**What is pseudocode?**

Pseudo-code is an alternative method of describing an algorithm that uses text instead of a diagram.

Pseudo-code can be thought of as a simplified form of programming code.

The prefix ‘pseudo’ means ‘false’ or ‘not genuine’.

Writing pseudo code allows us to lay down the logic of a problem in a “almost like real code” way without having to worry about the actual strict rules and syntax of a particular language.

**So what is this cheat sheet?**

This cheat sheet provides advice on the format in which you should write pseudocode when answering OCR exam questions. Practice makes perfect. Throughout your course practice producing pseudocode before you code up solutions to problems and it will become second nature in the exam.

**Rules for writing pseudo-code**

By definition, there are no hard and fast rules for writing pseudo-code, but certain guidelines will ensure that the algorithm is clear:

* Describe each step of the algorithm as briefly as possible.
* Use uppercase letters with keywords and other parts of the pseudo-code which are closer to a programming language.
* User lowercase letters with parts of the pseudo-code which are closer to English.
* If you use keywords to show the beginning and end of a block of code, then the code inside the block should be indented.

|  |  |  |
| --- | --- | --- |
| Concept | Example Pseudocode | Notes |
| Variables | x=3  name="Bob" | Variables are assigned using the = operator |
| global userid = 123 | Variables declared inside a function or procedure are assumed to be local to that subroutine.  Variables in the main program can be made global with the keyword global. |
| Casting | str(3) returns "3"  int("3") returns 3  float("3.14") returns 3.14 | Variables should be typecast using the int, str, and float functions. |
| Outputting to screen | PRINT("hello") | PRINT(string) |
| Taking input from user | name = INPUT("Please enter your name") | *Variable* = INPUT(*prompt to user*) |
| Iteration – Count controlled | FOR I = 0 to 7  PRINT("Hello")  NEXT i | This would print hello 8 times (0-7 inclusive). |
| Iteration – Condition controlled | WHILE answer != "computer”  answer = INPUT("What is the password?")  ENDWHILE | While Loop |
| DO  Answer = INPUT("What is the password?")  UNTIL answer == "computer" | Do Until Loop |
| Logical operators | WHILE x <=5 AND flag == FALSE | AND OR NOT |
| 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. 12MOD5 gives 2 |
| DIV | Quotient e.g. 17DIV5 gives 3 |
| ^ | Exponentiation e.g. 3^4 gives 81 |

|  |  |  |
| --- | --- | --- |
| Concept | Example Pseudocode | Notes |
| Selection | IF entry == "a" THEN  PRINT("You selected A")  ELSEIF entry == "b" then  PRINT("You selected B")  ELSE  PRINT("Unrecognised selection")  ENDIF | IF / ELSE selection |
| SWITCH ENTRY:  CASE "A":  PRINT("You selected A")  CASE "B":1  PRINT("You selected B")  DEFAULT:  PRINT("Unrecognised selection")  ENDSWITCH | SWITCH / CASE selection |
| String handling | stringname.LENGTH | To get the length of a string |
| stringname.SUBSTRING(startingPosition, numberOfCharacters) | To get a substring |
| Subroutines | FUNCTION triple(number)  RETURN number \* 3  ENDFUNCTION  Called from main program  Y =triple(7) | Function |
| PROCEDURE greeting(name)  PRINT("hello" + name)  ENDPROCEDURE  Called from main program  greeting("Hamish") | Procedure |
| PROCEDURE foobar(x:**byVal**, y:**byRef**)  …  …  ENDPROCEDURE | Unless stated values passed to subroutines can be assumed to be passed by value in the exam.  If this is relevant to the question **byVal** and **byRef** will be used. In the case shown here x is passed by value and y is passed by reference. |

|  |  |  |
| --- | --- | --- |
| Concept | Example Pseudocode | Notes |
| Arrays / Lists | ARRAY names[5]  names[0] = "Ahmad"  names[1] = "Ben"  names[2] = "Catherine"  names[3] = "Dana"  names[4] = "Elijah"  PRINT(names[3]) | Arrays should be 0 based and declared with the keyword array. |
| ARRAY board[8,8]  board[0,0] = "rook" | Example of 2D array |
| Reading to and writing from files | myFile = **OPENREAD**("sample.txt")  x = myFile.**READLINE**()  myFile.**CLOSE**() | To open a file to read you should use OPENREAD.  READLINE should be used to return a line of text from the file.  The example on the left makes x the first line of sample.txt |
| ENDOFFILE() | This is used to determine if the end of a file has been reached. |
| myFile = OPENREAD("sample.txt")  WHILE NOT myFile.**ENDOFFILE**()  PRINT(myFile.READLINE())  ENDWHILE  myFile.CLOSE() | The example on the left will print out the contents of sample.txt |
| myFile = **OPENWRITE**("sample.txt")  myFile.**WRITELINE**("Hello World")  myFile.CLOSE() | To open a file to write to openWrite is used and writeLine to add a line of text to the file. In the program below hello world is made the contents of sample.txt (any previous contents are overwritten). |
| Comments | PRINT("Hello World") //This is a comment | Comments are denoted by // |

|  |  |  |
| --- | --- | --- |
| Concept | Example Pseudocode | Notes |
| Methods and attributes | PUBLIC and PRIVATE  PRIVATE attempts = 3  PUBLIC PROCEDURE setAttempts(number)  attempts = number  ENDPROCEDURE  PRIVATE FUNCTION getAttempts()  RETURN attempts  END FUNCTION | Methods and attributes can be assumed to be public unless otherwise stated.  Where the access level is relevant to the question it will always be explicit in the code denoted by the keywords. |
| player.setAttempts(5)  PRINT(player.getAttempts()) | Methods should always be instance methods, you are not expected to be aware of static methods. You should call them using object.method as shown on the left. |
| Constructors and inheritance | CLASS Pet  PRIVATE name  PUBLIC PROCEDURE **NEW**(givenName)  Name = givenName  ENDPROCEDURE  ENDCLASS | You should write constructors as you would procedures with the name new |
| SUPER.methodName(parameters) | You should show Inheritance by using the keyword *inherits* keyword  *Superclass* methods should be called with the keyword *super*. |
| CLASS dog **INHERITS** Pet  PRIVATE breed  PUBLIC PROCEDURE NEW(givenName, givenBreed)  SUPER.NEW(givenName)  Breed = givenBreed  ENDPROCEDURE  ENDCLASS | In the case of the constructor the pseudocode would look like the example on the left. |
| objectName = NEW className(parameters)  e.g.  myDog = NEW Dog("Fido","Scottish Terrier") | To create an instance of an object the following format is used |