

Pseudocode

Pseudocode is a hybrid between explaining something to a person and explaining it to a computer. To read pseudocode, it's helpful to pretend to be a computer. To write pseudocode, it's helpful to imagine that you're explaining something to a very literal-minded person with no short term memory who has to write everything down.

Here's some simple pseudocode to generate all the ordered pairs consisting of two integers between 1 and 3:

```
for i from 1 to 3 do
  for j from 1 to 3 do
    print i, j
```

This is an example of a nested loop. There's an inner loop, where *i* is a fixed number and *j* ranges from 1 to 3, and within the inner loop you execute the print statement three times. There's an outer loop, in which you execute the inner loop three times. You should check that if you execute this pseudocode you get

```
1,1
1,2
1,3
2,1
2,2
2,3
3,1
3,2
3,3
```

Here's pseudocode for printing just the pairs *i, j* with *i* less than *j*.

```
for i from 1 to 3 do
  for j from 1 to 3 do
    if i < j then
      print i, j
```

It'll work, but it's wasteful. Better is

```
for i from 1 to 3 do
  for j from i+1 to 3 do
    print i, j
```

Better still:

```
for i from 1 to 2 do
  for j from i+1 to 3 do
    print i, j
```

(Do you see what the difference is, and why the second program is better?)

Indentation should be used to make the nested structure of the pseudocode clearer.

On the whole, pseudocode is supposed to clear to anyone who knows at least one computer language. The only exception that comes to mind is equality. In math we use the word “equal” ambiguously; when we write “Let x equal 2” we are assigning the value 2 to the variable x , whereas when we write “Does x equal y ?” we are asking a question. Do distinguish the two uses of =, we sometimes write assignment statements in the form $x:=2$ and tests of equality in the form $x==y$, although not everyone adheres to this.