Pseudocode/Flowchart/Algorithms Cheat Sheet

Standard Structure

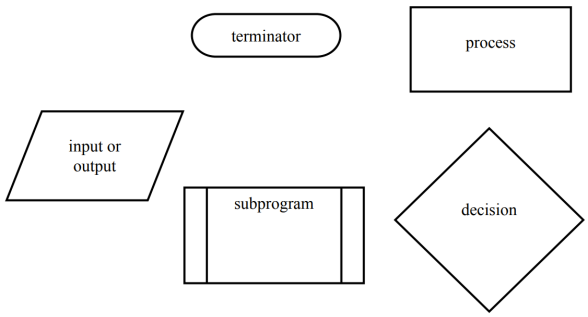
```
Unset
BEGIN
	algorithm
END

Example

BEGIN
	var = user input
	IF var = 10 THEN
		var += 1
	ELSE
		var -= 1
	ENDIF
END
```

BEGIN MAINPROGRAM
process 1
process 2
process 3
process 4
END MAINPROGRAM

BEGIN process 2
do this
do that
END process 2

Flowchart elements

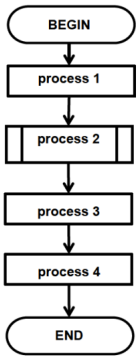
Flowcharts use the following symbols connected by lines with arrowheads to indicate the flow. It is common practice to show arrowheads to avoid ambiguity.



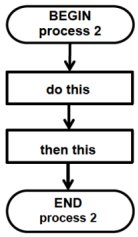
Subroutines

```
Unset
BEGIN name(parameters)
       algorithm
END name

Example

BEGIN InputAdder()
       var = userinput
       IF var = 10 THEN
              var += 1
       ELSE
              var -= 1
       ENDIF
END InputAdder
```

Flowcharts




```
BEGIN MAINPROGRAM
  Get First
  Get Second
  Biggest (First, Second, Result)
  Display Result
END MAINPROGRAM

BEGIN Biggest(Item1,Item2, Big)
  IF Item1 > Item2 THEN
    Big = Item1
  ELSE
     Big = Item2
  ENDIF
END Biggest
```



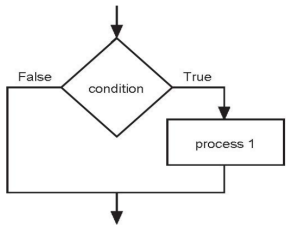
IF Statements

```
Unset
IF condition THEN
```

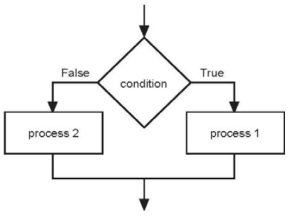
```
        statements
ELSE
        statements
ENDIF

Example

IF var = 10 THEN
        var += 1
ELSE
        var -= 1
ENDIF
```

Flowchart

1.



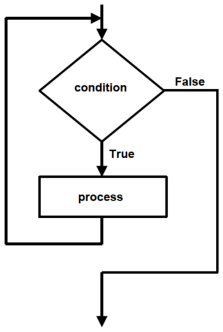
2.



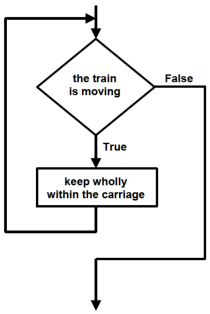
While loops

```
Unset
WHILE conditon
        statements

ENDWHILE

Example

WHILE var < 10
        var += 1
ENDWHILE
```

Flowchart



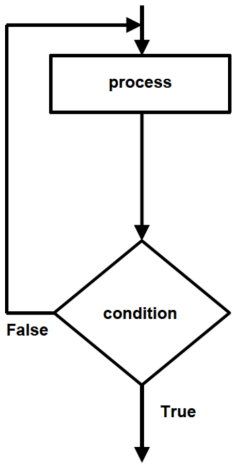
Flowchart

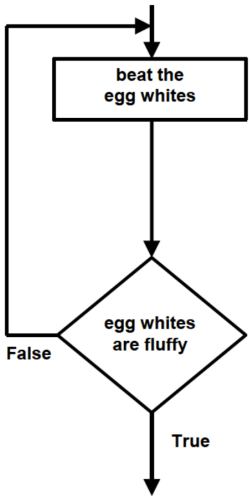


Post test repetition

```
Unset
REPEAT statements
UNTIL condition

Example

var = 1
REPEAT
      Print var
      var += 1
UNTIL var = 10
```

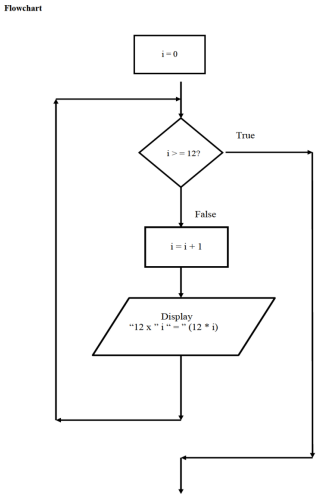
Flowchart





For/Next loops

```
Unset
FOR variable = start TO finish STEP increment
       statements
NEXT variable

Example

FOR var = 1 TO 10 STEP 1
       print var
NEXT var
```

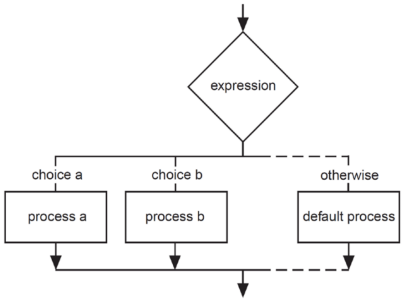


Casewhere

```
Unset
CASEWHERE expression evaluates to
       A: process A
       B: process B
OTHERWISE: process ...
ENDCASE

Example

CASEWHERE signal is
       red : stop the vehicle
       amber : stop the vehicle
       green : proceed through the intersection
OTHERWISE : proceed with caution
```

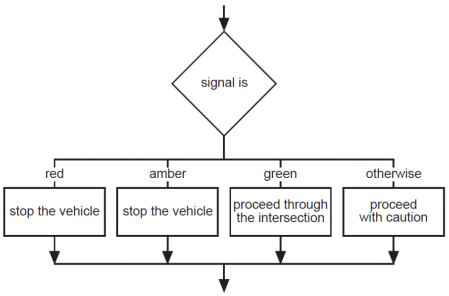
```
ENDCASE
```

Flowchart



Note: As the flowchart version of the multi-way selection indicates, **only one** process on each pass is executed as a result of the implementation of the multi-way selection.

Flowchart



Records

```
Unset
```

Arrays

Finding a maximum value in an array

The value in the first element is stored in a temporary variable called Max. Each element is then considered in turn to determine if its value is larger than that stored value. If so, the value in Max is replaced by this larger value, and the index of this element is stored in a temporary variable called MaxIndex.

When all elements have been considered, Max will contain the largest value, and MaxIndex will contain the index of the largest element.

```
BEGIN FindMAX
        Let Max = Element (1)
        Let MaxIndex = 1
        Let i = 2
        REPEAT
                IF Element (i) > Max THEN
                        Let Max = Element (i)
                        Let MaxIndex = i
                END IF
                Let i = i + 1
        UNTIL i > NumElementsInArray
        Display "The highest value is " Max " at position " MaxIndex
END FindMAX
```

Load an array and print its contents

An array can be loaded from data values or input from the keyboard. The following algorithm assumes that values are read from a list of data statements until a sentinel value of "xxx" is encountered.

```
BEGIN LoadArray
        Let i = 1
        Read DataValue
        WHILE DataValue <> "xxx"
                Let Element (i) = DataValue
                i = i + 1
                Read DataValue
        ENDWHILE
        Let NumElements = i
        Display " There are" NumElements " items loaded into the array"
END LoadArray
```

Note the use of a priming read to ensure that the sentinel value is not loaded into the array. The pre-test loop ensures that if there is no data in the list of data to be loaded (other than the sentinel value) then the loop will never be entered.

To print the array, it is assumed that there is a variable that stores the number of elements in the array.

```
BEGIN PrintArrayContents
        Let i = 1
        REPEAT
                Display Element (i)
                i = i + 1
        UNTIL i > = NumElements
END PrintArrayContents
```

Note that if the number of elements is not known, the sentinel value would be stored into the last element in the array, and printing would continue until the sentinel value of "xxx" is encountered.

Multidimensional arrays

To access each element in a multidimensional array, we use a series of nested FOR / NEXT loops:

```
BEGIN PrintProductSalesAndTotal
        FOR Town = 1 to 6
                FOR Month = 1 to 12
                        FOR Product = 1 to 4
                                Display "Town" Town, "Month" Month, "Product" Product;
                                Display Sales (Town, Month, Product)
                                Add Sales (Town, Month, Product) to Total
                        NEXT Product
                NEXT Month
        NEXT Town
        Display "Total sales across all towns for all products sold this year =" Total
END PrintProductSalesAndTotal
```

Files

Random Access/Relative Files

Creating a relative file

Relative files need to be opened for relative access when they are created and must be closed before the program ends. All records are accessed through the use of a key which specifies the relative position of that record within the file. The key field used must contain positive integer values only. There is no sentinel value written as the file is not accessed sequentially.

The following example writes 10 records to a relative file called ProductData. The data is entered by the user from the keyboard and the products are entered in no particular order:

```
BEGIN CreateARelativeFile
        Open ProductData for relative access
        FOR i = 1 to 10
                Display "Please enter the details for the next product: "
                Get ProdNumber, description, quantity, price
                Write ProductData from ProdNumber, description, quantity, price using
                ProdNumber

                'note the use of the variable ProdNumber as the key field, specifying where this
                record will be written in the file.
        NEXT i

        Close ProductData
END CreateARelativeFile
```

Opening Files

```
Unset
open filename for relative access

where: filename is the name of the file to be used

Example

open animalData for relative access
```

Reading from file

```
Unset
Read filename into field1, …, fieldN using key

where: field1, …, fieldN are fields of a record
key specifies the relative position of that record within the file

Examples

Read animalData into animalNumber, name, species using animalNumber
```

Writing to a file

```
Unset
Write filename from field1, …, fieldN using key

Example

Write animalData from animalNumber,name, species using animalNumber
```

Closing a file

```
Unset
close filename

where: filename is the name of the file previously opened

Example

close animalData
```

Sequential

Opening Files

```
Unset
open filename for method

Methods
input (read from file)
output (write to file – overwrites existing contents)
```

```
append (add to the end)

Example

open animalData for input
```

Reading from a file

```
Unset
Read field1, …, fieldN from filename

where: field1, …, fieldN are fields of a record in the file
Note 1: The file must have been opened for input
Note 2: It is also possible to read in data one character at a time.

Example

Read name, species from animalData
```

Writing to a file

```
Unset
Write filename from field1, …, fieldN

where: field1, …, fieldN are variables in the program
Note: The file must have been opened for output or append.

Example

Write animalData from name, species
```

Closing a file

```
Unset
close filename

where: filename is the name of the file previously opened

Example
```

```
close animalData
```

Sources ish

Page 58 to 74 in https://drive.google.com/file/d/1BFhc0upgU3f_gcd5juW2HvqRXmYJ1bxY/view

Pseudocode

Pseudocode uses English-like statements with defined rules of structure and keywords.

Pseudocode guidelines

The pseudocode keywords are:

for each procedure or subroutine

BEGIN *name*
END *name*

for binary selection

IF *condition* THEN
    statements
ELSE
    statements
ENDIF

for multi-way selection

CASEWHERE expression evaluates to
    A: process A
    B: process B
……..
OTHERWISE: process …
ENDCASE

for pre-test repetition

WHILE *condition*
    statements
ENDWHILE

for post-test repetition

REPEAT
    statements
UNTIL *condition*

For FOR / NEXT loops

FOR *variable* = *start* TO *finish* STEP *increment*
    statements
NEXT *variable*