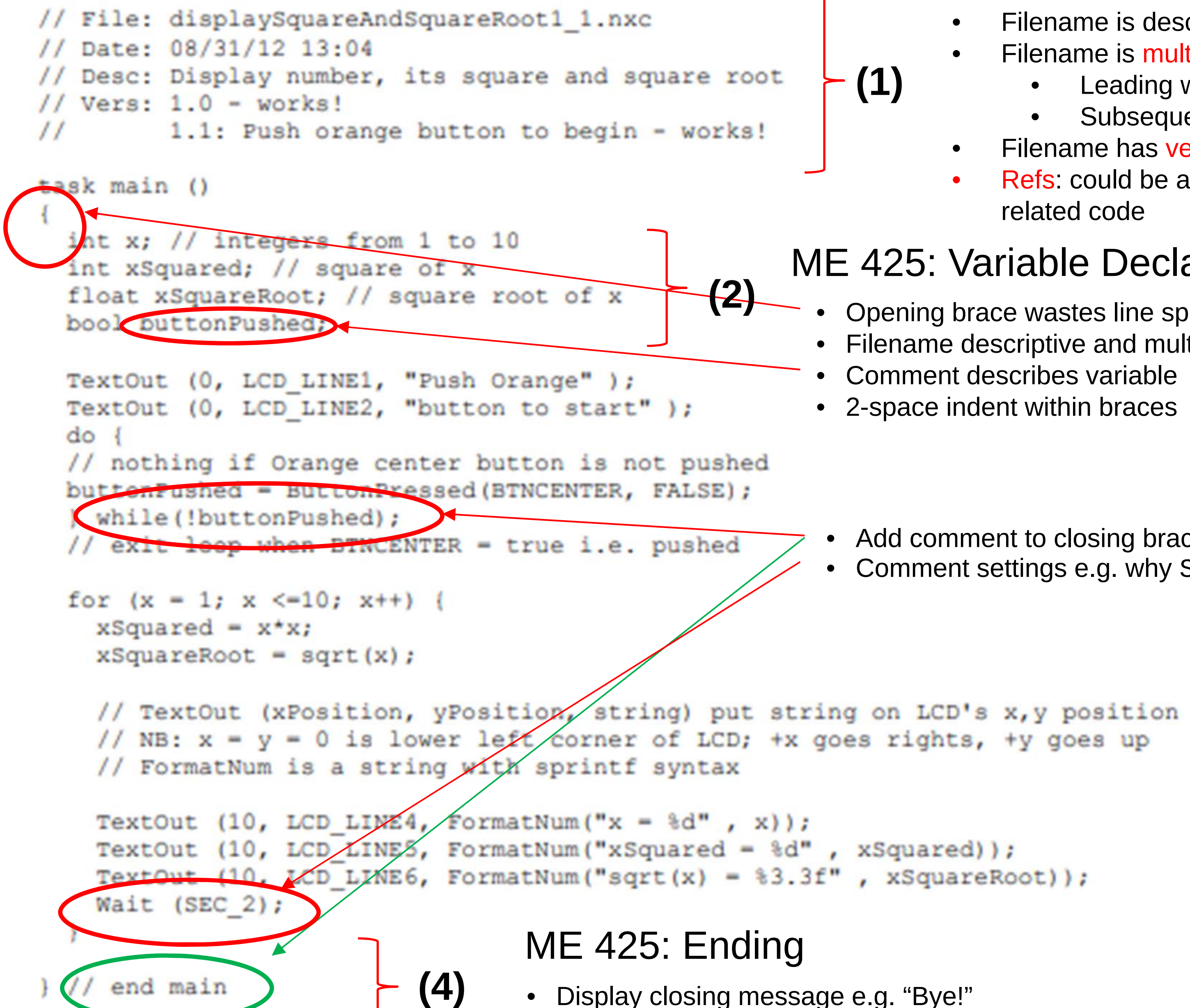
Slides: NXC Programming 1

Best Practices, Data Types, TextOut and FormatNum

Best Practices



```
// File: displaySquareAndSquareRoot1_1.nxc
// Date: 08/31/12 13:04
// Desc: Display number, its square and square root
// Vers: 1.0 - works!
//       1.1: Push orange button to begin - works!

task main ()
{
  int x; // integers from 1 to 10
  int xSquared; // square of x
  float xSquareRoot; // square root of x
  bool buttonPushed;

  TextOut (0, LCD_LINE1, "Push Orange" );
  TextOut (0, LCD_LINE2, "button to start" );
  do {
  // nothing if Orange center button is not pushed
  buttonPushed = ButtonPressed(BTNCENTER, FALSE);
  } while(!buttonPushed);
  // exit loop when BTNCENTER = true i.e. pushed

  for (x = 1; x <=10; x++) {
    xSquared = x*x;
    xSquareRoot = sqrt(x);

    // TextOut (xPosition, yPosition, string) put string on LCD's x,y position
    // NB: x = y = 0 is lower left corner of LCD; +x goes rights, +y goes up
    // FormatNum is a string with sprintf syntax

    TextOut (10, LCD_LINE4, FormatNum("x = %d" , x));
    TextOut (10, LCD_LINE5, FormatNum("xSquared = %d" , xSquared));
    TextOut (10, LCD_LINE6, FormatNum("sqrt(x) = %3.3f" , xSquareRoot));
    Wait (SEC_2);
  }

} // end main
```

(1)

ME 425: Always start all your NXC programs with these intro comments

- Filename is descriptive
- Filename is multiword
  - Leading word is not capitalized
  - Subsequent words capitalize 1st letter
- Filename has version e.g. 1_0a, 1_1b, etc
- Refs: could be an additional comment to recall related code

(2)

ME 425: Variable Declarations

- Opening brace wastes line space
- Filename descriptive and multiword
- Comment describes variable
- 2-space indent within braces

- Add comment to closing brace
- Comment settings e.g. why SEC_2

ME 425: Body

(3)

(4)

ME 425: Ending

- Display closing message e.g. “Bye!”
- Close any motors, sensors, etc with `StopAllTasks`

Data Types http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/vars.html

Data Type	Description	Examples
`bool`	Unsigned 8-bit value usually 0 or 1	`bool buttonPushed;`
`byte`	Unsigned 8-bit value for integers 0 to 255 i.e. 0 to $2^8 - 1$	`byte personAge;`
`char`	Signed 8-bit value for integers -128 to 127, usually for storing ASCII values	`char vowel = “A”;`
`int`	Signed 16-bit value for integers -32768 to +32767 (i.e. $-2^7\ to + (2^7-1)$	`int homesOnMarket;`
`short`	Signed 16-bit value	
`long`	Signed 32-bit value -2,147,483,648 to 2,147,483,647 (i.e. $-2^{31}\ to\ (2^{31}-1)$	`long currentDeficit;`
`unsigned`	Keyword to modify char, int, and long e.g. unsigned int is 0 to $2^{32}-1 = 4{,}294{,}967{,}296$	`unsigned int cityPopulation;`
`float`	A 32-bit real number	`float pi;`
`string`	For a string of characters (ending with a NULL)	`string mm = “Mad Max”;`

`arrays`	Single or multiple dimension arrays	`char vowels[];`
		`char vowels[] = {“A”, “E”, “I”, “O”, “U”};`

TextOut

http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/group___display_module_functions_ga9a070f70dbe14ebfb0b6b0c0abbef64c.html#ga9a070f70dbe14ebfb0b6b0c0abbef64c

```
char TextOut(int x, int y, string str, unsigned long options)
```

Usually use
`LCD_LINE1`
`LCD_LINE2`
`:`
`LCD_LINE8`

String must be a string variable or text within quotes E.G. `mm` or “`Mad Max`” where `mm` was declared as `string`

I never used.

FormatNum

http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/group__cstring_a_p_i_gae1e152293a956a9911b3940664b7b9f4.html#gae1e152293a956a9911b3940664b7b9f4

```
string FormatNum(string fmt, variant num)
```

Follows sprintf format specifier

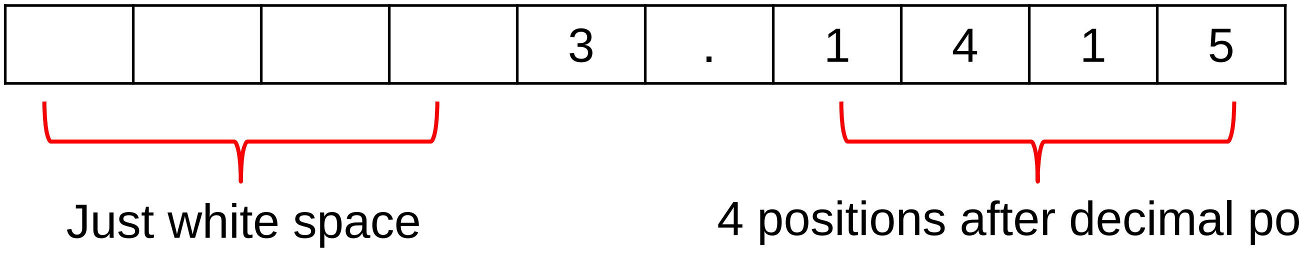
Often combine `TextOut` and `FormatNum` e.g.

```
    TextOut(0, LCD_LINE1, FormatNum(“Age = %d”, myAge));
    TextOut(0, LCD_LINE2, FormatNum(“Pi = %f”, piValue));
```

Specifier	Used for
%d	Display an int
%f	Display a float
%x	Display int in Hex

Often control the width e.g. minimum number of spaces (right justified)

```
  TextOut(0, LCD_LINE1, FormatNum(“Age = %3d”, myAge));
  TextOut(0, LCD_LINE2, FormatNum(“Pi = %10.4f”, piValue));
```



				3	.	1	4	1	5

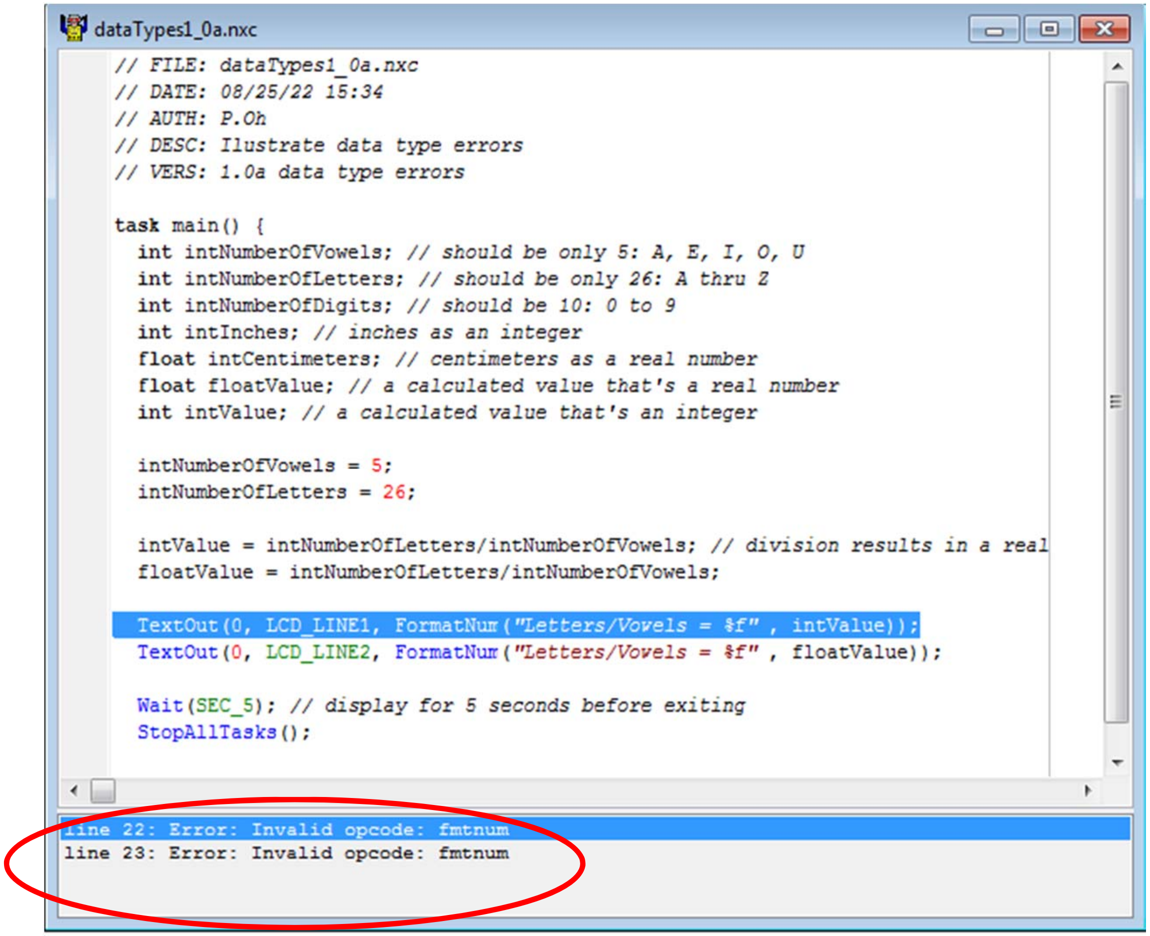
Just white space

4 positions after decimal point

Issues to Watch Out For

Error: “`Invalid opcode: fmtnum`”

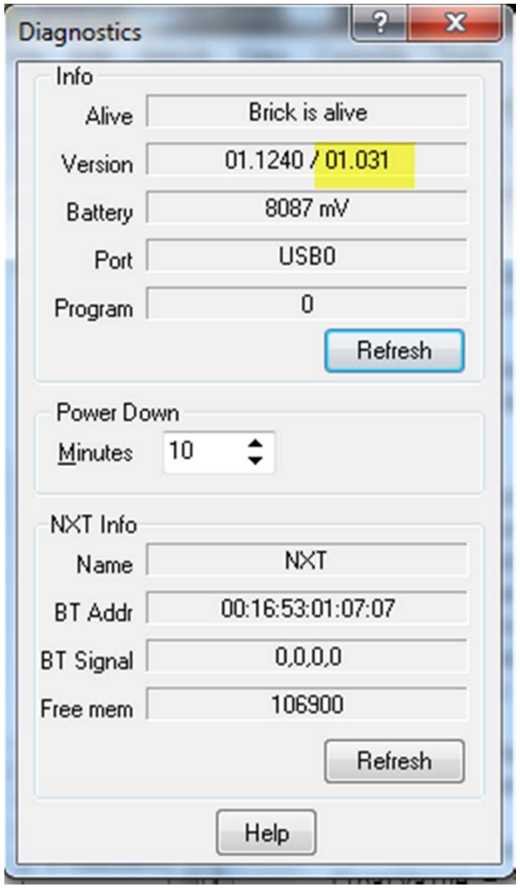
Culprit: Usually means needs firmware update



dataTypes1_0a.nxc

```
// FILE: dataTypes1_0a.nxc
// DATE: 08/25/22 15:34
// AUTH: P.Oh
// DESC: Ilustrate data type errors
// VERS: 1.0a data type errors

task main() {
  int intNumberOfVowels; // should be only 5: A, E, I, O, U
  int intNumberOfLetters; // should be only 26: A thru Z
  int intNumberOfDigits; // should be 10: 0 to 9
  int intInches; // inches as an integer
  float intCentimeters; // centimeters as a real number
  float floatValue; // a calculated value that's a real number
  int intValue; // a calculated value that's an integer

  intNumberOfVowels = 5;
  intNumberOfLetters = 26;

  intValue = intNumberOfLetters/intNumberOfVowels; // division results in a real
  floatValue = intNumberOfLetters/intNumberOfVowels;

  TextOut(0, LCD_LINE1, FormatNum("Letters/Vowels = %f" , intValue));
  TextOut(0, LCD_LINE2, FormatNum("Letters/Vowels = %f" , floatValue));

  Wait(SEC_5); // display for 5 seconds before exiting
  StopAllTasks();
```

```
line 22: Error: Invalid opcode: fmtnum
line 23: Error: Invalid opcode: fmtnum
```



Diagnostics

Info

Alive: Brick is alive

Version: 01.1240 / 01.031

Battery: 8087 mV

Port: USB0

Program: 0

Power Down

Minutes: 10

NXT Info

Name: NXT

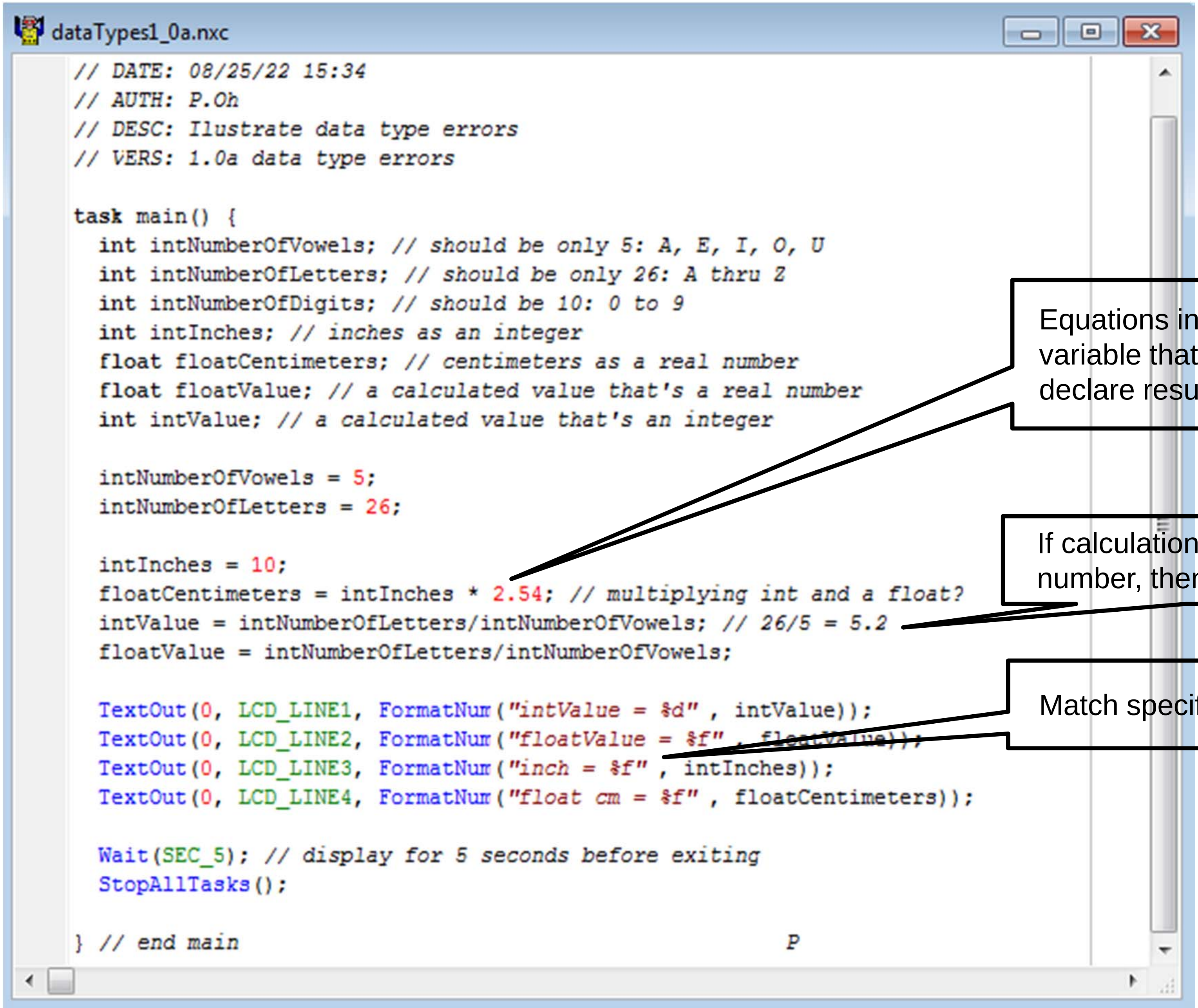
BT Addr: 00:16:53:01:07:07

BT Signal: 0,0,0,0

Free mem: 106900

Solution: `Tools – Download Firmware` and then use file from course site i.e. `lms_arm_nbcnxc_131.rfw`

Bad Practice: Mismatching data types



```
// DATE: 08/25/22 15:34
// AUTH: P.Oh
// DESC: Ilustrate data type errors
// VERS: 1.0a data type errors

task main() {
  int intNumberOfVowels; // should be only 5: A, E, I, O, U
  int intNumberOfLetters; // should be only 26: A thru Z
  int intNumberOfDigits; // should be 10: 0 to 9
  int intInches; // inches as an integer
  float floatCentimeters; // centimeters as a real number
  float floatValue; // a calculated value that's a real number
  int intValue; // a calculated value that's an integer

  intNumberOfVowels = 5;
  intNumberOfLetters = 26;

  intInches = 10;
  floatCentimeters = intInches * 2.54; // multiplying int and a float?
  intValue = intNumberOfLetters/intNumberOfVowels; // 26/5 = 5.2
  floatValue = intNumberOfLetters/intNumberOfVowels;

  TextOut(0, LCD_LINE1, FormatNum("intValue = %d" , intValue));
  TextOut(0, LCD_LINE2, FormatNum("floatValue = %f" , floatValue));
  TextOut(0, LCD_LINE3, FormatNum("inch = %f" , intInches));
  TextOut(0, LCD_LINE4, FormatNum("float cm = %f" , floatCentimeters));

  Wait(SEC_5); // display for 5 seconds before exiting
  StopAllTasks();

} // end main                                    P
```

Equations involving any variable that's a float, then declare results as float

If calculation will result in real number, then declare as float

Match specifier with data type