ADT.

① classes of object
- Attribute : data members.
- Behavior : methods

② Principle of Object-Oriented Programming.
- Encapsulation (hide inner detail).
  - object combine data and operation.
- Inheritance (reused).
  - class can inherit properties from other classes.
- Polymorphism
  - object can determine appropriate operations at execution time.

③ Operation contract.
- Purpose. (what actions take place?).
- Assumptions (what does the module assume?)
- Input (what data is available to a module?)
- Output (what effect does the module have on the data?)

△ Begin the contract during analysis, finish during design.
△ Use to document code, particularly in header files.

1-6. (Binary search)

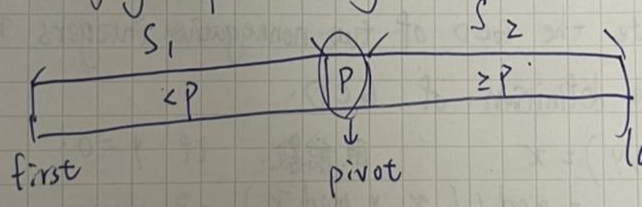
② Finding the largese item in an Array.

maxArray (left half of an Array) && maxArray (Right half of an Array) (= 分法)

1-7

② Finding the K-th Smallest item in an Array.

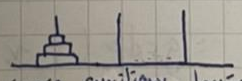
- select a pivot item (樞紐) in the array.
- Arranging => partitioning



- Resursively applying strategy:

1-8

- Tower of Hanoi




```
Algorithm towers (numDisks, source, dest, auxiliary)
if (numDisks == 1)
  move from "source" to "dest"
else
  towers (numDisks - 1, source, auxiliary, dest)

  towers (numDisks - 1, auxiliary, dest, source)
```

Ch3 - LINKED LIST.

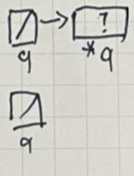
A pointer contains the locations, or address in memory

- Declaration of an inte pointer variable p.

(int *p; // 尚未添增新儲存空間.

p = &x; // 指向x位置.

delete p;
p = NULL; } same // 徹底消失.



- Modularity
  - Isolates errors.
  - Eliminats redundancies.

=> Achieve a better solution:

① Cohesion - module perform single well-defined tasks.
(高內聚)

② Coupling - measure of dependence among modules.
(低耦合).

Key issue in Programming.

1. Modularity (模組化).
2. style.
3. Modifiability
4. Easy of Use.
5. Fail-safe programming.
6. Debugging.
7. Testing.

1-4.

② Give two N=)a, b, where a>b, write a recursive function ~~the~~ to compute the sum of all the integer from a to b.

```
int sum(int a, int b)
{ if (a>b)
    return sum(a, b+1)+b;
  else if(a==b)
    return b;
}
```

1-5

② Compute the GCD of two nonnegative integers x and y.

recursive definition of GCD.

gcd1(x,y) = x 取餘數. if y=0.

= gcd1(x, y mod x) if y>x.

= gcd1(y, x mod y) otherwise.

gcd2(x,y) = y if x mod y=0.

= gcd2(y, x mod y) otherwise.

```
int gcd1(int x, int y).
if(y==0) return x
else if(y>x) return gcd1(x,y%x); else return gcd1(y,x%y);
```

```
int gcd2(int x, int y).
if !(x%y) return y;
else return gcd2(y, x%y);
```