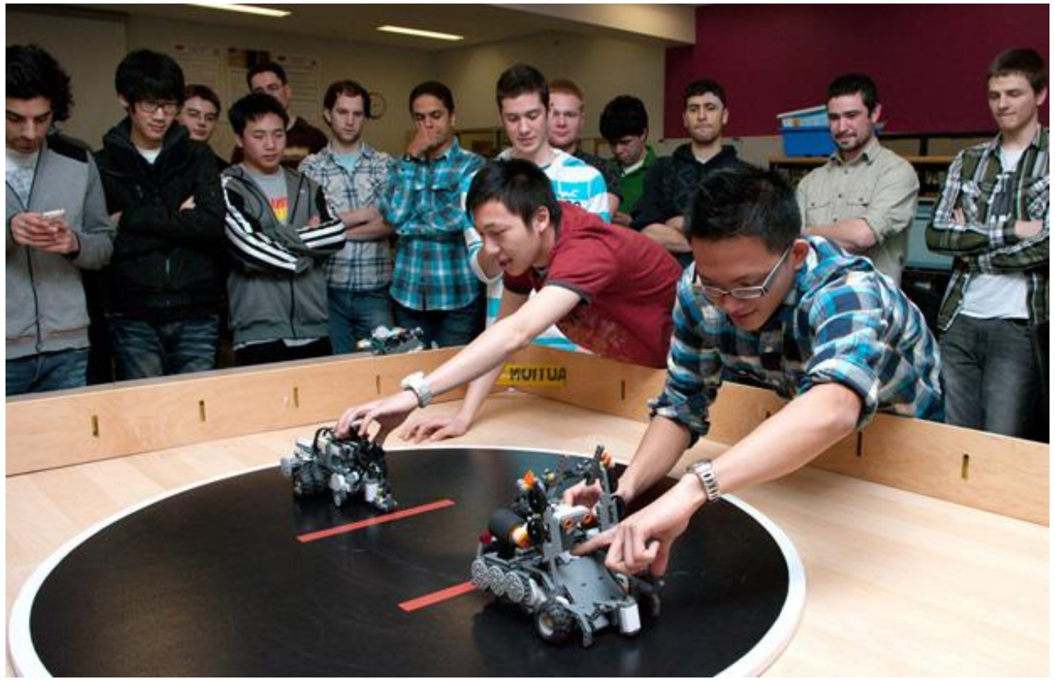
Mechatronics Design I
MSE 110

SIMON FRASER UNIVERSITY
THINKING OF THE WORLD



Instructor

Amr Marzouk

Teaching Assistants

Sina Salari

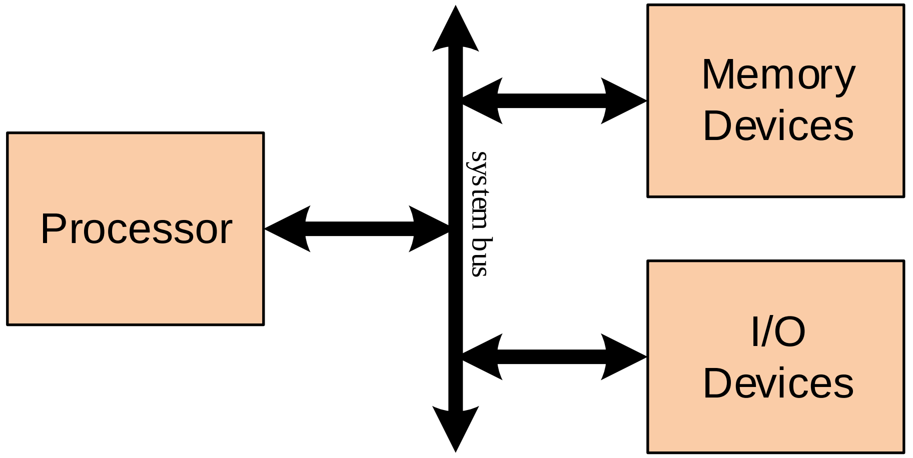
Mikel Garcia-Poulin

Lecture Outline

- Introduction to computer architecture
- Algorithm development: Flow Charts and Pseudocode
- Introduction to LEGO Mindstorms™ programming using RobotC

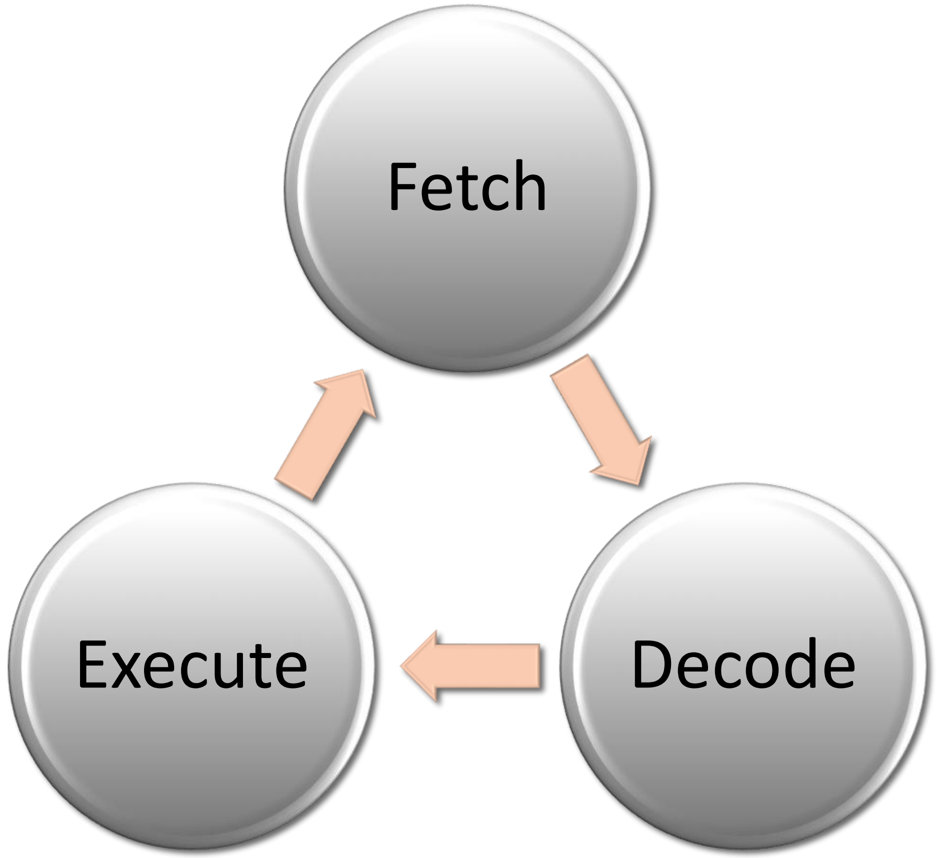
Computer Architecture

- ***Computer Structure:***
- The computers that you have encountered are the result of decades of evolution. The machines that we commonly refer to as "desktop" computers are incredibly complicated. The modern machines are all based upon a simple structure called "Von Neumann Architecture".



Computer Architecture

- ***The Processor:***
- The processor is a complex circuit that controls the transfer of data over the system bus. The pattern of bits that are transferred across the bus are viewed as representing either data or machine instructions. Associated with every processor design is a specific set of machine instructions called the "Processor Instruction Set". Each instruction is assigned a unique pattern of bits called the "Machine Instruction Code".
- The processor undergoes "Machine Cycles" immediately upon applying power to system. Machine cycles continue until power is removed from the system. A machine cycle consists of three stages.
- 1) Instruction Fetch.
- 2) Instruction Decode.
- 3) Instruction Execute.



Computer Architecture

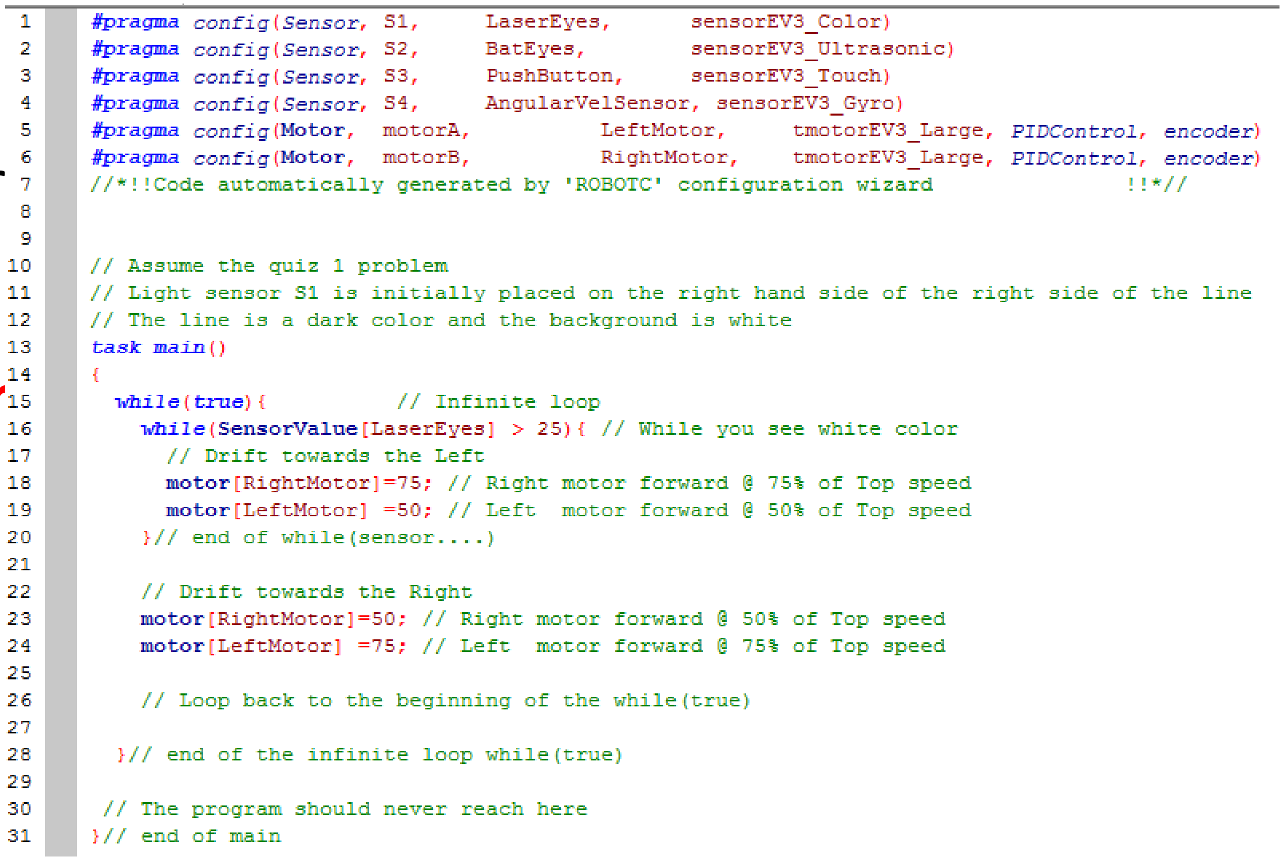
- ***The Memory Subsystem:***
- The memory subsystem is composed of many devices that are capable of retaining many patterns of bits. The devices are classified as either "volatile" or "non-volatile". depending on whether the device is capable of retaining a pattern when power is removed from the device. A magnetic disc subsystem is non-volatile. The main memory banks are volatile.

Computer Architecture

- ***The Input/Output Subsystem:***
- The Input/Output (I/O) subsystem is composed of many devices that perform conversions between the internal bit patterns and activity in real-world physical parameters.
- *A mouse*: is a device that converts real physical motion into bit patterns.
- *An LCD*: is a complicated "light bulb". Bit patterns are somehow converted into a pattern of glowing pixels.
- *An Audio Subsystem*: is a collection of devices that convert between sound pressure waves and streaming bit patterns.

Computer Architecture

- ***Computer Programs:***
- A computer program in its rawest form is a region of the main memor system containing machine instruction codes. Each machine instruction is executed, one per machine cycle in a **sequential order** Other regions of the main memory system may be reserved for storing bit patterns that represent meaningful data within the context of the programs task. These data objects are manipulated by the processor as it executes machine instructions.



```
#pragma config(Sensor, S1,     LaserEyes,       sensorEV3_Color)
#pragma config(Sensor, S2,     BatEyes,         sensorEV3_Ultrasonic)
#pragma config(Sensor, S3,     PushButton,      sensorEV3_Touch)
#pragma config(Sensor, S4,     AngularVelSensor, sensorEV3_Gyro)
#pragma config(Motor,  motorA,          LeftMotor,     tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor,  motorB,          RightMotor,    tmotorEV3_Large, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//


// Assume the quiz 1 problem
// Light sensor S1 is initially placed on the right hand side of the right side of the line
// The line is a dark color and the background is white
task main()
{
  while(true){          // Infinite loop
    while(SensorValue[LaserEyes] > 25){ // While you see white color
      // Drift towards the Left
      motor[RightMotor]=75; // Right motor forward @ 75% of Top speed
      motor[LeftMotor] =50; // Left  motor forward @ 50% of Top speed
    }// end of while(sensor....)

    // Drift towards the Right
    motor[RightMotor]=50; // Right motor forward @ 50% of Top speed
    motor[LeftMotor] =75; // Left  motor forward @ 75% of Top speed

    // Loop back to the beginning of the while(true)

  }// end of the infinite loop while(true)

 // The program should never reach here
}// end of main
```

Computer Programming

- ***Computer Programming:***
- Constructing a computer program is equivalent to choosing the bit patterns for the region of machine instructions and also allocating regions for storing data objects. This can be a laborious task. Nowadays we use computers to assist us with programming computers.
- "high-level compiled language" (C#, C++ is an example of a high-level language)
- "Language Compiler" checks the text files for correct conformity to the language rules. This is referred to as "Syntax Checking".
- The final executable machine instructions is generated by three programs, the compiler, the linker and the loader.
- Modern day computers have an advanced "Graphical User Interface", (GUI), as part of an operating system, (OS). Programming is accomplished with an "Integrated Development Environment", (IDE)

Flowcharts

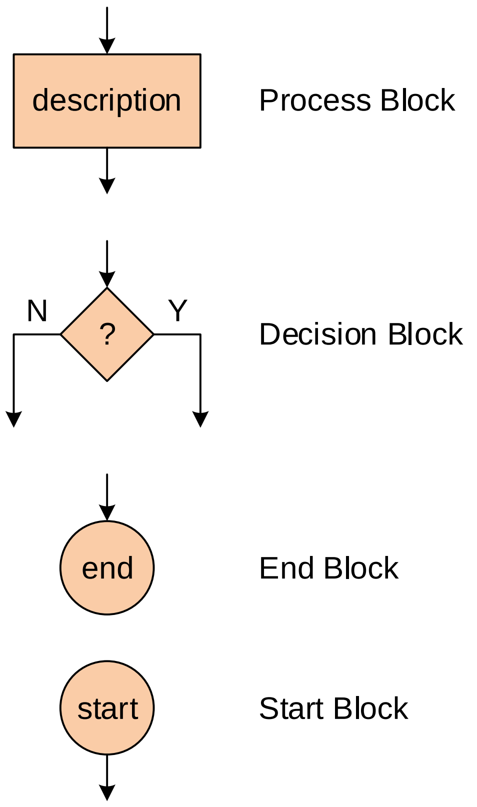
RULE 1: A flowchart must be created using the shapes,

RULE 2: All flowcharts must contain ONE and only ONE start block. A start block has ONE arrow pointing away from the block. (Exit arrow)

RULE 3: A flowchart may contain a SINGLE end block. The only exception to this occurs when a program never ends. In this case there will be no end block. An end block has ONE arrow pointing towards the block. (Entry arrow)

RULE 4: A flowchart may contain MANY decision blocks. Each decision block contains a simple question that has a YES/NO answer. A decision block has ONE entry arrow and TWO exit arrows. The exit arrows are labelled YES and NO to indicate the path taken as a result of answering the question.

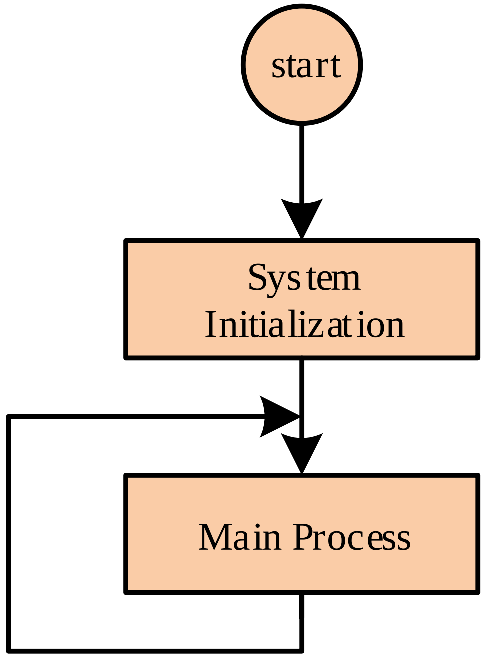
RULE 5: A flowchart may contain MANY process blocks. Each process block contains a brief description of the action taken when executing the block. Keep the text inside the block simple and descriptive. You should use footnotes to elaborate on the contents. A process block has ONE entry arrow and ONE exit arrow.

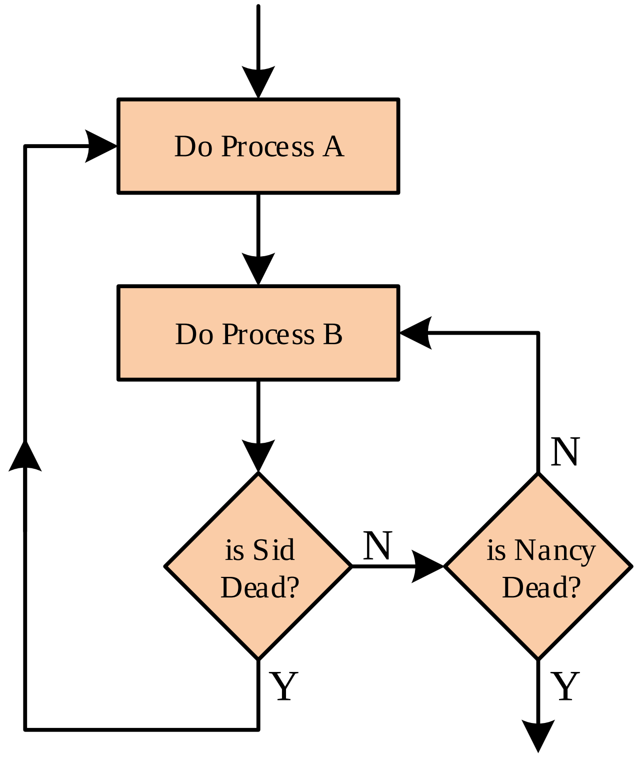


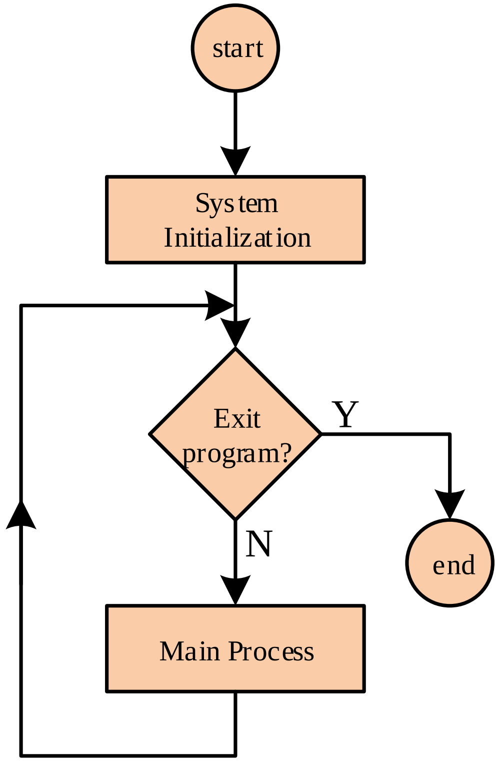
Challenge Question !

- ❑ Design a flow chart for a robot to move forward for three wheel rotations.
- Hint: You need to use the motor encoder

Examples of a common program structures: Notice the infinite loop.



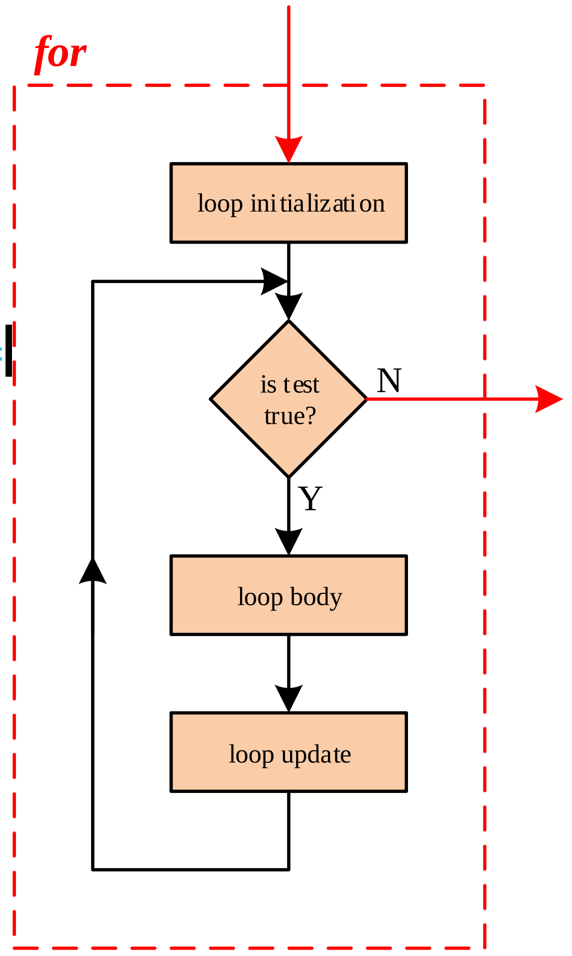




Example of the three basic loop structures: For-Loop

- Most languages implement loop structures called, *for*, *while* and *do while*.
- Notice the single entry arrow and a single exit arrow.

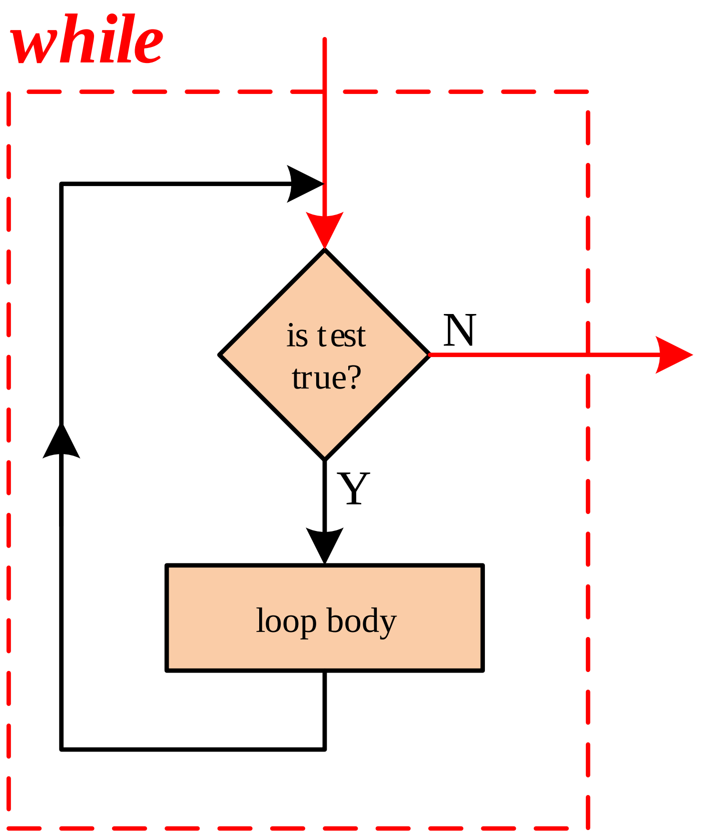
```
// An example for a "For" Loop displaying the numbers 0 through 4
// Check the flowcharts lecture
int i=0;
for (i = 0; i < 5; i++) {
  displayCenteredTextLine(4,"%d",i);
  wait1Msec(1000);
  eraseDisplay();
}
```



Example of the three basic loop structures: While Loop

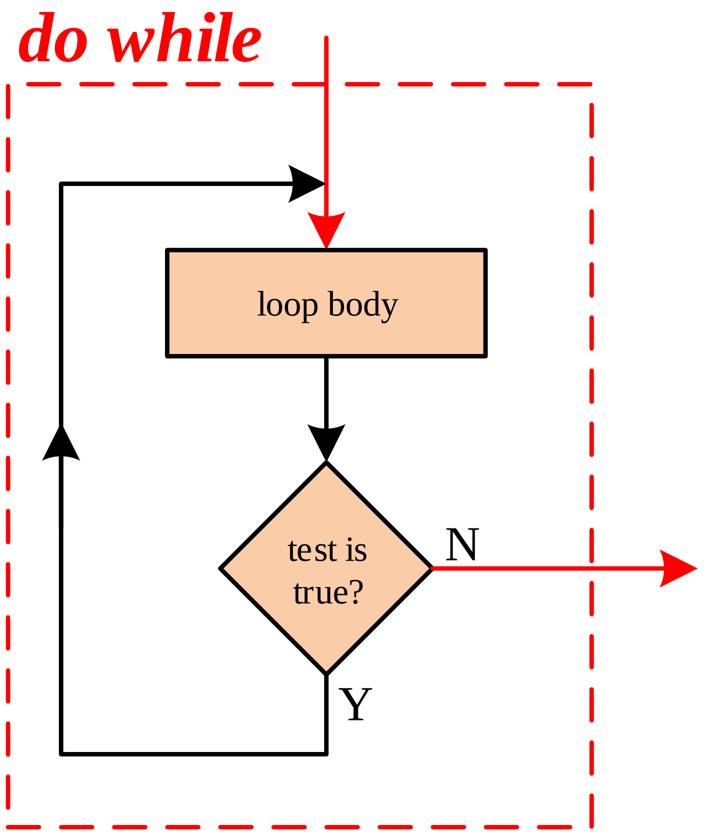
```
// An example for a "While" Loop displaying the numbers 0 trough 4
// Check the flowcharts lecture
int j=0;

while (j < 5) { // Execute the lines within this loop as long as
                // j<5.
                // In other words: Execute the loop body
                // until j>=5
  displayCenteredTextLine(4,"%d",j);
  wait1Msec(1000);
  eraseDisplay();
  j++;  // Without this line, this while loop would have never
      // exited because the value of j will never be updated
      // in the loop and j would have never reached the value of
      // 5 (i.e. the condition to exit the loop)
}
```



Example of the three basic loop structures: do-While Loop

```
// An example for a "do-While" Loop displaying the numbers 0 through 4
// Note that a do-while will execute the loop-body at least once
// even if the loop condition is false
// Check the flowcharts lecture
int k=0;
do{
  displayCenteredTextLine(4,"%d",k);
  wait1Msec(1000);
  eraseDisplay();
  k++;
}while (k < 5);
```



In-Class Challenge
Bouncing Rectangle (The Pong Game)

- Required:
  - A ball the bounces off right, left and top screen walls in a direction opposite to its angle of incidence.
  - The player controls a bat with the EV3 touch sensors (i.e. push buttons).
  - If the ball falls off the bat and hits the bottom screen wall, the player looses.
- Optional:
  - Every 3 hits successful hits, game level is increased and ball speed increases.
  - Score, level and game lives are kept track of and displayed on the EV3 screen
  - Students are encouraged to innovate in the implementation. For example, support for 1 player vs 2 player (one can use the built-in buttons on the brick, the other can use the touch sensors)