

Declaring Variables and Constants

Variables are assigned using the = operator e.g. $x = 3$.

Local variables Variables declared inside a function or procedure are local to that subroutine.

Global variables Variables in the main program can be made global with the keyword global. E.g. **GLOBAL** `userid = 123`.

Constants The values of constants do not change throughout the program.
E.g. **CONST** `Vat = 20`.

Data Types

Integer	VAR age as INTEGER	Whole numbers only	0, 6, 10293, - 999
Real or Float	VAR price as REAL	Numbers that have a decimal point	0.15, - 5.87, 100.0
Char	VAR letter as CHAR	A single letter, number, symbol	"A", "k", "- 5", "-", "\$"
String	VAR name as STRING	Used to represent text, it is a collection of characters	"FsT- mQ2", "- \$money\$"
Boolean	VAR numFound as BOOLEAN	Could take one of two values, usually TRUE or FALSE	True/F- alse, 1/0, Yes/No

Casting Variables

You can change the data type of a variable by using casting.

Converting integer 3 to string. `str(3)` returns "3"

Converting string "3" to integer. `int("3")` returns 3

Converting string "3.14" to float. `float("3.14")` returns 3.14

String Handling

Finding the length of a string `VAR name as STRING`
`name = INPUT("Enter your name")`
`PRINT("Your name has" + name.length + "characters")`

String Handling (cont)

Getting a substring `stringname.substring(startingPosition, numberOf)`
NB The string will start with the 0th character.

Example:

```
someText = "Computer Science"
PRINT(someText.length)
PRINT(someText.substring(3,3))
```

Will display:

```
16
put
```

**Extracting a
specific character
from a string** `name[i]`
Example:
`name = "Paloma"`
`name[3]` returns "o"

**Converting to
uppercase** `name.UPPER()`

**Converting to
lowercase** `name.LOWER()`

Taking inputs from user

Inputs taken from a user need to be stored in a variable.

Example: `VAR name as STRING`
`name = INPUT("Enter your name")`

Outputting to screen

Outputting a string `PRINT("Hello")`

**Outputting a variable set
by you** `word = ("Hello")`
`PRINT(word)`

**Outputting a variable
entered by the user** `VAR name as STRING`
`name = INPUT("What is your name?")`
`PRINT("Hello" + name)`

1-Dimensional Arrays

Declaring an array `ARRAY names[5]`

**Initialising an array - filling it up with
values** `names[0] = "Ahmad"`
`names[1] = "Ben"`
`names[2] = "Catherine"`
`names[3] = "Dana"`
`names[4] = "Elijah"`

Displaying a specific item from an array `PRINT(names[3])`
will display "Dana"

1-Dimensional Arrays (cont)

Displaying ALL items in an array - method 1

```
FOR i = 0 to 5
  PRINT(names[i])
NEXT i
```

Displaying ALL items in an array - method 2

```
ARRAY names[5]
names[0] = "Ahmad"
names[1] = "Ben"
names[2] = "Catherine"
names[3] = "Dana"
names[4] = "Elijah"
PRINT(names)
```

Dynamically inserting values in an array

E.g. Ask the user to enter 5 names

```
FOR i = 0 to 5
  names[i] = INPUT("Enter name:")
NEXT i
```

Performing calculations on one Array element

E.g. Increase element 2 of ARRAY age by 10: age[2] = age[2] + 10

Performing calculations on Array elements

E.g. Increase ALL the values in ARRAY ages by 2:

```
FOR i = 0 to 4
  age[i] = age[i] + 2
NEXT i
```

2-Dimensional Arrays

Note: *Refer to CGP Page 50*

Declaring a 2D array

A 2D array is built as

```
ARRAY(row, column)
```

ARRAY score[4,5]

builds an array of 4 rows, 5 columns.

This can be interpreted as 4 Tests, 5 Students

Initialising a 2D array - filling it up with values

```
score[0,0] = "15"
```

Sets score 15 to Test 0, Student 0

Displaying a specific item from a 2D array

```
PRINT(score[1,3])
```

will display 14

2-Dimensional Arrays (cont)

Dynamically inserting values in an array

E.g. Ask the user to enter all the scores

```
FOR i = 0 to 3
  FOR j = 0 to 4
    score[i,j] = INPUT("Enter score for Test " + i + " Student " + j)
  NEXT j
NEXT i
```

Sub Programs - Functions

Functions take at least one parameter and they must always return a value.

Example: Write a function to join two strings together with a space between them and show it working on the strings "computer" and "science".

```
FUNCTION join_strings(x as STRING, y as STRING) as STRING
  RETURN x + " " + y
ENDFUNCTION
```

Calling the function from the main program:

```
subject = join_strings("computer", "science")
PRINT(subject)
```

Sub Programs - Procedures

Procedures don't have to take parameters...	...but sometimes they w
PROCEDURE welcome()	PROCEDURE betterwe
PRINT("Hello and welcome.")	PRINT("Hello" + name
PRINT("Let's learn about procedures.")	PRINT("Let's learn ab
ENDPROCEDURE	ENDPROCEDURE

Procedures are called by typing their name... and giving an argume

welcome()	betterwelcome("Pablo")
-----------	------------------------

Will display:	Will display:
Hello and welcome.	Hello Pablo and welcon
Let's Learn about procedures.	Let's Learn about proce

Note that procedures **DO NOT** return a value

File Handling - Writing to a file

Adding a line of text to a file

```
myFile = openWrite("sample.txt")
myFile.writeline("Hello World")
myFile.close()
```

File Handling - Reading from a file

Reading and outputting a single line from the text file (see further details in CGP Pg 51)

```
myFile = openRead("sample.txt")
x = myFile.readLine()
myFile.close()
```

Reading and outputting the whole contents of a text file

```
myFile = openRead("sample.txt")
while NOT myFile.endOfFile()
    PRINT(myFile.readLine())
ENDWHILE
myFile.close()
```

Comparison operators

==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Arithmetic operators

+	Addition e.g. $x=6+5$ gives 11
-	Subtraction e.g. $x=6-5$ gives 1
*	Multiplication e.g. $x=12*2$ gives 24
/	Division e.g. $x=12/2$ gives 6
MOD	Modulus e.g. $12\text{MOD}5$ gives 2
DIV	Quotient e.g. $17\text{DIV}5$ gives 3
^	Exponentiation e.g. 3^4 gives 81

Boolean operators

AND	If two or more statements are true.
OR	If either statement is true.
NOT	To reverse the logical results of a statement.

Selection - if/else

Selection involves making decisions based on a comparison.

Comparison operators are used, sometimes with **boolean operators**.

```
IF entry == "A" THEN
    PRINT("You selected A")
ELSEIF entry == "B" THEN
    PRINT("You selected B")
ELSE:
    PRINT("Unrecognised selection")
ENDIF
```

Selection - switch/case

Selection involves making decisions based on a comparison.

Comparison operators are used, sometimes with **boolean operators**.

```
SWITCH entry:
    CASE "A":
        PRINT("You selected A")
    CASE "B":
        PRINT("You selected B")
    DEFAULT:
        PRINT("Unrecognised selection")
ENDSWITCH
```

Iteration - For Loop

FOR loops will repeat the code inside them a fixed number of times. The number of times that the code repeats will depend on an **initial value**, **end value**, and the **step count**.

Example:

```
FOR i = 0 to 7
    PRINT("Hello")
NEXT i
```

Will print hello 8 times (0-7 inclusive).

Iteration - Repeat Loop

This loop is controlled by a condition at the **end of the loop**. Keep going **until** the condition is **TRUE** (i.e. while it is false). Always runs the code inside it **at least once**. You get an **infinite loop** if the condition is **never true**.

Example: Write an algorithm that a supermarket self-scan machine could use to check if enough money has been fed into it and output the right amount of change.

```
VAR total as INTEGER
total = 0
VAR cost, coin, change as INTEGER
cost = total cost in pence
REPEAT
    coin = INPUT("Value of coin")
    total = total + coin
UNTIL total >= cost
change = total - cost
OUTPUT change
```

Iteration - While Loop

This loop is controlled by a condition at the **start of the loop**. Keep going **while** the condition is **TRUE** (i.e. until it is false). Never runs the code inside if condition is initially **false**. You get an **infinite loop** if the condition is **always true**.

Example: Write an algorithm that a supermarket self-scan machine could use to check if enough money has been fed into it and output the right amount of change.

```
VAR total as INTEGER
total = 0
VAR cost, coin, change as INTEGER
cost = total cost in pence
WHILE total < cost
    coin = INPUT("Value of coin")
    total = total + coin
ENDWHILE
change = total - cost
OUTPUT change
```

C

By **lcheong**
cheatography.com/lcheong/

Not published yet.
Last updated 30th April, 2018.
Page 3 of 5.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>

Iteration - Do While Loop

This loop is controlled by a condition at the **end of the loop**. Keep going **while** the condition is **TRUE** (i.e. until it is false). Always runs the code inside it **at least once**. You get an **infinite loop** if the condition is **always true**.

Example: Write an algorithm that a supermarket self-scan machine could use to check if enough money has been fed into it and output the right amount of change.

```
VAR total as INTEGER
total = 0
VAR cost, coin, change as INTEGER
cost = total cost in pence
DO
    coin = INPUT("Value of coin")
    total = total + coin
WHILE total < cost
OUTPUT change
```



By **lcheong**
cheatography.com/lcheong/

Not published yet.
Last updated 30th April, 2018.
Page 4 of 5.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>