Flowchart

Initial and terminal symbols

to indicate the start point and the end point of the flowchart

Input/read and output/write symbols

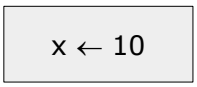
to indicate steps involving the exchange of data with the outside

Flow direction symbol

to indicate the origin and destination of the flow

Símbolo de processamento

to indicate data calculation or data handling



Check symbol ou alternatives symbol

to indicate the selection of one of the actions, according to the result of a logical condition (yes/no)

Connection symbol

to indicate the connection of two points in the flowchart (can be omitted)

Pseudocode

An algorithm must start with

```
algorithm <name-of-algorithm>
```

An algorithm must end with

```
end_algorithm
```

The data necessary to execute the algorithm is indicated by the command (optional)

```
input parameters: <list-of-variables>
```

where <list-of-variables> are the names of the variables that receive the initial data

The data determined (obtained) by the algorithm is indicated by the command (optional)

```
output parameters: <list-of-variables>
```

where <list-of-variables> are the names of the variables that receive the results

Writing comments to clarify the algorithm (this part isn't executable)

```
{ comments }
```

The ← symbol is used to assign the result of a expression to a variable

```
<variable> ← <expression>
```

Structure for input/read data

```
ler: <list-of-variables>
```

to indicate the <list-of-variables> that is available to receive the initial data (values)

Structure for output/write data

```
escrever: <list-of-variables, messages>
```

to indicate which and where certain values should be written and to show messages

Simple conditional structure

```
if <logical_condition> then
    <commands>
end_if
```

Composed conditional structure

```
if <logical_condition> then
    <commands_1>
else
    <commands_2>
end_if
```

Repetition structure that checks the condition at the entry of the cycle

```
while <logical_condition> do
    <commands>
end_while
```

Repetition structure that checks the condition at the exit of the cycle

```
do
    <commands>
while <logical_condition>
```