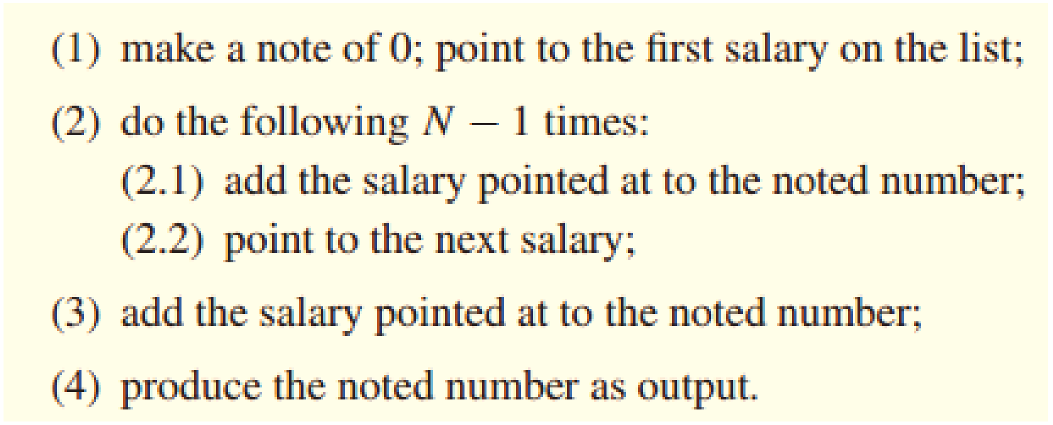
作业1-4

DH第2章练习1、2、3、4、5、6、7、8

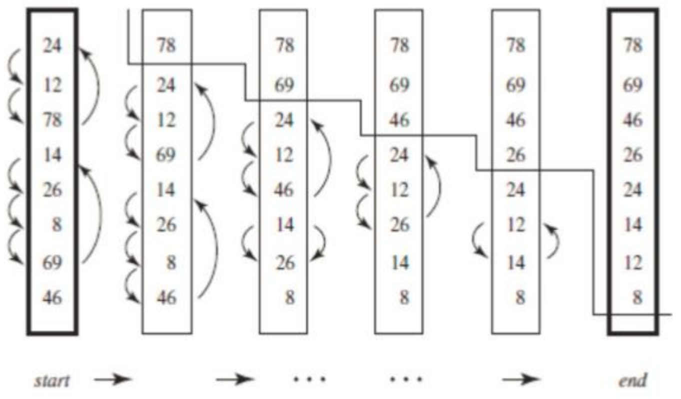
2.1. The algorithm for summing the salaries of *N* employees presented in the text performs a loop that consists of adding one salary to the total and advancing a pointer on the employee list *N* − 1 times. The last salary is added separately. What is the reason for this? Why don't we perform the loop *N* times?



```
(1) make a note of 0; point to the first salary on the list;
(2) do the following $N - 1$ times:
    (2.1) add the salary pointed at to the noted number;
    (2.2) point to the next salary;
(3) add the salary pointed at to the noted number;
(4) produce the noted number as output.
```

2.2. Consider the bubblesort algorithm presented in the text.
(a) Explain why the outer loop is performed only *N* – 1 times.
(b) Improve the algorithm so that on every repeated execution of the outer loop, the inner loop checks one element less.

- 内层循环的下标变化范围是什么？



(a)每次确定一个数位置，当N-1个数的位置确定，第N个必然也确定了。

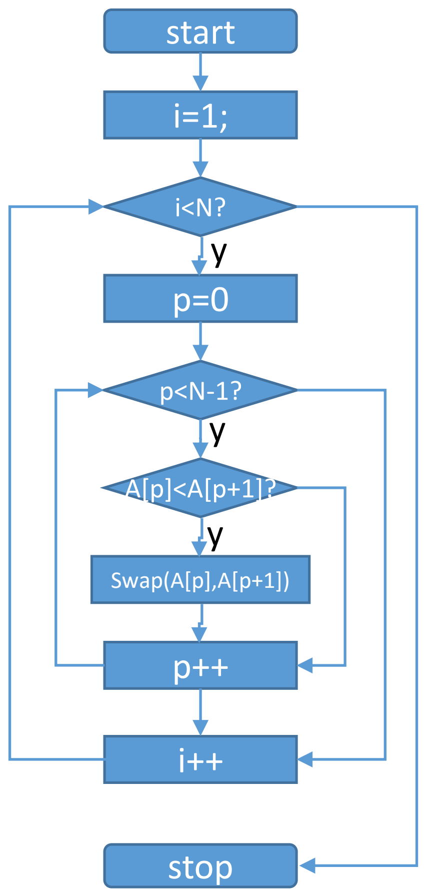
(b)

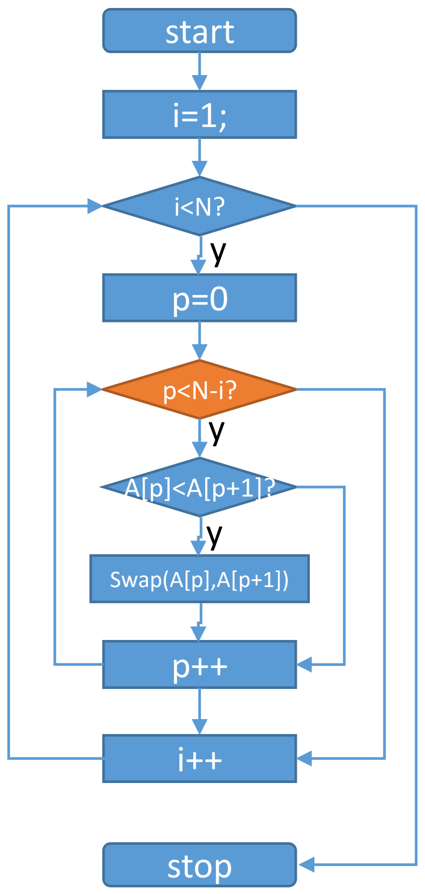
i=0;

i=i+1;

```
(1) do the following $N - 1$ times:
    (1.1) point to the first element;
    (1.2) do the following N-i times:
        (1.2.1) compare the element pointed to with the next element;
        (1.2.2) if the compared elements are in the wrong order, exchange them;
        (1.2.3) point to the next element.
```

2.3. Prepare flowcharts for the bubblesort algorithm presented in the text and for the improved version you were asked to design in Exercise 2.2.





2.4. Write algorithms that, given an integer *N* and a list *L* of *N* integers, produce in *S* and *P* the sum of the even numbers appearing in *L* and the product of the odd ones, respectively.
(a) Using bounded iteration.
(b) Using “goto” statements.

- 初始化：
    - S=0
    - P=1

2.5 Show how to perform the following simulations of some control constructs by others. The sequencing construct “and-then” is implicitly available for all the simulations. You may introduce and use new variables and labels if necessary.

(a) Simulate a “for-do” loop by a “while-do” loop.

```
for (A;B;C) do D=> A; while(B) do {D;C;}
```

(b) Simulate the “if-then” and “if-then-else” statements by “while-do” loops.

```
if A then B => while A do {B; break;}
if A then B else C=> while A do {B; break;} while !A do {C; break;}
```

(c) Simulate a “while-do” loop by “if-then” and “goto” statements.

```
                   F: if A then   begin
while A do B =>            B;
                           goto  F;
                        end
```

(d) Simulate a “while-do” loop by a “repeat-until” loop and “if-then” statements.

```
while A do B =>   if A then   repeat B until !A
```

2.8 Show how to simulate a “while-do” loop by conditional statements and a recursive procedure.

```
F(){
    If A then{
            B;
            F();
    }
}
```