Functions

CS 110 Bryn Mawr College

Review

- Drawing images/text on a sketch
- Variables and primitive data types
- Data type conversion
- Expressions and Operators
  - Mathematical
  - Relational
  - Logical
- Conditionals
  - if-statement
  - switch-statement

Functions

- A function names a block of code, making it reusable.
- Arguments can be "passed in" to function and used in body.
- Arguments are a comma-delimited set of variable declarations.
- Argument values are ***copies*** of passed values, not originals.
- Function must return a value that matches function declaration.

```
return_type function_name( argument_decl_list ) {
    statements;
    return value;
}
```

Function Examples

```
void setup() { … }
void draw() { … }


void line( float x1, float y1, float x2, float y2) { … }
… and other graphic functions


float float( … )
… and other type-conversion functions

… etc.
```

Functions

Modularity

- Functions allow the programmer to break down larger programs into smaller parts.
- Promotes organization and manageability.

Reuse

- Enables the reuse of code blocks from arbitrary locations in a program.

More Examples

```
/*************************************
** This function squares a number
** Inputs: a value to be squared
** Outputs: returns the square of the number
**                provided
*************************************/
double square (double n) {
   return  n * n;
}
```

```
/*************************************
** Function: FindMinimum ()
** Finds the minimum of two integers
** Inputs: integers n1 and n2 to be compared
** Outputs: returns the smaller of n1 and n2.
*************************************/
int findMinimum (int n1, int n2) {
   int min;
   if (n1 < n2)  {
      min = n1;
   }
   else  {
      min = n2;
   }
   return min;
}
```

Return statement (two forms)

return exp;

- Causes the function to stop and returns the value of exp back to the calling function

return;

- returns program contol back to the calling function.

What happens when we call a function?

1. The argument expressions are evaluated.
2. The resulting values are copied into the corresponding parameters.
3. The statements in the function's body are evaluated in order.
4. When a return statement is evaluated, the value of its argument is used as the function's result.
5. The calling function continues after "substituting" the function's returned value in place of the function call.

Calling Function Example

```
/*****************************
 * This program generates a table of
 * Celsius to Fahrenheit conversions
*************************/
int LOWER = 0;
int UPPER = 100;
int STEP = 5;

void setup() {

  // prints table headings
  println("Celsius to Fahrenheit table.");
  println("  C \t F");

 // generates the table
  for (int celsius = LOWER; celsius <= UPPER; celsius += STEP) {
    int fahrenheit = (int) celsiusToFahrenheit(celsius);
    println(celsius+"\t"+fahrenheit);
  }
}
```

```
/*********************************
 * Function: celsiusToFahrenheit()
 * Converts Fahrenheit to Celsius
 * Inputs: degrees Celsius
 * Output: Returns the Fahrenheit equivalent
*******************************/
double celsiusToFahrenheit(double celsius){
  double fahrenheit = 9.0 / 5 * celsius + 32;
  return fahrenheit;
}
```

Variable Scope

The part of the program from which a variable can be accessed.

Rules:

1. Variables declared in a block are only accessible within the block.
2. Variables declared in an outer block are accessible from an inner block.

Variable Lifetime

- Variables cannot be referenced before they are declared.
- Variables can be declared in…
  - the global scope
  - the body of a function or constructor
  - the arguments of a function or constructor
  - a statement block (for, while, if, …).
- A variable is created and initialized when a program enters the block in which it is declared.
- A variable is destroyed when a program exists the block in which it was declared.

```
int v1 = 1;

void setup() {
  int v2 = 2;

  for (int v3=3; v3 <= 3; v3++) {
    int v4 = 4;
    println("-------------");
    println("v1=" + str(v1));
    println("v2=" + str(v2));
    println("v3=" + str(v3));
    println("v4=" + str(v4));
    //println("v5=" + str(v5));
  }

  int v3 = 6;
  println("v3=" + str(v3));

  aFunction(v2);
}

void aFunction(int v5) {
  println("-------------");
  println("v1=" + str(v1));
  //println("v2=" + str(v2));
  //println("v3=" + str(v3));
  //println("v4=" + str(v4));
  println("v5=" + str(v5));
}

void draw() { }
```

- What is printed?
- What happens if the second v3 declaration is removed?
- What would happen if the v5 print statement is executed?
- What would happen if commented statements in aFunction were called?