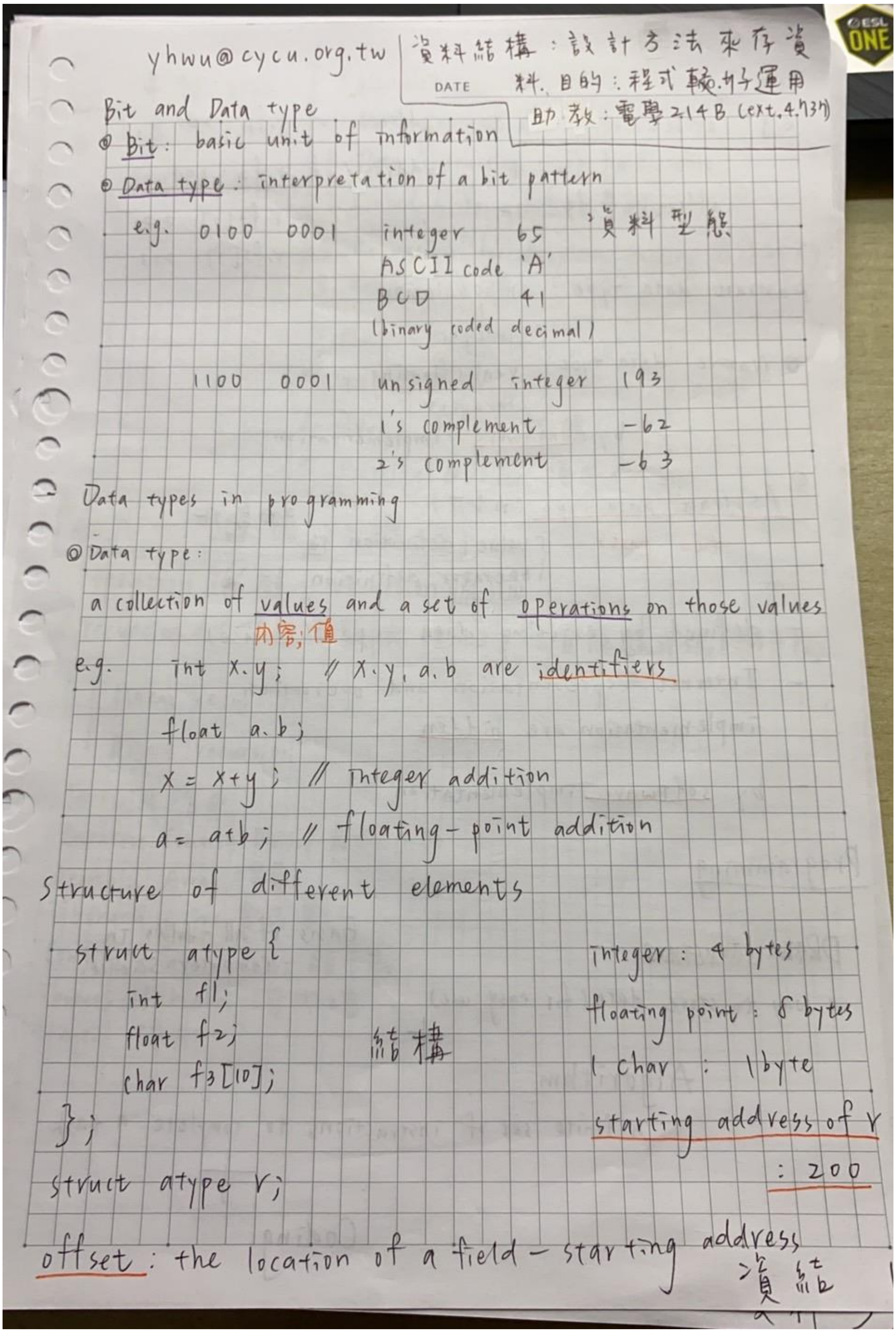


yhwu@cycu.org.tw

資料結構：設計方法來存資料。目的：程式較好運用

DATE

助教：電學214B (ext.4737)

Bit and Data type

◎ Bit : basic unit of information

◎ Data type : interpretation of a bit pattern

資料型態

e.g. 0100 0001	integer	65
	ASCII code	'A'
	BCD (binary coded decimal)	41
1100 0001	unsigned integer	193
	1's complement	-62
	2's complement	-63

Data types in programming

◎ Data type :

a collection of values and a set of operations on those values

內容;值

```
e.g.   int x,y;   // x,y, a,b are identifiers
       float a,b;
       x = x+y;   // Integer addition
       a = a+b;   // floating-point addition
```

Structure of different elements

```
struct atype {
    int f1;
    float f2;
    char f3[10];
};
struct atype r;
```

結構

Integer : 4 bytes

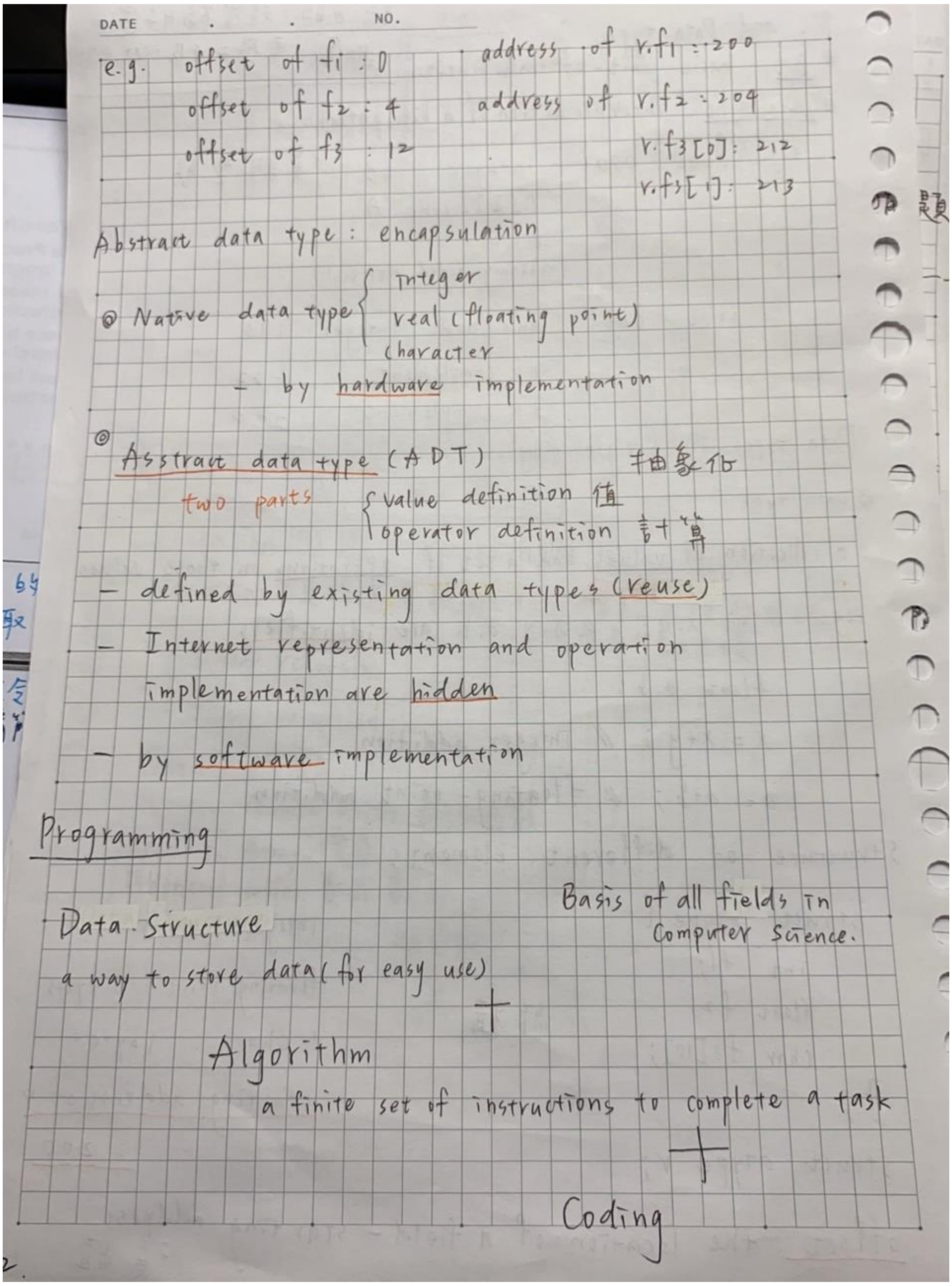
floating point : 8 bytes

1 char : 1 byte

starting address of r : 200

offset : the location of a field - starting address

資結



DATE . . NO.

e.g. offset of f_1 : 0 address of $r.f_1$: 200

offset of f_2 : 4 address of $r.f_2$: 204

offset of f_3 : 12 $r.f_3[0]$: 212

$r.f_3[1]$: 213

Abstract data type : encapsulation

◎ Native data type { Integer
real (floating point)
character

- by hardware implementation

◎ Asstract data type (ADT) 抽象化

two parts { value definition 值
operator definition 計算

- defined by existing data types (reuse)
- Internet representation and operation implementation are hidden
- by software implementation

Programming

Data Structure Basis of all fields in Computer Science.

a way to store data (for easy use)

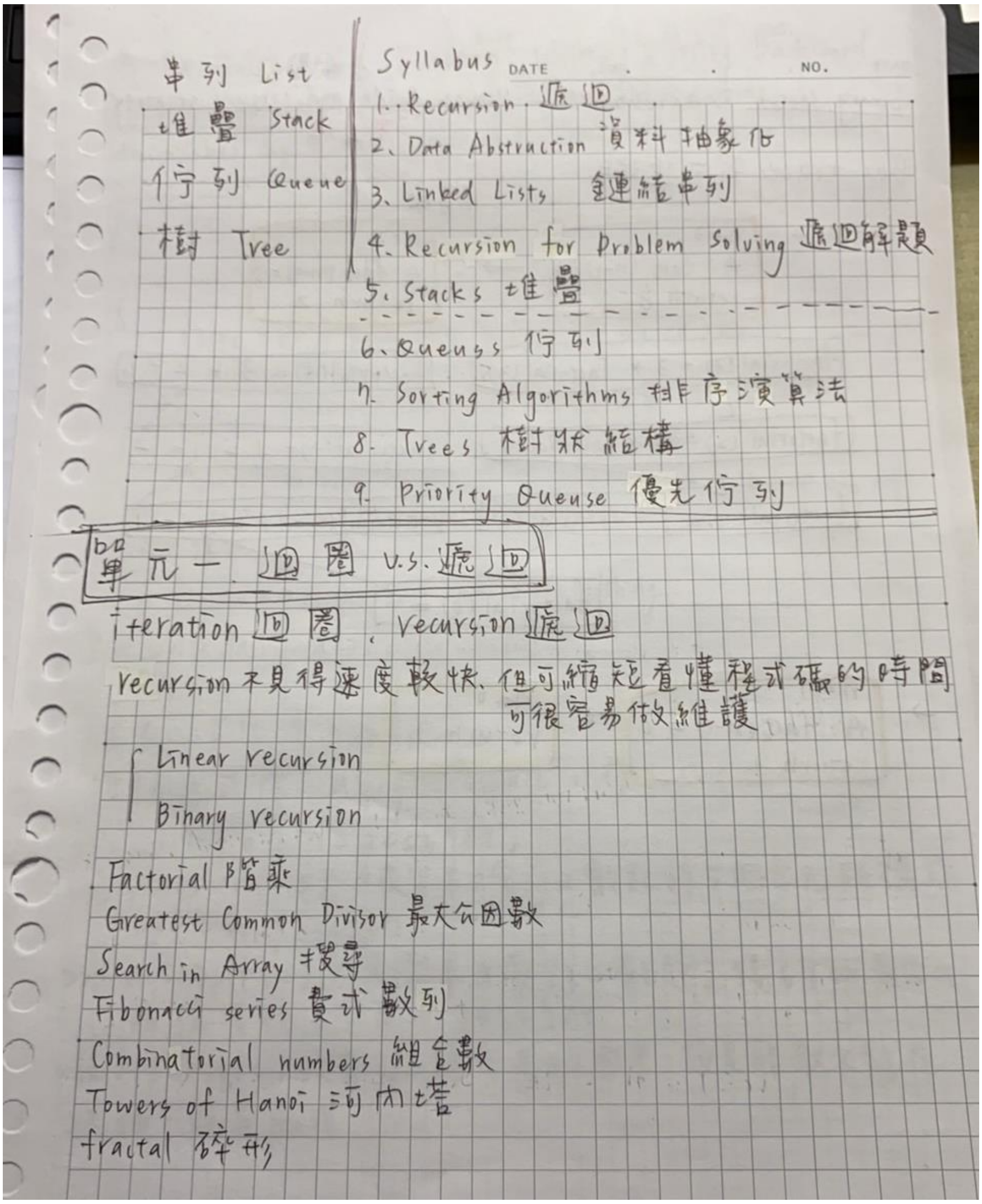
+

Algorithm

a finite set of instructions to complete a task

+

Coding



串列 List

堆疊 Stack

佇列 Queue

樹 Tree

Syllabus DATE . . NO.

1. Recursion 遞迴
2. Data Abstruction 資料抽象化
3. Linked Lists 鏈結串列
4. Recursion for Problem Solving 遞迴解題
5. Stacks 堆疊
6. Queuss 佇列
7. Sorting Algorithms 排序演算法
8. Trees 樹狀結構
9. Priority Queuse 優先佇列

單元一 迴圈 v.s. 遞迴

iteration 迴圈、recursion 遞迴

recursion 不見得速度較快，但可縮短看懂程式碼的時間
可很容易做維護

- Linear recursion
- Binary recursion

Factorial 階乘

Greatest Common Divisor 最大公因數

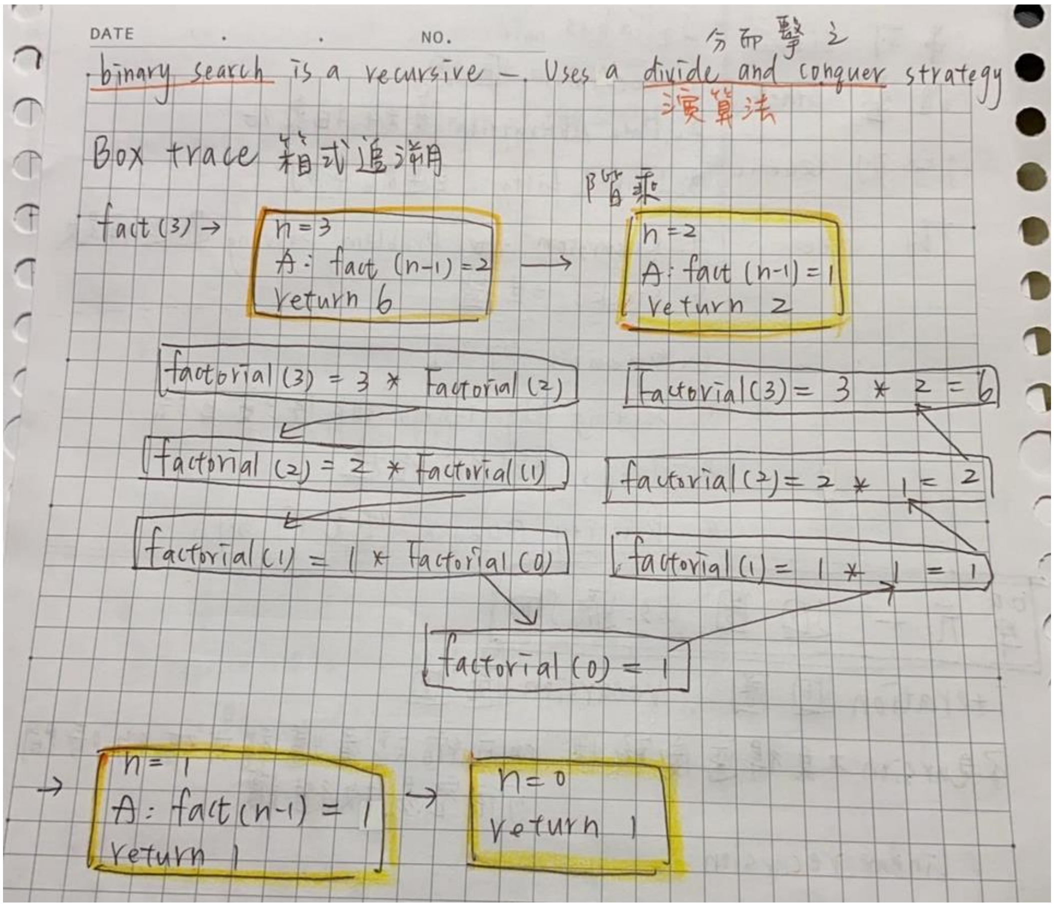
Search in Array 搜尋

Fibonacci series 費式數列

Combinatorial numbers 組合數

Towers of Hanoi 河內塔

fractal 碎形



DATE . . NO.

binary search is a recursive — Uses a divide and conquer strategy
分而擊之
演算法

Box trace 箱式追溯

階乘

fact(3) →

n = 3
A: fact (n-1) = 2
return 6

→

n = 2
A: fact (n-1) = 1
return 2

→

n = 1
A: fact (n-1) = 1
return 1

→

n = 0
return 1

factorial (3) = 3 * Factorial (2)

factorial (2) = 2 * Factorial (1)

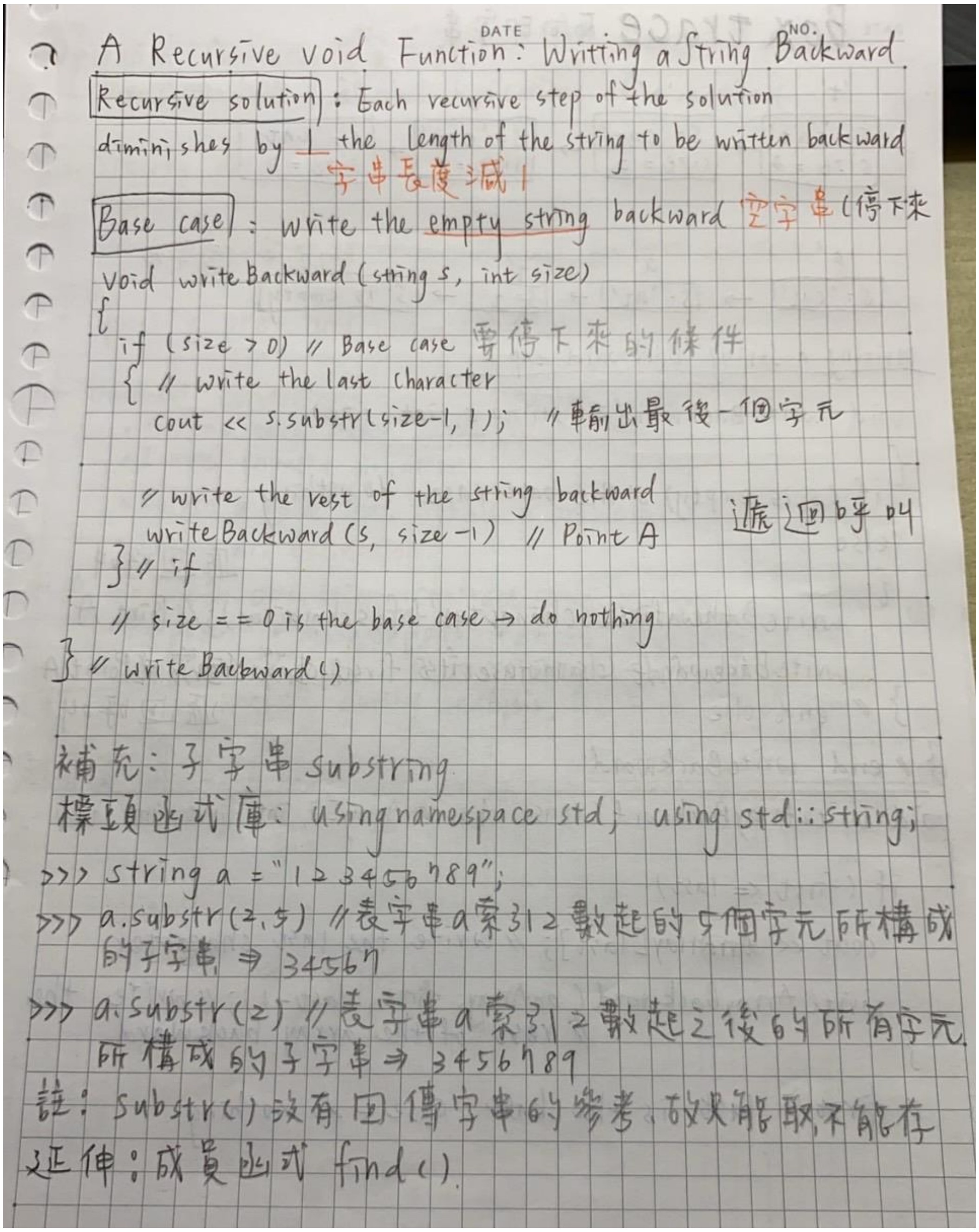
factorial (1) = 1 * Factorial (0)

factorial (0) = 1

factorial (1) = 1 * 1 = 1

factorial (2) = 2 * 1 = 2

factorial (3) = 3 * 2 = 6



A Recursive void Function: Writting a String Backward

Recursive solution: Each recursive step of the solution diminishes by 1 the length of the string to be written backward
字串長度減1

Base case: write the empty string backward 空字串(停下來

```
void writeBackward (string s, int size)
{
  if (size > 0) // Base case 要停下來的條件
  { // write the last character
    cout << s.substr(size-1, 1);   // 輸出最後一個字元

    // write the rest of the string backward
    writeBackward (s, size -1)  // Point A          遞迴呼叫
  } // if
  // size == 0 is the base case → do nothing
} // writeBackward()
```

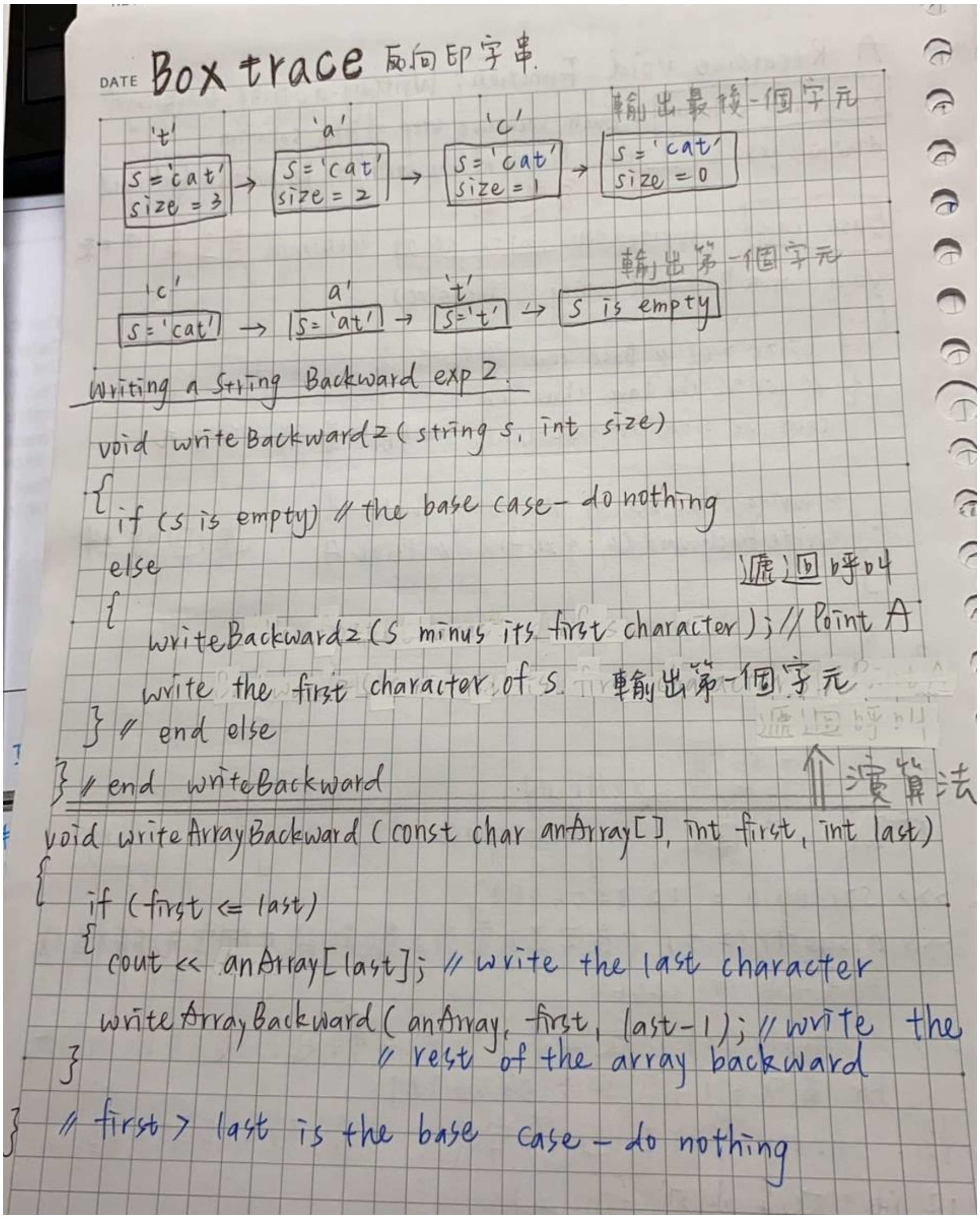
補充：子字串 substring

標頭函式庫：using namespace std; using std::string;

```
>>> string a = "123456789";
>>> a.substr(2,5) // 表字串a索引2數起的5個字元所構成
    的子字串 ⇒ 34567
>>> a.substr(2) // 表字串a索引2數起之後的所有字元
    所構成的子字串 ⇒ 3456789
```

註：substr()沒有回傳字串的參考，故只能取，不能存

延伸：成員函式 find()



DATE

Box trace 反向印字串.

輸出最後一個字元

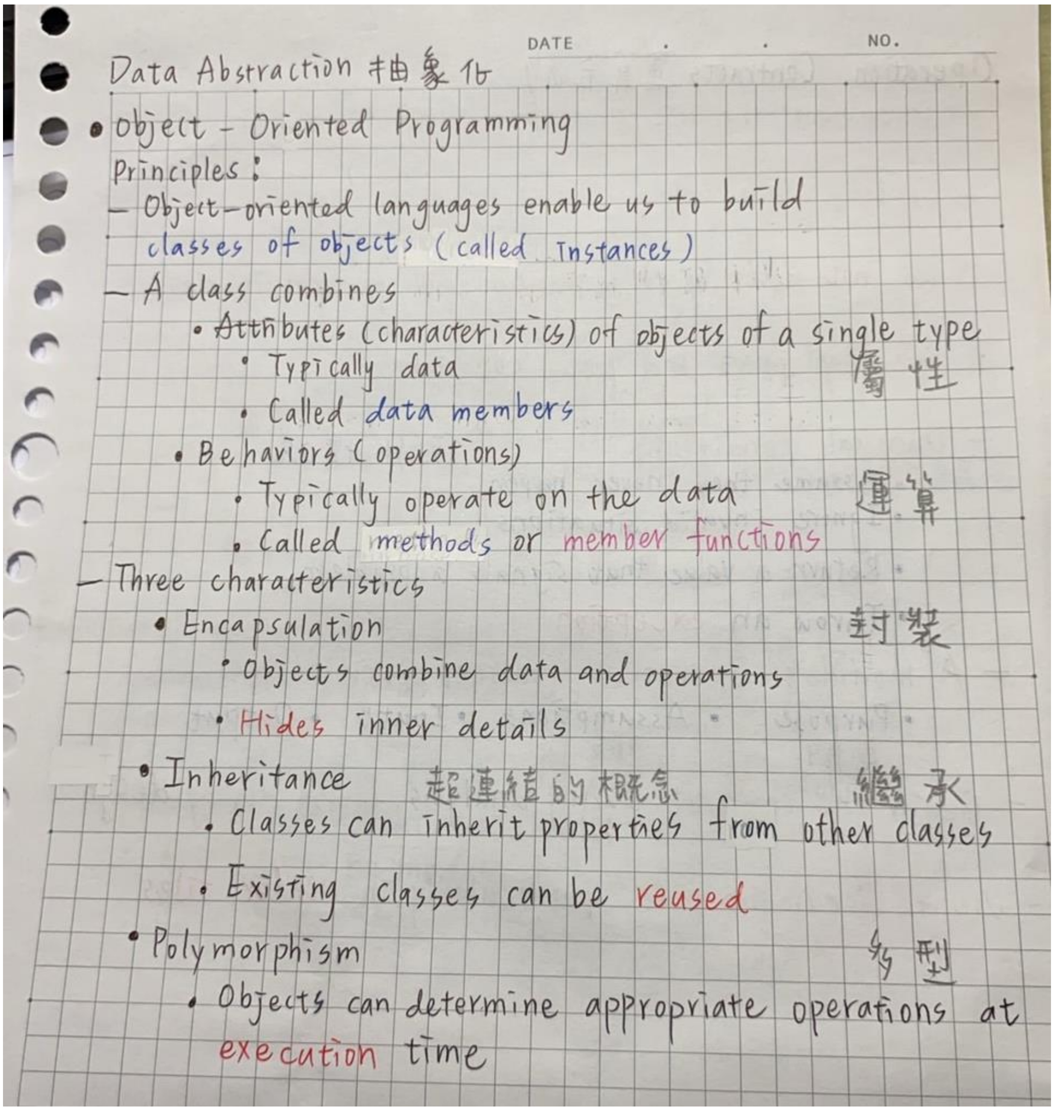
't' | s = 'cat' size = 3 → 'a' | s = 'cat' size = 2 → 'c' | s = 'cat' size = 1 → s = 'cat' size = 0

輸出第一個字元

'c' | s = 'cat' → a' | s = 'at' → 't' | s = 't' → s is empty

Writing a String Backward exp 2.

```
void writeBackward2(string s, int size)
{
  if (s is empty) // the base case - do nothing
  else
  {                                                   遞迴呼叫
    writeBackward2(S minus its first character); // Point A
    write the first character of s.   輸出第一個字元
  } // end else                                       遞迴呼叫
} // end writeBackward                                ⇑ 演算法
void writeArrayBackward(const char anArray[], int first, int last)
{
   if (first <= last)
   {
     cout << anArray[last]; // write the last character
     writeArrayBackward(anArray, first, last-1); // write the
                               // rest of the array backward
   }
} // first > last is the base case - do nothing
```



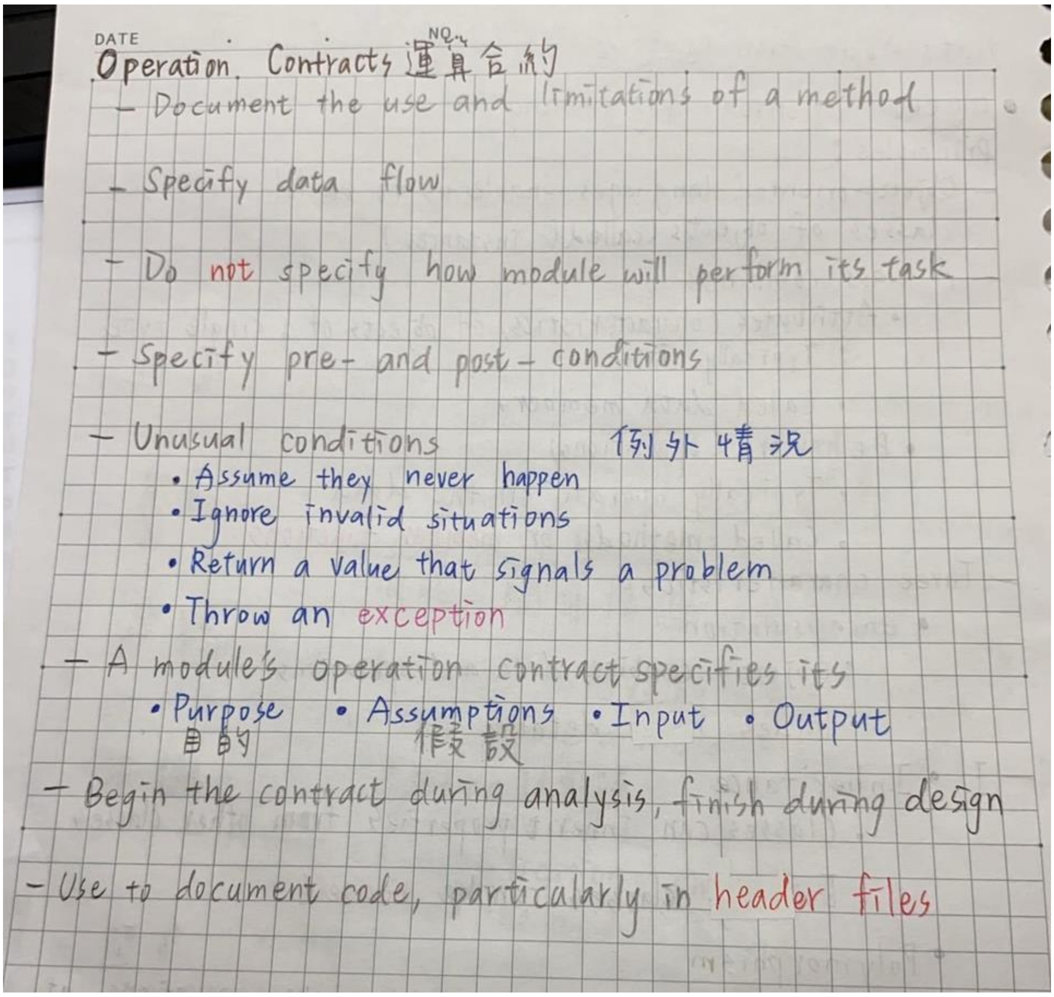
DATE . . NO.

Data Abstraction 抽象化

- Object - Oriented Programming

Principles:

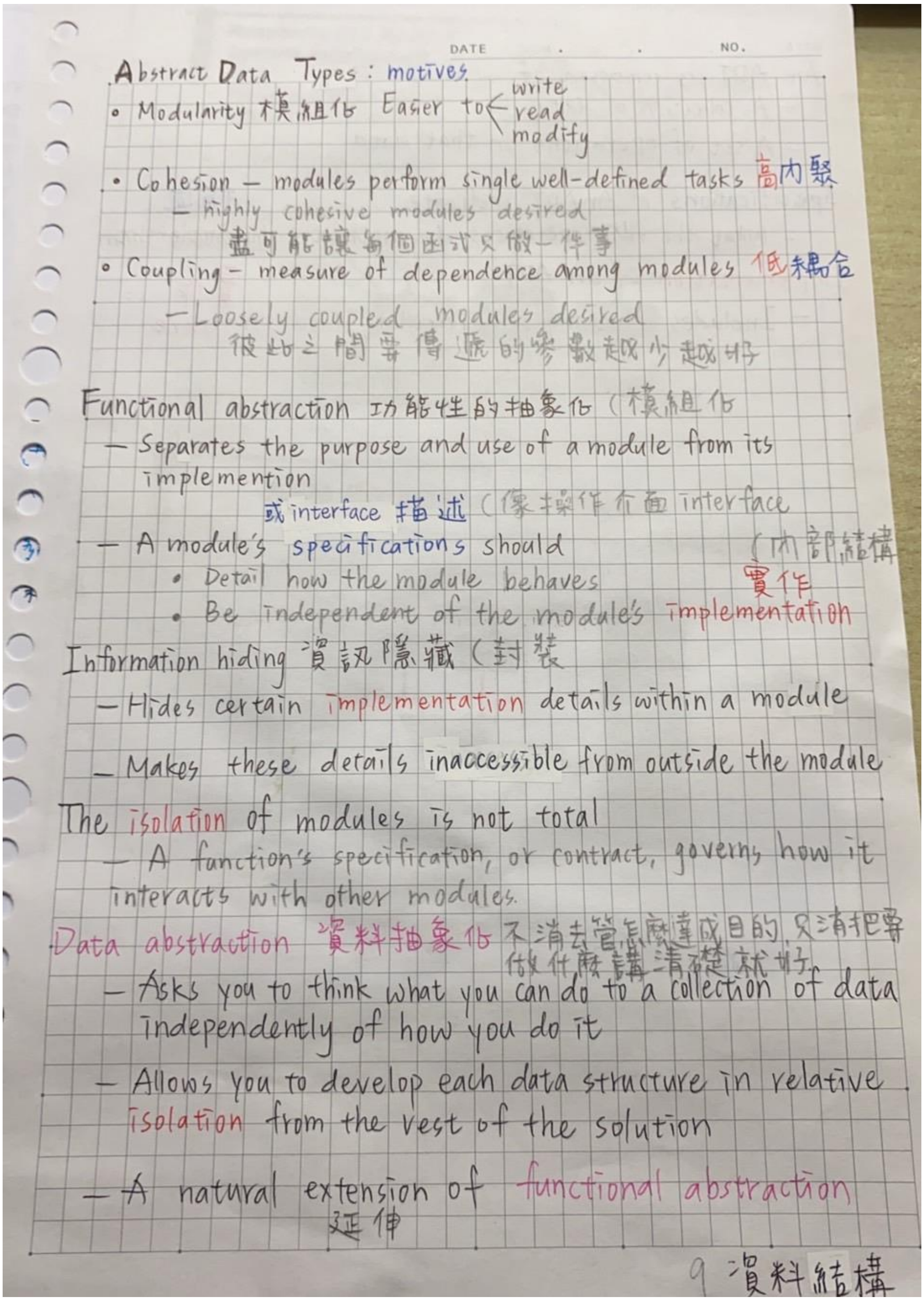
- Object-oriented languages enable us to build classes of objects (called Instances)
- A class combines
  - Attributes (characteristics) of objects of a single type 屬性
    - Typically data
    - Called data members
  - Behaviors (operations) 運算
    - Typically operate on the data
    - Called methods or member functions
- Three characteristics
  - Encapsulation 封裝
    - Objects combine data and operations
    - Hides inner details
  - Inheritance 超連結的概念 繼承
    - Classes can inherit properties from other classes
    - Existing classes can be reused
  - Polymorphism 多型
    - Objects can determine appropriate operations at execution time



DATE NO.

Operation Contracts 運算合約

- Document the use and limitations of a method
- Specify data flow
- Do not specify how module will perform its task
- Specify pre- and post- conditions
- Unusual conditions 例外情況
    - Assume they never happen
    - Ignore invalid situations
    - Return a value that signals a problem
    - Throw an exception
- A module's operation contract specifies its
    - Purpose 目的
    - Assumptions 假設
    - Input
    - Output
- Begin the contract during analysis, finish during design
- Use to document code, particalarly in header files



DATE . . NO.

Abstract Data Types : motives

- Modularity 模組化 Easier to ← write / read / modify

- Cohesion — modules perform single well-defined tasks 高內聚
  - highly cohesive modules desired
    盡可能讓每個函式只做一件事
- Coupling - measure of dependence among modules 低耦合
  - Loosely coupled modules desired
    彼此之間要傳遞的參數越少越好

Functional abstraction 功能性的抽象化（模組化

- Separates the purpose and use of a module from its implemention
  或interface 描述（像操作介面 interface
- A module's specifications should （內部結構
  - Detail how the module behaves 實作
  - Be independent of the module's implementation

Information hiding 資訊隱藏（封裝

- Hides certain implementation details within a module
- Makes these details inaccessible from outside the module

The isolation of modules is not total

- A function's specification, or contract, governs how it interacts with other modules.

Data abstraction 資料抽象化 不消去管怎麼達成目的，只消把要做什麼講清礎就好.

- Asks you to think what you can do to a collection of data independently of how you do it
- Allows you to develop each data structure in relative isolation from the vest of the solution
- A natural extension of functional abstraction
  延伸