

Flowcharts, pseudocode, and program documentation

A Flowchart is a flow diagram serving as map of how programmer intends to solve a particular problem, showing the logical steps required, the decisions to be reached, and the paths to follow: in other words, it provides a roadmap of the project's algorithm. Flowcharts historically have served as an important part of the "documentation" for a program. A flowchart can assist you greatly in debugging programs during their development in Visual Basic.

The flowchart illustrates major sections or pieces of program to be worked on separately, and lends itself to proper use of subroutines, library calls, and other preprogrammed pieces. Flowcharts can be very useful when trying to debug a program.

Different levels of detail can be provided in a flowchart: a high level flowchart has relatively few symbols, and provides a broad overview of program operation. A low level flowchart provides more richness in detail, and can give almost a 1:1 correspondence to the lines of code in the final program.

Specific flowchart symbols represent basic information processing steps, input/output, processing, branchpoints, flow direction, and annotation. Here are some of the major symbols used:

Input/output symbol: indicates data read into memory or passed out of memory; the symbol is a parallelogram. Use of this element indicates a call for keyboard entry via INPUT, a data READ statement, an INPUT or READ from disk files, etc.



Processing symbol: rectangle. This symbol implies execution of a defined operation or group of operations, resulting in changed values of variables, changed storage locations, etc. Example: create an array of 100 random numbers.



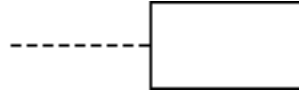
Branchpoints symbol: this is used in the logic of a program to make a decision to take one of two or more alternative pathways, i.e., test a condition to make decision. The decision symbol is most commonly a diamond, but an alternate symbol used is a rectangle with initial, increment and maximum values enclosed.



Flowlines: these are the connectors which link other symbols to show the sequence of executed operations; the normal flow is expected to be left to right, and top to bottom; otherwise, arrows are required to move from right to left, or bottom to top. If there is a break in continuity of the flowchart (say, at the bottom of the page), a connector symbol can be used to show where the flow proceeds onto the next page. Flowlines may cross without interaction, or may join from several sources.



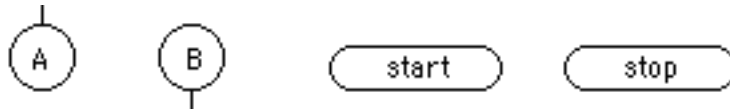
Annotation symbol: this symbol allows inclusion of descriptive comments and explanatory notes for clarification. They may be drawn to the left or to the right of the flowchart symbols.



Specialized I/O symbols (optional): can include symbols for a keyboard; for punched card (or mark sense cards, etc.); magnetic tape; magnetic disk; printed output on paper; output to a display screen or plotter; a telecommunications link; on online storage to a hard disk.



Other miscellaneous symbols include various connectors, circles (from page to page, or on same page to avoid long flowlines); terminal - box with rounded corners, or ellipse (start, stop, halt, pause).



Here is a simple flowchart example: write a program to determine the largest value of a set of N integers (where $N > 2$):

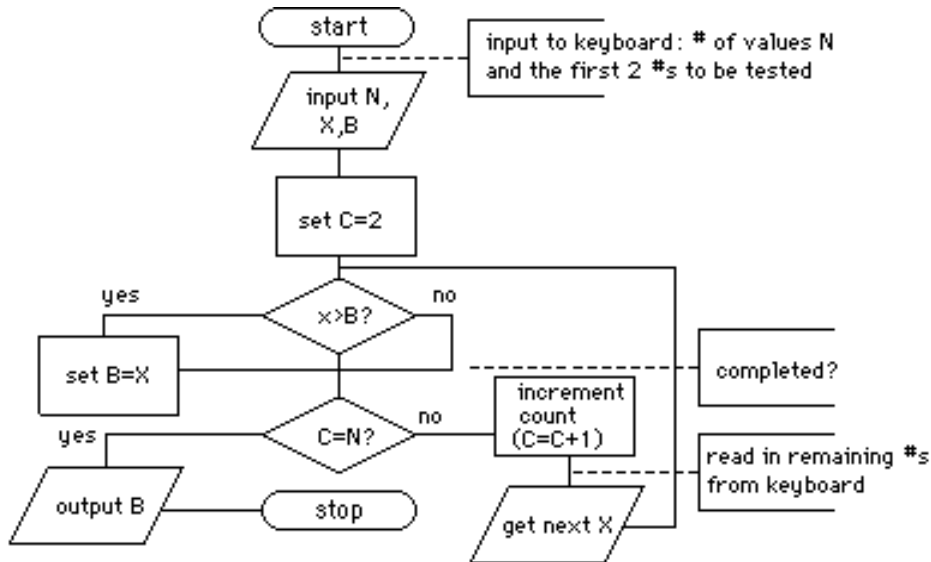
Here's a first crude description of the algorithm in pseudocode (see below):

input N (the number of integers to be examined)
 input each of the N integers one by one, keeping the largest so far.
 after N integers are inputted, finally outputting the largest one saved

Here's a slightly more detailed description of the algorithm:

input N (the number of integers to be examined)
 input first integer, save it as X
 input second integer, call it B
 keep count of how many values were inputted: set a counter $C = 2$
 compare each value X with B , then save the biggest as B
 input each of the remaining values: after N integers are inputted, C will = N
 output the largest value, which was saved as B

Next, draw the flowchart:



Finally, write the code. A generic Basic program might look like this:

```

INPUT "Number of values to be entered: ":N
INPUT "Enter first number: ":X
INPUT "Enter next number: ":B
LET C=2
IF X>B THEN B=X
IF C<N THEN
LET C=C+1
INPUT "Enter next number: ":X
ELSE
PRINT "The largest number was ":B
END IF
END
  
```

In Visual Basic, a reasonable design of the form for such a program might be the one shown below:

The corresponding code might look like this

```
Private Sub Command1_Click()
List1.AddItem Text1
End Sub
Private Sub Command2_Click()
Dim intX As Integer, intB As Integer
Dim intCtr As Integer, intNoOfItems As Integer
intNoOfItems = List1.ListCount
intX = Val(List1.List(1))
intB = Val(List1.List(2))
For intCtr = 3 To intNoOfItems
If intX > intB Then intB = intX
intX = Val(List1.List(intCtr))
Next intCtr
Text2 = intB
End Sub
```

Pseudocode description of an algorithm: pseudocode serves the same value as a flowchart, but is written as text rather than as diagram. Pseudocode consists of a sequence of statements that express how data is to be manipulated.

There are three basic control structures used: these are sufficient to write out any algorithm in a systematic way in a structured programming fashion. The resultant "code" looks very similar to Visual Basic.

Notations:

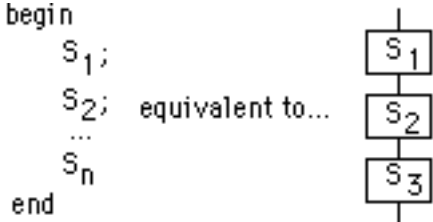
"type declaration" specifies type of data: eg. Integer B,X,N, Real A,B

"assignment": a variable \leftarrow expression eg. $B \leftarrow X$

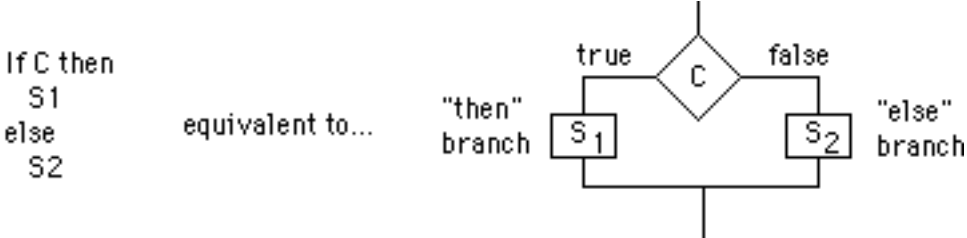
input, output: e.g. Output B

Basic control structures:

1. **Sequence:** the number of statements to be executed in order; use two delimiters: begin, end.



2. **Decision:** two or more alternative courses of action, depending on existence of some condition. "If this is true, do this, otherwise, do that"



3. **Loop:** repeat execution of a sequence of statements while a certain condition is true: requires that condition reverse after a while, otherwise it will be an "infinite loop."



The choice of amount of detail in each step is left to discretion of programmer. Actually, however, a flowchart allows reader to follow logic of algorithm more easily than a linear description in English.

Example of a program algorithm in pseudocode.

Problem: In s set of integers N, the positive numbers (including 0) and the negative numbers are to be separately counted.

Algorithm: use "decision" and "loop" control structures:

1. if the integer is greater than or equal to zero, then increment the counter of positive integers by 1
- else increment the counter of negative integers by 1
2. while fewer than N integers have been read in, do
 - begin read in the next integer
 - increment the appropriate counter by 1
 - increment the counter of integers read so far by 1
 - end

Here's the pseudocoded program:

```
begin
  Integer N, Poscnt, Negcnt, I; Num
  Input N
  Poscnt <-- 0 *Initialize counters
  Negcnt <-- 0
  1 <-- 0
  While I < N do Now count
    begin
      I <-- I + 1
      Input Num
      if Num ≥ 0 then
        Poscnt <-- Poscnt + 1
      else
        Negcnt <-- Negcnt + 1
      end
    end
  Output Poscnt, Negcnt
end
```

Note that * indicates comment in pseudocoded program.