# Module 2 - Program Flow

Alternate Wikibooks Text: [C Programming – Program Flow and Control](https://en.wikibooks.org/wiki/C_Programming/Control)

## [Make a Decision](http://learn.parallax.com/propeller-c-start-simple/make-decision)

Generally, any program flows from the first line to the last. But programs can conditionally repeat a block of code within the program until a specified condition exists and then exit that block of code. This is called looping.

The Program Flow section illuminates the way programs work with the first four exercises and then the last two as practice creating code that does something interesting.

First, we talk about how those conditions are determined.

Insert a picture here and paste in a copy of your program from **Try This** for **Make a Decision**. Don’t forget to comment the program.

Insert a picture here and paste in a copy of your program from **Your Turn** for **Make a Decision**. Don’t forget to comment the program.

## [Make Several Decisions](http://learn.parallax.com/propeller-c-start-simple/make-several-decisions)

C allows you to cascade the decision making process one test after another, each with its own program in response to the result.

Insert a picture here and paste in a copy of your program from **Try This** for **Make Several Decisions**. Don’t forget to comment the program.

Insert a picture here and paste in a copy of your program from **Your Turn** for **Make Several Decisions**. Don’t forget to comment the program.

## [Make Complicated Decisions](http://learn.parallax.com/propeller-c-start-simple/make-complicated-decisions)

With the addition of logical expressions, you can make “complicated” decisions. We take the logical functions from electronic digital logic, and test two or more input statements for true.

Insert a picture here and paste in a copy of your program from **Try This** for **Make Complicated Decisions**. Don’t forget to comment the program.

Insert a picture here and paste in a copy of your program from **Your Turn** for **Make Complicated Decisions**. Don’t forget to comment the program.

## [Code That Repeats (The While Statement)](http://learn.parallax.com/propeller-c-start-simple/code-repeats)

We use a **while** statement in a program to repeat a certain block of code (part of the program) repeatedly until a certain condition no longer exists. We do not care when the condition no longer exists, so **while** is kind of an “open ended” conditional statement. We have another statement that we use to make a block of code repeat for a certain number of times or for a certain length of time. As a matter of fact it is called a **for** statement.

The **while** statement is analogous to testing a cake for doneness. Usually a toothpick is inserted into the cake. If the toothpick comes out wet, the cake goes back into the oven. We might include a delay of about 5 minutes before the next test. The test is repeated and the toothpick checked. We’re not sure how many times this may have to repeat, but there’s no point in eating an underdone cake. We could include error checking by repeating the test a certain number of times and then check to make sure the oven is even on or broken. This avoids a “hung” process or endless loop.

In a computer, I could wait for a certain I/O pin to change state, repeating the same kind of checking process until it does.

Insert a picture here and paste in a copy of your program from **Try This** for **Code That Repeats**. Don’t forget to comment the program.

Insert a picture here and paste in a copy of your program from **Your Turn** for **Code That Repeats**. Don’t forget to comment the program.

## [Counting Loops](http://learn.parallax.com/propeller-c-start-simple/counting-loops)

As mentioned in the previous section, **for** loops are used for counting . You can also use them for delay timing if you add a pause statement in there somewhere.

Insert a picture here and paste in a copy of your program from **Try This** for **Counting Loops**. Don’t forget to comment the program.

Insert a picture here and paste in a copy of your program from **Your Turn** for **Counting Loops**. Don’t forget to comment the program.

**The next two sections are important applications of what you’ve learned. One is another look at arrays. The other is a topic that may shed light on some advanced topics. When you create variables with int statements, let’s say, and give them names, you never really know where they go in memory. Where did the compiler end up putting the numbers you’re using in your program? This section shows you how to find out. First, playing with indexed lists: arrays.**

## [Index Array Variables](http://learn.parallax.com/propeller-c-start-simple/index-array-variables)

Using a counting loop, you can march through an array and print out the list of values therein.

Insert a picture here and paste in a copy of your program from **Try This** for **Index Array Values**. Don’t forget to comment the program.

Insert a picture here and paste in a copy of your program from **Your Turn** for **Index Array Values**. Don’t forget to comment the program.

## [Variable Values and Addresses](http://learn.parallax.com/propeller-c-start-simple/variable-values-and-addresses)

Well this one tool you’ll need when you need it. Read carefully here.

Insert a picture here and paste in a copy of your program from **Try This** for **Variable Values and Addresses**. Don’t forget to comment the program.

Insert a picture here and paste in a copy of your program from **Your Turn** for **Variable Values and Addresses**. Don’t forget to comment the program.