

# IB Pseudocode Syntax

## Basic instructions

Name	Syntax	Description	Examples
Assign	<i>variable = value</i>	Assigns a <b>value</b> to the <b>variable</b>	<b>a = 1</b>
Input	input <i>variable</i>	Inputs the <b>variable</b>	input <b>a</b>
Output	output <i>variable/expression</i>	Outputs a <b>value</b> of the <b>variable</b> or <b>expression</b>	output <b>a</b> output <b>"hello"</b> output <b>2 + 2</b>
Create	create <i>type variable</i>	Creates the <b>variable</b> with standard <b>value</b> of given <b>type</b>	create <b>Boolean a</b> create <b>Number b</b> create <b>String c</b>
Delete	delete <i>variable</i>	Deletes the <b>variable</b>	delete <b>a</b>

## Conditions

Name	Syntax	Description	Examples
If	if <i>condition</i>	Indicates the start of a <b>condition</b> block and states the first <b>condition</b>	if <b>a = 1</b>
Else if	else if <i>condition</i>	States an additional <b>condition</b>	else if <b>a = 2</b>
Else	else	Indicates the start of the part of a <b>condition</b> block which will be executed if all <b>conditions</b> above are <b>false</b>	else
End if	end if	Indicates the end of a <b>condition</b> block	end if

## Loops

Name	Syntax	Description	Examples
While loop	loop while <i>condition</i>	Executes a loop block while the <b>condition</b> is <b>true</b>	loop while <b>a &lt; 5</b>
Until loop	loop until <i>condition</i>	Executes a loop block until the <b>condition</b> is <b>true</b>	loop until <b>a = 5</b>
For loop	loop <i>variable</i> from <i>start value</i> to <i>end value</i>	Executes a loop block for every <b>value</b> of the <b>variable</b> between <b>start value</b> and <b>end value</b>	loop <b>a</b> from <b>1</b> to <b>5</b>
End loop	end loop	Indicates the end of a <b>loop</b> block	end loop

## Operators

Name	Syntax	Description	Examples
Equal	<code>value1 = value2</code>	Checks is the first <code>value</code> equal to the second	<code>a = 1</code>
Not equal	<code>value1 != value2</code> <code>value1 &lt;&gt; value2</code>	Checks is the first <code>value</code> not equal to the second	<code>a != 1</code> <code>a &lt;&gt; 1</code>
Greater	<code>value1 &gt; value2</code>	Checks is the first <code>value</code> greater than the second	<code>a &gt; 1</code>
Greater or equal	<code>value1 &gt;= value2</code>	Checks is the first <code>value</code> greater or equal to the second	<code>a &gt;= 1</code>
Less	<code>value1 &lt; value2</code>	Checks is the first <code>value</code> less than the second	<code>a &lt; 1</code>
Less or equal	<code>value1 &lt;= value2</code>	Checks is the first <code>value</code> less or equal to the second	<code>a &lt;= 1</code>
Not	<code>NOT value1</code>	Executes logical or bitwise <code>NOT</code> for the <code>value</code>	<code>NOT a</code>
And	<code>value1 AND value2</code>	Executes logical or bitwise <code>AND</code> for the first and the second <code>values</code>	<code>a AND 1</code>
Or	<code>value1 OR value2</code>	Executes logical or bitwise <code>OR</code> for the first and the second <code>values</code>	<code>a OR 1</code>
Xor	<code>value1 XOR value2</code>	Executes bitwise <code>XOR</code> for the first and the second <code>values</code>	<code>a XOR 1</code>
Addition	<code>value1 + value2</code>	Adds the first and the second <code>values</code>	<code>a + 1</code>
Subtraction	<code>value1 - value2</code>	Subtracts the first and the second <code>values</code>	<code>a - 1</code>
Multiplication	<code>value1 * value2</code>	Multiplies the first and the second <code>values</code>	<code>a * 1</code>
Division	<code>value1 / value2</code>	Divides the first and the second <code>values</code>	<code>a / 1</code>
Modulo	<code>value1 mod value2</code>	Gets <code>modulo</code> of the first and the second <code>values</code>	<code>a mod 1</code>
Integer division	<code>value1 div value</code>	Gets integer part of the division of the first and the second <code>values</code>	<code>a div 1</code>

## Functions

Name	Syntax	Description	Examples
Function	function <i>name</i> ( <i>arg1</i> , ...)	Indicates the start of a function block with <i>name</i> and arguments	function f(a, b)
Return	return <i>variable/expression</i>	Returns <i>value</i> or <i>expression</i> from function	return a return "hello" return 2 + 2
End function	end function	Indicates the end of a function block	end function
Run function	<i>name</i> ( <i>arg1</i> , ...)	Runs a function block with given <i>name</i> and arguments	f(1, 2)

## Procedures

Name	Syntax	Description	Examples
Procedure	procedure <i>name</i> ( <i>arg1</i> , ...)	Indicates the start of a procedure block with <i>name</i> and arguments	procedure p(a, b)
End procedure	end procedure	Indicates the end of a procedure block	end procedure
Run procedure	<i>name</i> ( <i>arg1</i> , ...)	Runs a procedure block with given <i>name</i> and arguments	p(1, 2)

## Basic data types

Name	Syntax	Description	Examples
Boolean	true/false create Boolean <i>variable</i>	Boolean type that can contain only true or false values	a = true b = false create Boolean c
Number	<i>number</i> create Number <i>variable</i>	Number type that can contain any number value	a = 0 b = 1 create Number c
String	"string" create String <i>variable</i>	String type that can contain any text	a = "hello" create String b

## Arrays

Name	Syntax	Description	Examples
Create array	create Array <i>name</i>	Creates an empty array with given <i>name</i>	create Array a
Get item	<i>name</i> [ <i>index</i> ]	Returns an item with given <i>index</i> from the array	a[0]
Set item	<i>name</i> [ <i>index</i> ] = <i>value</i>	Assigns a <i>value</i> to given <i>index</i> from the array	a[0] = 1
Array size	<i>name</i> .size()	Returns a size of the array	a.size()
Assign array	<i>name</i> = [ <i>val1</i> , <i>val2</i> , ...]	Assigns an array with given <i>values</i> to a <i>variable</i>	a = [1, 2, 3]

## Dictionaries

Name	Syntax	Description	Examples
Create dictionary	create Dictionary <i>name</i>	Creates an empty dictionary with given <i>name</i>	create Dictionary a
Get item	<i>name</i> ["key"]	Returns an item with given <i>key</i> from the dictionary	a["a"]
Set item	<i>name</i> ["key"] = <i>value</i>	Assigns a <i>value</i> to given <i>key</i> from a dictionary with given <i>name</i>	a["a"] = 1
Assign dictionary	<i>name</i> = {"key1": <i>val1</i> , "key2": <i>val2</i> , ...}	Assigns a dictionary with given <i>values</i> to a <i>variable</i>	a = {"a": 1, "b": 2}

## Collections

Name	Syntax	Description	Examples
Create collection	create Collection <i>name</i>	Creates an empty collection with given <i>name</i>	create Collection <i>a</i>
Add item	<i>name</i> .addItem( <i>value</i> )	Adds a <i>value</i> to the end of the collection	<i>a</i> .addItem(1)
Get next	<i>name</i> .getNext()	Returns next value from the collection	<i>a</i> .getNext()
Reset next	<i>name</i> .resetNext()	Resets next element of the collection	<i>a</i> .resetNext()
Has next	<i>name</i> .hasNext()	Checks does the collection have next element	<i>a</i> .hasNext()
Is empty	<i>name</i> .isEmpty()	Check does the collection contains elements	<i>a</i> .isEmpty()
Collection size	<i>name</i> .size()	Returns a size of the collection	<i>a</i> .size()

## Stacks

Name	Syntax	Description	Examples
Create stack	create Stack <i>name</i>	Creates an empty stack with given <i>name</i>	create Stack <i>a</i>
Push	<i>name</i> .push( <i>value</i> )	Adds a <i>value</i> to the stack	<i>a</i> .push(1)
Pop	<i>name</i> .pop()	Gets a <i>value</i> from the stack	<i>a</i> .pop()
Is empty	<i>name</i> .isEmpty()	Check does the stack contains elements	<i>a</i> .isEmpty()
Stack size	<i>name</i> .size()	Returns a size of the stack	<i>a</i> .size()

## Queues

Name	Syntax	Description	Examples
Create queue	create Queue <i>name</i>	Creates an empty queue with given <i>name</i>	create Queue <i>a</i>
Enqueue	<i>name</i> .enqueue( <i>value</i> )	Adds a <i>value</i> to the queue	<i>a</i> .enqueue(1)
Dequeue	<i>name</i> .dequeue()	Gets a <i>value</i> from the queue	<i>a</i> .dequeue()
Is empty	<i>name</i> .isEmpty()	Check does the queue contains elements	<i>a</i> .isEmpty()
Queue size	<i>name</i> .size()	Returns a size of the queue	<i>a</i> .size()