







Flowchart Symbol



Basics of Flow chart

Flowchart is a diagrammatic representation of an algorithm. Flowchart are very helpful in writing program and explaining program to others.

Symbols Used In Flowchart

Different symbols are used for different states in flowchart, For example: Input/Output and decision making has different symbols. The table below describes all the symbols that are used in making flowchart

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Used to represent start and end of flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for airthmetic operations and data-manipulations.
	Desicion	Used to represent the operation in which there are two alternatives, true and false.
	On-page Connector	Used to join different flowline

Symbol	Purpose	Description
	Off-page Connector	Used to connect flowchart portion on different page.
	Predefined Process/Function	Used to represent a group of statements performing one processing task.

Flowchart Symbols and their usage

This is an overview of all the flowchart symbols that you will use when drawing flowcharts and process flow. All these objects are available in Creately and you can try out a [demo](#) or take a look at some [sample flowcharts](#) for more context.

[Terminal / Terminator](#)



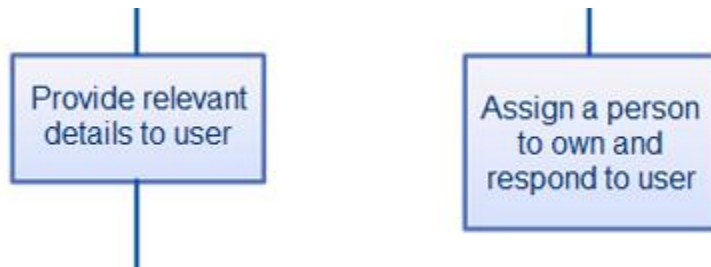
The terminator is used to show where your flow begins or ends. Ideally, you would use words like 'Start', 'Begin', 'End' inside the terminator object to make things more obvious.



[Process / Rectangle](#)



Flowchart Process object is used to illustrate a process, action or an operation. These are represented by rectangles; and the text in the rectangle mostly includes a verb. Examples include 'Edit video', 'Try Again', 'Choose your Plan'.



Data (I/O)



The Data object, often referred to as the I/O Shape shows the Inputs to and Outputs from a process. This takes the shape of a parallelogram.

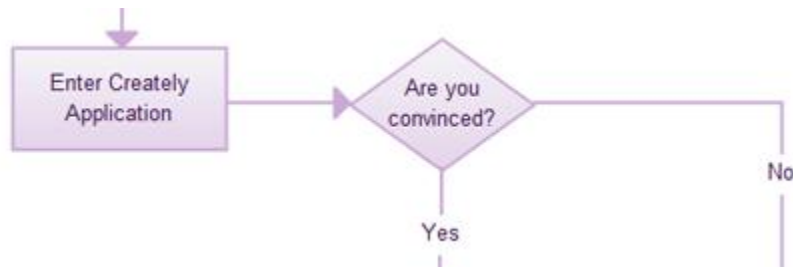


Decision / Conditional



Decision object is represented as a Diamond. This object is always used in a process flow to as a question. And, the answer to the question determines the arrows coming out of the Diamond. This shape is quite unique with two arrows coming out of it. One from the bottom point corresponding to Yes or True and one from either

the right/left point corresponding to No or False. The arrows should be always labelled to avoid confusion in the process flow.



Subroutine / Predefined Process :



This shape takes two names - 'Subroutine' or 'Predefined Process'. Its called a subroutine if you use this object in flowcharting a software program. This allows you to write one subroutine and call it as often as you like from anywhere in the code.

The same object is also called a Predefined Process. This means the flowchart for the predefined process has to be already drawn, and you should reference the flowchart for more information.



Advantages Of Using flowcharts:

- **Communication:** Flowcharts are better way of communicating the logic of a system to all concerned or involved.
- **Effective analysis:** With the help of flowchart, problem can be analyzed in more effective way therefore reducing cost and wastage of time.
- **Proper documentation:** Program flowcharts serve as a good program documentation, which is needed for various purposes, making things more efficient.
- **Efficient Coding:** The flowcharts act as a guide or blueprint during the systems analysis and program development phase.
- **Proper Debugging:** The flowchart helps in debugging process.
- **Efficient Program Maintenance:** The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part

Disadvantages Of Using flowcharts:

- **Complex logic:** Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy. This will become a pain for the user, resulting in a waste of time and money trying to correct the problem
- **Alterations and Modifications:** If alterations are required the flowchart may require re-drawing completely. This will usually waste valuable time.
- **Reproduction:** As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem.

Rules For Constructing Variable Name

1. Characters Allowed :

- o Underscore(_)
- o Capital Letters (A – Z)
- o Small Letters (a – z)
- o Digits (0 – 9)

2. Blanks & Commas are not allowed

3. No Special Symbols other than underscore(_) are allowed

4. First Character should be alphabet or Underscore
5. Variable name Should not be Reserved Word

Tip 1 : Use allowed Characters

Valid Names

```
num  
Num  
Num1  
_NUM  
NUM_temp2
```

Tip 2 : blanks are not allowed

Invalid Names

```
number 1  
num 1  
addition of program
```

Tip 3 : No special symbols other that underscore

Valid Identifier

```
num_1  
number_of_values  
status_flag
```

Tip 4 : First Character must be underscore or Alphabet

Valid Identifier

```
_num1  
Num  
Num_  
-
```

—

Invalid Identifier

```
1num
1_num
365_days
```

Tip 5 : Reserve words are not allowed

C is case sensitive.

Variable name should not be Reserve word.

However if we capitalize any Letter from Reserve word then it will become legal variable name.

Valid Identifier

```
iNt
Char
Continue
CONTINUE
```

Invalid Identifier

```
int
char
continue
```

Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division (modulo division)

Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is =

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b

Operator	Example	Same as
<code>*=</code>	<code>a *= b</code>	<code>a = a*b</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a/b</code>
<code>%=</code>	<code>a %= b</code>	<code>a = a%b</code>

Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in [decision making](#) and [loops](#).

Operator	Meaning of Operator	Example
<code>==</code>	Equal to	<code>5 == 3</code> is evaluated to 0
<code>></code>	Greater than	<code>5 > 3</code> is evaluated to 1
<code><</code>	Less than	<code>5 < 3</code> is evaluated to 0
<code>!=</code>	Not equal to	<code>5 != 3</code> is evaluated to 1

Operator	Meaning of Operator	Example
<code>>=</code>	Greater than or equal to	<code>5 >= 3</code> is evaluated to 1
<code><=</code>	Less than or equal to	<code>5 <= 3</code> is evaluated to 0

Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in [decision making in C programming](#).

Operator	Meaning	Example
<code>&&</code>	Logical AND. True only if all operands are true	If <code>c = 5</code> and <code>d = 2</code> then, expression <code>((c==5) && (d>5))</code> equals to 0.
<code> </code>	Logical OR. True only if either one operand is true	If <code>c = 5</code> and <code>d = 2</code> then, expression <code>((c==5) (d>5))</code> equals to 1.
<code>!</code>	Logical NOT. True only if the operand is 0	If <code>c = 5</code> then, expression <code>!(c==5)</code> equals to 0.

Resources :

<https://www.owlgen.com/question/what-is-an-algorithm-explain-characteristics-of-an-algorithm-with-the-help-of-an-example>