Teaching Pseudocode and Flowcharts

Introduction

Pseudocode - is a native language description of what the robot is required to do. With practice pseudocode eventually resembles ROBOTC code.
Flowchart - is a graphical representation of program flow.

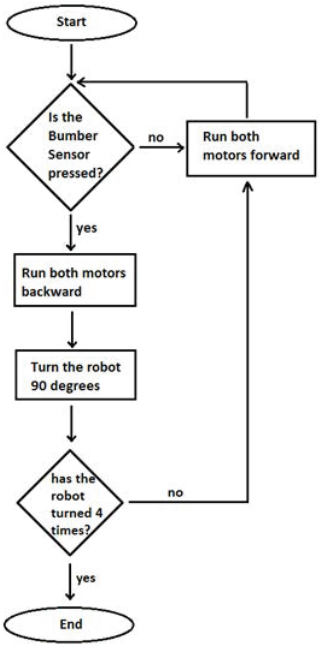
It is a very good practice to have students begin each programming task by breaking the task into its smallest parts and develop pseudocode that describes the robot's behaviors. You will find much more detail on this practice in the *Introduction to Pseudocode lesson*.

The practice of developing pseudocode is informally introduced on the first day of class when students are asked to write a program to control a humanoid robot to make a sandwich. See *Introduction to Robot Programming* for a full description of that activity. In that activity, students will not yet know a robot programming language, but they can begin to write pseudocode to describe the behaviors the robot will need to complete to make the sandwich. (E.g., lift the arm, open the hand, reach for the bread, grab the bread, etc.).

Example pseudocode

In this teacher's guide, we use the term *flowchart* in a very general way to convey that students are constructing visual representations of the problem and its solution. The advantage to flowcharts is that they help a student identify robot decision making.

Example flowchart



This curriculum doesn't formally introduce flowcharts until the Program Flow Chapter because the first two chapters, Basic Movement and Sensors, only require simple behaviors (movingForward, turning, moveUntil a sensor value, etc.). During these activities, students will learn the basic lexicon of the programming language allowing them to be begin writing robot behaviors using pseudocode. Teachers should encourage students to use pseudocode - even if it initially starts as common descriptions and slowly improves to reflect a more accurate programming language. Pseudocode will eventually be used to describe individual parts within a flowchart since each pseudocode phrase describes a unique robot behavior.

In the Program Flow Chapter, students will begin programming their robots to solve problems that contain multiple steps with decisions, that is when this curriculum formally introduce flowcharts. Additionally, you will find an activity called Tic-Tac-Toe which can be used to help students learn to think algorithmically. Students learn are tasked to write a tic-tac-toe to break down the game into its basic parts and then describe the algorithm used to play the game. This can be a group activity centered around developing a flowchart to determine how to play the game. With practice, students will begin to see that flowcharts provide a visual representation of the robot's decision making process and then begin using flowcharts to solve their programming challenges.

Overview

In order to plan a program and write efficient code students need to be able to write clear instructions for the robot. This lesson introduces students to the idea of writing clear instructions and then introduces them to pseudocode.

Objectives

Students will be able to:

- Listen carefully and follow instructions
- Communicate clear instructions
- Break tasks down into smaller pieces
- Write pseudocode for a simple maze
- Understand the necessity of planning clear steps

Materials

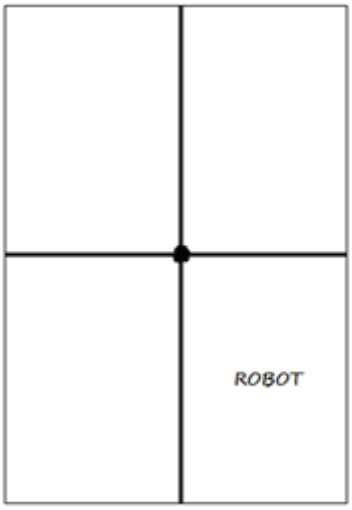
- Blank paper for each student
- Pencil and a ruler for each student
- A large table top or floor surface to setup a simple maze
- Tape to mark out maze boundaries
- Small box or object to represent a robot

Clear Communications Lesson

Procedure

1. Introduce the following “Clear Communications” drawing activity. Tell students:

Robots will follow the program that they are given, even if that program doesn’t make sense. The robot needs to be given a program that it can understand to produces the desired outcome. Let’s begin with a drawing activity to help us think about this idea of giving clear instructions.



Have all students begin with a blanks sheet of paper and then give them the following verbal instructions:

1. *Draw a dot in the center of your page*
2. *Draw a vertical line from the top of your page to the bottom of your page, passing through the center dot*
3. *Draw a horizontal line from the top of your page to the bottom of your page, passing through the center dot*
4. *Write the word ROBOT center of the square created at the bottom right of your page*

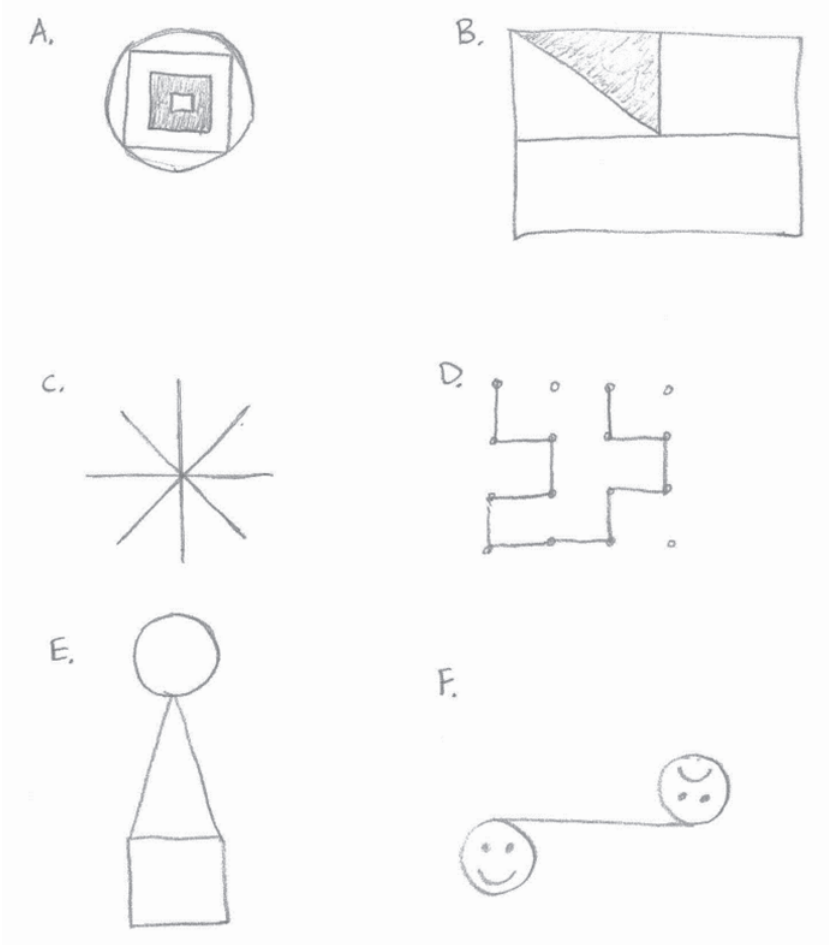
Have students hold up their drawings when everyone is finished. Check to see that everyone’s is the same. Discuss any differences if they arise.

Introduction to Clear Communications

Choose a student to be the communicator and give them a simple drawing (see examples at the bottom of this page) making sure no one else sees the picture that you've given the student communicator. Have the communicator describe the drawing for the others to reproduce on their paper. The other students may ask questions for clarification and the communicator may adjust their instructions if they see mistakes being made. Encourage students to see how quickly and accurately the picture can be reproduced – each drawing should not take more than 5 minutes.

Repeat the exercise with a new picture and a new communicator student. This time the other students are not allowed to talk or ask any questions, but the communicator may still adjust their instructions if they see mistakes being made.

Repeat the exercise one last time with a new picture. Now the communicator must sit behind a screen or with their back turned and no questions may be asked. The communicator must clearly give their instructions one time for the others to reproduce the drawing.



Discuss this last exercise with the class.
Ask the students who had to be the communicator:

How did you decide the order in which to give instructions?

How confident were you that your instructions would work?

Ask the students who were reproducing the drawing:

Did you find the instructions easy to follow?

Was there anything you drew differently because the instructions were unclear?

What makes a good set of instructions?

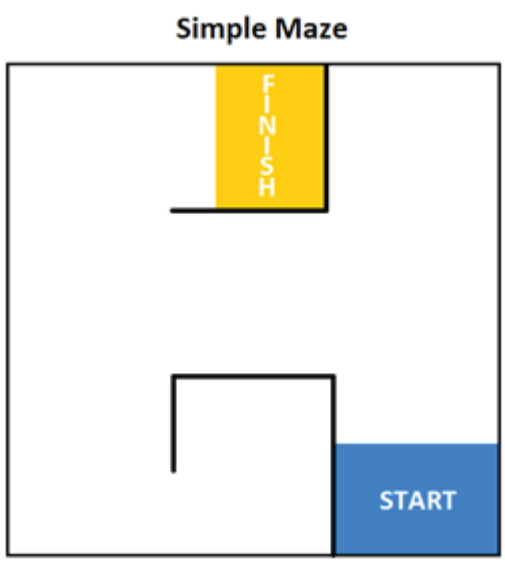
[Emphasize this question, and encourage students to conclude that the instructions need to be broken down into simple steps because instructions that are too big can be confusing and lead to error.]

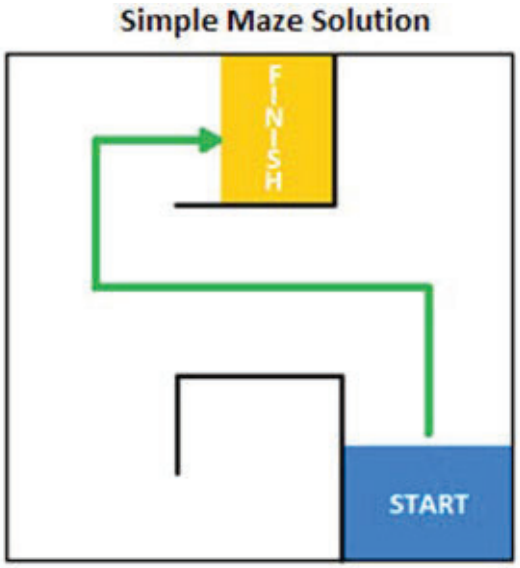
Introduction to Robot Programming

Tell students: *This last exercise is most like programming robots – the programmer (or communicator) gives a set of instructions which will be followed exactly. The instructions need to be broken down into the simplest possible pieces so that they can be performed accurately and without confusion.*

Tell students: ***Behaviors** are a very convenient way to talk about what a robot is doing and what it must do. Every action of a robot can be described as a behavior: move forward, turn on a motor, look for an obstacle, stop, and solve a maze are all examples of robot behaviors.*

Direct students to the simple maze shown below on the left. The goal is for the robot to solve the maze by following the path below on the right. Tell students: *To do this, we will think about the robot's actions in terms of behaviors.*

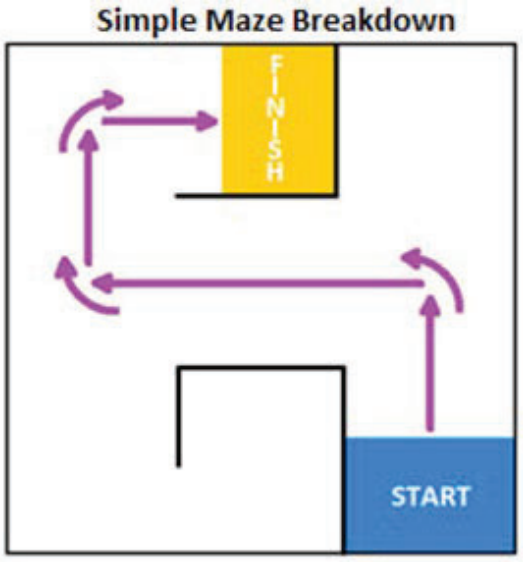




Give a student a robot. Have a student volunteer move the object along the path that would solve the maze. Point out that we can easily see what behavior the robot needs in order to solve the maze.

Tell students:

Some behaviors are too big to give to the robot as instructions, so we need to think about breaking them down into smaller behaviors. "Solve the maze" is actually a very complex behavior because it involves many steps and the robot wouldn't know what to do.



Instruct students to:
Write down the behavior "Solve the maze" on the top of your paper.

Now, tell them to:
Break down that behavior into smaller behaviors and write them below in order.

If students are unsure, ask:
What does the robot need to do to follow the solution path?

(Move forward, turn left, move forward, turn right, etc.)

Ask students:
Is your list of behaviors is clear enough to instruct the robot through the maze.

They should realize that we are close, but the robot doesn't know how far to move forward each time or how much to turn.

Finally, instruct students to:
Use your rulers to figure out the distance the robot needs to move for each behavior.

(Exact distances will depend on your maze setup).
Write this new specific behavior next to their last list of behaviors.

We now have a list of behaviors specific enough to give as instructions to the robot.
Tell students:
By starting with a very large solution behavior and breaking it down into smaller and smaller sub-behaviors, you have a logical way to figure out what a robot needs to do to accomplish its task.

Talking about and writing the code in English is the first step in good pseudocode practice, which allows us to plan robot behaviors before we translate them to code.

Example Pseudocode

Pseudocode Exercise

What is Pseudocode?

Robots need very detailed and organized instructions in order to perform their tasks. Before a programmer can begin programming they need to break a robot's behaviors down into simple behaviors and figure out when each behavior should run. Some programmers like to use pseudocode to begin ed-constructing the programming problem.

pseudo

adj : not genuine but having the appearance of;

Source: WordNet ® 1.6, © 1997 Princeton University

Pseudocode is a hybrid language, halfway between English and code. It is not real code

yet, but captures the details that will be important in translating your ideas to code, while still allowing you to think and explain things in plain language. Good pseudocode will make it very straightforward to write real code afterwards, because all the behaviors and logic will already be contained in the pseudocode.

Pseudocode example

If you wanted to program a robot to stop when it saw and object and move forward when it didn't see and object your pseudocode might look like:

pseudocode

1. Move forward
2. If (sonar sensor detects and object)
   stop
3. When the sonar sensor no longer
   sees and object move forward.
4. Do this forever

Exercise

1. Convert these instructions to pseudocode and into a flowchart:
   a. "If it's raining, bring an umbrella."
   b. "Keep looking until you find it."
   c. "Take twenty paces, then turn and shoot."
   d. "Go forward until the touch sensor (on port 1) is pressed in."
   e. "Turn on oven. Cook the turkey for 4 hours or until meat thermometer reaches 180 degrees."
   f. "Crossing the street" Hint, make sure that you look both ways!

2. Compare the advantages and disadvantages of flowcharts and pseudocode.
   Explain in your own word why you believe one is better than the other.
   Is one of them always better than the other, or are both good in different situations?
   Can you use both to help solve the same problem? Should you?

Introduction to Flowcharts

Overview

A flowchart is a visual representation of program flow and is used by programmers to break down and model robot behaviors. This lesson provides teachers with a guide to introduce flowcharts in the *Sensors - Forward Until Touch Unit*, and then this lesson should be applied to all of the other Sensor Units. Students have already been introduced to pseudocode and program flow in the Introduction to Robot Programming activity; this lesson focuses on how to graphically describe robot decision making. In this lesson the teacher will model how integrate pseudocode and decision making into a flowchart.

Objectives

Students will be able to:

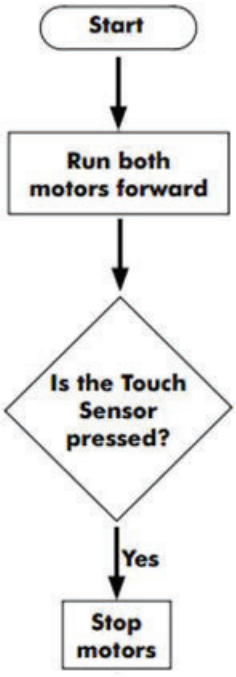
- Break down a problem into simple components
- Organize the components into a proper sequence
- Represent a sequence of behaviors in a flowchart
- Use a flowchart to analyze robot behaviors

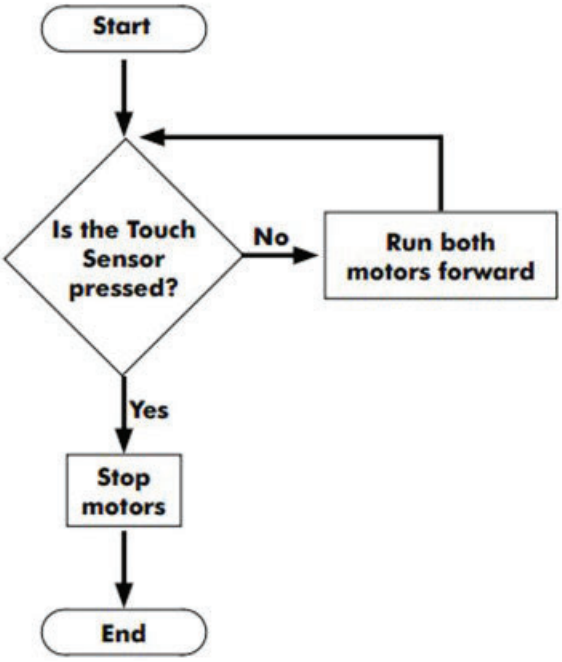
Procedure:

Have students watch the video and complete the questions in the Sensors - Forward Until Touch 3: Move Until Touch Unit. In this example students will write a program that requires the robot to make a decision and includes multiple steps. This is a good time to introduce flowcharts. Note: All students should be able to see a picture of the flowcharts below.

Tell students: *Now that the robot is using a sensor to make decisions, we can use a flowchart to understand how our robot's behavior is broken down into steps within a program. So far, our flowcharts have just been single steps of pseudocode, but now we will need to add decision blocks which ask a "yes or no" question.*

Show both of the following flowcharts to the class and ask:
Which flowchart best represents the robot behavior seen in the Move Until Touch video?





Draw the example flowcharts on the left on the board or project them onto the screen for students to see and discuss.

If students struggle or disagree, ask:

When does the robot stop moving forward? *(When the bumper sensor is pressed in.)*

When does the robot check the bumper sensor to make the decision to move forward or stop?
(The robot continually decides to move forward when the bumper is not pushed, and decides to stop when the bumper is pushed)

Which flowchart represents a robot continually making decisions based on the bumper sensor?
(The flowchart on the right requires the robot to keep asking if the bumper is pressed)

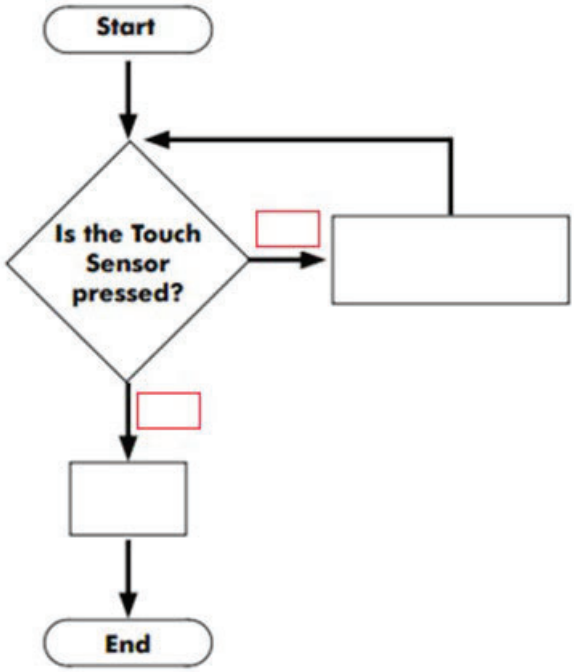
Read through the flowchart on the right (on the previous page) to the class to show how each robot action is represented in the flowchart. Teach students the difference between start/stop, action, and decision blocks.
Tell students: *The robot must continually check the value of the bumper sensor to decide what to do. If the bumper sensor is not pressed, the robot continues running both motors forward. If the bumper sensor is pressed, the robot stops all motors. The flow of these decisions is given in the flowchart on the right, but missing from the chart of the left.*

Go to **Forward Until Touch 3 -Try it! 2** activity. Refer to the same flowchart as the class thinks about the following question:

What happens if you start the program with the bumper switch pressed in?"

Give them a moment to think, and then ask students to explain their answer to the class using a flowchart. Tell the students: *Use a flowchart to break down each step of the robot's behavior so that we can see what actions the robot will perform in any situation.*

Next, scroll down to the **Forward Until Touch - Try it! 3** activity.
Take down the previous flowchart, and show this new flowchart with missing entries. Have students copy it onto their papers and fill in the blanks to properly represent program flow while a robot is pushing a box until its touch sensor is released.
If students struggle, ask: *Now when will the robot stop moving forward?* (When the bumper sensor is released)



When students have successfully filled out their flowcharts, ask the class:

What would happen if the box was not all the way against the bumper sensor when the program started?

(The robot would not move)
Use a flowchart to show this pathway.

Scroll down to **Forward Until Touch - Mini Challenge 1: Vacuum**. On a new piece of paper, have students design a flowchart to represent the robot's behavior in this challenge. The robot will need to perform more actions after the bumper sensor is read each time.
Give the students several minutes to work on their flowchart. Have them work in small groups if necessary.

If students struggle, ask:
What does the robot need to do after the bumper sensor is pressed? (Backup and turn towards another wall)

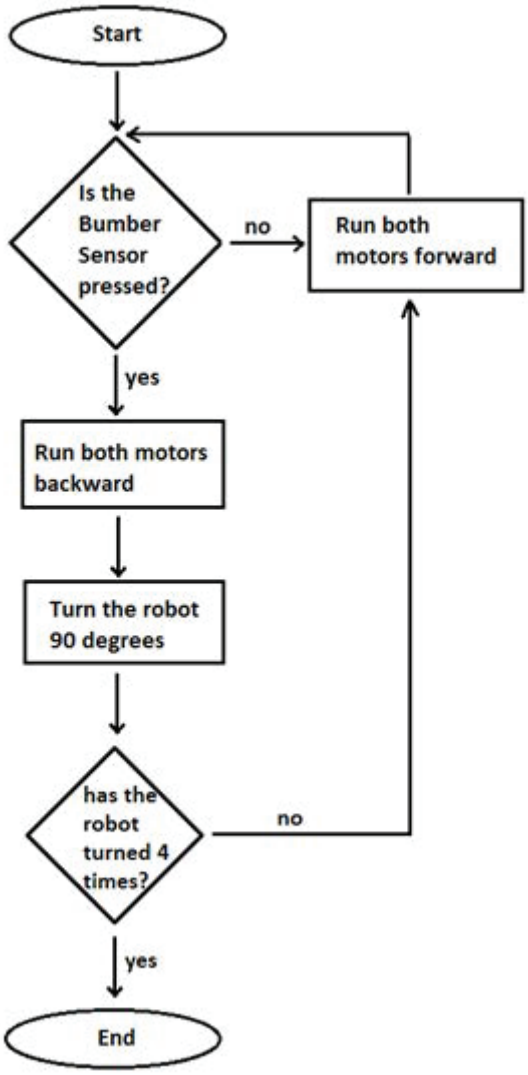
Does the robot ever need to repeat its behaviors?
(Yes, it will have the same behavior for all 4 walls)

How will the robot know when it has contacted all the walls? (When it has repeated its behavior 4 times)

After students have finished, ask a group to draw their solution for the class. Follow the flow and check the logic for any missing steps or problem areas.
Ask if other students created a different flowchart they would like to share.
Discuss aspects of each solution until the class comes to an agreement on a correct flowchart. It's possible to have multiple correct solutions.

Tell students:
Flowcharts help us understand the decision making process of a robot. The answer to a "yes or no" decision will determine what action the robot does. By creating a properly organized flowchart, we can see the plan a robot needs to follow to be successful.

At this point, either have students use their flowcharts to write the code and try each challenge, or move on to the tic-tac-toe activity to transfer the critical thinking.



Pictured above is the solution for the Forward Until Touch - Mini Challenge 1: Vacuum flowchart

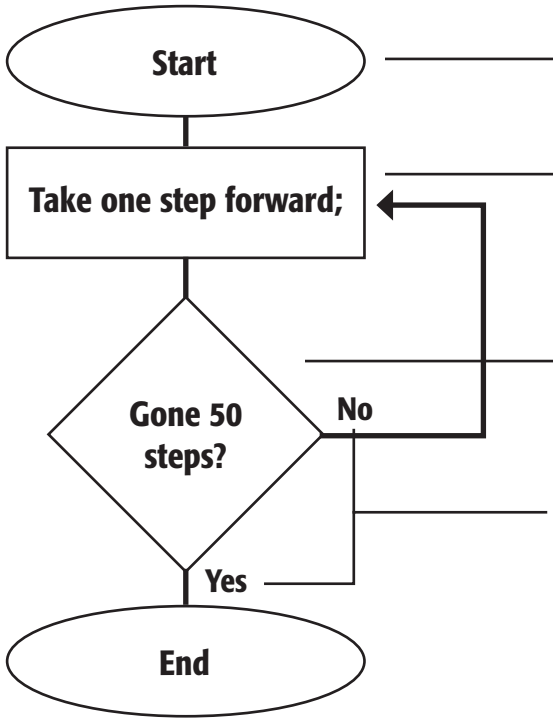
Flowchart Exercise

What are Flowcharts?

Robots need very detailed and organized instructions in order to perform their tasks. The programmer must break things down into simple behaviors and figure out when each behavior should run. A flowchart is a tool that can be used by programmers to determine program flow.

A flowchart provides a way of visually representing and organizing individual behaviors and decisions within a program -- it provides a diagram of the "flow" of the program. Programmers use flowcharts to lay out the steps that will be needed in their final program, and to help determine how the robot's behaviors should be broken down.

Parts of a Flowchart



Start of Program - Marks the beginning of the program, begin here. Follow the line to get to the next block.

Statement Block - A statement to execute, or a behavior to perform.

Decision Block - A decision point in your program. Ask a simple question, and do different things depending on the answer.

Yes/No - Answers to the question posed in the decision block. Follow the line labeled with the appropriate answer.

End of Program - Marks the end of the program. If you reach this point, the program is done!

Exercise

1. Make a flowchart organizing the "flow" of getting ready to go to school in the morning. Be sure to include the following steps in your chart, but don't be afraid to add other things if you need them!

Select something to wear	Look for your shoes	Put your shoes on
Take a shower	Brush your teeth	Hit snooze button
Eat breakfast	Put toast in the toaster	Get dressed
Walk or get a ride to school	Check your alarm clock	Comb your hair
Get out of bed	Turn on shower	Check the time

Tic-Tac-Toe Challenge

This teacher-led activity helps students to extend their computational thinking to an everyday problem. In this activity, students are asked to develop an algorithm to play tic-tac-toe. Teachers will find a prepared slide presentation designed to help students think about solving the game. The game is a simple game that most students quickly find that there is no way to win. To ensure that all students are familiar with the game, have them play several games with each other and discuss their strategy before beginning the activity.

The following is the suggested discussion surrounding the introduction to the tic-tac-toe programming presentation, teachers will find a Power Point Presentation in the teacher resources folder:

Slide 1: The teacher says:
The algorithmic thinking skills that we've been developing can be applied to problems outside of robotics. Let's take a look at how we can use what logic to develop a strategy for winning at tic-tac-toe.

(Teacher can read through the big ideas of the unit with examples of how each is applied to this problem.)

Let's start by considering simple aspects of playing tic-tac-toe.

How many ways can you win horizontally?

(Discuss the term, horizontally, if necessary.)

Slide 2: The teacher says:
Here we see the three ways a player can win horizontally.

How many ways can you win vertically?

(Discuss the term, vertically, if necessary.)

Slide 3: The teacher says:
Good, and here are the three ways to win vertically.

How many ways can you win diagonally? (Discuss term, diagonally, if necessary.)

Slide 4: The teacher says:
Right, there are only two ways to win diagonally.

So how many ways is there to win total?

Slide 5: The teacher says:
Right, 8 ways total. Let's organize and label those ways in an array or grid like the one used for tic-tac-toe.

Slide 6: The teacher says:
Here's the nine-boxed array like the grid used in the game. You can see that each possible way to win is labeled. If I'm the first player, how can I decide where to put my X first?

Continued next page.

(Students might respond in a variety of ways but they should eventually recognize that as the first player, the X should be placed where it provides the greatest number of opportunities to win. This is explained to students on the next slide.)

Slide 7: The teacher says:
Right, considering which box will provide the most possible wins is a good strategy.

Slide 8: The teacher says:
Here we can see that the center box has the greatest number of possible wins - 4.

Which are those four possible ways to win?

Slide 9: The teacher says:
Right, here we see those four ways.

So we place our X in the middle square.

The Student Challenge

Students are now challenged to create a flowchart that documents the decisions involved when deciding where to place the first X in a game of tic-tac-toe. The flowcharts will include pseudocode that describes how they would program a robot to play tic-tac-toe.