Data Structures and Algorithms

Pseudocode

Dr. Lina Xu
lina.xu@ucd.ie

School of Computer Science,
University College Dublin

September 3, 2018

Learning outcomes

After this lecture and the related practical students should...

- understand an algorithm defined in pseudocode
- be able to trace the values of variables through an algorithm
- be able convert a pseudocode algorithm into Java

Table of Contents

Pseudocode

- Pseudocode is a simple language that is used to describe algorithms
- Pseudocode cannot be executed
- Pseudocode is not as strict as programming languages

Syntax

- The syntax of pseudocode is not precisely defined
- It combines programming concepts, mathematical notation and natural language
- It should be clear enough to be easily converted into a programming language

Typical Pseudocode

Typically pseudocode combines the following features:

- Operators: + - * / % & | != < > <= >= <>
- The assignment operator: $\leftarrow$
- If statements: `if ... then ... [else ...]`
- While loops: `while ... do ...`
- For loops: `for ... do ...`
- Repeat loops: `repeat ... until ...`
- Array indexing: `A[i]`
- Algorithm declarations: `Algorithm algName(params, ...)`
- Procedure calls: `algName(params, ...)`
- Return a value: `return ...`
- Print something: `print(...)`
- User input: `read()`

No Brackets

Pseudocode does not use brackets, this means that you have to pay close attention to the indentation of the code

- Code that belongs within a larger statement such as an algorithm or for loop is indented by one step more than the statement it belongs to
- For example a print statement designed to print the numbers 1 to 10 would be indented more than the loop that comes before it

Example

```
for every integer value i in the range [0, 10] do
  print(i)
```

- After anything that comes after the loop but is not to be repeated should be indented at the same level

Example

SumRange

```
1 Algorithm SumRange(lower, upper):
2   Input: Two integers specifying the range
3   Output: The sum of the integers in the
     given range
4
5   sum ← 0
6   for every integer value i in the range
     [lower, upper] do
7     sum ← sum + i
8   return sum
```

Table of Contents

Algorithm Tracing

- Algorithm tracing is a technique that is used to study how variables change as an algorithm is executed
- This is an excellent way of understanding how an algorithm, works
- There are three steps to perform a trace
  1. Give each line in the algorithm a number
  2. Create a variable table where the columns are variables and the rows represent state changes
  3. Identify sample values (test cases) for parameters

Example trace

Sum Range

```
1   sum ← 0
2   for every integer value i in the range
     [lower, upper] do
3     sum ← sum + i
4   return sum
```

Test Case: lower = 5, upper = 8

Line	sum	i		Line	sum	i
1	0	-		2	11	7
2	0	5		3	18	7
3	5	5		2	18	8
2	5	6		3	26	8
3	11	6		4	26	-

Example Trace

Is prime

```
1  Algorithm IsPrime(number):
2    Input: An integer number to be checked
3    Output: true if the number is prime, false
      if not
4
5    if number < 4 then
6      return true
7    for every integer value i in the range [2,
      √number] do
8      if number % i = 0 then
9        return false
10   return true
```

```
1  if number < 4 then
2    return true
3  for every integer value i in the range [2, √number] do
4    if number % i = 0 then
5      return false
6  return true
```

Calculating for number = 11.

Line	number	i
1	11	-
3	11	2
4	11	2
3	11	3
4	11	3
6	11	-

Calculating for number = 16.

Line	number	i
1	16	-
3	16	2
4	16	2
5	16	2

Tracing with arrays

- Usually when tracing an array we use a sequence of boxes
- Each box is labelled with the index
- Example: an array of size 10

0	1	2	3	4	5	6	7	8	9

```
1 Algorithm SumArray(A, n):
2   Input: An integer array A of size n.
3   Output: The Sum of the values in A.
4
5   sum ← 0
6   for k in the range 0 to n-1 do
7     sum ← sum + A[k]
8   return sum
```

```
1  sum ← 0
2  for k in the range 0 to n-1 do
3     sum ← sum + A[k]
4  return sum
```

Line	sum	k	0	1	2	3	4
1	0	-	5	12	4	6	2
2	0	0	5	12	4	6	2
3	5	0	5	12	4	6	2
2	5	1	5	12	4	6	2
3	17	1	5	12	4	6	2
2	17	2	5	12	4	6	2
3	21	2	5	12	4	6	2
2	21	3	5	12	4	6	2
3	27	3	5	12	4	6	2
2	27	4	5	12	4	6	2
3	29	4	5	12	4	6	2

Array Maximuim Example

```
1 Algorithm MaxArray(A, n):
2   Input: An integer array A of size n.
3   Output: The maximum value in A.
4
5   currentMax ← A[0]
6   for k in the range 1 to n-1 do
7     if currentMax < A[k] then
8       currentMax ← A[k]
9   return currentMax
```

```
1  currentMax ← A[0]
2  for k in the range 1 to n-1 do
3    if currentMax < A[k] then
4      currentMax ← A[k]
5  return currentMax
```

Line	currentMax	k	0	1	2	3	4
1	5	-	5	12	4	6	2
2	5	0	5	12	4	6	2
3	5	0	5	12	4	6	2
4	12	1	5	12	4	6	2
2	12	1	5	12	4	6	2
3	12	2	5	12	4	6	2
2	12	2	5	12	4	6	2
3	12	3	5	12	4	6	2
2	12	3	5	12	4	6	2
2	12	4	5	12	4	6	2
3	12	4	5	12	4	6	2
5	12	-	5	12	4	6	2

Table of Contents

Converting Pseudocode to Java

When converting pseudocode into a programming language there are some difficulties

- Pseudocode has no variable declaration or type checking
- Some expression and statements may not be precise

Here are some basic guidelines for converting pseudocode to Java

- Write 1 method for every algorithm
- Algorithms are not associated with any data structure, they should be declared as static
- Add `System.out.println(...)` to print out variable trace
- Use test cases to make sure it is correct

Further Information and Review

If you wish to review the materials covered in this lecture or get further information, read the following sections in Data Structures and Algorithms textbook.

- 1.9 - Writing a Java Program