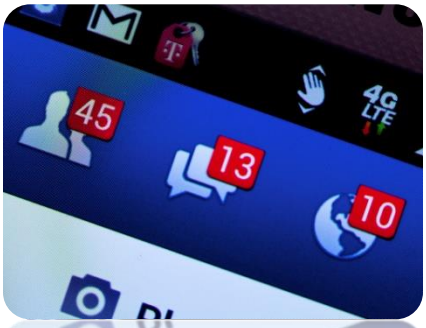
Computational Problem Solving (CS100)

Shafay Shamail, Arif Zaman and

Safee Ullah Chaudhary



Lab 9 Objectives:

- Functions

Lab Guidelines

1. Make sure you get your work graded before the lab time ends.
2. You put all your work onto the LMS folder designated for the lab (i.e. “Lab09”) before the time the lab ends.
3. Talking to each other is NOT permitted. If you have a question, ask the lab assistants.
4. If you are a hot-shot C++ expert, you are still not allowed to use any feature of C++ that has not been covered in class or in the lab.
5. The object is not simply to get the job done, but to get it done in the way that is asked for in the lab.

Functions

Functions allow to structure programs in segments of code to perform individual tasks.

In C++, a function is a group of statements that is given a name, and which can be called from some point of the program. The most common syntax to define a function is:

```
type name ( parameter1, parameter2, ...)
{
      statements
}
```

Where:

- `type` is the type of the value returned by the function.

- `name` is the identifier by which the function can be called.

- `parameters` (as many as needed or none at all): Each parameter consists of a type followed by an identifier, with each parameter being separated from the next by a comma. Each parameter looks very much like a regular variable declaration (for example: `int x`), and in fact acts within the function as a regular variable which is local to the function. The purpose of parameters is to allow passing arguments to the function from the location where it is called from.

- `statements` is the function's body. It is a block of statements surrounded by braces { } that specify what the function actually does.

http://www.cplusplus.com/doc/tutorial/functions/

Task 1

a) Write a function with no parameters. When called, this function should just print:

```
I am a function, I was executed
```

b) Modify this function so that it takes a number N as input and displays the above statement, N times. For example if N = 6, the program should output:

```
I am a function, I was executed
I am a function, I was executed
I am a function, I was executed
I am a function, I was executed
I am a function, I was executed
I am a function, I was executed
```

CHECKPOINT 1

STOP

Show your program and result to the TA.

Task 2

a) You have to design a calculator which can perform four basic arithmetic operations: addition, subtraction, multiplication and division. You have to write a separate function for each operation.

The functions should take 2 doubles as input and return the correct answer.

b) Make a function named calculate.

This function takes three arguments: The left number, the operation and the right number.

The operation is a character that can be '+', '-', '*' or '/'
The calculate function must use the four functions made previously and run the appropriate function on the two numbers depending on what the character is.

CHECKPOINT 2
STOP

Show your program and result to the TA.

Task 3

You want to send the coordinates of a secret location to a friend. For this purpose you have to convert the coordinates into morse code (Assume the coordinates are whole numbers). Each digit is represented by the following pattern in morse code.

Number	Morse code	Number	Morse code
0	- - - - -	5	. . . . .
1	.- - - -	6	- . . . .
2	..- - -	7	- - . . .
3	. . . - -	8	- - - . .
4	. . . . -	9	- - - - .

For example, if the x and y coordinates are 132 and 591 respectively, then the morse code that should be generated is:

```
.----...--..---     .....----..----
```

Write a function in C++ that takes x and y coordinates as input and returns the morse code as a string.

CHECKPOINT 3

STOP

Show your program and result to the TA.

Task 4

A number p>1 is prime if the only numbers that divide it evenly (without a remainder) are 1 and the number itself. Since we are going to be looking for large primes, declare p to be a long long integer (nope, not a typo).

Note that if a number is not prime, then at least one of the divisors is going to be less than or equal to sqrt(n).

The trial division algorithm to test if a number p is prime can be described by the following idea:

> Consider all trial divisors, d, from 2 to sqrt(p)
> If any d divides p evenly, then p is not prime.
> If none of the d's could divide p evenly then p is a prime.

a) Write a function in C++ that checks whether a number is a prime number or not. The function should take a number as input and return true when the number is prime and false when the number is not prime.

b) Write another function that prints all the prime number between a range provided by the user. For instance, if the user enters 10 and 50, the function should output 11 13 17 19 23 29 31 37 41 43 47.

c) Find all the prime numbers between 10,000,000,000,000 and 10,000,000,000,100. Since these are large numbers, your program may be slow. Try to be efficient, so that you get the smallest runtime.

For example: Note that as soon as you find one divisor that divides p, you know it is not prime, so you should stop the loop. If you are continuing to loop after the first divisor, you are wasting your time, so this will take too long. If you have a good loop this process should only take a few seconds at the most.

Another way to save time is to avoid computing the square root many times. How could you do that?

The list should be

10000000000037

10000000000051

10000000000099

CHECKPOINT 4

STOP

Show your program and result to the TA.

Zip your tasks into one folder with format:
example "**18100012-Lab9**" and upload on LMS before the tab is closed. You will not be given extra time