**COSC175 functions:  void vs value-returning**

* void function:  name begins with a verb
* value-returning function name is noun or adjective describing the returned result

1. Is each of the following void or value-returning?
2. Write the header declaration for each:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Function | void/value returning? | Write the declaration for each: |
|  | DisplayMenu | *void* | *void DisplayMenu()* |
|  | CalcSquare |  |  |
|  | Square |  |  |
|  | CalcSum |  |  |
|  | Sum |  |  |
|  | GetDimensions |  |  |
|  | PrintDimensions |  |  |
|  | CalcPerimeter |  |  |
|  | Perimeter |  |  |
|  | Max |  |  |

|  |  |
| --- | --- |
| **void function** | **value-returning function** |
| Name begins with Verb | Name usually a noun or adjective |
| May return 0, 1, or many values as arguments | Returns 1 value only |
| General purpose | Usually mathematical |
| Standalone call | Call is part of an expression |

**Some example value-returning functions:** Min, Max, Cos, Sin, Sum, Avg, Diff

**Some example void functions:** ShowDiff. CalcAvg, GetIput, ValidateDate, DisplayHeader**Arguments:**

* **Formal arguments** are declared with the function
* A type (int, float, char, etc) must be included with the formal argument
* There may be 0, 1, or many arguments

**Function Definition**

* Include declaration and body
* body should contain code to implement what their name describes (no more – no less)
* value-returning functions must have return statement
* Any variables used must be declared as arguments or local variables

1. Write a complete declaration, fully specifying each **formal argument**
2. Write the code for the function

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Arguments | Write the declaration for each: | Call the function |
| 1. DisplayMenu | none | *void DisplayMenu()*  *{*  *cout << “Choose a menu option” << endl;*  *cout << “1. Spanish” << endl;*  *cout << “2. French” << endl;*  *cout << “3. German” << endl;*  *cout << “4. English” << endl;*  *cout << “5. Exit” << endl;*  *}* |  |
| 1. CalcSquare | num,square |  |  |
| 1. Square | num |  |  |
| 1. CalcSum | num1,num2, sum |  |  |
| 1. Sum | num1,num2 |  |  |
| 1. GetDimensions | length,width |  |  |
| 1. PrintDimensions | length,width |  |  |
| 1. CalcPerimeter | length, width,perim |  |  |
| 1. Perimeter | length,width |  |  |
| 1. Max | num1,num2 |  |  |

**Calling a Function**

* To call a function, you send **actual arguments**
* Actual arguments do not need to have the same name as the formal arguments
* Formal arguments and actual arguments must match in number and in type
* void function calls are standalone:

exa:      DisplayMenu();

CalcSquare(x, square) ;

CalcCalcSquare(5,ans);

* Function calls are part of an expression, the value returned must be used

Exa:

cout << “The square is “ Square(x);

cout << “The square is “ Square(5);

x = Square(5);

y = 2 \* Square(x) + x + 3;

Show a sample call to each of the procedures/functions you have implemented above:

**Parameter passing methods**

* Specify arguments as /\* in \*/  /\* out \*/  /\* in-out \*/
* In  - pass by value
  + passed into the function
  + default
  + values are known before calling the function
  + values are not changed inside the function
  + example: DisplayName (/\* in \*/string name)
* Out - pass by reference
  + passed out of the function
  + values are not known before calling the function
  + values are known after calling the function
  + values are changed inside the function
  + example: InputName (/\* out \*/string& name)
* In/Out  - pass by reference
  + passed into and out of the function
  + values are changed inside the function
  + example: IncrementX (/\* in/out \*/ int& x)