Code That Repeats

Submitted by Andy Lindsay on Thu, 03/21/2013 - 16:52

Lesson edited to work with **CodeBlocks** IDE by Jeff La Favre 3/12/26

*(Updated 2013-08-08 for SimpleIDE 0.9.4 and its Learn folder's Simple Libraries and Examples) [SimpleIDE is the IDE for use with the robot. This lesson is edited so that we can use **CodeBlocks** as the IDE, which does not require a robot – J. La Favre]*

Sometimes we need to be able to program our robot microcontroller to repeat certain operations. This is useful for simple things, such as blinking a light, and more complex tasks, such as robotic sensor navigation.

Blocks of repeating code are called *loops*. **`The C while`** loop can be set up to repeat a code block while some condition is true. A **`while`** loop can also be set up to repeat endlessly, but there is a trick for escaping endless loops as well. Let's try all of these.

- Start **CodeBlocks** (or the IDE you are using).
- Open the **File** menu and select **New**. Then select **Empty File**.
- Click the mouse in the text window of **CodeBlocks** and use the keyboard to enter the following text: **#include <stdio.h>**
- Open the **File** menu and select **Save file**, which opens a **Save file** dialog box.
- In the dialog box, open the drop-down labeled **Save as type** and select **C/C++ files**. In the **file name** slot enter this name for the file: **repeat while.** At the top of the dialog box there is a **Save in** slot, which determines where the file will be saved. Make sure you know the location where you are saving your file so that you can find it later. Now click the **Save** button to save your program file.
- Use your keyboard to type the text in the box on the next page into the text window of **CodeBlocks** under the first line of text you have already entered.

- Click the **Save** button to save the code you just entered with keyboard.
- Take a look at the code and predict the output.
- Run the program by opening the **Build** menu and selecting **Build and Run**. If there are no errors in the program, a new program window will open. The program will pause after a line is displayed until you press the Enter key. Then the while loop will run again and print the next line. Keep pressing the Enter key until the program lets you know it is done. Compare the actual output to your predicted output. Verify that the program display counts up to 100 in units of 5.

```
int main()
{
 int n = 0;
 while(n < 100)   // while the variable named n is less than 100, execute the code block below
 { // start of while code block
  n = n + 5;    // each time this code block is executed, the value of n increases by 5 and eventually will equal 100
  printf("n = %d\n", n);    // display the value of n
  //pause(200);         //this line in original program has been suppressed with a comment due to lack of support in Dev-C++
  getchar();  //this line is substituted for line above…program will pause here until user presses the Enter key
 } // end of while code block, now program jumps back up to while line to test if n is still below 100, if not, loop stops executing
 printf("All done!");    //this displays after while loop stops executing
}
```

How it Works

Inside `main`, a variable `n` is declared and initialized to zero. Then, `while (n < 100)` starts a loop that will keep repeating the code block below it as long as `(n < 100)` is true.

Each time through the loop, `n = n +5` increases the value of `n` and then a `printf` displays the new value.

Eventually, `n = 100` so `(n < 100)` is no longer true. At that point, the code execution skips to the first instruction below the `while` loop's code block. In this case, it's `printf("All done!");`.

[Ed. Note: the original code includes the statement **pause(200);**, which means pause the program execution for 200 milliseconds. This would cause the display to show five sets of numbers every second on the screen to help reinforce the fact that a loop is working. Unfortunately, the **pause** function is not available in the **CodeBlock** libraries, and so this line is commented out. A one line comment is marked using // and any code to the right of // will not execute, or in our case, give an error when attempting to compile the code. I have selected the function **getchar** as a substitute for the function **pause**. The **getchar** function gets a character input from the keyboard. When the program comes to this line, it pauses, waiting for the user to press a key on the keyboard then the Enter key. Just pressing the Enter key is enough to make the program continue. It is possible to write a custom function and give it the name pause and this will be the subject of another lesson – J. La Favre]

Did You Know?

Keep in mind that conditions like `a < b` and `n < 100` evaluate to 1 if they are true or 0 if they are false. Like `if` statements, `while` statements execute their code block when the condition is 1, or really any value other than zero.

Try This – Endless Loop

Think about the Did You Know section for a minute. Can you guess how to make a **`while`** loop repeat forever? If you change **`while(n < 100)`** to **`while(1)`**, you'll have an endless loop.

- Open the **File** menu and select **Save file As**, and save your project as **Repeat Endless**.
- Change **`while(n < 100)`** to **`while(1)`**.
- Run the modified program and verify that it just keeps going and going and going

Your Turn - Break Out of an Endless Loop

Even an endless loop doesn't have to be endless thanks to the C language's **`break`** keyword.

- Add **`if(n == 50) break;`** inside the **`while`** loop's code block.
- Test and verify that the program breaks out of the loop when **`n`** reaches 50.

```
    getchar();
    if (n == 50) break;     // add this line
  }
  printf("All done!");
}
```

What do you think would happen if you move the **`break`** statement to just underneath **`n = n + 5`** inside the **`while`** loop's code block? Try it!