Pseudocode

From Wikipedia, the free encyclopedia

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm.

It uses the structural conventions of a programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program. Pseudocode resembles, but should not be confused with skeleton programs which can be compiled without errors. Flowcharts, drakon-charts and Unified Modeling Language (UML) charts can be thought of as a graphical alternative to pseudocode, but are more spacious on paper.

Contents

- 1 Application
- 2 Syntax
- 3 Mathematical style pseudocode
  - 3.1 Common mathematical symbols
  - 3.2 Example
- 4 Machine compilation of pseudocode style languages
  - 4.1 Natural language grammar in programming languages

4.2 Mathematical programming languages
5 See also
6 References
7 External links

Application

Textbooks and scientific publications related to computer science and numerical computation often use pseudocode in description of algorithms, so that all programmers can understand them, even if they do not all know the same programming languages. In textbooks, there is usually an accompanying introduction explaining the particular conventions in use. The level of detail of the pseudo-code may in some cases approach that of formalized general-purpose languages.

A programmer who needs to implement a specific algorithm, especially an unfamiliar one, will often start with a pseudocode description, and then "translate" that description into the target programming language and modify it to interact correctly with the rest of the program. Programmers may also start a project by sketching out the code in pseudocode on paper before writing it in its actual language, as a top-down structuring approach, with a process of steps to be followed as a refinement.

Syntax

As the name suggests, pseudocode generally does not actually obey the syntax rules of any particular language; there is no systematic standard form, although any particular writer will generally borrow style and syntax; for example, control structures from some conventional programming language. Popular syntax sources include Fortran, Pascal, BASIC, C, C++, Java, Lisp, and ALGOL. Variable declarations are typically omitted. Function calls and blocks of code, such as code contained within a loop, are often replaced by a one-line natural language sentence.

Depending on the writer, pseudocode may therefore vary widely in style, from a near-exact imitation of a real programming language at one extreme, to a description approaching formatted prose at the other.

This is an example of pseudocode (for the mathematical game fizz buzz):

Fortran style pseudo code

```
program fizzbuzz
  Do i = 1 to 100
    set print_number to true
    If i is divisible by 3
      print "Fizz"
      set print_number to false
    If i is divisible by 5
      print "Buzz"
      set print_number to false
    If print_number, print i
    print a newline
  end do
```

Pascal style pseudo code

```
procedure fizzbuzz
  For i := 1 to 100 do
    set print_number to true;
    If i is divisible by 3 then
      print "Fizz";
      set print_number to false;
    If i is divisible by 5 then
      print "Buzz";
      set print_number to false;
    If print_number, print i;
    print a newline;
  end
```

C style pseudo code:

```
void function fizzbuzz {
  for (i = 1; i <= 100; i++) {
    set print_number to true;
    If i is divisible by 3
      print "Fizz";
      set print_number to false;
    If i is divisible by 5
      print "Buzz";
      set print_number to false;
    If print_number, print i;
    print a newline;
  }
}
```

Basic style pseudo code

```
Sub fizzbuzz()
  For i = 1 to 100
    print_number = True
    If i is divisible by 3 Then
      Print "Fizz"
      print_number = False
    End If
    If i is divisible by 5 Then
      Print "Buzz"
      print_number = False
    End If
    If print_number = True Then print i
    Print a newline
  Next i
End Sub
```

Mathematical style pseudocode

In numerical computation, pseudocode often consists of mathematical notation, typically from set and matrix theory, mixed with the control structures of a conventional programming language, and perhaps also natural language descriptions. This is a compact and often informal notation that can be understood by a wide range of mathematically trained people, and is frequently used as a way to describe mathematical algorithms. For example, the sum operator (capital-sigma notation) or the product operator (capital-pi notation) may represent a for-loop and a selection structure in one expression:

Return $\sum_{k \in S} x_k$

Normally non-ASCII typesetting is used for the mathematical equations, for example by means of markup languages, such as TeX or MathML, or proprietary formula editors.

Mathematical style pseudocode is sometimes referred to as pidgin code, for example *pidgin ALGOL* (the origin of the concept), *pidgin Fortran*, *pidgin BASIC*, *pidgin Pascal*, *pidgin C*, and *pidgin Lisp*.

Common mathematical symbols

Type of operation	Symbol	Example
Assignment	← or :=	$c \leftarrow 2\pi r$, $c := 2\pi r$
Comparison	$=, \neq, <, >, \leq, \geq$	
Arithmetic	$+, -, \times, /$, mod	
Floor/ceiling	$\lfloor, \rfloor, \lceil, \rceil$	$a \leftarrow \lfloor b \rfloor + \lceil c \rceil$
Logical	**and**, **or**	
Sums, products	$\sum \prod$	$h \leftarrow \sum_{a \in A} 1/a$

Example

Here follows a longer example of mathematical style pseudo-code, for the Ford-Fulkerson algorithm:

```
algorithm ford-fulkerson is
    input: Graph G with flow capacity c,
           source node s,
           sink node t
    output: Flow f such that f is maximal from s to t

    (Note that f(u,v) is the flow from node u to node v, and c(u,v) is the flow capacity from node u to node v)

    for each edge (u, v) in GE do
        f(u, v) ← 0
        f(v, u) ← 0

    while there exists a path p from s to t in the residual network Gf do
        let cf be the flow capacity of the residual network Gf
        cf(p) ← min{cf(u, v) | (u, v) in p}
        for each edge (u, v) in p do
            f(u, v) ←  f(u, v) + cf(p)
            f(v, u) ← −f(u, v)

    return f
```

Machine compilation of pseudocode style languages

Natural language grammar in programming languages

Various attempts to bring elements of natural language grammar into computer programming have produced programming languages such as HyperTalk, Lingo, AppleScript, SQL, Inform and to some extent Python. In these languages, parentheses and other special characters are replaced by prepositions, resulting in quite talkative code. These languages are typically dynamically typed, meaning that variable declarations and other boilerplate code can be omitted. Such languages may make it easier for a person without knowledge about the language to understand the code and perhaps also to learn the language. However, the similarity to natural language is usually more cosmetic than genuine. The syntax rules may be just as strict and formal as in conventional programming, and do not necessarily make development of the programs easier.

Mathematical programming languages

An alternative to using mathematical pseudocode (involving set theory notation or matrix operations) for documentation of algorithms is to use a formal mathematical programming language that is a mix of non-ASCII mathematical notation and program control structures. Then the code can be parsed and interpreted by a machine.

Several formal specification languages include set theory notation using special characters. Examples are:

- Z notation
- Vienna Development Method Specification Language (VDM-SL).

Some array programming languages include vectorized expressions and matrix operations as non-ASCII formulas, mixed with conventional control structures. Examples are:

- A programming language (APL), and its dialects APLX and A+.
- MathCAD.

See also

- Concept programming
- Drakon-chart
- Flowchart
- Literate programming
- Program Design Language
- Short Code
- Structured English

References

- Justin Zobel (2004). "Algorithms" in *Writing for Computer Science* (second edition). Springer. ISBN 1-85233-802-4.

External links

- A pseudocode standard (http://www.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html)
- Collected Algorithms of the (http://calgo.acm.org/) ACM
- Pseudocode Guidelines (http://www.cs.cornell.edu/Courses/cs482/2003su/handouts/pseudocode.pdf), PDF file.
- Pseudocode Programming Process (http://www.coderookie.com/2006/tutorial/the-pseudocode-programming-process/) base on data from Code Complete book
- Pseudocode generation tool (http://www.gnstudio.com/open-source/apdt/) from a model tree learn how to generate pseudocode in a second