## Flowchart building blocks

A simple flowchart for computing factorial N (N!)



###  Symbols

A typical flowchart from older basic computer science textbooks may have the following kinds of symbols:

Start and end symbols

Represented as circles, ovals or rounded rectangles, usually containing the word "Start" or "End", or another phrase signaling the start or end of a process, such as "submit inquiry" or "receive product".

Arrows

Showing "[flow of control](http://en.wikipedia.org/wiki/Control_flow)". An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to. The line for the arrow can be solid or dashed. The meaning of the arrow with dashed line may differ from one flowchart to another and can be defined in the legend.

Generic processing steps

Represented as [rectangles](http://en.wikipedia.org/wiki/Rectangles). Examples: "Add 1 to X"; "replace identified part"; "save changes" or similar.

Subroutines

Represented as rectangles with double-struck vertical edges; these are used to show complex processing steps which may be detailed in a separate flowchart. Example: PROCESS-FILES. One subroutine may have multiple distinct entry points or exit flows); if so, these are shown as labeled 'wells' in the rectangle, and control arrows connect to these 'wells'.

Input/Output

Represented as a [parallelogram](http://en.wikipedia.org/wiki/Parallelogram). Examples: Get X from the user; display X.

Prepare conditional

Represented as a [hexagon](http://en.wikipedia.org/wiki/Hexagon). Shows operations which have no effect other than preparing a value for a subsequent conditional or decision step .

Conditional or decision

Represented as a diamond ([rhombus](http://en.wikipedia.org/wiki/Rhombus)) showing where a decision is necessary, commonly a Yes/No question or True/False test. The conditional symbol is peculiar in that it has two arrows coming out of it, usually from the bottom point and right point, one corresponding to Yes or True, and one corresponding to No or False. (The arrows should always be labeled.) More than two arrows can be used, but this is normally a clear indicator that a complex decision is being taken, in which case it may need to be broken-down further or replaced with the "pre-defined process" symbol.

Junction symbol

Generally represented with a black blob, showing where multiple control flows converge in a single exit flow. A junction symbol will have more than one arrow coming into it, but only one going out.

In simple cases, one may simply have an arrow point to another arrow instead. These are useful to represent an [iterative](http://en.wikipedia.org/wiki/Iteration) process (what in Computer Science is called a [loop](http://en.wikipedia.org/wiki/Control_flow#Loops)). A loop may, for example, consist of a connector where control first enters, processing steps, a conditional with one arrow exiting the loop, and one going back to the connector.

For additional clarity, wherever two lines accidentally cross in the drawing, one of them may be drawn with a small semicircle over the other, showing that no junction is intended.

Labeled connectors

Represented by an identifying label inside a circle. Labeled connectors are used in complex or multi-sheet diagrams to substitute for arrows. For each label, the "outflow" connector must always be unique, but there may be any number of "inflow" connectors. In this case, a junction in control flow is implied.

Concurrency symbol

Represented by a double transverse line with any number of entry and exit arrows. These symbols are used whenever two or more control flows must operate simultaneously. The exit flows are activated concurrently when all of the entry flows have reached the concurrency symbol. A concurrency symbol with a single entry flow is a *fork*; one with a single exit flow is a *join*.

It is important to remember to keep these connections logical in order. All processes should flow from top to bottom and left to right.

**Wyróżnia się następujące rodzaje bloków:**



Graficzne przedstawienie bloków w kolejności ich opisywania.

* a) **Blok graniczny** − oznacza on początek, koniec, przerwanie lub wstrzymanie wykonywania działania, np. *blok startu programu*.
* b) **Blok wejścia-wyjścia** − przedstawia czynność wprowadzania danych do programu i przyporządkowania ich zmiennym dla późniejszego wykorzystania, jak i wyprowadzenia wyników obliczeń, np. *czytaj z, pisz z+10*.
* c) **Blok obliczeniowy** − oznacza wykonanie operacji, w efekcie której zmienią się wartości, postać lub miejsce zapisu danych, np. *z := z + 1*.
* d) **Blok decyzyjny, warunkowy** − przedstawia wybór jednego z dwóch wariantów wykonywania programu na podstawie sprawdzenia warunku wpisanego w ów blok, np. *a = b*.
* e) **Blok wywołania podprogramu** − oznacza zmianę wykonywanej czynności na skutek wywołania podprogramu, np. *MAX(x,y,z)*.
* f) **Blok fragmentu** − przedstawia część programu zdefiniowanego odrębnie, np. *sortowanie*.
* g) **Blok komentarza** − pozwala wprowadzać komentarze wyjaśniające poszczególne części schematu, co ułatwia zrozumienie go czytającemu, np. *wprowadzenie danych*.
* h) **Łącznik wewnętrzny** − służy do łączenia odrębnych części schematu znajdujących się na tej samej stronie, powiązane ze sobą łączniki oznaczone są tym samym napisem, np. *A1, 7*.
* i) **Łącznik zewnętrzny** − służy do łączenia odrębnych części schematu znajdujących się na odrębnych stronach, powinien być opisany jak łącznik wewnętrzny, poza tym powinien zawierać numer strony, do której się odwołuje, np. *4.3, 2,B2*.

Przykładowy algorytm opisany za pomocą schematu blokowego.
Iteracyjna wersja obliczania silni. silnia(0)=1 silnia(n)=1\*2\*...\*n-1\*n

