Variables and Math

Submitted by Andy Lindsay on Thu, 03/21/2013 - 17:21
original source: http://learn.parallax.com/propeller-c-start-simple/variables-and-math

Lesson edited to work with **Dev-C++** IDE by Jeff La Favre 10/22/15

(Updated 2013-08-08 for SimpleIDE 0.9.4 and its Learn folder's Simple Libraries and Examples) [Ed. note: SimpleIDE is IDE used with robot, for this lesson we will use ***Dev-C++*** *as the IDE, which does not require a robot to run programs. There are some minor differences in some function names between SimpleIDE and Dev-C++. For example, the print function used to display text on the computer screen is written like this:* ***print*** *using SimpleIDE and like this:* ***printf*** *using Dev-C++. The function name in Dev-C++ adds the character f to the end of print. While this is a minor difference, the exact function name must be used for the IDE you are using. When members move to SimpleIDE to write robot code, if they want to include the print function, they must NOT add the f to the end of print, as they must do when using Dev-C++ – J. La Favre]*

Variables are named sections of memory that make it possible for your program to "remember" values. Your program can also use them in math expressions to perform calculations.

This example program, Variables and Calculations.c, declares variables named **`a`**, **`b`**, and **`c`**. It stores 25 in **`a`**, 17 in **`b`**, and the result of **`a + b`** in the variable named **`c`**.

- 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>** Remember that all programs must reference the library or libraries for functions used in the program. In this program you will use the **printf** function to display text on the computer screen. That function is stored in the standard input/output library named **stdio.h**, which is why you need to include this line at the top of your code.
- 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: **variables and calculations** 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 below and paste it into the text window of **Dev-C++** under the first line of text you have already entered. Alternatively, you can type the code in yourself, not including text starting with // of each line.
- Click the **Save** button to save the code you just pasted or entered with the keyboard.
- Now examine the program code, and try to predict what will be displayed when the program runs.
- 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. Compare the actual output to your predicted output.

```
int main()  //remember, the first function for all programs must be main()
{           //start of code block for main() function
 int a = 25;  //create a variable named a and give it a value of 25
  int b = 17;  //create a variable named b and give it a value of 17
 int c = a + b;  //create a variable named c and give it a value of a + b, which would be 42 in this example
printf("c = %d ", c);  //display on screen the value of c, like this: c = 42.  %d is a placeholder for value stored in c
 }          //end of code block for main() function
```

First, a few words about the format of the code listed in the text box above. Notice that the text has been color coded. This is the same coloring scheme that is used in **Dev-C++**. The parentheses for functions **()** and braces for code blocks **{}** are in red.

Also in red are mathematical operators like **=** and **+** . Also, the semicolon marking the end of a statement is in red **;** . The first argument inside the parentheses of the printf function is colored in a blue color. Numbers assigned to variables are colored violet.

Notice that each line has some text colored in a lighter blue color, starting with //. Two forward slashes, like this //, mark the beginning of a comment. Any text on a line, starting at the point where // occurs, is ignored by the computer when the program is compiled and run. The comments are only for people who are reading the code, trying to understand it. The programmer adds comments to help others understand his or her code. It also helps the programmer remember what she or he was trying to accomplish with the code when he or she reviews it at a later time.

How Variables and Calculations.c Works

Variables and Calculations.c declares an integer variable named **`a`** and assigns it the value 25 with **`int a = 25`**. Then, it declares a second variable named **`b`** and initializes it to 17 with int **`b = 17`**. The last integer variable it declares is named **`c`**, and stores the result of **`a + b`** in it.

Finally, it displays the value of **`c`** with **`printf("c = %d", c)`**. This variation on **`printf`** displays a sequence of characters called a *string*, followed by a variable. The **`%d`** is called a *format placeholder*, and it tells **`printf`** how to display the value stored in that variable as a decimal number, 42 in this case.

Did You Know?

The term + is a *binary operator*, meaning that it performs an operation on two inputs. Examples of binary operators include:

`+`	Add
`-`	Subtract
`*`	Multiply
`/`	Divide
`%`	Modulus (remainder of a division calculation)

Try This

On the next page is a modified version of the main routine that displays **`"a = , b ="`** with their values, and then **`"a + b ="`** and the value of **`c`** on a new line. Then, it repeats for **`a - b`**.

Notice that the second time it calculates the value of **`c`**, we don't need to declare it with **`int`**. It's just **`c = a - b`**. Notice also that **`printf`** allows you to display more than one numeric value within your string. All it takes is two format placeholders in the string and the names of two variables, separated by commas, after the string.

- Select **Save As** in the **File** menu, name it **Test Binary Operators**, save as type **c source files(*.c)** and click **Save** button.
- Modify the current code in the text window of **Dev-C++** with the code in the box on the next page.
- Run the program and verify the output.

```
#include <stdio.h>

int main()

{

 int a = 25;

  int b = 17;

 int c = a + b;

 printf("a = %d, b = %d\n", a, b);  // now we have two %d placeholders, the first one for value stored in a and second one for value in b

// code on above line will result in display like this: a = 25, b = 17

printf("a + b = %d\n", c);

c = a – b;

printf("a = %d, b = %d\n", a, b);

printf("a – b = %d\n", c);

 }
```

Your Turn

- Expand Test Binary Operators.c so that it goes through testing all five binary operators in the Did You Know section.

PRO TIP: Displaying % with printf

To display the output of the Modulus operator, use **`("a mod b = ...")`** or ("**`a  %% b = ...`**) in the **`printf`** function. Since `%` has another purpose in **`print`** strings, just saying **`("a % b = ...`**) will give unexpected results.

- Try changing **`a`** to 17 and **`b`** to 25, then re-run.
- Try declaring **`int`** variables of **`y`**, **`m`**, x and **`b`**, and then use them to calculate and display **`y = m*x + b`**.