LEARN PROGRAMMING

STUDY GROUP- SESSION #6

Weekly: Wednesday 19:15 to 22:15
E037 G29

TODAY?

1. Functions again
    a. Functions return values
    b. Functions arguments (parameters)
    c. Function signatures and their value
2. Defining and calling functions
3. Uses of functions
4. Function overloading
5. Identifying pieces of code to be made a function
6. Examples

Functions

A function is a set of code that is put together to perform a certain task

It can be looked at as:

- A grouping of code relevant to a particular task
- A delegation of srots
- Something that hides the intricacies of a particular task

A function needs to be called to make it do its task

To describe it in terms of code it is simply a function name with a block of code in curly braces

The code needed to perform the particular task is inside the block

```
void Func1()
{

    // do something here
}
```

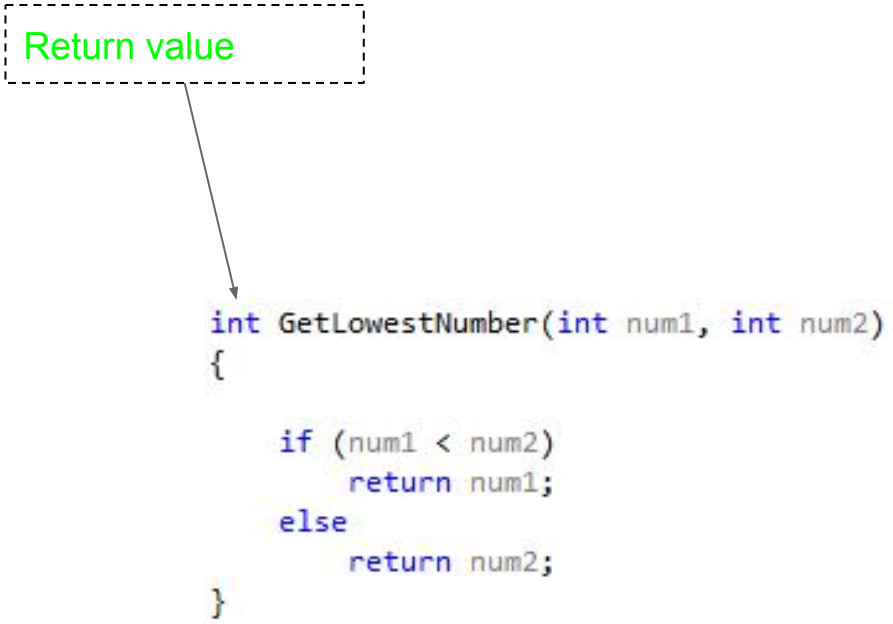
Return values of functions

Functions perform a certain task and return the result of their processing that may be of use to you

Return value may be of the data types you have seen or void

Functions can return only one value

The return statement precedes the return value



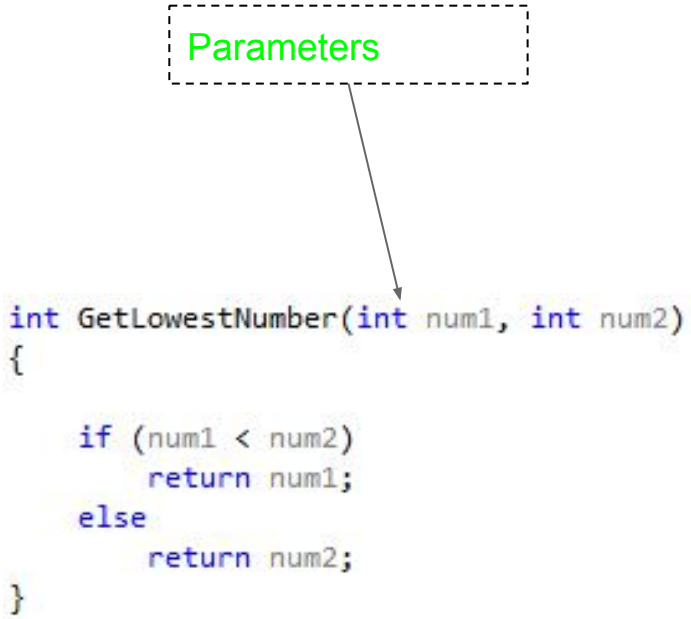
Function parameters

Often to perform a certain task a function needs some input to work on

It can also be in the form of an instruction to the the function in the way we have to perform a task

Functions can have multiple parameters

Parameters can be of any of the data types



A function signature is defined by

- The name of the function
- The return value
- The parameters

Each function has to have a unique signature. Functions can have the same name but the same signature

The function signature has to be kept in mind when calling a function.

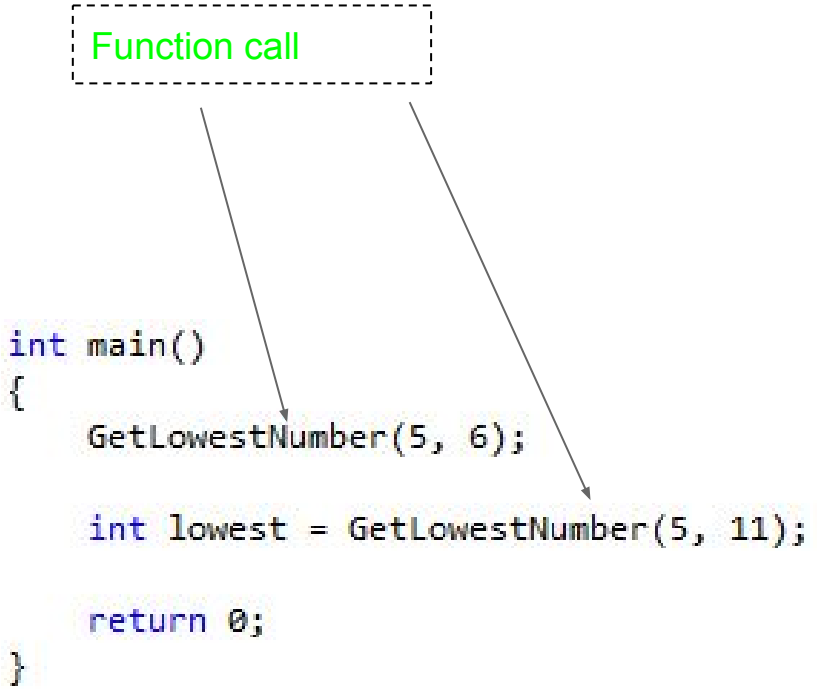
The function’s identity is the signature not the name

Function definitions and calls

Functions are defined by giving the signature of the function and the block of code that performs the task with the return statement if needed

A function is called by giving the function name and the parameters it needs. Capturing the return value is always optional

Based on the signature in the function call the compiler searches for a function with a matching signature and calls it.



USES OF FUNCTIONS

- Re-usability
  - We can call the same function multiple times without re-writing the code
- Organization
  - Grouping things together is an important part of organizing anything
- Readability
  - Often times function names describe what a piece of code is doing which is easier to understand
- Reducing complexity and bugs
  - All you have to do is make sure the code is working fine at one place

IDENTIFYING FUNCTIONS

- Needs practice to identify which piece of code can be converted to functions
- Identify which parts of the code will be needed elsewhere
- Identify parts of the code that are too complex that you want to wrap up
- Identify parts of the code that can be easily generalized
- Identify parts of the code that are not clear on first read

Task For Today? (using functions)

Let us go back to calculating the GPA with functions

$$(grades * credits)/credits = gpa$$

Task For Today? (using functions)

Function Overloading:

WRITE a program using functions:

1. Set 3 variables and based on that type of variable call a relevant function (pass those variable as value to a function).
2. Hint: You have to make 3 user defined functions and call them accordingly.
3. Name of the function should be same.

Function Overloading