Pseudocode & Flow Charts

Pseudocode is a **shorthand notation for programming** which uses a combination of **informal programming structures and verbal descriptions of code**. Emphasis is placed on expressing the behavior or outcome of each portion of code rather than on strictly correct syntax (it does still need to be reasonable, though).

In general, pseudocode is used to outline a program before translating it into proper syntax. This helps in the initial planning of a program, by creating the logical framework and sequence of the code. An additional benefit is that because pseudocode does not need to use a specific syntax, it can be translated into different programming languages and is therefore somewhat universal. It captures the **logic and flow of a solution** without the bulk of strict syntax rules.

Below is some pseudocode written for a program which moves as long as a touch sensor is not pressed, but stops and turns to the right if its sonar detects an object less than 20in away.

```
task main()
{
 while ( touch sensor is not pressed )
  {
    Robot runs forward

    if (sonar detects object < 20in away)
    {
      Robot stops

      Robot turns right
    }
  }
}
```

Some intact syntax
The use of a while loop in the pseudocode is fitting because the way we read a while loop is very similar to the manner in which it is used in the program.

Descriptions
There are no actual motor commands in this section of the code, but the pseudocode suggests where the commands belong and what they need to accomplish.

This pseudocode example includes elements of both programming language, and the English language. Curly braces are used as a visual aid for where portions of code need to be placed when they are finally written out in full and proper syntax.

Pseudocode & Flow Charts

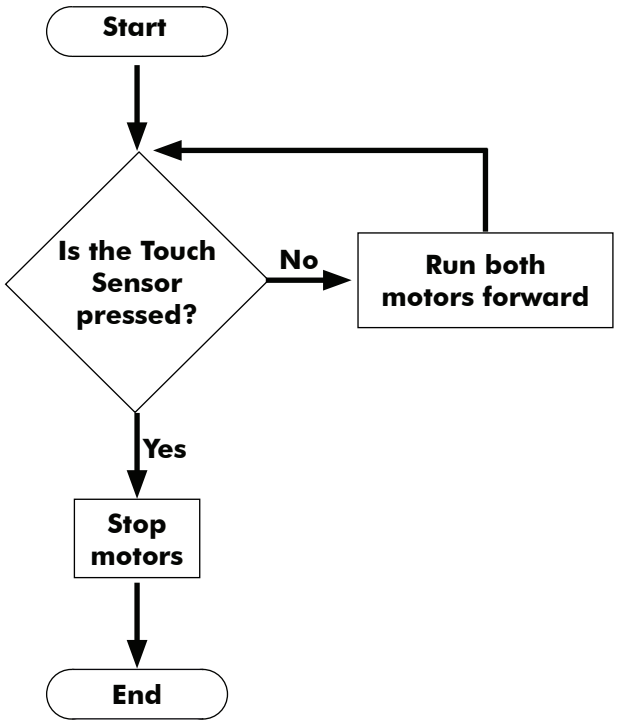
Flow Charts are a **visual representation of program flow**. A flow chart normally uses a combination of **blocks** and **arrows** to represent actions and sequence. Blocks typically represent **actions**. The **order** in which actions occur is shown using arrows that point from statement to statement. Sometimes a block will have multiple arrows coming out of it, representing a step where a **decision** must be made about which path to follow.

Start and End symbols are represented as rounded rectangles, usually containing the word "Start" or "End", but can be more specific such as "Power Robot Off" or "Stop All Motors".

Actions are represented as rectangles and act as basic commands. Examples: "wait1Msec(1000)"; "increment LineCount by 1"; or "motors full ahead".

Decision blocks are represented as diamonds. These typically contain Yes/No questions. Decision blocks have two or more arrows coming out of them, representing the different paths that can be followed, depending on the outcome of the decision. The arrows should always be labeled accordingly.

To the right is the flow chart of a program which instructs a robot to run forward as long as its touch sensor is not pressed. When the touch sensor is pressed the motors stop and the program ends.



To read the flow chart:

- Start at the "**Start**" block, and follow its arrow down to the "Decision" block.
- The **decision block** checks the status of the touch sensor against two possible outcomes: the touch sensor is either pressed or not pressed.
- If the touch sensor is not pressed, the program follows the **"No" arrow** to the action block on the right, which tells the motors to run forward. The arrow leading out of that block points back up and around, and ends back at the Decision block. This forms a **loop**!
- The **loop** may end up repeating many times, as long as the Touch Sensor remains unpressed.
- If the touch sensor is pressed, the program follows the "Yes" arrow and stops the motors, then ends the program.