

Exercise walk-through

11.1

Programming Basics

A-Level Computer Science

11.1 Programming Basics

Name: _____ Class: _____ Date: _____ **Total: 53 marks**

KEY WORDS variables • constants • data type • assignment
• identifier

Objective

Write your first **pseudocode** statements: declare **variables** and **constants**, give them values with **assignment**, and use INPUT and OUTPUT. This is the first sheet of 11.1 and it needs no earlier programming. Sheet 11.1.1 then deals with operators and expressions, and sheet 11.1.2 with built-in functions.

You must be able to:

- write pseudocode statements that declare variables, and that declare and initialise constants
- choose the correct **data type** for a variable, and write a value of each type correctly
- write assignment statements, and follow what a sequence of assignments does
- write statements for input from the keyboard and output to the console

1 Worked examples

Study these first. Each one shows a **method** that a question later on this sheet needs.

■ Variables and constants

A **variable** is a named place in memory. Its value **can change** while the program runs. A **constant** has a value that is set once and **never changes**.

DECLARE Count : INTEGER

Count $\leftarrow$ 7

Count $\leftarrow$ Count + 1



a box named Count
is made

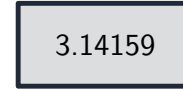


the value 7
is put in the box



7 + 1 is worked out, and
8 **replaces** the 7

CONSTANT Pi = 3.14159



a constant: its value
can **never** change

```
DECLARE Count : INTEGER
```

```
DECLARE Price : REAL
```

```
DECLARE Name : STRING
```

```
CONSTANT TaxRate = 0.2
```

- A variable is declared with `DECLARE`, its **identifier** , a colon and its **data type**.
- A constant is declared with `CONSTANT`, its identifier, the sign = and its value. It has no arrow, and its type is not written.
- **Why use a constant?** The value has a meaningful name, it cannot be changed by mistake, and if the tax rate changes it is changed in **one** place only.

■ The data types, and how to write a value

data type	holds	a value is written like this
INTEGER	a whole number	25 -3
REAL	a number with a fraction part	4.5 -0.01
CHAR	one character	'A' '7' (single quotes)
STRING	text: zero or more characters	"Hello" "" (double quotes)
BOOLEAN	true or false	TRUE FALSE
DATE	a date	25/12/2026

"25" is a `STRING`, not a number: it is text that happens to contain digits. You cannot do arithmetic with it.

■ Assignment

```
Count ← 0
Price ← 4.50
Name ← "Ada"
Count ← Count + 1
```

- The arrow `←` means "**becomes**". The **right** side is worked out first, and the result is stored in the variable on the **left**.
- `Count ← Count + 1` is not an equation. It means: take the value in `Count`, add 1, and store the result back in `Count`.
- The left side is always **one variable**. `5 ← Count` and `A + B ← 10` are errors.
- Give a variable a value before you use it. This first value is its **initial value** ; setting it is called **initialising** the variable.

Method: follow a sequence of assignments. Use one column for each variable, and one row for each statement.

statement	A	B	Temp
A ← 3	3		
B ← 8	3	8	
Temp ← A	3	8	3
A ← B	8	8	3
B ← Temp	8	3	3

The last three statements **swap** the values of A and B. The third variable is needed: after `A ← B` the old value of A would be lost.

■ Input and output

```
OUTPUT "Enter your name"
INPUT Name
OUTPUT "Hello, ", Name
OUTPUT "You have ", Count, " messages"
```

- INPUT waits for the user to type a value, and stores it in the variable.
- OUTPUT displays one or more values, separated by **commas**. Text is in double quotes; a variable is not.
- An OUTPUT before an INPUT that tells the user what to type is called a **prompt**

■ A whole program: declare, input, process, output

Input the price of a meal. Add a service charge of 10%, and output the amount to pay.

```
CONSTANT ServiceRate = 0.1
DECLARE MealPrice : REAL
DECLARE AmountToPay : REAL
OUTPUT "Enter the price of the meal"
INPUT MealPrice
AmountToPay ← MealPrice + MealPrice * ServiceRate
OUTPUT "Amount to pay: ", AmountToPay
```

Past-paper practice: complete sheets 11.1, 11.1.1 and 11.1.2 before attempting past-paper set 11.1.2.

Exercise 2.1

2.1 State whether each item should be a **variable** or a **constant** in a program for a school shop. [4]

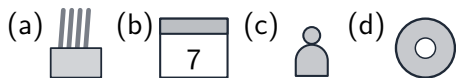
- (a) the number of pens left in the shop
- (b) the number of days in a week
- (c) the name of the customer who is being served
- (d) the value of π , used to find the area of a round badge



Exercise 2.1

2.1 State whether each item should be a **variable** or a **constant** in a program for a school shop. [4]

- (a) the number of pens left in the shop
- (b) the number of days in a week
- (c) the name of the customer who is being served
- (d) the value of π , used to find the area of a round badge



2.1

- (a) it changes each time a pen is sold → **variable**
- (b) it never changes → **constant**
- (c) it changes with each customer → **variable**
- (d) it never changes → **constant**

Exercise 2.2

2.2 State the data type of each value.

[5]

(a) "Computer"

(b) 'C'

(c) 18

(d) 18.0

(e) FALSE

(a) "Computer"

(b) 'C'

(c) 18

(d) 18.0

(e) FALSE

Exercise 2.2

2.2 State the data type of each value.

[5]

(a) "Computer"

(b) 'C'

(c) 18

(d) 18.0

(e) FALSE

(a) "Computer"

(b) 'C'

(c) 18

(d) 18.0

(e) FALSE

2.2

(a) text in double quotes → **STRING**

(b) one character in single quotes → **CHAR**

(c) a whole number → **INTEGER**

(d) it has a fraction part (a decimal point) → **REAL**

(e) **BOOLEAN**

Exercise 2.3 (a)

2.3 Write a pseudocode statement for each of these.

[4]

(a) declare a variable **Age** that holds a whole number

Age		Surname	
Height		MaxPlayers	

Exercise 2.3 (a)

2.3 Write a pseudocode statement for each of these.

[4]

(a) declare a variable **Age** that holds a whole number

Age		Surname	
Height		MaxPlayers	

(a) DECLARE Age : INTEGER

Exercise 2.3 (b)

2.3 Write a pseudocode statement for each of these.

[4]

(b) declare a variable **Surname** that holds text

Exercise 2.3 (b)

2.3 Write a pseudocode statement for each of these.

[4]

(b) declare a variable **Surname** that holds text

(b) **DECLARE Surname : STRING**

Exercise 2.3 (c)

2.3 Write a pseudocode statement for each of these.

[4]

(c) declare a constant `MaxPlayers` with the value 11

Exercise 2.3 (c)

2.3 Write a pseudocode statement for each of these.

[4]

(c) declare a constant `MaxPlayers` with the value 11

(c) `CONSTANT MaxPlayers = 11`

Exercise 2.3 (d)

2.3 Write a pseudocode statement for each of these.

[4]

(d) store the value 3.5 in the variable `Height`

Exercise 2.3 (d)

2.3 Write a pseudocode statement for each of these.

[4]

(d) store the value 3.5 in the variable `Height`

(d) `Height` $\leftarrow$ 3.5

Exercise 2.4 (a)

2.4 Each statement contains an error. Write the correct statement.

[4]

(a) DECLARE Total = INTEGER



Exercise 2.4 (a)

2.4 Each statement contains an error. Write the correct statement.

[4]

(a) `DECLARE Total = INTEGER`

(a) a declaration uses a colon, not =: `DECLARE Total : INTEGER`

Exercise 2.4 (b)

2.4 Each statement contains an error. Write the correct statement.

[4]

(b) `CONSTANT Rate : 0.05`

Exercise 2.4 (b)

2.4 Each statement contains an error. Write the correct statement.

[4]

(b) `CONSTANT Rate : 0.05`

(b) a constant is given its value with `=`, not with a colon: `CONSTANT Rate = 0.05`

Exercise 2.4 (c)

2.4 Each statement contains an error. Write the correct statement.

[4]

(c) `10 ← Score`

Exercise 2.4 (c)

2.4 Each statement contains an error. Write the correct statement.

[4]

(c) `10 ← Score`

(c) the variable is on the left of the arrow: `Score ← 10`

Exercise 2.4 (d)

2.4 Each statement contains an error. Write the correct statement.

[4]

(d) `OUTPUT "The total is, Total"`

Exercise 2.4 (d)

2.4 Each statement contains an error. Write the correct statement.

[4]

(d) `OUTPUT "The total is, Total"`

(d) the variable must be outside the quotes, after a comma: `OUTPUT "The total is ", Total`

Exercise 2.5

2.5 Complete the table to show the value of each variable after each statement. [4]

X

Y

statement	X	Y
$X \leftarrow 4$		
$Y \leftarrow X + 2$		
$X \leftarrow X * Y$		
$Y \leftarrow X - Y$		

Exercise 2.5

2.5 Complete the table to show the value of each variable after each statement. [4]

X

Y

statement	X	Y
$X \leftarrow 4$		
$Y \leftarrow X + 2$		
$X \leftarrow X * Y$		
$Y \leftarrow X - Y$		

2.5

- work down the table, one statement at a time; a variable keeps its value until it is assigned again (one mark for each correct row)

statement	X	Y
$X \leftarrow 4$	4	
$Y \leftarrow X + 2$	4	6
$X \leftarrow X * Y$	24	6
$Y \leftarrow X - Y$	24	18

Exercise 2.6

2.6 Write pseudocode that asks the user for their name and their favourite sport, and then outputs one sentence that contains both. Declare your variables. [4]



Exercise 2.6

2.6 Write pseudocode that asks the user for their name and their favourite sport, and then outputs one sentence that contains both. Declare your variables. [4]



2.6

- two **STRING** variables declared
- a prompt and an **INPUT** for each value
- one **OUTPUT** that joins text and both variables with commas

Full solution on next slide

Full solution 2.6

2.6

- two `STRING` variables declared
- a prompt and an `INPUT` for each value
- one `OUTPUT` that joins text and both variables with commas

```
DECLARE Name : STRING
DECLARE Sport : STRING
OUTPUT "Enter your name"
INPUT Name
OUTPUT "Enter your favourite sport"
INPUT Sport
OUTPUT Name, " likes ", Sport
```

Exercise 3.1 (a)

3.1 A program for a cinema uses these values: the name of a film, the number of tickets sold, the price of one ticket, whether the film is in 3D, and the letter of the row of a seat.

(a) Write a pseudocode declaration for a variable to hold **each** of the five values. [5]



Exercise 3.1 (a)

3.1 A program for a cinema uses these values: the name of a film, the number of tickets sold, the price of one ticket, whether the film is in 3D, and the letter of the row of a seat.

(a) Write a pseudocode declaration for a variable to hold **each** of the five values. [5]



(a) one mark for each declaration with a meaningful identifier and the correct type, for example:

```
DECLARE FilmName : STRING
DECLARE TicketsSold : INTEGER
DECLARE TicketPrice : REAL
DECLARE Is3D : BOOLEAN
DECLARE RowLetter : CHAR
```

Exercise 3.1 (b)

3.1 A program for a cinema uses these values: the name of a film, the number of tickets sold, the price of one ticket, whether the film is in 3D, and the letter of the row of a seat.

(b) The booking fee is always 1.50. Write the pseudocode statement that declares this as a constant. [1]

Exercise 3.1 (b)

3.1 A program for a cinema uses these values: the name of a film, the number of tickets sold, the price of one ticket, whether the film is in 3D, and the letter of the row of a seat.

(b) The booking fee is always 1.50. Write the pseudocode statement that declares this as a constant. [1]

(b) `CONSTANT BookingFee = 1.50`

Exercise 3.1 (c)

3.1 A program for a cinema uses these values: the name of a film, the number of tickets sold, the price of one ticket, whether the film is in 3D, and the letter of the row of a seat.

(c) Give **two** benefits of using a constant for the booking fee. [2]

Exercise 3.1 (c)

3.1 A program for a cinema uses these values: the name of a film, the number of tickets sold, the price of one ticket, whether the film is in 3D, and the letter of the row of a seat.

(c) Give **two** benefits of using a constant for the booking fee. [2]

(c) any **two** of:

- the value cannot be changed by mistake while the program runs
- if the fee changes, the program is changed in one place only
- the name makes the program easier to read than the number 1.50

Exercise 3.2 (a)

3.2 Study this pseudocode.

```
DECLARE First : INTEGER
DECLARE Second : INTEGER
First ← 6
Second ← First * 2
First ← First + Second
Second ← Second - 2
OUTPUT First, " ", Second
```



(a) State what is output.

[2]

Exercise 3.2 (a)

3.2 Study this pseudocode.

```
DECLARE First : INTEGER
DECLARE Second : INTEGER
First ← 6
Second ← First * 2
First ← First + Second
Second ← Second - 2
OUTPUT First, " ", Second
```



(a) State what is output.

[2]

(a) **First** becomes 6, **Second** becomes 12, **First** becomes $6 + 12 = 18$, **Second** becomes $12 - 2 = 10$

- the output is **18 10**

Exercise 3.2 (b)

3.2 Study this pseudocode.

```
DECLARE First : INTEGER
DECLARE Second : INTEGER
First ← 6
Second ← First * 2
First ← First + Second
Second ← Second - 2
OUTPUT First, " ", Second
```



(b) A student adds the statement `Second ← "ten"` at the end. Explain why this is an error.

[2]

Exercise 3.2 (b)

3.2 Study this pseudocode.

```
DECLARE First : INTEGER
DECLARE Second : INTEGER
First ← 6
Second ← First * 2
First ← First + Second
Second ← Second - 2
OUTPUT First, " ", Second
```



(b) A student adds the statement `Second ← "ten"` at the end. Explain why this is an error. [2]

(b) `Second` was declared as an `INTEGER`

- `"ten"` is a `STRING`, and a value of one data type cannot be stored in a variable of another

Exercise 3.3 (a)

3.3 Two variables `Left` and `RightValue` hold the strings "red" and "blue". A student tries to swap the two values with this pseudocode:

```
Left ← RightValue  
RightValue ← Left
```

Left "red"
Right "blue"

(a) State the value of `Left` and the value of `RightValue` after the two statements. [1]

Exercise 3.3 (a)

3.3 Two variables `Left` and `RightValue` hold the strings "red" and "blue". A student tries to swap the two values with this pseudocode:

```
Left ← RightValue  
RightValue ← Left
```

Left "red"
Right "blue"

(a) State the value of `Left` and the value of `RightValue` after the two statements. [1]

(a) both variables hold **"blue"**

Exercise 3.3 (b)

3.3 Two variables `Left` and `RightValue` hold the strings "red" and "blue". A student tries to swap the two values with this pseudocode:

```
Left ← RightValue  
RightValue ← Left
```

Left	"red"
Right	"blue"

(b) Explain why the swap does not work.

[2]

Exercise 3.3 (b)

3.3 Two variables `Left` and `RightValue` hold the strings "red" and "blue". A student tries to swap the two values with this pseudocode:

```
Left ← RightValue  
RightValue ← Left
```

Left	"red"
Right	"blue"

- (b) Explain why the swap does not work. [2]
- (b) the first statement replaces the value of `Left` with "blue", so the value "red" is lost
- the second statement then copies "blue" back into `RightValue`

Exercise 3.3 (c)

3.3 Two variables `Left` and `RightValue` hold the strings `"red"` and `"blue"`. A student tries to swap the two values with this pseudocode:

```
Left ← RightValue  
RightValue ← Left
```

Left "red"
Right "blue"

(c) Write pseudocode that swaps the two values correctly. Declare any extra variable you use. [3]

Exercise 3.3 (c)

3.3 Two variables `Left` and `RightValue` hold the strings "red" and "blue". A student tries to swap the two values with this pseudocode:

```
Left ← RightValue  
RightValue ← Left
```

Left "red"
Right "blue"

(c) Write pseudocode that swaps the two values correctly. Declare any extra variable you use. [3]

(c) a third variable of the same type is declared

- the value of `Left` is saved first
- then `Left ← RightValue`, and the saved value goes into `RightValue`

```
DECLARE Temp : STRING  
Temp ← Left  
Left ← RightValue  
RightValue ← Temp
```

Exercise 3.4 (a)

3.4 A mobile phone company charges 0.08 for each text message and 0.15 for each minute of calls. These two prices do not change. A program inputs a customer's name, the number of text messages and the number of minutes of calls. It outputs the customer's name and the total cost.

(a) Complete the identifier table for the program.

[3]

identifier	data type	description
CustomerName		the customer's name
Texts		the number of text messages
Minutes		the number of minutes of calls
TotalCost		the total cost

Exercise 3.4 (a)

3.4 A mobile phone company charges 0.08 for each text message and 0.15 for each minute of calls. These two prices do not change. A program inputs a customer's name, the number of text messages and the number of minutes of calls. It outputs the customer's name and the total cost.

(a) Complete the identifier table for the program.

[3]

identifier	data type	description
CustomerName		the customer's name
Texts		the number of text messages
Minutes		the number of minutes of calls
TotalCost		the total cost

(a) CustomerName is a **STRING**; Texts and Minutes are **INTEGER**

- TotalCost is a **REAL**

Exercise 3.4 (b)

3.4 A mobile phone company charges 0.08 for each text message and 0.15 for each minute of calls. These two prices do not change. A program inputs a customer's name, the number of text messages and the number of minutes of calls. It outputs the customer's name and the total cost.

(b) Write pseudocode for the program. Use constants for the two prices, and declare your variables. [7]



Exercise 3.4 (b)

3.4 A mobile phone company charges 0.08 for each text message and 0.15 for each minute of calls. These two prices do not change. A program inputs a customer's name, the number of text messages and the number of minutes of calls. It outputs the customer's name and the total cost.

(b) Write pseudocode for the program. Use constants for the two prices, and declare your variables. [7]



(b) two constants declared with = and the correct values

- the four variables declared
- the three inputs

Full solution 3.4 (b)

(b) two constants declared with = and the correct values

- the four variables declared
- the three inputs
- the cost worked out as $\text{texts} \times \text{text price} + \text{minutes} \times \text{minute price}$
- the result stored in `TotalCost`
- the name and the total cost output

```
CONSTANT TextPrice = 0.08
CONSTANT MinutePrice = 0.15
DECLARE CustomerName : STRING
DECLARE Texts : INTEGER
DECLARE Minutes : INTEGER
DECLARE TotalCost : REAL
INPUT CustomerName
INPUT Texts
INPUT Minutes
TotalCost ← Texts * TextPrice + Minutes * MinutePrice
OUTPUT CustomerName, " ", TotalCost
```