CSCI1300 - Introduction to Computer Programming
Instructor: Dr Knox
Due by Sunday 5PM

Recitation 6: For Loops

In this recitation we will go through `for` loops and then compare them with `while` loops and provide a few examples. The previous programs in class used `while` loops. `for` loops are built on the same core principle, but with different syntax. They will repeat execution of a single statement or group of statements as long as a specified condition continues to be satisfied. If the condition is not true, then the statement will not be executed.

Working With For Loops

In addition to `while` loops, the `for` loop is another alternative way to add repetition to your program. As opposed to `while` loops, all the necessary values/conditions - the initial values, stopping condition and updating of the counter variables is all packed in a single line. The syntax for the `for` loop is as follows -

```
for (start value; end condition; update variable value ) {
  //Code to execute until the above mentioned condition is false
}
```

Start value

- You can either declare the variable and assign a value to it or give a value to a variable which had been already declared.

Condition

- This is an important aspect of the for loop where you instruct the `for` loop to repeat the block of code until the condition is false.

Update variable value

- We can update the value by any number and is dependent on your task.

Example of For Loop:

```
#include <iostream>
using namespace std;
int main()
{
  // The loop runs the block of code until i is less than 10 (i<10)
  //The value of i increases by one in each loop.
  for ( int i = 0; i < 10; i++ ){
     // Keep in mind that the loop condition checks
     //  the conditional statement before it loops again.
     //  When the i value equals 10 the loop breaks
    cout << “The value of i is :” << i <<endl;
  }
}
```

The equivalent of this loop can be written in a while loop as:

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

int main()
{
  int i = 0;                            // declare the variable
  while ( i < 10 ){                     // While x is less than 10
    cout<< “The value of i is :”<< i <<endl;
    i++;         //If the value of i not updated it will end up in infinite loop
  }
}
```

Modulus Operator:

A modular operator, mod, is used when we’re interested in finding the remainder when dividing two integers. In normal division we see equations such as:

$$A/B = Q \text{ remainder } R$$

In modular math, for the same statement we would have: A mod B = R i.e. A, when divided by B gives remainder R. While programming we use % symbol for mod operations.

String Operations using For Loops

So far, we've learnt to use strings as a datatype and using the built in functions of the string library such as **`length`, `replace` and `substr`**. In this recitation, we'll use the same concepts and extend them using for loops. To review, strings are sequences of characters and can be accessed using the syntax `name[index]`. **Remember- Indices begin from 0 and not 1.** So, a string of length N would have indices from 0 to N-1.

Example

```
string str = "Hello CS1300!";
cout<< str[0] ; //This would print 'H' on the screen.
cout<< str[1]; //This would print 'e' on the screen.
cout<< str[9]; //This would print '3' on the screen.
```

We can now extend these concepts to for loops. Recall that a typical while loop to traverse a string character by character would look like this.

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

int main()
{
  string str = "Hello CS1300";           //declare the string variable
  int i = 0;                             //declare the index variable
  //notice how the condition is < and not <=, this is because str.length returns N and i goes only from 0 to N-1
    while ( i < str.length() ){         //while loop and stopping condition
  //Perform operation here
      i++;            //Remember to update the index variable
  }
}
```

An equivalent for loop to this would be as follows

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

int main()
{
  string str = “Hello CS1300”;           //declare the string variable
   //Declare index variable, specify  starting and  stopping condition and update in a
single line
   for(int i =0; i < str.length(); i++){
   //Do your processing here
    }
}
```

Recitation 6 Activity:

There is a link of codeRunner for Recitation Activity. Please write and run your code there.

For this recitation activity we’ll be doing two programs. You will be evaluated for your use of For loops in these tasks.

1. The first program simulates the word game “FizzBuzz”. The task is to create a function **`fizzBuzz`** that takes in three arguments **- `N`** (the number we are going to count upto), **`divisor_1`** (the first divisor) and **`divisor_2`** (the second divisor). The function would check if all numbers starting from 1 to N are divisible by both the divisors. The function returns no value. For your output-
   a. If the number is divisible the **`divisor_1`** print “Fizz”.
   b. If the number is divisible by the **`divisor_2`** print “Buzz”
   c. If the number is divisible by both the first and the second divisor, print “FizzBuzz”
   d. For everything else, print the number itself.

   For example if `N= 15, divisor_1 = 3` and `divisor_2 = 5,` then the output would be-
   ```
   1
   2
   Fizz
   4
   Buzz
   Fizz
   7
   8
   Fizz
   Buzz
   11
   Fizz
   13
   ```

```
14
FizzBuzz
```

2. Given a character search for all occurrences of the character in the string and print the character as well as the next 3 characters(hint: use `substr` for this). Create a function `findChar` for the above task. This function takes in two arguments- the string and the character and prints out the result on the screen. It returns no value.

   Example: Given the string **`abracadabra`** and the character **`a`**, your output should be-

   ```
   abra
   acad
   adab
   abra
   a
   ```