Aror University of Art, Architecture, Design & Heritage Sukkur.

Department of Artificial Intelligence and Multimedia Gaming

Fundamentals of Programming (Fall-2023)

LAB No. 07

Prepared by: Abdul Haseeb Shaikh

Objective of Lab No. 07:

After performing lab 7, students will be able to:

- Use While Loop in C++
- Use Do While Loop in C++
- Use Nested Loops in C++
- Use Jump Statements in C++

While Loop:

While Loop in C++ is used in situations where we do not know the exact number of iterations of the loop beforehand. The loop execution is terminated on the basis of the test condition.

The various parts of the While loop are:

1. **Test Expression:** In this expression, we have to test the condition. If the condition evaluates to true then we will execute the body of the loop

and go to update expression. Otherwise, we will exit from the while loop.

2. **Update Expression:** After executing the loop body, this expression increments/decrements the loop variable by some value.
3. **Body:** This is a group of statements that include variables, functions, and so on. With the while loop, code, and simple names can be printed, complex algorithms can be executed, or functional operations can be performed.

Syntax:

```
while (test_expression)
{
   // statements

  update_expression;
}
```

Example:

```cpp
// C++ program to illustrate while loop

#include <iostream>
using namespace std;

int main()
{
    // initialization expression
    int i = 1;

    // test expression
    while (i < 6) {
        cout << "Hello World\n";

        // update expression
        i++;
    }

    return 0;
}
```

Do While Loop:

Like while the do-while loop execution is also terminated on the basis of a test condition. The main difference between a do-while loop and a while loop is in the do-while loop the condition is tested at the end of the loop body, i.e do-while loop is exit controlled whereas the other two loops are entry-controlled loops.

```
do
{
   // loop body

   update_expression;
}
while (test_expression);
```

Example:

```
// Initialization expression
int i = 2;

do {
    // Loop body
    cout << "Hello World\n";

    // Update expression
    i++;

}
// Test expression
while (i < 1);
```

Nested Loop:

Loop inside a Loop is called Nested Loop, we can nest any loop inside any other loop, and there is also no any restriction on the number of nesting levels, One example of nested loop is shown below:

```
int i=1;
int j;

while(i<=5){
    j=1;
    cout<<"Multiplication Table of: "<<i<<endl;
    while(j<=10){
        cout<<i<<"*"<<j<<"="<<i*j<<"\n";
        j+=1;
    }
    i++;
}
```

The Loop shown in the given image is a nested loop, this loop prints the multiplication table of first five even numbers.

Jump Statements:

Jump statements manipulate the flow of program if some conditions are met.

- Four Types:

- **Break:** It forces the loop to terminate, upon meeting a condition, or jump out of a switch case statement.

```cpp
// C++ program to demonstrate the
// break statement
#include <iostream>
using namespace std;

// Driver Code
int main()
{
    for (int i = 1; i < 10; i++) {

        // Breaking Condition
        if (i == 5)
            break;
        cout << i << " ";
    }
    return 0;
}
```

- **Continue:** Executes some parts of the loop, while skipping others

```cpp
// C++ program to demonstrate the
// continue statement
#include <iostream>
using namespace std;

// Driver code
int main()
{
    for (int i = 1; i < 10; i++) {

        if (i == 5)
            continue;
        cout << i << " ";
    }
    return 0;
}
```

- **Goto:** C++ goto statement can take control to any part of the program.

```cpp
    int n = 4;

    if (n % 2 == 0)
        goto label1;
    else
        goto label2;

label1:
    cout << "Even" << endl;
    return 0;

label2:
    cout << "Odd" << endl;
```

- **Return:** It terminates the entire function, and it is stronger than a break statement

```cpp
// C++ program to demonstrate the
// return statement
#include <iostream>
using namespace std;

// Driver code
int main()
{
    cout << "Begin ";

    for (int i = 0; i < 10; i++) {

        // Termination condition
        if (i == 5)
            return 0;
        cout << i << " ";
    }
    cout << "end";

    return 0;
}
```

Lab Exercises:

1. Write a C++ program that calculates the factorial of a number entered by the user using a while loop. Factorial is the multiplication of any given number with all the numbers before it until 1, like factorial of 5 will be (5*4*3*2*1)
   Note: For Negative Numbers there doesn't exists a factorial.

2. Create a program that calculates the result of a base raised to an exponent using a while loop.

3. Write a C++ program that calculates the sum of all even numbers between two numbers entered by the user using a do while loop.

4. Guess a Number Game: Write a C++ program that generates a random number between 1 and 100 and asks the user to guess it. Use a while loop to provide hints (higher or lower) until the user guesses the correct number.

5. Write down C++ code to print the following pattern using a nested loop:

6. Create a program that calculates the sum of all numbers from 1 to 100, excluding the multiples of 5, Note: You must use a Jump statement in this task, and use a while or do while loop.

7. Build a simple calculator with a menu that allows the user to perform basic operations (addition, subtraction, multiplication, division). Use a do-while loop to keep the program running until the user chooses to exit.