Nested
For Loops

اَللّٰهُمَّ اُرْزُقْنِى عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ اُرْزُقْنِى رِزْقًا وَاسِعًا حَلَالًا طَيِّبًا
مُبَارَكًا مِنْ عِنْدِكَ

Single Loop: Review

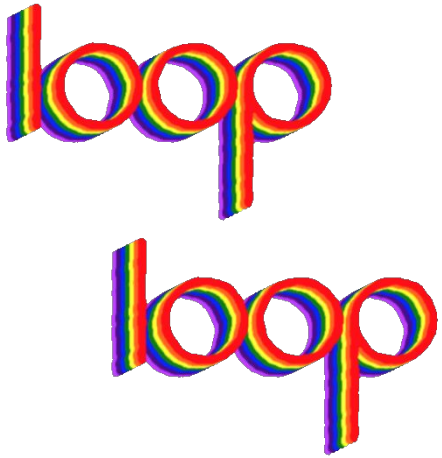
Previously, we had to iterate a single loop to solve a problem.

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

main(){

 for(int x = 0; x < 5; x = x + 1)
 {
    cout << "Welcome to UET!!" << endl;
 }
}
```

```
#include <iostream>
using namespace std;
main(){
    for (int num = 0; num <= 100; num = num + 1){
            if (num % 13 == 0)
            {
                cout << num << endl;
            }
    }
}
```

Multiple Loops

There can be some more complex problems that need a solution by iterating multiple loops.



Multiple Loops: Working Example

Suppose a teacher is teaching a course for a semester of 3 weeks. Within each week there are 7 days. For some absurd reason, the teacher needs to print all 7 days for each and every week of the semester.

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1 Day : 1
Week 1 Day : 2
Week 1 Day : 3
Week 1 Day : 4
Week 1 Day : 5
Week 1 Day : 6
Week 1 Day : 7

Week 2 Day : 1
Week 2 Day : 2
Week 2 Day : 3
Week 2 Day : 4
Week 2 Day : 5
Week 2 Day : 6
Week 2 Day : 7

Week 3 Day : 1
Week 3 Day : 2
Week 3 Day : 3
Week 3 Day : 4
Week 3 Day : 5
Week 3 Day : 6
Week 3 Day : 7
```

Working Example: Solution

For solving this problem, one needs to maintain the information of two things.

1. Number of Week
2. Number of Day

Solution: Step 1

Lets print only the Weeks now.

```
#include <iostream>
using namespace std;
main()
{
    for (int week = 1; week <= 3; week = week + 1)
    {
        cout << "Week " << week << endl;
    }
}
```

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1
Week 2
Week 3

C:\C++>
```

Solution: Step 2

Now we want to print all the days of Week 1 and then Week 2 and then Week 3.

```
#include <iostream>
using namespace std;
main()
{
    for (int week = 1; week <= 3; week = week + 1)
    {
        cout << "Week " << week << endl;
    }
}
```

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1
Week 2
Week 3

C:\C++>
```

Solution: Step 2

We definitely need a for loop for all the 7 days.

```
#include <iostream>
using namespace std;
main()
{
    for (int week = 1; week <= 3; week = week + 1)
    {
        cout << "Week " << week << endl;
    }
}
```

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1
Week 2
Week 3

C:\C++>
```

Solution: Step 2

We definitely need a for loop for all the 7 days. But where to write that for loop ???

```
#include <iostream>
using namespace std;
main()
{
    for (int week = 1; week <= 3; week = week + 1)
    {
        cout << "Week " << week << endl;
    }
}
```

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1
Week 2
Week 3

C:\C++>
```

Solution: Step 2

We would need to write the second for loop within the first for loop.

```
#include <iostream>
using namespace std;
main()
{
    for (int week = 1; week <= 3; week = week + 1)
    {
        cout << "Week " << week << endl;
    }
}
```

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1
Week 2
Week 3

C:\C++>
```

Solution: Step 2

We would need to write the second for loop within the first for loop.

```
#include <iostream>
using namespace std;
main(){
    for (int week = 1; week <= 3; week = week + 1)      // outer loop
    {
        for (int day = 1; day <= 7; day = day + 1)       // inner loop
        {
            cout << "Week " << week << " Day : " << day << endl;
        }
        cout << endl;
    }
}
```

Solution: Output

```
#include <iostream>
using namespace std;
main(){
    for (int week = 1; week <= 3; week = week + 1)   // outer loop
    {
        for (int day = 1; day <= 7; day = day + 1)   // inner loop
        {
            cout << "Week " << week << " Day : " << day << endl;
        }
        cout << endl;
    }
}
```

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1 Day : 1
Week 1 Day : 2
Week 1 Day : 3
Week 1 Day : 4
Week 1 Day : 5
Week 1 Day : 6
Week 1 Day : 7

Week 2 Day : 1
Week 2 Day : 2
Week 2 Day : 3
Week 2 Day : 4
Week 2 Day : 5
Week 2 Day : 6
Week 2 Day : 7

Week 3 Day : 1
Week 3 Day : 2
Week 3 Day : 3
Week 3 Day : 4
Week 3 Day : 5
Week 3 Day : 6
Week 3 Day : 7
```

Nested Loop

```cpp
#include <iostream>
using namespace std;
main(){
    for (int week = 1; week <= 3; week = week + 1)   // outer loop
    {
        for (int day = 1; day <= 7; day = day + 1)   // inner loop
        {
            cout << "Week " << week << " Day : " << day << endl;
        }
        cout << endl;
    }
}
```

Loop inside the loop is called Nested Loop.

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Week 1 Day : 1
Week 1 Day : 2
Week 1 Day : 3
Week 1 Day : 4
Week 1 Day : 5
Week 1 Day : 6
Week 1 Day : 7

Week 2 Day : 1
Week 2 Day : 2
Week 2 Day : 3
Week 2 Day : 4
Week 2 Day : 5
Week 2 Day : 6
Week 2 Day : 7

Week 3 Day : 1
Week 3 Day : 2
Week 3 Day : 3
Week 3 Day : 4
Week 3 Day : 5
Week 3 Day : 6
Week 3 Day : 7
```

Nested Loop

Nested loop means a loop inside another loop body. We can have any of the **counter** or **conditional loop** as **outer** or **inner loop**.

```
for ( ; ; )
{
      while ( )
      {
      }
}
```

Nested Loop

Nested loop means a loop inside another loop body. We can have any of the **counter** or **conditional loop** as **outer** or **inner loop**.

```
while ( )
{
      while ( )
      {
      }
}
```

Alter the normal flow of Loop

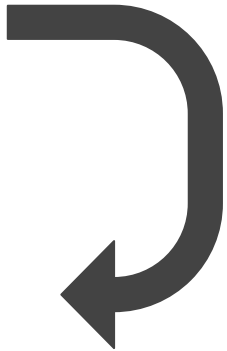
Sometimes, one needs to alter the normal flow of the execution of the loops. Maybe the requirement is to skip a single iteration of the loop or maybe skip the complete loop. For such conditions, we have two statements.

1. Break
2. Continue

Break Statement

Break statement is used to stop the execution of the loop.

```
for(initial statement; loop condition; update statement)
{
      if(breaking condition)
      {
            break;
      }
      //code
}

// code
```



Break Statement

```
#include <iostream>
using namespace std;
main(){
    for (int i = 0; i < 10; i = i + 1)
    {
        if (i == 4)
        {
            break;
        }
        cout << i << endl;
    }
}
```

```
C:\C++>c++ break.cpp -o break.exe

C:\C++>break.exe
0
1
2
3

C:\C++>
```

Continue Statement

Continue statement is used to skip a single iteration of the loop.

```
for(initial statement; loop condition; update statement)
{
      if(condition)
      {
            continue;
      }
      //code
}

// code
```

Continue Statement

```
#include <iostream>
using namespace std;
main(){
    for (int i = 0; i < 10; i = i + 1)
    {
        if (i == 4)
        {
            continue;
        }
        cout << i << endl;
    }
}
```

```
C:\C++>c++ continue.cpp -o continue.exe

C:\C++>continue.exe
0
1
2
3
5
6
7
8
9

C:\C++>
```

Learning Objective

In this lecture, we learnt how to write a C++ Program that repeats a complex set of instructions using nested loops and alter the normal flow of execution of loops.

Conclusion

- Just like nested if statements we can have nested loops.
- If a loop exists inside the body of another loop, it's called a nested loop.
- We can have any of the counter or conditional loop as outer or inner loop.
- For a single iteration of the outer loop, complete iterations of the inner loop are executed.

Conclusion

- **Break** and **Continue** statements are used to alter the normal flow of the execution of the loops.
- **Break statement** is used to stop the execution of the loop.
- **Continue statement** is used to skip a single iteration of the loop.

Self Assessment

1. What is the output of the following nested for loop?

```
for (int i = 1; i <= 10; i = i + 1)
{
    for (int j = 1; j <= i; j = j + 1)
    {
    cout << i;
    }
    cout << endl;
}
```

Self Assessment

3. What is the expected output?

```
int i = 0;
while (i < 10)
{
     cout << i << endl;
     i = i  + 1;
     if (i == 4)
     {
     break;
     }
}
```

Self Assessment

4. What is the expected output?

```
int n = 1;
while(n < 10)
{
    if (n == 5)
    {
        n = n + 1;
        continue;
    }
    cout << "n = " << n << endl;
    n = n + 1;
}
```

Self Assessment

2. Write a C++ program separately that prints the following patterns separately one below the other. Use nested for loops to generate the patterns.

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

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

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

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