CS1002 – Programming Fundamentals

Lecture # 18
Tuesday, October 25, 2022
FALL 2022
FAST – NUCES, Faisalabad Campus

Muhammad Yousaf

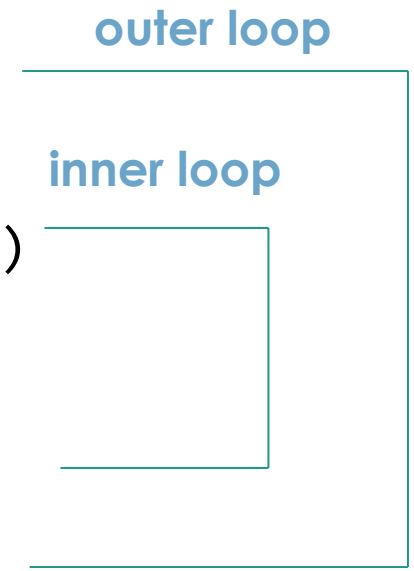
Outline

- Nested Control Structures
- Nested Loops
- Examples

Nested Control Structures (Nested Loops)

- A nested loop is a loop inside the body of another loop
- Example:

```
for (row = 1; row <= 3; row++)
{
   for (col = 1; col <= 3; col++)
   {
      cout << row * col << endl;
   }
}
```



Nested Control Structures

- To create the following pattern:

```
*****
*****
*****
*****
*****
```

- We can use the following code:

```
for (int i = 1; i <= 5 ; ++i)
{
    for (int j = 1; j <= 5; ++j)
        cout << "*";
    cout << endl;
 }
```

Nested Control Structures

- To create the following pattern:

```
*
**
***
****
*****
```

- We can use the following code:

```
for (int i = 1; i <= 5 ; ++i)
{
    for (int j = 1; j <= i; ++j)
        cout << "*";
    cout << endl;
}
```

Nested Control Structures (cont'd.)

- What is the result if we replace the first for statement with the following?

```
for (int i = 5; i >= 1 ; i--)
{
    for (int j = 1; j <= i; ++j)
        cout << "*";
    cout << endl;
}
```

Answer:

```
*****
****
***
**
*
```

Example

Suppose you want to create the following multiplication table:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50

Step-1

Count the number of rows?

That's External loop?

Step-2

Count the noumber of columns?

That's Inner Loop!

```
for (   ???     )
{
    ???
}
```

Step-3

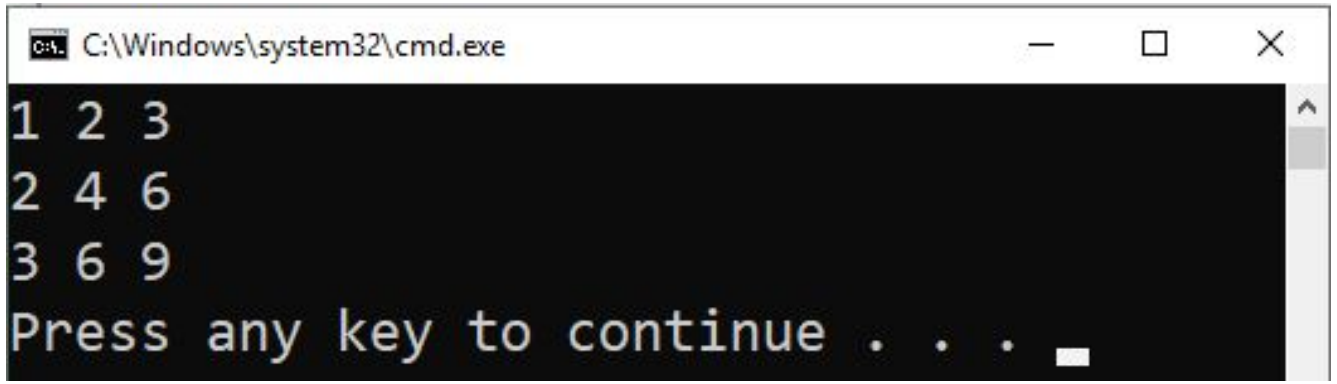
What pattern to print?
Play with i and j!

Notes on Nested Loops

- Inner loop goes **through all its repetitions** for each repetition of outer loop
- Inner loop repetitions complete **sooner** than outer loop
- Total number of repetitions for inner loop is product of number of repetitions of the two loops

Output = ?

```
for (int y = 1; y <= 3; ++y)
{
   for (int x = 1; x <= 3; ++x)
  {
      cout << x * y << " ";
   }
   cout << "\n";
}
```



Loop Structures

- **Problem statement**

  A college has a list of test results (1 = pass, 2 = fail) for 10 students. Write a program that analyzes the results. If more than 8 students pass, print "Raise Tuition".

- **Notice that**
  - **Program processes 10 results**
    - Fixed number, use counter-controlled loop
  - **Two counters can be used**
    - One counts number that passed
    - Another counts number that fail
  - **Each test result is 1 or 2**
    - If not 1, assume 2

Loop Structures

- **Top level outline**
  Analyze exam results and decide if tuition should be raised
- **First refinement**
  Initialize variables
  Input the ten quiz grades and count passes and failures
  Print a summary of the exam results and decide if tuition should be raised
- **Refine**
  Initialize variables
  to
  Initialize passes to zero
  Initialize failures to zero
  Initialize student counter to one

Loop Structures

- Refine

Input the ten quiz grades and count passes and failures

to

While student counter is less than or equal to ten

Input the next exam results

If the student passed

Add one to passes

Else

Add one to failures

Add one to student counter

Loop Structures

- Refine

Print a summary of the exam results and decide if tuition should be raised

to

Print the number of passes

Print the number of failures

If more than eight students passed

Print "Raise tuition"

```
#include<iostream>
using namespace std;
int main()
{
    int passes = 0;       //number of passes
    int failures = 0;     //number of failures
    int stdCounter = 1;   //student counter
    int result;           //one exam result
    //process 10 users using counter-controlled loop
    while (stdCounter <= 10)
    {
        //prompt user for input and obtain value from user
        cout << "Exam Result ( 1 = pass , 2 = fail): ";
        cin >> result;

        //if 1 increment passes
        if (result == 1) //if-else nested in while loop
            ++passes;
        else             //if result not one increment failures
            ++failures;

        //increment student counter so that loop can terminate
        ++stdCounter;
    } //end while
```

```
    //termination phase
    cout << "Passes = " << passes << endl;
    cout << "Failures = " << failures << endl;

    // if more than 8 students passed then raise tution
    if (passes > 8)
        cout << "Raise Tution" << endl;

    return(0);
}
```

```
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Passed 6
Failed 4
```

Avoiding Bugs by Avoiding Patches

- Software patch
  - Piece of code written on top of an existing piece of code
  - Intended to fix a bug in the original code
- Some programmers address the symptom of the problem by adding a software patch
- Should instead resolve underlying issue

Debugging Loops

- Loops are harder to debug than sequence and selection structures
- Use loop invariant
  - Set of statements that remains true each time the loop body is executed
- Most common error associated with loops is off-by-one

Examples

Example program

What is the out put of following code segment?

```
int count = 0;
while (count++ < 10)
    cout << "This loop repeats statements." <<endl;
```

```
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
This loop can repeat statements.
```

What is the out put of following code segment?

```
int count = 5;
while(--count > 0)
    cout << count << " " ;
cout << endl ;
```

```
4 3 2 1
```

What is the out put of following code segment?

```
int count = 5 ;
while (count-- > 0)
    cout << count << " " ;
cout << endl;
```

```
4 3 2 1 0
```

What is the out put of following code segment?

```
int count = 1 ;
while (count++ <= 5)
    cout << count * (count - 2) << " " ;
cout << endl;
```

```
0 3 8 15 24
```

State what output, if any, results from each of the following statements:

a.
```
for (i = 1; i <= 1; i++)
    cout << "*";
cout << endl;
```

```
*
```

b.
```
for (i = 2; i >= 1; i++)
    cout << "*";
cout << endl;
```

Infinite loop

c.
```
for (i = 1; i <= 1; i--)
    cout << "*";
cout << endl;
```

Infinite loop

d.
```
for (i = 12; i >= 9; i--)
    cout << "*";
cout << endl;
```

```
****
```

e.
```
for (i = 0; i <= 5; i++)
    cout << "*";
cout << endl;
```

```
******
```

f.
```
for (i = 1; i <= 5; i++)
{
    cout << "*";
    i = i + 1;
}
cout << endl;
```

```
***
```

What is the output of the following code? Is there a relationship between the variables **x** and **y**? If yes, state the relationship? What is the output?

```
int x = 19683;
int i;
int y = 0;

for (i = x; i >= 1; i = i / 3)
    y++;
cout << "x = " << x << ", y = " << y << endl;
```

```
x = 19683, y = 10
```

Suppose that the input is 5 3 8. What is the output of the following code? Assume all variables are properly declared.

```
cin >> a >> b >> c;
for (j = 1; j < a; j++)
{
    d = b + c;
    b = c;
    c = d;
    cout << c << " ";
}
cout << endl;
```

```
5
3
8
11 19 30 49
```

Suppose that the input is **38 35 75 44 -1**. What is the output of the following code? Assume all variables are properly declared.

```
sum = 0;
cin >> num;
for (j = 0; j <= 3; j++)
{
    cin >> num;
    sum = sum + num;
}
cout << "Sum = " << sum << endl;
```

```
38
35
75
44
-1
Sum = 153
```

```
int i = 0, value = 0;
while (i <= 20)
{
    if (i % 2 == 0 && i <= 10)
        value = value + i * i;
    else if (i % 2 == 0 && i > 10)
        value = value + i;
    else
        value = value - i;
    i = i + 1;
    cout << "value = " << value << endl;
}
```

```
value = 200
```

What is the output

```
int num = 12;
while (num >= 0)
{
    if (num % 5 == 0)
    {
        num++;
        continue;
    }
    cout << num << " ";
    num = num - 2;
}
cout << endl;
```

```
12 11 9 7 6 4 2 1
```

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

int main()
{
    int x, y, z;
    x = 4; y = 5;
    z = y + 6;
    do
    {
        cout << z << " ";
        z = z + 7;
    }
    cout << (((z - x) % 4) != 0);
    cout << endl;
}
```

```
11 18 25
```

The do...while loop in the following program is supposed to read some numbers until it reaches a sentinel (in this case, -1). It is supposed to add all of the numbers except for the sentinel. If the data looks like:

```
12    5    30    48    -1
```

the program does not add the numbers correctly. Correct the program so that it adds the numbers correctly.

```
#include <iostream>

using namespace std;
int main()
{
   int total = 0,
       count = 0,
       number;
   do
   {
       cin >> number;
       total = total + number;
       count++;
   }
   while (number != -1);

   cout << "The number of data read is " << count << endl;
   cout << "The sum of the numbers entered is  " << total
        << endl;

   return 0;
}
```

```
12
5
30
48
-1
The number of data read is 5
The sum of the numbers entered is 94
```

a.

```
int i, j;
for (i = 1; i <= 5; i++)
{
    for (j = 1; j <= 5; ++j)
        cout << setw(3) << i;
    cout << endl;
}
```

```
  1  1  1  1  1
  2  2  2  2  2
  3  3  3  3  3
  4  4  4  4  4
  5  5  5  5  5
```

b.

```
int i, j;
for (i = 1; i <= 5; i++)
{
    for (j = (i + 1); j <= 5; ++j)
        cout << setw(3) << j;
    cout << endl;
}
```

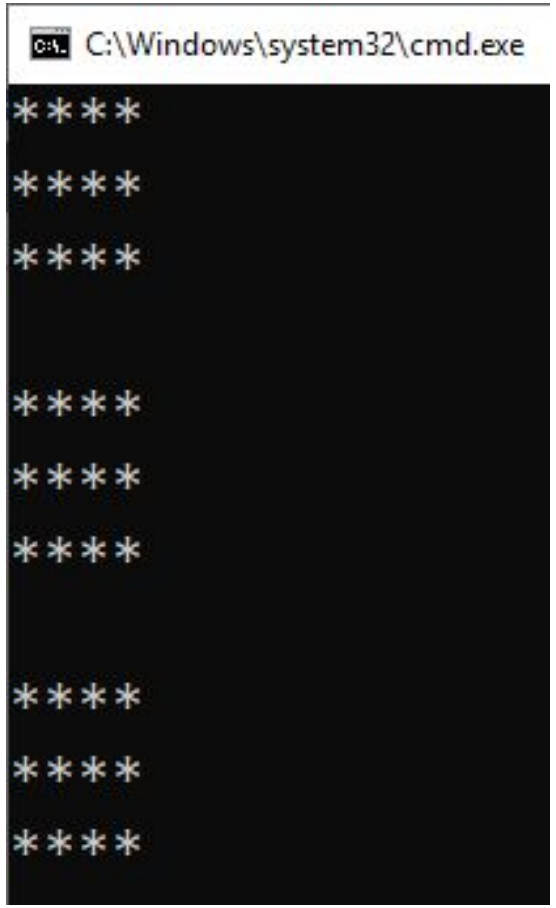
```
  2  3  4  5
  3  4  5
  4  5
  5
```

Three-Nested-Loops

What will be the output of the following code

```
int main()
{
    for (int i = 1; i <= 3; ++i)
    {
        for (int j = 1; j <= 3; ++j)
        {
            for (int k = 1; k <= 4; ++k)
                cout << '*';
            cout << endl;
        } // end inner for

        cout << endl;
    } // end outer for
}
```



```
int i, j;
for (i = 1; i <= 5; ++i)
{
   for (j = 1; j <= i; ++j)
      cout << setw(3) << j;
   cout << endl;
}
```

C:\Windows\system32\cmd.exe

```
  1
  1   2
  1   2   3
  1   2   3   4
  1   2   3   4   5
```

```
const int M = 10;
const int N = 10;
int i, j;
for (i = 1; i <= M; ++i)
{
    for (j = 1; j <= N; ++j)
        cout << setw(3) << M * (i - 1) + j;
    cout << endl;
}
```

C:\Windows\system32\cmd.exe

```
  1  2  3  4  5  6  7  8  9 10
 11 12 13 14 15 16 17 18 19 20
 21 22 23 24 25 26 27 28 29 30
 31 32 33 34 35 36 37 38 39 40
 41 42 43 44 45 46 47 48 49 50
 51 52 53 54 55 56 57 58 59 60
 61 62 63 64 65 66 67 68 69 70
 71 72 73 74 75 76 77 78 79 80
 81 82 83 84 85 86 87 88 89 90
 91 92 93 94 95 96 97 98 99100
```

```
int i, j;
for (i = 1; i <= 9; ++i)  //Outer loop
{
    for (j = 1; j <= (9 - i); ++j) //first loop
        cout << " ";

    for (j = 1; j <= i; ++j) //second loop
        cout << setw(1) << j;

    for (j = (i - 1); j >= 1; --j) //third loop
        cout << setw(1) << j;
    cout << endl;
}
```

C:\Windows\system32\cmd.exe

```
1
121
12321
1234321
123454321
12345654321
1234567654321
123456787654321
```

Summary

- C++ has three looping (repetition) structures:
- **while, for**, and **do...while**
- **while, for**, and **do** are reserved words
- **while** and **for** loops are called pretest loops
- **do...while** loop is called a posttest loop
- **while** and **for** may not execute at all, but **do...while** always executes at least once

Summary (cont'd.)

- **while**: expression is the decision maker, and the statement is the body of the loop
- A while loop can be:
  - Counter-controlled
  - Sentinel-controlled
  - Flag-controlled
  - EOF-controlled
- In the Windows console environment, the end-of-file marker is entered using Ctrl+z

Summary (cont'd.)

- **for** loop: simplifies the writing of a counter-controlled while loop
  - Putting a semicolon at the end of the for loop is a semantic error
- Executing a **break** statement in the body of a loop immediately terminates the loop
- Executing a **continue** statement in the body of a loop skips to the next iteration

Questions