iteration V.S recursion
迴圈 遞迴
問題不會變小 問題變小
反覆做一樣的事 程式只要寫一份 變更精簡

{ Linear recursion
{ Binary recursion

應用 Factorial 階乘
Greatest Common Divisor 最大公因數
Search in array 搜尋
Fibonacci series 費式數列
Combinatorial numbers 組合數
Towers of Hanoi 河內塔
fractal 碎形

binary search
divide and conquer 分而擊之

Box trace 箱式追蹤

substr 將字串的某部分抓出來
substr(size-1,1) 輸出最後一個字元
base case 終止條件

Selecting a pivot item in array 只找其中一邊
樞紐
pivot
(30) 7 25 39 19 48 2 16 12
30 7 25 19 2 16 12 | 39 48
12 7 25 19 2 16 30 ~~39 48~~
12 7 2 | 16 19 25
2 7 12 16 19 25

Linear Recursion 線性遞迴
1. Test for base case
2. Recur once 只擇一遞迴
ReverseArray( ) 反轉陣列元素次序
Swap 交換

Binary Recursion
Occurs whenever there are two recursion calls for each non-base case
draw Ticks
An interval with a central tick length L consists of

Multiplying Rabbits
1. 遞迴定義
2. 問題減化
3. 終止條件
4. 保證終止

Fibonacci Sequence
1. Note that n_k at least doubles every other time.
2. That is $n \geq 2^{\frac{k}{2}}$ It is exponential 呼叫次數以指數成長
Better Fibonacci : Use linear recursion instead
呼叫次數以線性成長

Three ways to compute x^n for nonnegative integer n:

1. iterative 以迴圈求n次方
2. recursive 以遞迴求n次方

   $x^0 = 1$ $x^n = x * x^{n-1}$, if $n > 0$
3. recursive function 二元遞迴

   $x^0 = 1$

   $x^n = (x^{\frac{n}{2}})^2$, if $n > 0$ and n is even 偶數

   $x^n = x \cdot (x^{\frac{n}{2}})^2$, if $n > 0$ and n is odd 奇數

Organizing a Parade

1. 花車殿後: $F(n) = P(n-1)$
2. 樂隊殿後: $B(n) = F(n-1) = P(n-2)$

Number of acceptable parades of length n $P(n) = P(n-1) + P(n-2)$

Mr. Spock's Dilemma (Choosing k out of n Things)

$C(n,k)$ = 包含地球人 + 不包含地球人

$C(n,k) = C(n-1, k-1) + C(n-1, k)$

Base case

There is one group of everything: $C(k,k) = 1$

There is one group of nothing: $C(n,0) = 1$

Although k cannot exceed n here, we want our solution to be general

$C(n,k) = 0$ if $k > n$

Property of Binary Trees

Leaf nodes: recursive calls to base cases 葉節點

Internal nodes: recursive calls to non-base cases 內部節點

|leaf nodes| - |internal node| = 1 數量差1

The number of recursive calls to base cases must be equal to |leaf nodes| = $C(n,k)$

The number of recursive calls to non-base cases is equal to |internal nodes|

$C(n,k) - 1$

Data Abstraction 資料抽象化
class of objects (call instances)
Attributes: data members
Behaviors: methods

Principles of Object-Oriented Programming
1. Encapsulation 封裝：hides inner details 資訊隱藏
2. Inheritance 繼承：reused
3. Polymorphism 多型

Operation Contracts 運算合約
1. Purpose 目的
2. Assumpeions 假設
3. Input
4. Output

Key Issues in Programming
1. Modularity 模組化：Isolates errors, eliminates redundancies
2. Style 程式風格
3. Modifiability 可修改性
4. Ease of Use
5. Fail-safe programming
6. Debugging
7. Testing

Achieve a Better Solution
1. Cohesion 高內聚：一個程式只做一件事
2. Coupling 低耦合：不同程式間傳遞少量參數

An ADT is composed of a collection of data and set of operations on data
Specifications 描述 of an ADT indicate
Implementation 實作 of an ADT

Grocery List

predecessor 先行者

successor 後繼者

Head 沒有 predecessor

Tail 沒有 successor

ADT List Operations

1. Create List 建構：createList()
2. Destory List 解構：destroyList()
3. Determine a list is empty 是否為空：isEmpty() (boolean)
4. Determine number of item in list 計算個數：getLength() (integer)
5. Insert an item 插入：insert(index、newItem、success) (原位置資料往後)
6. Delete 刪除 ：remove(index、success) (後面資料往前移)
7. Retrieve 本會索：retrieve(index、dataItem、success)
8. Replace 置換：replace(list、integer、newItem、boolean) (先刪除後插入)
9. sortedIsEmpty() (boolean) 是否為空
10. SortedGetLength() (integer) 計算個數
11. sortedInsert (newItem、boolean) 新增
12. sortedRemove (index、boolean) 移除
13. sortedRetrieve (index、dataItem、boolean) 檢索
14. locatePosition (anItem、boolean) : integer 定位

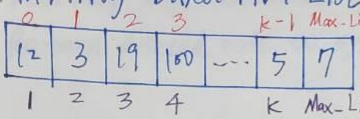
Define an appointment book 行事曆抽象化資料型態

建構→新增→取消→是否有約→約會目的

以約會簿完成：①改變指定日期&時間的約會目的

②顯示指定日期的所有約會

An Array-Based ADT List



底層實作：陣列索引
高階描述：串列位置

shifting items for insertion at position 後面位置的資料都往右移
Deletion causes a gap, fill gap by shifting 後面位置的資料都往左移
Reverse the entire list 反轉整個序列：先刪除後插入
Number of data movements (remove): 4+4+4+4=16
Number of data movements (insert): 0+1+2+3=6

ADT Polynomial
1. degree() 問最高項次方數
2. coefficient(in power) power次項的係數
3. changeCoefficient(in newCoefficient, in power) 將Power次項的係數改寫

Implemented by ADT List 利用List來實作Polynomial
isEmpty()、getLength()、insert()、remove()、retrieve()

C++ Namespaces 命名空間
※ The contents of the namespace can be accessed by code inside or outside
使用 scope resolution operator(::) 去取得命名空間外的元素 (範圍解析)
try：設定保護範圍
catch：捕捉例外狀況

Linked List
Pointer-based Implementations 鏈連結串列
Circular Linked Lists 環狀鏈連結串列
Doubly Linked Lists 双向鏈連結串列

Pointer 指標=門牌
Initially undefined, but Not NULL 還沒有房子
Static allocation 一般變數、直接配給
p=&x (房子x的門牌)
p=new int 動態配置 Daynamic allocation

Implementing ADT

Violating the wall of ADT operations

C++ classes

encapsulation 封裝

private 私密

public 公開

Each class definition is placed in a header file 描述 : Classname.h

The implementation of a class's methods are placed in an implementation file,

Classname.cpp 實作

Constructors : create and initialize new instances of a class

```
Sphere();  // Default Constructor
Sphere(double initialRadius);  // Constructor
```

Destructor : each class has one destructor 預設解構

父類別 : Sphere (base class)　子類別 : ColorSphere (derived class)

An instance of derived class can invoke public methods of the base class.

Overloading in class Rational 多載 // same function name for different functions

Private : only class instances

Protected : subclass instance (可繼承)

Public : any class instances

```
void setRational(long, long);  // 覆載 overriding
void setRational(long);  // 多載 overloading
```

delete p 歸還房子
p = NULL 徹底遺忘門牌

(a) Declaring pointer variables ex: int *p 申請空白的門牌
(b) pointing to statically allocated memory p = &x 抄寫別人的門牌
(c) assigning a value ex: *p = 6 搗佔鵲巢
(d) allocating memory dynamically ex: p = new int 緊急配置
(e) assigning a value ex: *p = 7 堆放家當
(f) copying a pointer ex: q = p 抄寫更多一張門牌
(g) allocating memory dynamically and assigning a value q = new int, *q = 8
緊急配置並堆放家當
(h) assigning NULL to a pointer variable ex: p = NULL 遺忘門牌
(i) deallocating memory ex: delete q、q = NULL 歸還房子並遺忘門牌

Dynamic Allocation of Arrays 動態配置陣列

```
int arraySize = 50
double *anArray = new double [arraySize]
int anArray[2] ≡ *(anArray+2)
```
陣列名稱三指標

Pointer Based Linked Lists

A node in a linked list is usually a struct.
The head pointer points to the first node in a linked list
If head is NULL, the linked list is empty.
A node is dynamically allocated

Displaying the Contents of a Linked List
Reference a node member with the → operator
A traverse operation visits each node in the linked list.
走訪 → 刪除 → 新增

Deleting a specified Node from a Linked List
⇒ Deleting an interior node : `prev->next = cur->next;`
`prev->next = prev->next->next;`

Deleting the first node : head = cur→next;
Return deleted node to system : cur→next = NULL;
delete cur;
cur = NULL;
Avoid : dangling reference

Inserting a Node into a Specified Position of a Linked List

To insert a node between two nodes

⇒ newPtr→next = cur;
prev→next = newPtr;
次序能互換

To insert a node at the begining of a linked list

⇒ newPtr→next = head;
head = newPtr;
次序不能換

Inserting at the end of a linked list is not a special case if cur is NULL

⇒ newPtr→next = cur;
prev→next = newPtr;
次序能交換

Visiting a Node on a Sorted Linked List

Finding the point of insertion or deletion for a sorted linked list of objects
走訪至欲增刪的節点、改変指標

A Pointer-Based Implementation of the ADT List.

Public methods	Private data members	Local variables to methods
-isEmpty	head 串列首	-cur 現在節点
-getLength	size 節点数	-prev 前一節点
-insert	Private method	
-remove	-find	
-retrieve		

ListNode* find(int index) const 定位搜尋

Constructors and Destructors
default constructor initializes size and head
A destructor is required for dynamically (動態) allocated memory

Copy constructor creates a deep copy 深層複製 (只複製門牌)
⇒ Copies size, head, and the linked list
The copy of head points to the copied Linked list

In constrase, a shallow copy 淺層複製 (複製整個 linked list)
⇒ Copyies size and head
The copy of head points to the original linked list

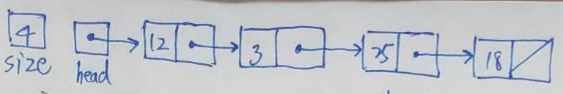
If you omit a copy constructor, the compiler generates one, but it is only sufficient for implementations that use statically allocated arrays.

A Pointer-Based Implementation of the ADT List
Size: Increasing the size of a resizable array can waste storage and time
Linked list grows and shrinks as necessary
Storage requirements: Array-based implementation reguires less memory than a pointer-based one for each item in the ADT

Retrieval 檢索: The time to access the i^{th} item
- Array-based: Constant (independent of i) 陣列: 常數時間
- Pointer-based: Depends on i 串列: 線性時間

Insertion and deletion 新增和刪除
- Array-based: Reguires shifting of data 陣列: 搬走
- Pointer-based: Reguires a traversal 串列: 走訪

Saving and Restoring a Linked List by Using a File
1. Use an external file to preserve the list between runs of a program
2. Write only data to a file, not pointers 只存資料、不留指標
3. Recreate the List from the file by placing each item at the end of the Linked list 重建: 循序加入尾端
⇒ Use a tail pointer to facilitate adding nodes to the end of the link list.
Treat the first insertion as a special case by setting the tail to head.



鏈連結串列

	item	next
0	12	1
1	3	2
2	25	3
3	18	-1
4		5
5		6
6		-1

head: 0

free(存空間大小): 4

Passing a Linked List to a Method

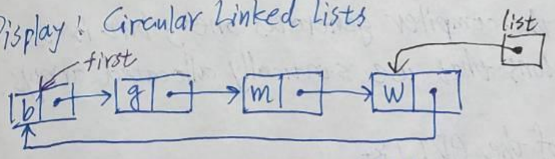
- A method with access to a linked list's head pointer has access to the entire list
- Pass the head Pointer to a method as a reference argument
⇒ Enables method to change to value of the head pointer itself (value argument would not) 傳址&：更改指標內容
- Recursive strategy to display a list
⇒ Write the first item in the list、write the rest of the list
- Recursive strategies to display a list backward
⇒ First strategy: Write the last item in the list 每次遞迴前印最後一個
Write the list minus its last item backward
⇒ Second strategy: Write the list minus its first item backward
Write the first item in the list 每次遞迴後印第一個
- Recursive view of a sorted linked list
⇒ The linked list to which head points is a sorted list if
1. head is NULL
2. head→next is NULL
3. head→item < head→next→item、and head→next points to a sorted linked list

扣除最後一個串列 (X)
扣除第一個串列 (O)

Variations : Circular linked lists
- Last node points to the first node
- Every node has a successor
- No node in circular linked lists contains NULL
- Access to last node requires a traversal
- Make external pointer point to last node instead of first node

最後一個節點的下一個指標指向第一個節點

Display : Circular Linked Lists



Variations : Dummy(空) Head Node
- Always present, even when the linked list is empty
- Insertion and deletion algorithms initialize prev to point to the dummy head node, rather to NULL (Eliminates the special case 沒有例外狀況)

1. 走訪至欲刪除的節點 2. 改變指標 3. 釋放空間

Insert : Dummy Head Node

1. 走訪至欲新增的前一個位置 2. 改變指標

Variations : Doubly (双向) Linked Lists
- Each node points to both its predecessor and its successor

⇒ precede(上一個) pointer and next(下一個) pointer

Insertions/deletions more involved than for a singly linked list

⇒ often has a dummy head node, often circular to eliminate special cases

1. 虛擬串列首的上一個指向最後節點
2. 最後節點的下一個指向虛擬串列首

Delete : Doubly Linked lists
- To delete the node to which cur points

1. 前一節點的下一個指向後一個節點
2. 後一節點的上一個指向前一節點

↕ 次序可互換

Insert: Doubly Linked lists

• To insert a new node pointed to by newPtr before the node pointed by cur

1、新節点的下一個指向目前節点
2、新節点的上一個指向前一節点
3、目前節点的上一個指向新節点
4、前一節点的下一個指向新節点

} 次序不能換

Application: Maintaining an Inventory

Local DVD store

• A list of movie titles
• Each title is associated with
  Have value 庫存量
  Want value 預訂量
  Wait list 訂戶名單

Program Input: 舊貨單、到貨單、指令
Program Output: 更新過的貨單、指令輸出

• Operations on the inventory

1、列表 2、搜尋 3、置換 4、新增

Polynomial addition 多項式加法

solution (top-down)

1、x or y is empty
2、a dummy head node
3、create a new node
4、copy p & c, then find the next
5、remaining x or y