Notes on Introduction to Computer Science

Vinh Dang

August 10, 2024

Contents

Part I

The Algorithmic Foundations of Computer Science

Chapter 1

Algorithm Discovery and Design

1.1 Representing Algorithms

1.1.1 Pseudocode

Definition 1.1: Pseudocode

Pseudocode is a set of English-language constructs designed to resemble statements in a programming language but do not actually run on a computer.

Remark.

> Pseudocode represents a compromise between the two extremes of natural and formal languages.

In the following sections, we will introduce a set of constructs for three types of algorithmic operations: sequential, conditional and iterative.

1.1.2 Sequential Operations

The three basic sequential operations are *computation*, *input* and *output*.

The pseudocode instruction for performing a computation and saving the result is:

```
Set the value of variable to arithmetic expression
```

Or more concisely,

```
variable $\leftarrow$ arithmetic expression
```

The pseudocode instructions for input and output are

```
Get values for variable,variable, ....
Print the values of variable, variable, ....
```

Or

```
input variable,variable, ....
output variable,variable, ....
```

Practice Problems

Write pseudocode for the following:

Problem 1.1.2.1

An algorithm that gets three data value x, y and z as input and output the average of those three values.

Algorithm for problem 1.1.2.1

```
1: input $x$, $y$, $z$
2: $average \leftarrow (x + y + z)/3$
3: output average
```

Problem 1.1.2.2

An algorithm that gets the radius r of a circle as input. Its output is both the circumference and the area of a circle of radius r.

Algorithm for problem 1.1.2.2

```
1: input $r$
2: $circumference \leftarrow 2\pi r$
3: $area \leftarrow \pi r^2$
4: output $circumference$, $area$
```

Problem 1.1.2.3

An algorithm that gets the amount of electricity used in kilowatt-hours and the cost of electricity per kilowatt-hour. Its output is the total amount of the electric bill, including an 8% sale tax.

Algorithm for problem 1.1.2.3

```
1: input $amount\_kwh$ and $cost\_per\_kwh$
2: $amount\_bill \leftarrow amount\_kwh \cdot cost\_per\_kwh \cdot 1.08$
3: output $amount\_bill$.
```

Problem 1.1.2.4

An algorithm that inputs your current credit card balance, the total dollar amount of new purchases, and the total dollar amount of all payments. The algorithm computes the new balance, which includes a 12% interest charge on any unpaid balance.

Algorithm for problem 1.1.2.4

```
1: input $balance$, $purchases$, and $payments$
2: $new\_balance \leftarrow 1.12 \cdot (balance + purchases - payments)$
3: output $new\_balance$
```

Problem 1.1.2.5

An algorithm that is given the length and width, in feet, of a rectangular carpet and determines its total cost given that the material cost is $23 per square yard.

Algorithm for 1.1.2.5

```
1: input $length$ and $width$
2: $area \leftarrow (length \cdot width)/9$
3: $cost \leftarrow area \cdot 23$
4: output $cost$
```

Problem 1.1.2.6

An algorithm that is given three numbers corresponding to the number of times a race car driver has finished first, second, and third. The algorithm computes and displays how many points that driver has earned given 5 points for a first, 3 points for a second and 1 point for a third place finish.

Algorithm for problem 1.1.2.6

```
1: input $first$, $second$, and $third$
2: $points \leftarrow 5 \cdot first + 3 \cdot second + third$
3: output $points$
```

1.1.3 Conditional and Iterative Operations

Conditional statements allow an algorithm to ask a yes/no question and select the next operation to perform on the basis of the answer to that question. The most common conditional statement is the if/then/else statement. It has the following format:

```
if condition then
    first set of algorithmic operations
else
    second set of algorithmic operations
```

Or

```
if condition then
    first set of algorithmic operations
else if condition then
    second set of algorithmic operations
else
    third set of algorithmic operations
```

Iteration or *looping* is the repetition of a block of instructions.

A *pretest* loop is a loop where the continuation condition is tested at the beginning of each pass through the loop.

The pretest loop *while* has the following format:

```
while condition do
    operation
    ⋮
    operation
```

A *posttest* loop is one where the test is done at the end of the loop body. It has the following format:

```
repeat
    operation
    ⋮
    operation
until condition
```

Remark.

> In the repeat/until loop, the loop body is always executed at least once, whereas the while loop can execute 0, 1 ir more times.

Remark.

> If a problem can be solved algorithmically, it can be expressed by using only the sequential, conditional and iterative operations.

Practice Problems

Problem 1.1.3.1

Write an algorithm that gets as input three data values x, y and z and output the average of these values if the value of x is positive. If the value of x is either 0 or negative, your algorithm should print the error message "Bad Data" instead.

Algorithm for problem 1.1.3.1

```
input $x, y, z$
if $x > 0$ then
    $average \leftarrow (x + y + z)/3$
else
    output "Bad Data"
```

Problem 1.1.3.2

Write an algorithm that gets as input your current credit card balance, the total dollar amount of new purchases, and the total dollar amount of all payments. The algorithm computes the new balance, which includes an 8% interest on any unpaid balance below \$100, 12% interest on any unpaid balance between \$100 and \$500, inclusive, and 16% on any unpaid balance above \$500.

Algorithm for Problem 1.1.3.2

```
**input** $balance$, $purchases$, and $payments$
$unpaid \leftarrow balance + purchases - payments$
**if** $unpaid < 100$ **then**
    $new\_balance \leftarrow 1.08 \cdot unpaid$
**else if** $100 \leq unpaid \leq 500$ **then**
    $new\_balance \leftarrow 1.12 \cdot unpaid$
**else**
    $new\_balance \leftarrow 1.16 \cdot unpaid$
**output** $new\_balance$
```

Problem 1.1.3.3

Write an algorithm that gets as input a single nonzero data value x and outputs the three values x^2, $\sin x$, and $1/x$. This process is repeated until the input value for x is equal to 999, at which time the algorithm terminates.

Algorithm for Problem 1.1.3.3

```
**repeat**
    **input** $x$
    $value\_1 \leftarrow x^2$, $value\_2 \leftarrow \sin x$, $value\_3 \leftarrow 1/x$
    **output** $value\_1, value\_2, value\_3$
**until** $x = 999$
```