Introduction to Programming and Computing for Scientists (2019 HT)

Tutorial-6: First steps with C++ programming (part 2)

Functions in C++

- Theoretically all the code could be written inside a single main() function ...
- However, for maintainability and manageability reasons, it is better to break it into smaller procedures. These are called functions.
- **Implementing a C++ function** involves the following elements:
    - Function definition
        - Consists of header and body
        - Body is the source code that makes up the function
        - Header specifies return value, name and parameter list
    - Function prototype (declaration)
        - Functions must be declared before they are called
        - Prototypes usually specified in header files that are called via the #include statement
        - Declare the function BEFORE the main(){}
    - Function call
        - The statement that executes a function is called a function call
        - Function calls can be specified any time
        - Can be used in assignments
- later topics: pass by reference or pointers

```
int sumup(int x)
{
 int sum, y = 5;
 sum = x + y;
 return sum;
}
```

```
int sumup(int );
```

```
int bignumber, inputnumber;
inputnumber = 12;
bignumber = sumup ( inputnumber );
```

Variable scope within function

- A variable is a “name” that is associated with memory reserved for storing the variable's value.
- Every variable has a name, a type, a value and a scope/lifetime:
- Scope: a variable can be global or local:
    - Variables declared outside functions, including main(), are global. They exist for the duration of a program and can be accessed from anywhere in the code.
- Variables declared inside functions are local to those functions.
    - Local variables may be accessed only inside the block in which they are declared.
    - When a function begins, it allocates space on the stack to hold its local variables.
    - This space exists only while the function is active, after the function returns, it deletes the allocated stack space, including all local variables.

Functions

Exercise 5: In this exercise, you're required to create a user-defined function to capture the program logic of the main program and call that function from main().

Step 1) write a small program that asks for two numbers, compares those numbers and prints out the larger one:

```
// small progrom to find the larger number

int main()
{
// ask the user to  enter two numbers on the keyboard

"enter the first number:"

"enter the second number:"

// compare the two numbers and find out which is the larger

 if ( write here the condition){
   write here what should happen in case the condition is true
 }

//print out the larger number

"The larger number is "
}
```

Functions

Exercise 5: In this exercise, you're required to create a user-defined function to capture the program logic of the main program and call that function from main().
Step 2) Rewrite your monolithic code so that it captures the „logic” in a function

```
#include <iostream>
using namespace std;

int main()
{
 int first, second, larger;
 cout<<"enter the first number:" << endl;
 cin>>first;
 cout<<"enter the second number:" << endl ;
 cin>>second;

 // The program logic that can be turned into a function
 larger = second;

 if (first > second){
  larger= first;
 }
 // Printing the result
 cout << "The larger number is " << larger  << endl ;
}
```

Functions

Exercise 5: In this exercise, you're required to create a user-defined function to capture the program logic of the main program and call that function from main(). Step 2) Rewrite your m

```
#include <iostream>

//function declaration

type name_of_the_function(type parameter 1, type parameter 2);
------------------------------------------------------------------------------------------
int main() {

 //function call

 larger =  name_of_the_function();

 cout << "The larger number is " << larger  << endl ;
}

------------------------------------------------------------------------------------------
//function definition

type name_of_the_function(type parameter 1, type parameter 2) {

// Write the actual function code here

return some_variable;

}
```

Functions and scope of variables

<u>Exercise 6:</u> The program below will not compile because of scope errors. Investigate which variables are used out-of-scope and comment out the corresponding lines.

```
#include <iostream>
using namespace std;
int globalScope = 0; //This is a global variable, visible everywhere.

void foo() {
   int fooScope = 1; //Only visible within foo function
   cout << "fooScope: " << fooScope << endl;
   cout << "localScope: " << localScope << endl;
}
int main() {
   cout << "globalScope: " << globalScope << endl;

   { //Any block declares a scope, even this useless one
      int localScope = 3;
      cout << "localScope: " << localScope << endl;
      foo();
      cout << "fooScope: " << fooScope << endl;
      int globalScope = 100; // variable hiding, very bad practice!
      cout << "globalScope: " << globalScope << endl;
   }
   cout << "localScope: " << localScope << endl;
   cout << "globalScope: " << globalScope << endl;
}
```

Homework

- You are asked to fix a broken code. See details in Canvas.