Recursion

Page:
Date: #1 (迭代)

iteration 反覆運算 recursion 遞迴

Recursive Functions

Factorial 階乘 Fibonacci series

Greatest Common Divisor Combinatorial numbers

Search in Array Towers of Hanoi

Recursion

→ breaks problem into smaller identical problems

→ An alternative to iteration

Binary search 二元搜尋法 (recursive)

Facts about a recursive solution

→ calls itself (recursive function)

→ Each recursive call solves an identical (smaller problem)

→ solution at least one smaller problem, base case (is known)

→ smaller problems must be (base case)

reaching (base case) → stop!

Factorial (n)

$n^* (n-1)^* (n-2)^* \cdots ^* 1$

return n^* fact $(n-1)$

#2 Data Abstraction

Page:
Date:

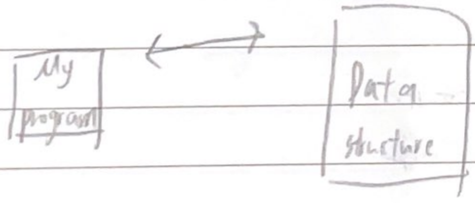
Cohesion (高內聚) - highly cohesive modules desired

Coupling (低耦合) - Loosely coupled modules desired

☆ Functional abstraction 功能性的抽象化
→ a module's $\overset{\text{描述}}{\text{specifications}}$ should
module behaves (detail)
Be independent of the module's $\overset{\text{實作}}{\text{implementation}}$

☆ Information hiding 資訊隱藏
hides certain implementation details within a module
details inaccessible from outside

Abstract Data Type (ADT)



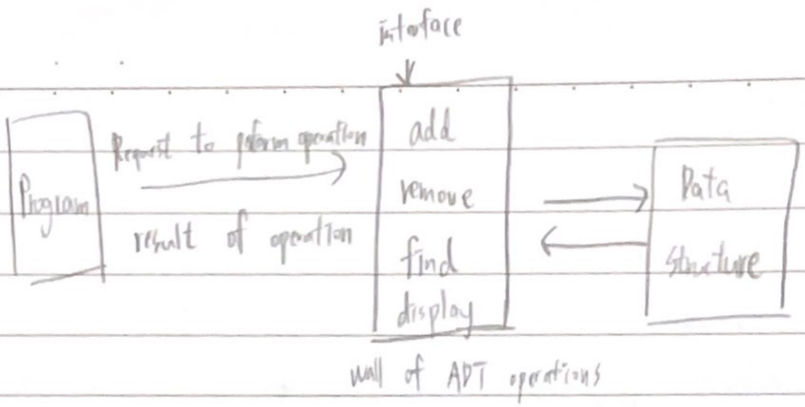
ADT is composed of
→ a collection of data
→ a set of $\boxed{\text{operations}}$ on that data

Specifications (描述) of an ADT indicate
→ what the ADT operations do, not how to implement them

Implementation of an ADT
→ particular data structure

Page:

Date:



The ADT list

items are referenced by their $\boxed{\text{position}}$ within the list

Specification ADT operations

→ define operation contract (運算合約) for the ADT list

→ $\boxed{\text{don't}}$ specify how to { store the list / perform the operations

Operation

- create an empty list 建構
- destory a list 解構
- Is empty ? (determine)
- number ?
- Insert
- Delete
- Look at (retrieve) 檢索

(Insert, Delete, Look at: given point)

Page:

Date:

C++ classes

Encapsulation combines ADT's date with its objections from an object
→ instance of a class
→ defines a new data type
→ contains (data members and methods), member functions
→ members are $\boxed{\text{private}}$ (Default), can specify them as public
→ encapsulation hides implementation details

C++ namespace

Creating a namespace 自訂命名空間
ex: namespace A
C++ standard library (std)

☆ Private = only class instances
Protected = subclass instances
Public = any class instances

#3 Linked List

Page:
Date:

Dynamic Allocation of Arrays

use (new) operator to allocate an array dynamically 動態(配置陣列)

ex: int arrsize = 50;

double *Arr = new double[arrsize];

↳ is a pointer to the its first element

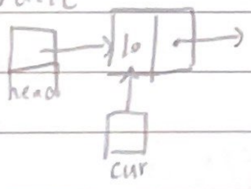
Pointer - Based Linked Lists

struct

```
Struct Node {          先 delete
    int x;             再 → NULL
    Node *next;
}
```

後面都要刪乾淨以免 memory leak

Delete



```
cur → next = NULL;
delete cur;
cur = NULL
```
} 順序不能反!

delete / insert 注意最前面各情況

Constructors and Destructors

Copy constructor create a $\boxed{\text{deep copy}}$ 深層複製

→ copies (size, head, linked list), points to copied linked list

$\boxed{\text{shallow copy}}$ 淺層複製

→ copies (size, head), points to the original linked list

Page:

Date:

Comparing	Array-Based	and Pointer-Based
Size	固定	依數量改變
Storage requirements	小	多 (存一個資料所花的空間)
Retrieval (檢索)	快 (constant 係數)	慢 (線性)
Insertion and deletion	搬移 慢 中間	走訪
		快 頭
	快 尾	

shifting of data ↑ , pointer 快一點

Saving and Restoring a Linked List by using a file

存 data , 不存 pointer (每次都不同)

use a tail pointer

Circular Linked List (環狀)

→ Last node points to the first node

Dummy Head Node (空)

→ 頭是一個空的節點 (可以不用考慮第一個 (delete / insert))

Doubly Linked Lists (雙向)

```
struct Node {                          → delete / insert 要寫兩向的
    int x ;
    Node * precede ;  ←
    Node * next ;     →
} ;
```

Page:

Date:

#4 Defining Languages

The Basics of Grammars

C++ identifier letter + zero or more letters and digits

Algebraic Expressions

☆ Infix expressions Ex = a+b 中序

☆ Prefix expressions Ex = +ab 前序

☆ Postfix expressions Ex = ab+ 後序

* 一個前序式後面再接上非空字串一定不是前序式

Backtracking (回溯)

[illegible]常和 recursion 一起用

Eight Queens Maze Number Maze

☆ Board

☆ Queen $\left\{ \right.$
8 $\left\{ \text{Board <vector>} \right.$

A Search Problem

→ use recursion $\left\{ \begin{array}{l} \text{有無路徑} \\ \text{是否已經過} \end{array} \right.$

$$\begin{tikzcd}
 & B \arrow[dr] & \\
A \arrow[ur] \arrow[dr] & & D \\
 & C \arrow[uu] &
\end{tikzcd}$$

setting | origin
destination

Page:

Date: Recursion and Mathematical Induction (數學歸納法)

→ Both use $\boxed{\text{base case}}$ to solve a problem

→ solve smaller problem

Induction can be used to

→ Prove properties about recursive algorithms
(特性 / 工作量)

Ex = recursive factorial

```
if (n==0)
    return 1;
else
    return n*fact(n-1);
```

Induction

最簡 $fact(0) = 0! = 1$

假設 $fact(n) = n! = n^*(n-1)^*\cdots{}^*1 \quad (n>0)$

歸納 $fact(n+1) = (n+1)^*n^*\cdots{}^*1 \quad (n>0)$