

Get the TurningPoint app for your phone. See

<http://community.dur.ac.uk/lt.team/?portfolio=turning-technologies>

or use the web

<https://student.turningtechnologies.eu/>

We will use the channel “DurhamADS2021”.

Algorithms and Data Structures Part 1

Topic 1: Introduction and Pseudocode

Matthew Johnson

matthew.johnson2@dur.ac.uk

Algorithms

What is an “algorithm”?

An algorithm is a method or a process followed to solve a problem.

What properties must an algorithm have?

- Correctness.
- Composed of concrete unambiguous steps.
- The number of steps must be finite.
- Must terminate.

Data Structures

What is a “data structure”?

A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.

Why do we study data structures and algorithms?

*We want to solve problems **efficiently** – to make the best use of resources such as **space** and **time**. Our choice of data structure or algorithm can make the difference between a program running in a few seconds or many days.*

5 / 42

In this module

- Learn the commonly used **data structures** and when to use them. These form a programmer’s basic data structure “toolkit.”
- Study well-known **algorithmic techniques** and demonstrate their application.
- Understand how to measure the **cost** of a data structure or an algorithm. These techniques also allow you to judge the merits of new data structures and algorithms that you or others might invent.

6 / 42

Module Content

- Introduction & pseudocode
 - Basic data structures
 - Recursive algorithms
 - Analysing algorithms (asymptotic classes)
 - Sorting
 - Binary search
 - Graph algorithms
 - String matching
- (see the syllabus on duo)

7 / 42

Module Information

- 40 one-hour lectures. 19 one-hour practicals.
- Lecturers: Matthew Johnson, Konrad Dabrowski, Andrei Krokhin and George Mertzios
- Course text: [Data Structures and Algorithms in Python \(or Java\)](#) by Goodrich et al.
- Course website: [duo](#)
- Assessment:
 - Assignment: deadline is 2pm on 15 January 2021.
 - End of year exam.

8 / 42

Machine model

Random access machine:

- 1 Memory consists of an infinite array.
- 2 Instructions executed sequentially one at a time.
- 3 All instructions take unit time. Running time is the number of instructions executed.

9 / 42

Pseudocode

To describe algorithms we will use generic pseudocode, not any one programming language.

No (very) strict rules

Typical “framework”:

Algorithm: Foo

Input: numbers a_1, a_2, \dots, a_n

Output: $\max\{a_i \mid 1 \leq i \leq n\}$

“Code that solves the problem”

11 / 42

Pseudocode: variables

Will (often) need to use variables

Basic types: **integer**, **float**, **char**(acter), **string**

If you've got an "important" variable then explicitly **declare** it, i.e., say what type it will be storing

```
integer i
float f
char c
string s
```

12 / 42

Pseudocode: variables

Of course, you will want to assign values to variables

No real convention here; some (real) languages use "=", some use ":", and others use "←"

```
integer i = 0
string s := "test string"
char c ← 'a'
```

(Many languages use double quotes for strings, and single quotes for characters)

13 / 42

Pseudocode: variables

May also have declaration separate from initialisation, e.g.

```
integer i
i = 12
```

Also, may have multiple variables in one go:

```
integer x=0, y=2, z=3
```

14 / 42

Arithmetic operations

- (...) parentheses (or brackets) for grouping
- + - * / add, subtract, multiply, divide

Examples

```
volume = length * width * height
z = (x+1) * y / (a-b)
```

Logical operations: AND, OR, NOT

15 / 42

Output (Printing)

Keyword "print"

Examples

```
print x
print "Hello world"
print "value of z is ", z
```

may produce outputs

```
0
Hello world
value of z is 12
```

Notice comma for concatenation (elsewhere may see '+')

16 / 42

If-then-else

Simple if-then

```
if condition then
    statement
    statement
end if
```

Many conditions involve comparisons:

< > <= (≤) >= (≥) == (=) != (≠)

```
if x ≠ 0 then
    z = y/x
end if
```

It's good style to visually indent the "body" of a conditional statement

17 / 42

If-then-else: Question 1

```
i =  
if i ≥ 5 then  
    i = i + 5  
    print i  
end if
```

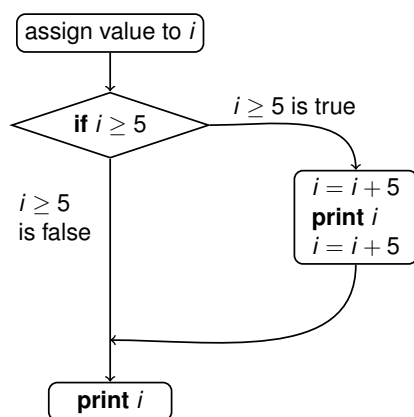
18 / 42

If-then-else: Question 2

```
i =  
if i ≥ 5 then  
    i = i + 5  
    print i  
    i = i + 5  
end if  
print i
```

19 / 42

If-then flow diagram for Question 2



20 / 42

If-then-else

What if we want to do something when the condition is false?
Could, of course, write

```
if  $x \neq 0$  then
     $z = y/x$ 
end if
if  $x = 0$  then
    print "division by zero error"
end if
```

However, this is nicer:

```
if  $x \neq 0$  then
     $z = y/x$ 
else
    print "division by zero error"
end if
```

Nested conditionals

Can of course construct more complicated things:

```
if condition then
    statement
    if condition then
        statement
        statement
    else
        statement
    end if
else
    if condition then
        statement
    end if
end if
```

22 / 42

Nested conditionals

But care needed with the presentation

```
 $n = 0$ 
integers  $a, b$  and  $c$ 
if  $a > b$  then
    if  $a > c$  then
         $n = 1$ 
    else
         $n = 2$ 
    print  $n$ 
```

23 / 42

If-then-else: Question 3

```
integers  $a, b, c$ 
if  $a > b$  then
     $x = a$ 
     $y = b$ 
else
     $x = b$ 
     $y = a$ 
end if
if  $c > x$  then
    print  $x$ 
else
    if  $c > y$  then
        print  $c$ 
    else
        print  $y$ 
    end if
end if
```

For loop

If you want to iterate some (numerical) variable through some range

Great many variations in how languages do this, simplest is probably

```
for  $variable = lower$  to  $upper$  do
    body (will often depend on  $variable$ )
end for
```

“Body” is simply a sequence of statements (and may contain If-Then-Elses, other loops, whatever)

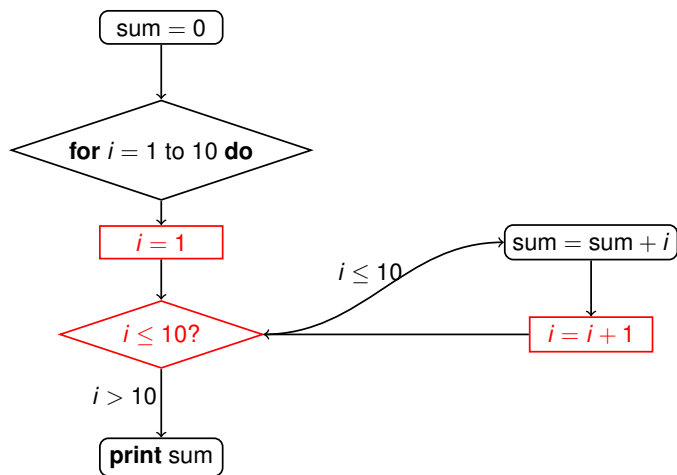
25 / 42

For loop example

```
integer  $sum = 0$ 
for  $i = 1$  to  $10$  do
     $sum = sum + i$ 
end for
print  $sum$ 
```

26 / 42

for loop flow diagram for previous example



You'll have noticed that the previous **for** loop can only iterate consecutive integers.

There's another, more generic one: iterate over given base set

```

for value in {value1, value2, ...} do
    body (will often depend on value)
end for
  
```

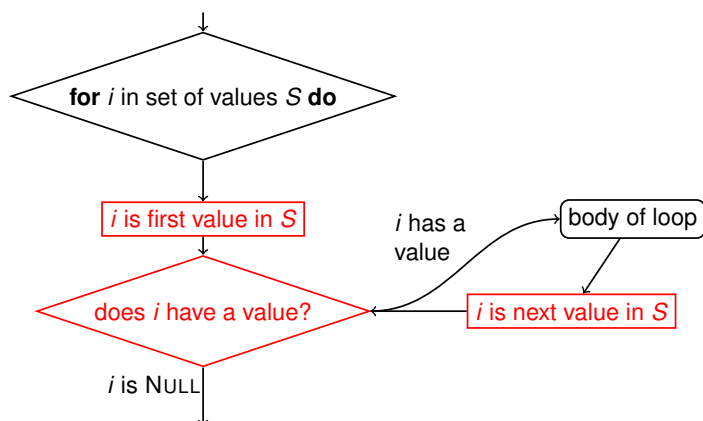
Example

```

integer sum = 0
for prime in { 2, 3, 5, 7 } do
    sum = sum + prime
end for
  
```

28 / 42

generic for loop flow diagram



29 / 42

for loops: Question 4

Or we can iterate some variable through a range, but increment by a stated value.

```
for  $i = 0$  to 9;  $i += 2$  do  
    print  $i$   
end for
```

30 / 42

for loops: Question 5

```
for  $i = 0$  to 9;  $i += -1$  do  
    print  $i$   
end for
```

31 / 42

While loop

Do something while some condition is true

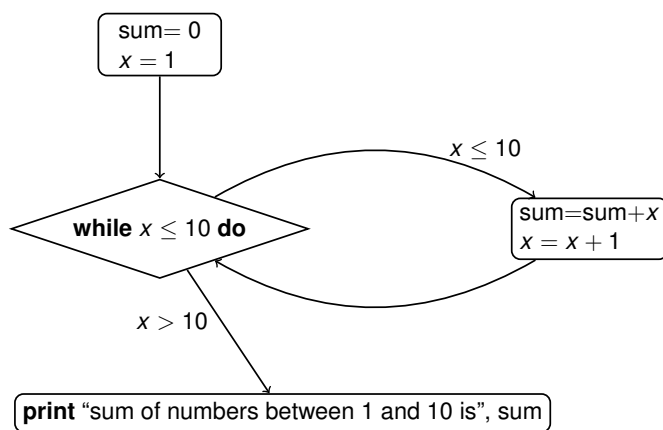
```
while condition do  
    body  
end while
```

E.g.

```
sum = 0  
x = 1  
while  $x \leq 10$  do  
    sum = sum + x  
    x = x + 1  
end while  
print "sum of numbers between 1 and 10 is", sum
```

32 / 42

while loop flow diagram for previous example



Important difference between **for** and **while** over numerical values:

for increments loop-variable automatically; **while** doesn't

```
for i = 1 to 10 do
  print i
end for
```

```
i = 1
while i ≤ 10 do
  print i
  i = i + 1  ← need to increment i by hand
end while
```

34 / 42

while loops: Question 6

```
product = 1
x = 0
while x ≤ 5 do
  product = product × x
end while
print product
```

35 / 42

while loops: Question 7

```
positive integer  $a$   
positive integer  $b$   
integer  $s = 0$   
integer count = 1  
while count  $\leq a$  do  
     $s = s + b$   
    count = count + 1  
end while  
print  $s$ 
```

36 / 42

while loops: Question 8

```
positive integer  $n$   
integers  $a_1, a_2, a_3, \dots, a_n$   
integer  $s = 0$   
for  $i = 1$  to  $n$  do  
     $s = s + a_i$   
end for  
 $s = s/n$   
print  $s$ 
```

37 / 42

Everything can be nested: Question 9

```
for  $x = 1$  to 4 do  
    for  $y = 1$  to 4 do  
        go to coordinate  $(x, y)$   
        plot red circle of diameter 0.1  
    end for  
end for
```

What if the first two lines were swapped?

38 / 42

We have a few further practice questions. If you find yourself waiting for others then do this:

Let S be a collection of numbers; (for example $S = \{3, 8, 17, 2, 9\}$). Write pseudocode to print the smallest and largest numbers in the collection. Use a for loop.

Or this:

Consider the simple game where you try to guess a number and are told whether your guess is too high or too low (or is correct)? Write some pseudocode that plays this game. Assume that the number to be guessed, say x , is given and for the guesses you can just include something like

positive integer y obtained by requesting input from user

A while loop and a couple of if statements should be all you

39 / 42

Question 10

```
positive integer  $x$ 
positive integer  $n$ 
integer  $z = 0$ 
while  $n > 0$  do
    if  $n$  is odd then
         $z = z + x$ 
         $n = n - 1$ 
    end if
     $x = x * 2$ 
     $n = n / 2$ 
end while
print  $z$ 
```

40 / 42

Question 11

```
positive integer  $x$ 
positive integer  $n$ 
integer  $z = 1$ 
while  $n > 0$  do
    if  $n$  is odd then
         $z = z * x$ 
         $n = n - 1$ 
    end if
     $x = x * x$ 
     $n = n / 2$ 
end while
print  $z$ 
```

41 / 42

Question 12

```
positive integer  $a$   
positive integer  $b$   
if  $a < b$  then  
     $a, b = b, a$  {exchange the values}  
end if  
while  $b \neq 0$  do  
    integer  $r$  is the remainder when  $a$  is divided by  $b$   
     $a = b$   
     $b = r$   
end while  
print  $a$ 
```