Computer Science
First Exams 2014

Pseudocode in Examinations

- ***Standard Data Structures***
- ***Examples of Pseudocode***

*Candidates are **NOT** allowed a copy of this document during their examinations.*

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

Introduction

The purpose of this document is to show the standard data structures and methods which may be displayed in IB Diploma Programme Computer Science examinations.

These methods, in their pseudocode format, may be used without explanation or clarification in examination questions. Teachers should ensure that candidates will be able to interpret these methods when presented as part of an examination question.

This list is not exhaustive. Other methods may be used in examination questions; however any additional methods will be fully explained within the examination.

This information is supported by a series of pseudocode examples, demonstrating how the pseudocode will be formatted and displayed during IB examinations.

Where answers are to be written in pseudocode, the examiners will be looking for clear algorithmic thinking to be demonstrated. In examinations, this type of question will be written in the approved notation, so a familiarity with it is expected.

It is accepted that under exam conditions candidates may, in their solutions, use pseudocode similar to a programming language with which they are familiar. This is acceptable. The markscheme will be written using the approved notation. Provided the examiners can see the logic in the candidate's response, regardless of language, it will be credited.

No marks will be withheld for syntax errors.

Candidates are not permitted to invoke a powerful command if it trivializes the question. For example, in response to a question asking the candidate to "*Construct an algorithm to arrange the elements of an array in increasing order*", the pseudocode "*sort the array*" is inappropriate and will receive no marks.

Arrays

An array is an indexed and ordered set of elements. Unless specifically defined in the question, the index of the first element in an array is 0.

```
NAMES[0]  // The first element in the array NAMES
```

Strings

A string can contain a set of characters, or can be empty. Strings can be used like any other variable.

```
MYWORD = "This is a string"
if MYWORD = "the" then
    output MYWORD
end if
```

Strings should be regarded as a class of objects. However no methods belonging to that class are part of this standard. Instead, if a specialized method such as `charAt()` or `substring()` is to be used in an examination, it will be fully specified as part of the question in which it is needed.

Collections

Collections store a set of elements. The elements may be of any type (numbers, objects, arrays, Strings, etc.).

A collection provides a mechanism to iterate through all of the elements that it contains. The following code is guaranteed to retrieve each item in the collection exactly once.

```
// STUFF is a collection that already exists
STUFF.resetNext()
loop while STUFF.hasNext()
    ITEM = STUFF.getNext()
    // process ITEM in whatever way is needed
end loop
```

Method name	Brief description	Example: HOT, a collection of temperatures	Comment
`addItem()`	Add item	`HOT.addItem(42)` `HOT.addItem("chile")`	Adds an element that contains the argument, whether it is a value, String, object, etc.
`getNext()`	Get the next item	`TEMP = HOT.getNext()`	`getNext()` will return the first item in the collection when it is first called. Note: `getNext()` does not remove the item from the collection.
`resetNext()`	Go back to the start of the collection	`HOT.resetNext()` `HOT.getNext()`	Restarts the iteration through the collection. The two lines shown will retrieve the first item in the collection.
`hasNext()`	Test: has next item	`if HOT.hasNext() then`	Returns TRUE if there are one or more elements in the collection that have not been accessed by the present iteration: The next use of `getNext()` will return a valid element.
`isEmpty()`	Test: collection is empty	`if HOT.isEmpty() then`	Returns TRUE if the collection does not contain any elements.

Higher Level Only

Stacks

A stack stores a set of elements in a particular order: Items are retrieved in the reverse order in which they are inserted (Last-in, First-out). The elements may be of any type (numbers, objects, arrays, Strings, etc.).

Method name	Brief description	Example: OPS, a stack of integers	Comment
`push()`	Push an item onto the stack	`OPS.push(42)`	Adds an element that contains the argument, whether it is a value, String, object, etc. to the top of the stack.
`pop()`	Pop an item off the stack	`NUM = OPS.pop()`	Removes and returns the item on the top of the stack.
`isEmpty()`	Test: stack contains no elements	`if OPS.isEmpty() then`	Returns TRUE if the stack does not contain any elements.

Queues

A queue stores a set of elements in a particular order: Items are retrieved in the order in which they are inserted (First-in, First-out). The elements may be of any type (numbers, objects, arrays, Strings, etc.).

Method name	Brief description	Example: WAIT, a queue of Strings	Comment
`enqueue()`	Put an item into the end of the queue	`WAIT.enqueue("Mary")`	Adds an element that contains the argument, whether it is a value, String, object, etc. to the end of the queue.
`dequeue()`	Remove an item from front of the queue	`CLIENT = WAIT.dequeue()`	Removes and returns the item at the front of the queue.
`isEmpty()`	Test: queue contains no elements	`if WAIT.isEmpty() then`	Returns TRUE if the queue does not contain any elements.

Examples of Pseudocode

AVERAGING AN ARRAY

The array STOCK contains a list of 1000 whole numbers (integers). The following pseudocode presents an algorithm that will count how many of these numbers are non-zero, adds up all those numbers and then prints the average of all the non-zero numbers (divides by COUNT rather than dividing by 1000).

```
COUNT = 0
TOTAL = 0

loop N from 0 to 999
  if STOCK[N] > 0 then
    COUNT = COUNT + 1
    TOTAL = TOTAL + STOCK[N]
  end if
end loop

if NOT COUNT = 0 then
  AVERAGE = TOTAL / COUNT
  output "Average = " , AVERAGE
else
  output "There are no non-zero values"
end if
```

COPYING FROM A COLLECTION INTO AN ARRAY

The following pseudocode presents an algorithm that reads all the names from a collection, NAMES, and copies them into an array, LIST, but eliminates any duplicates. That means each name is checked against the names that are already in the array. The collection and the array are passed as parameters to the method.

```
COUNT = 0   // number of names currently in LIST

loop while NAMES.hasNext()

  DATA = NAMES.getNext()

  FOUND = false
  loop POS from 0 to COUNT-1
    if DATA = LIST[POS] then
      FOUND = true
    end if
  end loop

  if FOUND = false then
    LIST[COUNT] = DATA
    COUNT = COUNT + 1
  end if
end loop
```

FACTORS

The following pseudocode presents an algorithm that will print all the factors of an integer. It prints two factors at a time, stopping at the square root. It also counts and displays the total number of factors.

```
// recall that
//   30 div 7 = 4
//   30 mod 7 = 2

NUM = 140  // code will print all factors of this number
F = 1
FACTORS = 0

loop until F*F > NUM  //code will loop until F*F is greater than NUM

  if NUM mod F = 0 then

    D = NUM div F
    output NUM , " = " , F , "*" , D

    if F = 1 then
      FACTORS = FACTORS + 0
    else if F = D then
      FACTORS = FACTORS + 1
    else
      FACTORS = FACTORS + 2
    end if

  end if

  F = F + 1

end loop

output NUM , " has " , FACTORS , " factors "
```

COPYING A COLLECTION INTO AN ARRAY IN REVERSE

The following pseudocode presents an algorithm that will read all the names from a collection, SURVEY, and then copy these names into an array, MYARRAY, in reverse order.

```
// MYSTACK is a stack, initially empty

COUNT = 0 // number of names

loop while SURVEY.hasNext()
  MYSTACK.push( SURVEY.getNext() )
  COUNT = COUNT + 1
end loop

// Fill the array, MYARRAY, with the names in the stack

loop POS from 0 to COUNT-1
  MYARRAY[POS] = MYSTACK.pop()
end loop
```