Pseudocode

Christophe Rhodes

Motivation

- describe programs
- independent of programming language
- intentionally as simple as possible

Definition

Pseudocode is an informal, high-level description of the operation of a computer program or other algorithm

Implications

- use simplest way to describe things
  - even if that is in English
- not executable by a computer
  - walk-through by humans
  - reasonable

Variable assignment

Variable assignment is indicated by the ⟵ symbol:

x ⟵ 1

Variables in pseudocode do not need to be declared

Sequencing

Vertical space

Statements separated by vertical space happen in sequence

$x \leftarrow 1$

$y \leftarrow x$

$x \leftarrow 2$

What value does x have after this? What about y?

Semicolons

Space sometimes gets tight, and more than one thing needs to go on a line. Semicolons separate statements in a sequence:

$x \leftarrow 1; y \leftarrow x; x \leftarrow 2$

Conditionals

if

Use **if then** to decide whether to do a sequence or not; end the sequence with **end if**

```
x ⟵ 0
if x > −6 then
    x ⟵ x + 1
end if
```

What value does x have after this?

Conditional Operators

Use mathematical notation (not code notation) in pseudocode:

- =, <, >
- ≤, ≥ (not <=, =>, >=)
- ∨, ∧, ¬

Conditionals

else

Use **else** to delimit a sequence to execute if the conditional is not true

```
x ⟵ 0
if x > 17 then
    x ⟵ x + 1
else
    x ⟵ x – 1
end if
```

What value does x have after this?

Conditionals

else if

Define chains of conditionals using **else if**. At most one of the sequences is executed.

```
x ← 0
if x > 3 then
    x ← 5
else if x > −3 then
    x ← 7
else if x > −8 then
    x ← 9
else
    x ← 11
end if
```

What is the value of x after this?

Work

1. Reading
   - CLRS, section 2.1
   - DPV, sections 0.1, 0.2
2. Quiz
   - available now on learn.gold
   - open until 16:00 Friday 12th October
   - try multiple times
   - mark is $30 + 70 \times (\text{score}/10)^2$

For loops

To loop with a variable bound to a series of numbers, use **for** with a description of the series.

```
x ← 0
for 0 ≤ i < 100 do
    x ← x + 1
end for
```

what is the value of x after this? 100

For loops

To loop with a variable bound to a series of numbers, use **for** with a description of the series.

```
x ← 0
for 0 ≤ i < 100 do
    x ← x + i
end for
```

what is the value of x after this? 4950

For loops

The order might matter: start with the left-hand bound and move towards the right-hand one.

```
x ← 0
for 0 ≤ i < 100 do
    x ← i
end for
```

```
x ← 0
for 100 > i ≥ 0 do
    x ← i
end for
```

what is the value of x after each of these?

99

0

For loops

Use **continue** to proceed directly to the next iteration of the innermost loop, and **break** to finish the innermost loop

```
x ← 0
for 0 ≤ i < 10 do
    x ← x + 1
    if x > 3 then
        break
    end if
    x ← x + i
end for
```

```
x ← 0
for 0 ≤ i < 10 do
    x ← x + 1
    if x > 3 then
        continue
    end if
    x ← x + i
end for
```

what is the value of x after each of these?

Nested loops

Loops nest: with nested loops, for each iteration of an outer loop, do the whole inner loop:

```
x ⟵ 0
for 0 ≤ i < 3 do
    for 0 ≤ j < 4 do
        x ⟵ x + 1
    end for
end for
```

what is the value of x after this? 12

Forall

Iterate over members of a collection using **forall**

```
x ⟵ 0
for all p ∈ prime numbers below 10 do
    x ⟵ x + 1
end for
```

what is the value of x after this? 4

ordering

There may be a natural order to the iteration (*e.g.* when iterating over a linear collection), but usually there won't be. Don't rely on a particular order!

While

Use **while** to express a loop which tests a condition at the start of a sequence, and if that condition is true does another iteration of the loop.

```
x ⟵ 0
y ⟵ 3
while y > 0 do
    x ⟵ x + 1
    y ⟵ y − 1
end while
```

what value does x have after this? 3

Repeat

Use **repeat until** to express a loop which tests a condition at the end of a sequence, and if that condition is false does another iteration of the loop.

```
x ⟵ 0
y ⟵ 3
repeat
    x ⟵ x + 1
    y ⟵ y − 1
until y < 0
```

what value does x have after this? 4

Loop

Use **loop** to express an unconditional loop. (You will need to use **break** to terminate the loop).

```
x ⟵ 11342
loop
    if x = 1 then
        break
    else if x is even then
        x ⟵ x ÷ 2
    else
        x ⟵ 3 × x + 1
    end if
end loop
```

Function calls

Functions have zero or more arguments, and return one result. Call them using their name, with arguments in brackets

$n \longleftarrow 5$

$x \longleftarrow \textsc{fact}(n)$

Functions

Define functions using **function**, and return a value using **return**.

```
function FACT(n)
    if n = 0 then
        return 1
    else
        return n × FACT(n−1)
    end if
end function
```

Pre- and post-conditions

Make it clear to the reader what conditions a function requires to operate correctly, and what it does if those conditions are met

Require: $n \in \mathbb{N}_0$
Ensure: Compute and return n!

```
function FACT(n)
    if n = 0 then
        return 1
    else
        return n × FACT(n−1)
    end if
end function
```