Pseudocode Conventions

Peter Occil

Pseudocode Conventions

This version of the document is dated 2023-06-13.

Peter Occil

1 Introduction

This document explains the conventions and common functions used in some of my articles that use pseudocode.

2 Symbols

In addition to the familiar `+`, `-`, `*` (multiplication), and `/` (division) operators, other symbols are defined below.

- `pi` is the constant π , the ratio of a circle's circumference to its diameter.
- `nothing` indicates the absence of a value. It corresponds to `null` in Java, C#, and JavaScript, `nil` in Ruby, and `None` in Python.
- `true` and `false` are the two Boolean values.
- `==` means "is equal to".
- `!=` means "is not equal to".
- A minus before a variable means 0 minus that variable. For example, `-a` means `(0 - a)`.
- The `<<` operator in the pseudocode is a bitwise left shift, with both sides of the operator being integers. If each side is 0 or greater, it is the same as multiplying the left-hand side by $2n$, where n is the right-hand side.
- The `>>` operator in the pseudocode is a bitwise right shift, with both sides of the operator being integers. If each side is 0 or greater, it is the same as dividing the left-hand side by $2n$, where n is the right-hand side, and discarding the fractional part of the result.
- The `|` operator in the pseudocode is a bitwise OR operator between two integers. It combines the bits of both integers so that each bit is set in the result if the corresponding bit is set on either or both sides of the operator.

3 Loops

Pseudocode may use `while` loops, which are self-explanatory.

Pseudocode may also use `for` loops, defined as follows:

- `for X in Y...Z; [[Statements]] ; end` is shorthand for `X = Y; while X < Z; [[Statements]]; X = X + 1; end`.
- `for X in Y...Z: [[Single-Statement]]` is shorthand for `X = Y; while X < Z; [[Single-Statement]]; X = X + 1; end`.

4 Lists and Files

In the pseudocode, lists are indexed starting with 0. That means the first item in the list has index 0, the second item in the list has index 1, and so on, up to the last item, whose index is the list's size minus 1.

In this context, a *list* is to be understood as a resizable array of items, not as a linked list.

A *list* can be expressed by wrapping items in brackets; for example, `[0, 1, 2]` is a three-item list.

- `NewList()` or `[]` creates a new empty list.
- `AddItem(list, item)` adds the item `item` to the list `list`.
- `size(list)` returns the size of the list `list`.
- `list[k]` refers to the item at index `k` of the list `list`.
- `GetNextLine(file)` is a method that gets the next line from a file, or returns `nothing` if the end of the file was reached.

5 Functions

- `sqrt(a)` is the square root of `a`, and is equivalent to `pow(a, 0.5)`.
- `ln(a)` is the natural logarithm of `a`.
- `exp(a)` is the inverse natural logarithm of `a`. Also known as the base of natural logarithms raised to the power `a`, so that `exp(1)` is the base of natural logarithms commonly denoted e.
- `pow(a, b)` is the number `a` raised to the power `b`.
- `sin(a)`, `cos(a)`, and `tan(a)` are the sine, cosine, and tangent of the angle `a`, respectively, where `a` is in radians.
- `atan2(y, x)` is—
  - the inverse tangent of `y/x`, in radians, if `x > 0`,
  - π plus the inverse tangent of `y/x`, in radians, if `y >= 0 and x < 0`,
  - $-\pi$ plus the inverse tangent of `y/x`, in radians, if `y < 0 and x < 0`,
  - $-\pi$ divided by 2 if `y < 0 and x == 0`,
  - π divided by 2 if `y > 0 and x == 0`, and
  - 0 if `y == 0 and x == 0`.

- `abs(a)` is the absolute value of `a`; it makes negative numbers nonnegative.
- `min(a, b)` is the smaller of `a` and `b`.
- `max(a, b)` is the larger of `a` and `b`.
- `floor(a)` is the highest integer that is less than or equal to `a`.
- `rem(a, b)` is the part of `b` that does not divide evenly into `a`, where the result has the sign of `b`. This operation is equivalent to `a - floor(a / b) * b`.
- `ceil(a)` is the lowest integer that is greater than or equal to `a`. This function is equivalent to `(0 - floor(0 - a))`.

Notes:

- The inverse sine, in radians, of `a` is equivalent to `atan2(a, sqrt(1.0 - a * a))`.
- The inverse cosine, in radians, of `a` is equivalent to `atan2(sqrt(1.0 - a * a), a)`.
- An integer `n` is *odd* if `rem(n, 2)` is 1.
- An integer `n` is *even* if `rem(n, 2)` is 0.

6 Pseudocode Notes

In the pseudocode:

- Divisions do not round to an integer. (In some programming languages, division of two integers results in an integer, which may be rounded differently depending on the language. For instance, Python's and Ruby's integer division does a floor rounding on the result of division, while Java's discards the fractional part of the result of division.)
- The pseudocode shown is not guaranteed to cover all error handling, such as recovery from overflows, out-of-bounds memory accesses, divisions by zero, unexpected infinity values, and other errors that might happen in a particular implementation.
- The pseudocode shown is not guaranteed to yield high performance in a particular implementation, either in time or memory.
- In general, computer implementations of the operators and functions above risk numerical errors, since computers generally can't operate "exactly" on real numbers. (This is less of an issue if the implementation uses an arbitrary-precision rational number format and the pseudocode uses only rational arithmetic and inputs. In addition, an implementation can work with *symbolic* representations of real numbers instead of those numbers.)

7 License