COMP108 Data Structures and Algorithms

Pseudo code (Part I)

Professor Prudence Wong

pwong@liverpool.ac.uk

2024-25

Outline

Describing algorithms using pseudo code

- Algorithm vs Program
- Pseudo code
 - Trace pseudo code
 - Develop simple pseudo code

Learning outcome:

▶ Able to use pseudo code to describe algorithm

What is an algorithm

A sequence of precise and concise instructions that guide you (or a computer) to solve a specific problem in a finite amount of time



- Daily life examples: cooking recipe, furniture assembly manual
 - What are input / output in each case?

Algorithm vs Program

An algorithm is a sequence of precise and concise instructions that guide a person/computer to solve a specific problem

Algorithms are free from grammatical rules

- Content is more important than form
- Acceptable as long as it tells people how to perform a task

Programs must follow some syntax rules

- Form is important
- Even if the idea is correct, it is still not acceptable if there is syntax error

Compute the n-th power

Input: a number x & a non-negative integer n Output: the n-th power of x Algorithm:

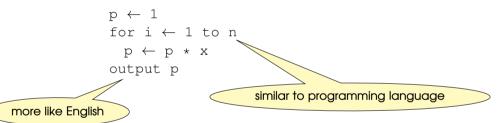
- 1. Set a temporary variable p to 1
- **2.** Repeat the multiplication $p \leftarrow p * x$ for n times
- 3. Output the result p.

Pseudo code

```
p = 1:
                            for (i=1; i \le n; i++)
Pseudo code:
                                                               Pascal:
                             p = p * x;
 p \leftarrow 1
                                                                 p := 1;
                            printf("%d\n", p);
 for i \leftarrow 1 to n
                                                                 for i := 1 to n
  p \leftarrow p * x
                                                                  p := p * x;
 output p
                                                                 writeln(p);
Python:
 p = 1
                                                               Java
                           C++:
 for i in range(n):
                                                                 p = 1;
                            p = 1;
  p = p * x
                                                                 for (i=1; i<=n; i++)
                            for (i=1; i<=n; i++)
 print p
                                                                  p = p * x;
                             p = p * x;
                                                                 System.out.println(p);
                            cout < < p < < endl;
```

Pseudo Code

One way to describe algorithm is by pseudo code



Pseudo code uses combination of both

Control flow

Expectations (refer to COMP101 / COMP122)

if-then-else

```
if <condition> then
    <statement>
else
    <statement>
```

► for-loop

```
for <variable> \leftarrow <value1> to <value2> do <statement>
```

while-loop

```
while <condition> do <statement>
```

block of statements

```
begin
<statement1>
<statement2>
:
end
```

```
OR

{
    <statement1>
    <statement2>
    :
}
```

Loops

Sum of first n +ve integers

$$\begin{array}{|c|c|} sum \leftarrow 0 \\ i \leftarrow 1 \\ while i \leq n \ do \\ begin \\ sum \leftarrow sum + i \\ i \leftarrow i + 1 \\ end \\ output sum \\ \end{array}$$

How to find product?

(i) 0 change to 1 (ii) + i change to * i

Trace table - how variables change

suppose n = 4

	-		
iteration	i before	sum	i after
before loop	1	0	1
1	1	1	2
2	2	3	3
3	3	6	4
4	4	10	5

suppose n = 6

iteration	i before	sum	i after
before loop	1	0	1
1	1	1	2
2	2	3	3
3	3	6	4
4	4	10	5
5	5	15	6
6	6	21	7

Example — What is being computed? common factors

Trace table

Suppose 0 < x < y & both are +ve integers

$$i \leftarrow 1$$
while $i \le x$ do
begin
if $x\%i == 0$ AND $y\%i == 0$ then
output i
 $i \leftarrow i + 1$
end

Operator % finds remainder: a%b gives remainder of a divided by b suppose x = 4, y = 12iteration i before output

i after 4 5

suppose x = 6 y = 15

iteration	i before	output	i after
	1	-	1
1	1	1	2
2	2	-	3
3	3	3	4
4	4	-	5
5	5	-	6
6	6	-	7

What happen if we change from "i \leftarrow i+1" to "else i \leftarrow i+1"?

Example 2 — What is being computed? HCF/GCD

Suppose 0 < x < y & both are +ve integers

```
i \leftarrow x
found ← false
while i > 1 AND found \neq true do
begin
  if x\%i == 0 AND y\%i == 0 then
     found ← true
  else
     i \leftarrow i - 1
end
output i
```

- found is a flag variable
- What value is output?
 - Questions:
 - What value of found makes the loop stop?
 - When does found change to such value?
 - What happens if we change the relational operator of the if-then-else from AND to OR?

Example 3

Consider the following algorithm.

// Assume x < y are two integers
$$r \leftarrow y$$
 $q \leftarrow 0$ while $r \ge x$ do begin $r \leftarrow r - x$ $q \leftarrow q + 1$ end output r and q

What is computed?

remainder & quotient

Trace table

Suppose x=4, y=14

q
0
) 1
2
3

Suppose x=5, y=14

(@ end of) iteration	r	q
	14	0
1	9	1
2	4	2

Suppose x=7, y=14

(@ end of) iteration	r	q
	14	0
1	7	1
2	0	2

Summary

Summary: Understanding and tracing pseudo code

Next: Developing pseudo code

For note taking