Control Structures 2: Loops

In this Week

- Motivation: Why Loops?
- The ***while*** loop

Why Loops?

- So far all our program statements have been executed at most once; unless they were skipped because of conditional control structures such as **if**, **else-if** or **else** statements
- Sometimes, we may require certain statements in our program to be executed more than once
- For example, suppose we would like to print the integers 0,1,2,…n where n is a user input value
- Based on our previous lessons, we would require n **cout** statements in order to achieve this simple task

Why Loops?

- Yet, this task is so simple that it would make sense to execute ONE **cout** statement n times making sure that a certain variable starts at a value 0 and keeps incrementing by 1 after the execution of each cout statement
- C++ provides control structures that enable us to achieve such repeated execution of C++ statement(s)
- Such control structures are called **loops**
- The number of times a **loop** executes given statements in its block is known as the number of ***iterations*** of the loop

Loops

- C++ provides three types of loops
- These are
  - **The *while* loop**
  - **The *for* loop, and**
  - **The *do-while* loop**

The *while* loop

- **Syntax**

```
while (Boolean_expression)
{
        Block of the while loop
}
```

- The while loop block is executed as long as the Boolean expression is evaluated to true; and execution goes below the block only when the Boolean expression is evaluated to false
- The curly brackets designate the block of the while loop. They can be omitted if the block contains only one statement
- If the curly brackets are omitted then **only the first** statement belongs to the block

The *while* loop

- The following program reads an integer value **n** from the user and then uses a while loop to print all the integers 0, 1, 2, ..., **n**

```
int main()
{
    int n;
    cout << "Please enter a number ";
    cin >> n;
    int k = 0;
    while (k <= n)
    {
        cout << k << endl;
        k = k + 1;
    }

    system("Pause");
    return 0;
}
```

- **How many iterations doe the loop perform? What will be the output of the program if n is negative?**

Increment/Decrement Operators inside Boolean Expressions

- We may use the unary pre/post increment/decrement operators inside Boolean expressions in order for the updating of a variable to be part of the Boolean expression
- Analyze the following examples and determine their outputs

int n = 0;
while (n++ < 5)
cout << ++n << endl;
	int n = 0;
	while (++n < 5)
	cout << n++ << endl;
		int n = 0;
		while (n++ < 5)
		cout << n++ << endl;

int n = 0;
while (++n < 5)
cout << ++n << endl;
	int n = 0;
	while (n+1 < 5)
	cout << n++ << endl;
		int n = 0;
		while (++n < 5)
		cout << n+1 << endl;

Casting and Boolean Expressions

- Sometimes we may use an expression that is not strictly speaking a Boolean expression in the place of the Boolean expressions and rely on automatic casting
- Analyze the following program and determine its output

```
int main()
{
    int k = 5;
    while (k)
    {
        k *= -1;
        cout << k << endl;
        k > 0 ? k-- : k++;
    }
    system("Pause");
    return 0;
}
```

Infinite Loops

- One of the most common mistakes in looping structures is getting into an infinite loop
- An infinite loop is a looping structure that does infinite iterations and therefore will at some point run out of memory and end up with a **runtime error**
- Most often the cause of an infinite loop mistake is not paying attention to the updating of a variable inside the while loop block or in the Boolean expression of the while loop block; that is if we do not modify the value of a variable being used to control the while loop structure, then we will end up with an infinite loop
- Consider the following program and determine its output if the user input for n is 1.

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout << "Demonstrating an infinite C++ while loop." << endl;
    cout << "Please enter an integer ";
    cin >> n;
    int k = 0;
    while (k <= n)
    {
        cout << "The value of k is " << k << endl;
        k+1;
    }

    system("Pause");
    return 0;
}
```

while loop: practice questions

1. Write a program to print the even integers 0,2,4,6,…28 using a *while* loop
2. Write a program that declares an integer variable ***n***, assigns ***n*** an integer input from from the user, and finally prints the integers ***n, n-1, n-2, …, 1.*** What is the output of the program if the input value for n ***n*** is less than ***1.***
3. Write a program that reads an integer value **n** and then prints **n** randomly generated integers in the range [-10, 10].
4. Write a program that reads an integer value **n** and then prints **n** randomly generated integer numbers in the range [-10, 10] and finally prints the minimum of the numbers. Assume the user input value for **n** is a positive integer.
5. Write a C++ program that declares two integers ***a*** and ***b***, assigns each of the variables a random integer in the range [-10, 15], and then prints all the integers between a and b (or vice versa) **exclusive** (that is without including **a** and **b**).
6. Write a program that prints the integers 7,10, 13, 16,… **n** exclusive (that is without including **n**) where **n** is the first integer divisible by 41.
7. Write a program that reads an integer **n** greater than 1 from user. If the user input value for **n** is not greater than 1 then your program must keep on asking the user for the input until a number greater than 1 is entered. Finally your program must print the message **the number is prime** or the message **the number is not prime**. **N.B.:-** An integer number **n** is prime if none of the integers 2,3,4,…,**n**-1 divides **n**.