Index Array Variables

Submitted by Andy Lindsay on Thu, 03/21/2013 - 17:20

original source: http://learn.parallax.com/propeller-c-start-simple/index-array-variables

Lesson edited to work with **Dev-C++** IDE by Jeff La Favre 10/23/15

---

*(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 **Dev-C++** as the IDE, which does not require a robot – J. La Favre]*

Ed. Note; you must have **Dev-C++** set to compile programs in **ISO C99** format for this lesson. The instructions for doing this are in lesson 9. If you have already completed lesson 9, then you have already set Dev-C++ to compile in ISO C99 format.

A great use of **`for`** loops is performing operations on elements in an array variable. The **`for`** loop can increment through the array's index values to access each element of the array in succession. (See the Array Variables lesson for a refresher.)

- Start **Dev-C++**.
- Open the **File** menu and select **New**. Then select **Source File**.
- Click the mouse in the text window of **Dev-C++** and use the keyboard to enter the following text: **#include <stdio.h>**
- Open the **File** menu and select **Save**, which opens a **Save As** dialog box.
- In the dialog box, open the drop-down labeled **Save as type** and select **c source files(*.c)**. In the **file name** slot enter this name for the file: **index array variables.** 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.
- Copy the text in the box on the next page and paste it into the text window of **Dev-C++** under the first line of text you have already entered. Alternatively, you can enter the text using the keyboard.
- Click the **Save** button to save the code you just pasted or entered with keyboard.

- Run the program by opening the **Execute** menu and selecting **Compile 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 for loop will run again and print the next line. Keep pressing the Enter key until the program is done. Verify that it prints each of the `p` array variable's elements.

```
int main()
{
 int p[] = {1, 2, 3, 5, 7, 11};          // Initialize the array
 for(int i = 0; i < 6; i++)              // Count i from 0 to 5
 {  // start of for code block
   printf("p[%d] = %d\n", i, p[i]);        // Display array element & value
  // pause(500);                          //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 for code block
}
```

How it Works

First, the main routine sets up an integer array named `p` with six elements, 1 followed by the first five prime numbers.

Next comes a **`for`** loop with a start index of **`int i = 0`**, a condition of **`i < 6`**, and an increment of **`i++`**.

Now, take a close look at the **`printf`** statement in the for loop's code block. It prints two items each time through the loop: the value of the i variable displayed as a **`p`** index, and the value of the element in the **`p`** array that has an index equal to i.

So, the first time through the **`for`** loop, **`printf("p[%d] = %d\n", i, p[i])`** becomes **`printf("p[%d] = %d\n", 0, p[0])`**. That prints **`"p[0] = 1"`** in the program window.

The second time through the loop, **`i++ increments the value of i`**, so the **`printf`** statement becomes **`printf("p[%d] = %d\n", 1, p[1])`**.

The **`for`** loop continues to count up through 5, displaying all the array's elements, ending with **`"p[5] = 11"`**.

Did You Know?

sizeof — C has a unary operator called **`sizeof`** that returns the number of bytes consumed by an object in memory, such as an array.

In the example program above, we knew our array had 5 elements, so we used i < 6 as the for loop's condition to access all of the elements in the **`p`** array. But what if we didn't know how many elements were in the **`p`** array? Here's a **`for`** loop that would work:

```
for(int i = 0; i < sizeof(p)/sizeof(int); i++)
```

Here, **`sizeof(p)`** gets the total number of bytes in the array. **`sizeof(int)`** gets the number of bytes used by one **`int`** element in the array. So, **`sizeof(p)/sizeof(int)`** equals the number of elements in the array.

Try This

- Open the **File** menu and select **Save As** and name the file **index array variables more** .
- Add several more prime numbers as elements in the `p` array using example below.
- Modify the `for` loop using the `sizeof` trick as the condition parameter.
- Run it, and see that it works.

```
int main()

{

 int p[] = {1, 2, 3, 5, 7, 11, 13, 17, 19, 23};          // modify this line to be like this

 for(int i = 0; i < sizeof(p) / sizeof(int); i++)          // modify this line to be like this

 {  // start of for code block

   printf("p[%d] = %d\n", i, p[i]);

  // pause(500);                    //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 for code block

}
```

Your Turn

- Write a program that multiplies each array element by 100 and then stores the result back to the array element. Then, have a second loop display all the values. If you did it right, that second loop should display 100, 200, 300, 500, 700, 1100, etc.