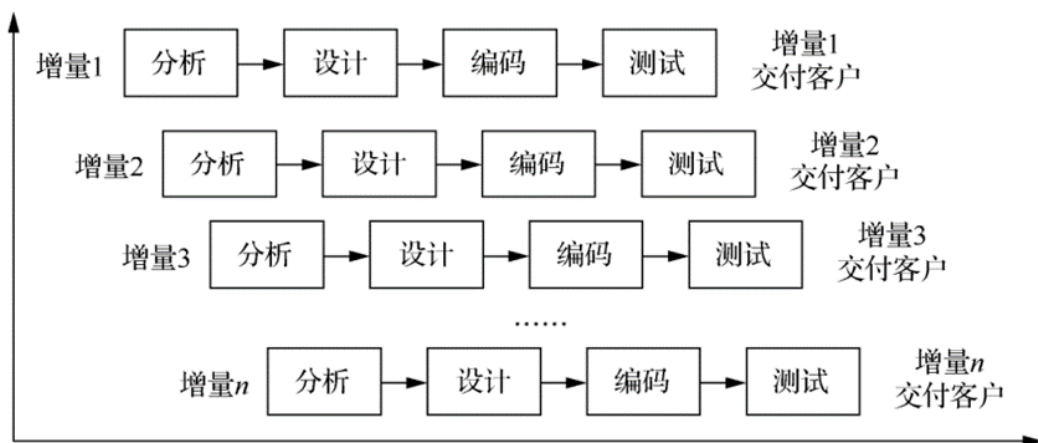
### 4.5.1 瀑布模型



- 瀑布模型中每個階段的工作都以上一個階段工作的結果為依據，同時又為下一個階段的工作提供了前提。它們自上而下、逐級下落、相互銜接，如同瀑布流水
- 瀑布模型在實踐中也逐漸暴露出了它的不足和問題，如各項活動實際并非完全是自上而下呈綫性變化的，由于是固定的順序，前期階段工作中所造成的差錯越拖到後期階段，則造成的損失和影響也越大，為了糾正差錯而花費的代價也越大

## 4.5 軟件開發模型與方法

### 4.5.2 增量模型

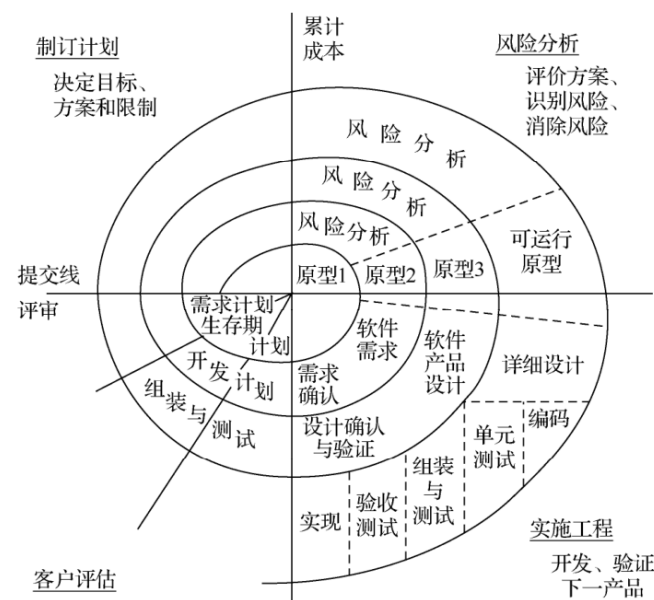


- 增量模型，是把待開發的軟件系統模塊化，將每個模塊作為一個增量組件，從而分批次地分析、設計、編碼和測試這些增量組件。運用增量模型的軟件開發過程是遞增式過程
- 采用增量模型進行開發，開發人員不需要一次性地把整個軟件產品交付給用戶，而是可以分批次進行交付
- 增量模型在面向市場的產品開發中是很有用的，並且該模型還可以有效地管理和控制軟件開發的風險

## 4.5 軟件開發模型與方法

### 4.5.3 螺旋模型

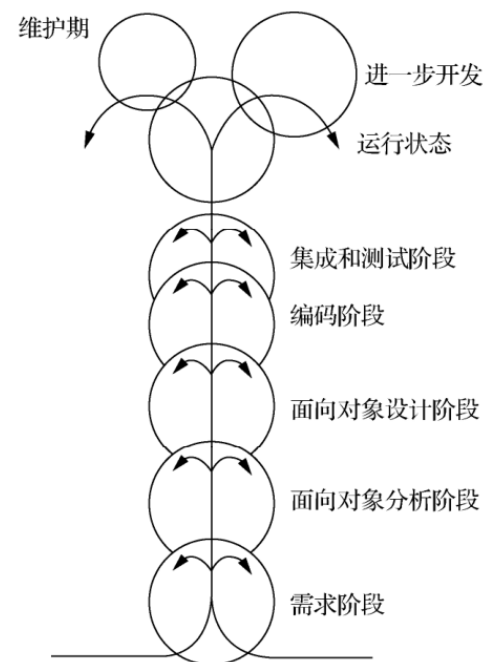
- 螺旋模型，可以看作瀑布模型和增量模型的結合。它像螺旋一樣不斷迭代，又像螺旋一樣不斷前進，即每次迭代都不是在原有水平上進行的
- 螺旋模型也要對系統目標進行分解，規定每一次螺旋的目標。在每一次螺旋周期的開發中采用簡化的瀑布模型并且加入風險分析和原型化的方法，然後用一次次螺旋上升的迭代實現最終目標
- 螺旋模型很好地將瀑布模型和增量模型的優點結合了起來，同時又能很好地管理軟件開發的過程，因此它更適合大型軟件的開發



## 4.5 軟件開發模型與方法

### 4.5.4 噴泉模型

- 噴泉模型，主要用于采用面向對象技術的軟件開發項目，是典型的面向對象的生命周期模型
- 由于噴泉模型屬面向對象軟件開發模型，因而可以較為容易地實現活動的迭代和無間隙
- 噴泉模型的各個階段之間並沒有明顯的界限，開發人員可以同步進行開發，這樣可以提高軟件項目開發效率、節省開發時間






## 4.5 軟件開發模型與方法

---

### 4.5.5 軟件開發方法：結構化方法

- 軟件生命周期的概念被提出後，軟件的開發開始有章可循。基于軟件生命周期的結構化方法的出現為成功開發大型軟件奠定了基礎
- 結構化方法主要包括：系統分析階段的結構化分析（SA）、系統設計階段的結構化設計（SD）、系統實現階段的結構化程序設計（SP）

### 4.5.6 軟件開發方法：面向對象方法

- 面向對象，就是以對象為中心、以類和繼承為構造機制來認識、理解和描述客觀世界并設計、構造出相應的軟件系統。面向對象（OO）方法，則是一種圍繞對象進行系統分析和系統設計、用面向對象的工具建立系統的方法
- 

## 4.6 統一建模語言 ( UML )

### 4.6.1 UML的基本概念

- UML，是一種通用的、用于對實際問題建立一個易于實現的、以圖形化爲主的、關於模型的標準建模語言，也是一種富有表達力的、繪製軟件藍圖的標準語言
- UML由信息系統和面向對象領域的3位著名的專家Grady Booch、James Rumbaugh和Ivar Jacobson共同提出。目前已經得到了廣泛的支持和應用，并成爲了國際標準



Grady Booch



James Rumbaugh



Ivar Jacobson



## 4.6 統一建模語言 ( UML )

---

### 4.6.1 UML的基本概念

- UML不是一種程序設計語言，而是一種可視化建模語言。它提供了一整套用于交流的詞匯及規則，從而方便不同用戶與同一軟件進行無障礙的交流，使不同的用戶對於同一事物產生相同的認識
- UML是為支持當今大部分面向對象的開發過程而設計的，它並沒有定義一種標準的開發過程，但它更適用於迭代式開發過程，並且是獨立于過程的

➤ *UML實例說明*








## 4.6 統一建模語言 ( UML )

---

### 4.6.2 UML的產生和發展

- 衆多的建模語言各有千秋，雖大多雷同但仍存在細微的差別，用戶難以區分，從而妨礙了用戶之間的溝通和交流。因此極有必要組織聯合設計小組，根據應用需求，對建模語言進行統一
  - 1995年10月，Booch和Rumbaugh聯合推出了Unified Method 0.8；
  - 1996年6月和10月分別發布了UML 0.9 和UML 0.91；
  - 1997年1月和9月陸續推出了UML 1.0和UML 1.1；
  - 1997年11月17日，對象管理組織（OMG）宣布接受UML，并認定其為標準的建模語言
  - 目前 UML 仍在不斷地發展和完善
- 



## 思考題

---

- 常見的軟件開發模型有哪些，請列舉幾個。
- 爲什麼程序設計語言需要注釋？
- 什麼是UML，請簡潔的解釋一下。



---

休息一下

**Take a break**



感謝觀賞

Thank you for listening.