C++

Prepared by: Mohamed Ayman

facebook.com/sw.eng.MohamedAyman

sw.eng.MohamedAyman@gmail.com

wuzzuf.net/me/engMohamedAyman

codeforces.com/profile/Mohamed_Ayman

Loops

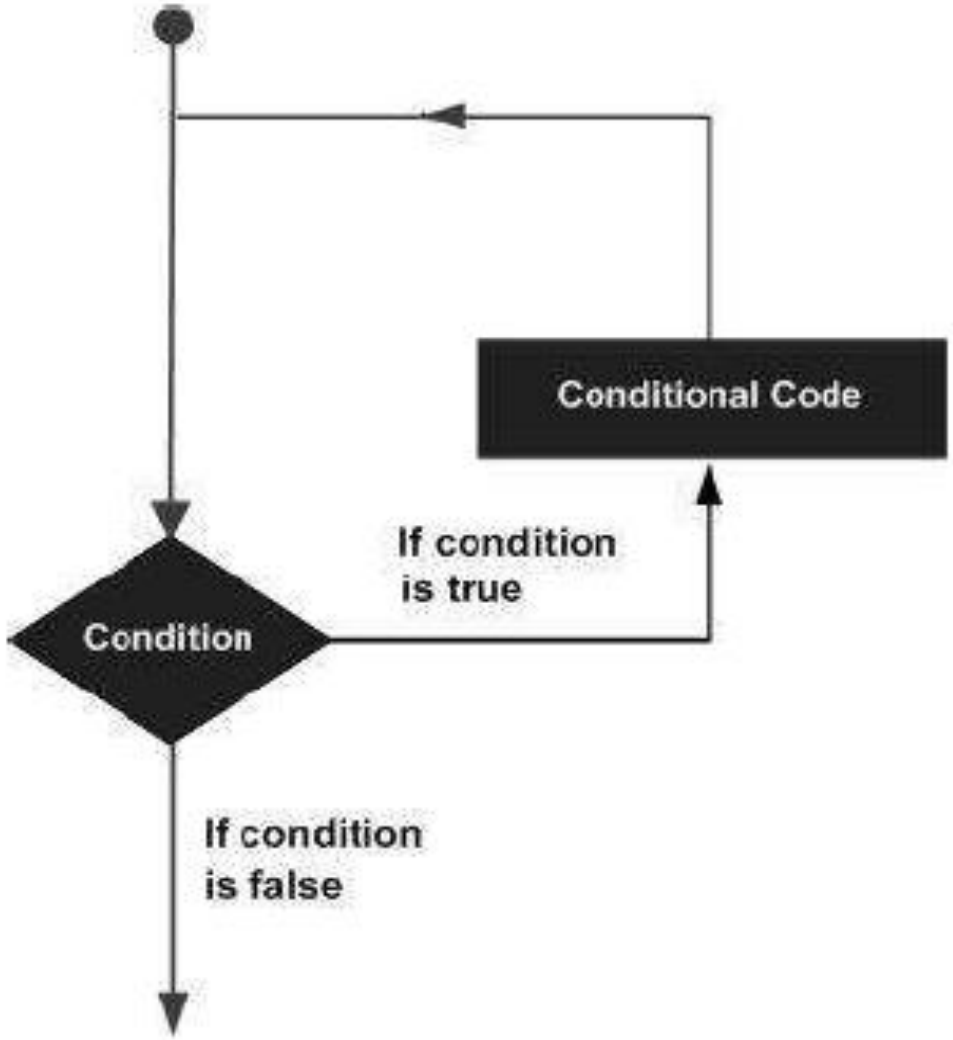
Outline

1) Loop Definition
2) Types of Loops
3) while Loop Statements
4) do while Loop Statements
5) for Loop Statements
6) Nested loops
7) Loop Control Statements

Loop Definition

- In general, statement are executed sequentially, The first statement in a function is executed first, followed by the second, and so on.
- There may be a situation when you need to execute a block of code several number of times.
- Programming language provide various control structure that allow more complicated execution paths.
- A loop statement allows us to execute a statement of group of statement multiple times.

Loop Definition



Types of Loops

- C++ programming language provides the following types of loops to handle looping requirement.

Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop	Like a 'while' statement, except that it tests the condition at the end of the loop body.
nested loops	You can use one or more loop inside any another 'while', 'for' or 'do..while' loop.

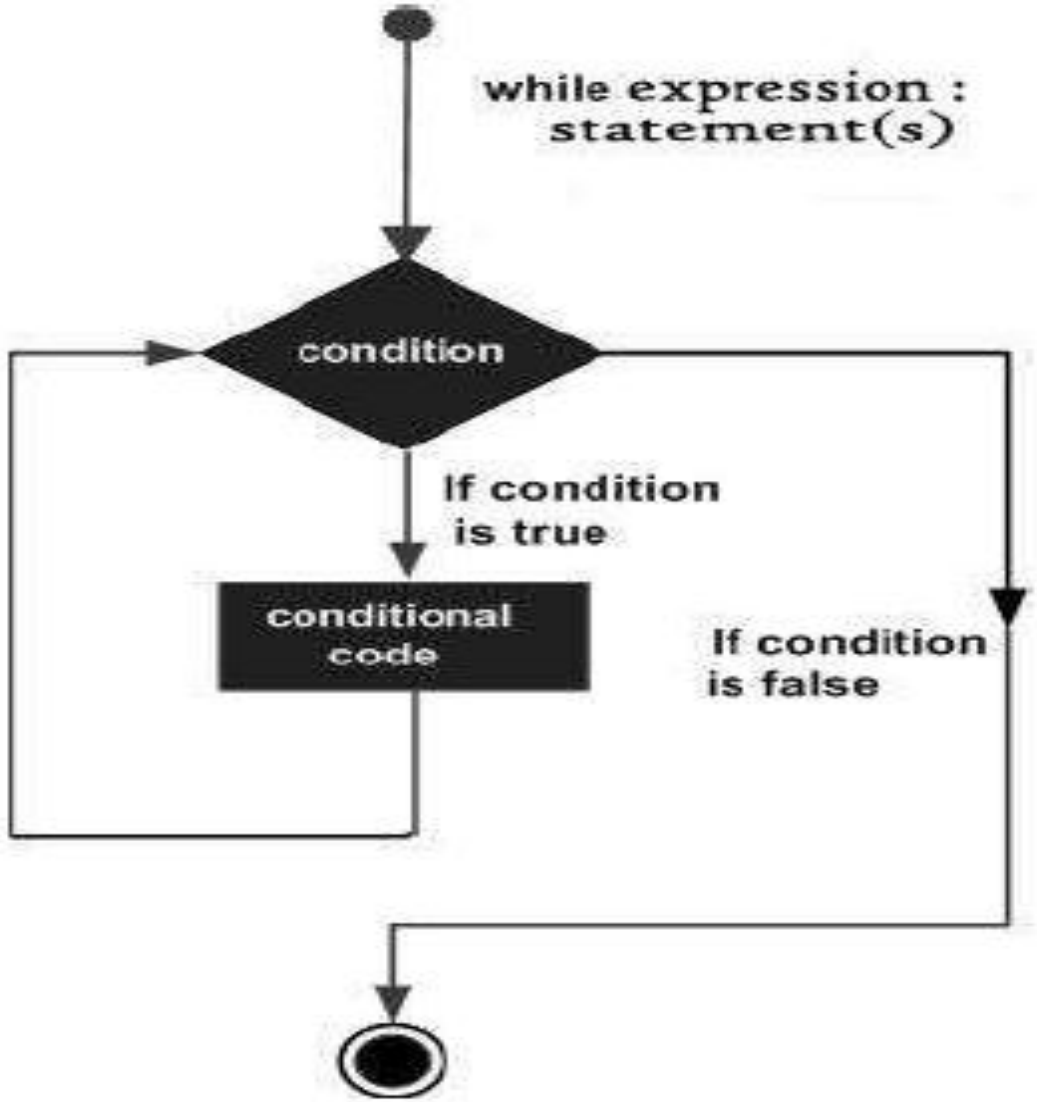
while Loop Statements

- A while loop statement in C++ programming language repeatedly executes a target statement as long as given condition is true.

```
while(condition)
{
    statement(s);
}
```

- Expression can be TRUE by any non-zero value. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

while Loop Statements



while Loop Statements Example

- Source code: https://repl.it/repls/TransparentLightblueGentoopenguin

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int i = 0;
7       while ( i < 5 )
8       {
9           cout << i << '\n';
10          i++;
11      }
12      cout << "finish\n";
13  }
```

```
0
1
2
3
4
finish
```

Practice

- Take as an input n and print 2^0, 2^1, 2^2 and so on ... 2^n using while loop such that n >= 0 and integer

- Test Cases:

```
 5
1 2 4 8 16 32
```

```
 10
1 2 4 8 16 32 64 128 256 512 1024
```

```
 3
1 2 4 8
```

```
 1
1 2
```

```
 12
1 2 4 8 16 32 64 128 256 512 1024 2048 4096
```

Practice Solution

- Source code: https://repl.it/repls/AcrobaticSerpentineFattaileddunnart

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

int main()
{
    int n;
    cin >> n;

    int i = 0, res = 1;
    while ( i <= n )
    {
        cout << res << ' ';
        res *= 2;
        i++;
    }
}
```

Practice

- Take as an input n and print it's factors using while loop such that n >= 1 and integer
- Test Cases:

```
 10
1 2 5 10
```

```
 12
1 2 3 4 6 12
```

```
 20
1 2 4 5 10 20
```

```
 25
1 5 25
```

```
 120
1 2 3 4 5 6 8 10 12 15 20 24 30 40 60 120
```

```
 13
1 13
```

Practice Solution

- Source code: https://repl.it/repls/AlienatedExtrasmallPheasant

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int n;
7       cin >> n;
8
9       int i = 1;
10      while ( i <= n )
11      {
12          if ( n % i == 0 )
13          {
14              cout << i << ' ';
15          }
16          i++;
17      }
18  }
```

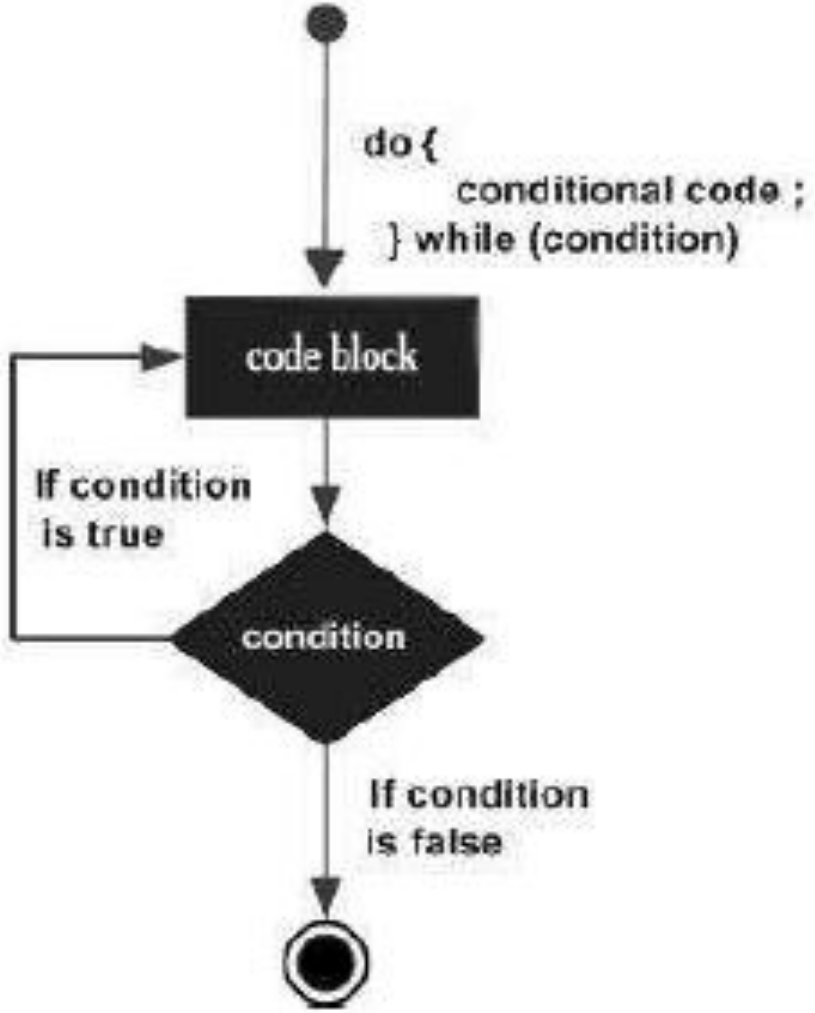
do while Loop Statements

- A do while loop is similar to a while loop, except that a do while loop is guaranteed to execute at least one time.

```
do
{
    statement(s);
}
while( condition );
```

- Expression can be TRUE by any non-zero value. The loop iterates do while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

do while Loop Statements



do while Loop Statements Example

- Source code: https://repl.it/repls/SlimWordyEagle

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  int main()
 5  {
 6      int i = 0;
 7      do
 8      {
 9          cout << i << '\n';
10          i++;
11      }
12      while( i < 5);
13
14      cout << "finish\n";
15  }
```

```
0
1
2
3
4
finish
```

Practice

- Take as an input n and print 2^0, 2^1, 2^2 and so on ... 2^n using do while loop such that n >= 0 and integer
- Test Cases:

```
 5
1 2 4 8 16 32
```

```
 10
1 2 4 8 16 32 64 128 256 512 1024
```

```
 3
1 2 4 8
```

```
 1
1 2
```

```
 12
1 2 4 8 16 32 64 128 256 512 1024 2048 4096
```

Practice Solution

- Source code: https://repl.it/repls/ScaryCrushingMammoth

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

int main()
{
    int n;
    cin >> n;

    int i = 0, res = 1;
    do
    {
        cout << res << ' ';
        res *= 2;
        i++;
    }
    while ( i <= n );
}
```

Practice

- Take as an input n and print it's factors using do while loop such that n >= 1 and integer
- Test Cases:

```
 10
1 2 5 10
```

```
 12
1 2 3 4 6 12
```

```
 20
1 2 4 5 10 20
```

```
 25
1 5 25
```

```
 120
1 2 3 4 5 6 8 10 12 15 20 24 30 40 60 120
```

```
 13
1 13
```

Practice Solution

- Source code: https://repl.it/repls/TimelyFavoriteBlackmamba

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

int main()
{
    int n;
    cin >> n;

    int i = 1;
    do
    {
        if ( n % i == 0 )
        {
            cout << i << ' ';
        }
        i++;
    }
    while ( i <= n );
}
```

for Loop Statements

- A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

```
for ( init; condition; increment )
{
   statement(s);
}
```

for Loop Statements

Here is the flow of control in a for loop:

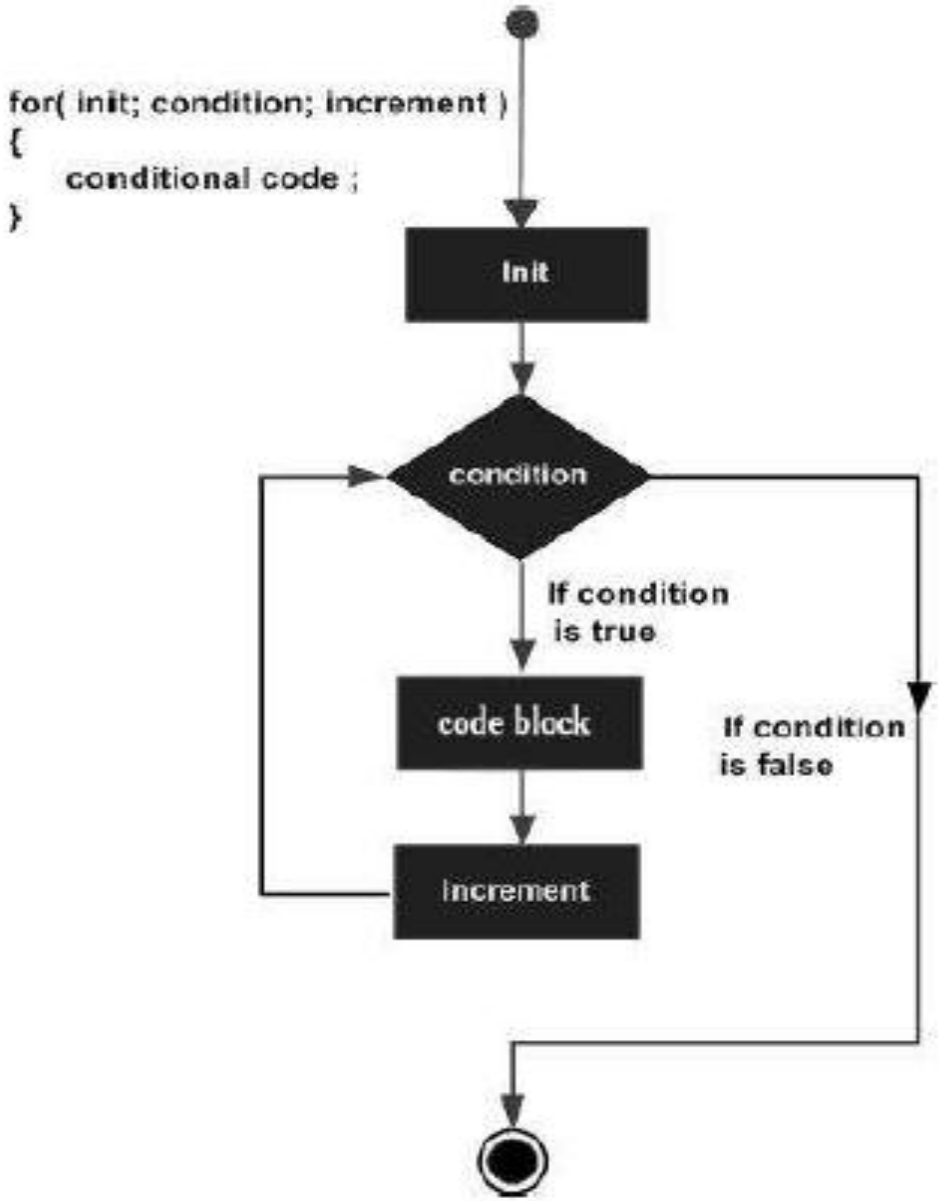
1. The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.

2. Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the for loop.

3. After the body of the for loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any C++ loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.

4. The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the for loop terminates.

for Loop Statements



for Loop Statements Example

- Source code: https://repl.it/repls/FlamboyantGlumBull

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

int main()
{
    for (int i = 0; i < 5; i++)
    {
        cout << i << '\n';
    }

    cout << "finish\n";
}
```

```
0
1
2
3
4
finish
```

Practice

- Take as an input n and print 2^0, 2^1, 2^2 and so on ... 2^n using for loop such that n >= 0 and integer
- Test Cases:

```
 5
1 2 4 8 16 32
```

```
 10
1 2 4 8 16 32 64 128 256 512 1024
```

```
 3
1 2 4 8
```

```
 1
1 2
```

```
 12
1 2 4 8 16 32 64 128 256 512 1024 2048 4096
```

Practice Solution

- Source code: https://repl.it/repls/HollowArtisticPig

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

int main()
{
    int n;
    cin >> n;

    int res = 1;
    for (int i = 0; i <= n; i++)
    {
        cout << res << ' ';
        res *= 2;
    }
}
```

Practice Solution

- Source code: https://repl.it/repls/LeadingTightAnnelida

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

int main()
{
    int n;
    cin >> n;

    for (int i = 0, res = 1; i <= n; i++, res*=2)
    {
        cout << res << ' ';
    }
}
```

Practice

- Take as an input n and print it's factors using for loop such that n >= 1 and integer
- Test Cases:

```
 10
1 2 5 10
```

```
 12
1 2 3 4 6 12
```

```
 20
1 2 4 5 10 20
```

```
 25
1 5 25
```

```
 120
1 2 3 4 5 6 8 10 12 15 20 24 30 40 60 120
```

```
 13
1 13
```

Practice Solution

- Source code: https://repl.it/repls/CriticalBeigeStag

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

int main()
{
    int n;
    cin >> n;

    for (int i = 1; i <= n; i++)
    {
        if ( n % i == 0 )
        {
            cout << i << ' ';
        }
    }
}
```

Nested loops

- C++ programming language allows the use of one loop inside another loop.
- The syntax for a nested while loop statement in C++ is as follows:

```
while(condition)
{
   while(condition)
   {
      statement(s); // you can put more statements.
   }
   statement(s); // you can put more statements.
}
```

Nested loops

- C++ programming language allows the use of one loop inside another loop.
- The syntax for a nested do while loop statement in C++ is as follows:

```
do
{
   statement(s); // you can put more statements.
   do
   {
      statement(s); // you can put more statements.
   }
   while( condition );
}
while( condition );
```

Nested loops

- C++ programming language allows the use of one loop inside another loop.
- The syntax for a nested for loop statement in C++ is as follows:

```
for ( init; condition; increment )
{
   for ( init; condition; increment )
   {
      statement(s); // you can put more statements.
   }
   statement(s); // you can put more statements.
}
```

Nested loops Example

- Source code: https://repl.it/repls/SwiftAshamedAcornwoodpecker

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int i = 1;
7       while( i <= 3 )
8       {
9           int j = 1;
10          while( j <= 3 )
11          {
12              cout << i*j << ' ';
13              j++;
14          }
15          cout << '\n';
16          i++;
17      }
18  }
```

```
1 2 3
2 4 6
3 6 9
```

Nested loops Example

- Source code: https://repl.it/repls/FuzzyWhichLamb

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int i = 1;
7       do
8       {
9           int j = 1;
10          do
11          {
12              cout << i*j << ' ';
13              j++;
14          }
15          while( j <= 3 );
16          cout << '\n';
17          i++;
18      }
19      while( i <= 3 );
20  }
```

```
1 2 3
2 4 6
3 6 9
```

Nested loops Example

- Source code: https://repl.it/repls/PerfectRigidLorikeet

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       for (int i = 1; i <= 3; i++)
7       {
8           for (int j = 1; j <= 3; j++)
9           {
10              cout << i*j << ' ';
11          }
12          cout << '\n';
13      }
14  }
```

```
1 2 3
2 4 6
3 6 9
```

Practice

- Take as an input n and print identity matrix using while loop, such that n >= 1 and integer
- Test Cases:

```
5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

```
4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

```
3
1 0 0
0 1 0
0 0 1
```

```
2
1 0
0 1
```

```
1
1
```

Practice Solution

- Source code: https://repl.it/repls/JealousBustlingSeaslug

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

int main()
{
    int n;
    cin >> n;

    int i = 0;
    while ( i < n )
    {
        int j = 0;
        while ( j < n )
        {
            if( i == j )
                cout << 1 << ' ';
            else
                cout << 0 << ' ';
            j++;
        }
        cout << '\n';
        i++;
    }
}
```

Practice

- Take as an input n and print identity matrix using do while loop, such that n >= 1 and integer
- Test Cases:

```
 5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

```
 4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

```
 3
1 0 0
0 1 0
0 0 1
```

```
 2
1 0
0 1
```

```
 1
1
```

Practice Solution

- Source code: https://repl.it/repls/VigilantSinfulPoodle

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

int main()
{
    int n;
    cin >> n;

    int i = 0;
    do
    {
        int j = 0;
        do
        {
            if( i == j )
                cout << 1 << ' ';
            else
                cout << 0 << ' ';
            j++;
        }
        while ( j < n );

        cout << '\n';
        i++;
    }
    while ( i < n );
}
```

Practice

- Take as an input n and print identity matrix using for loop, such that n >= 1 and integer
- Test Cases:

```
 5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

```
 4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

```
 3
1 0 0
0 1 0
0 0 1
```

```
 2
1 0
0 1
```

```
 1
1
```

Practice Solution

- Source code: https://repl.it/repls/StarryDimgraySunbittern

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

int main()
{
    int n;
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if( i == j )
                cout << 1 << ' ';
            else
                cout << 0 << ' ';
        }
        cout << '\n';
    }
}
```

Loop Control Statements

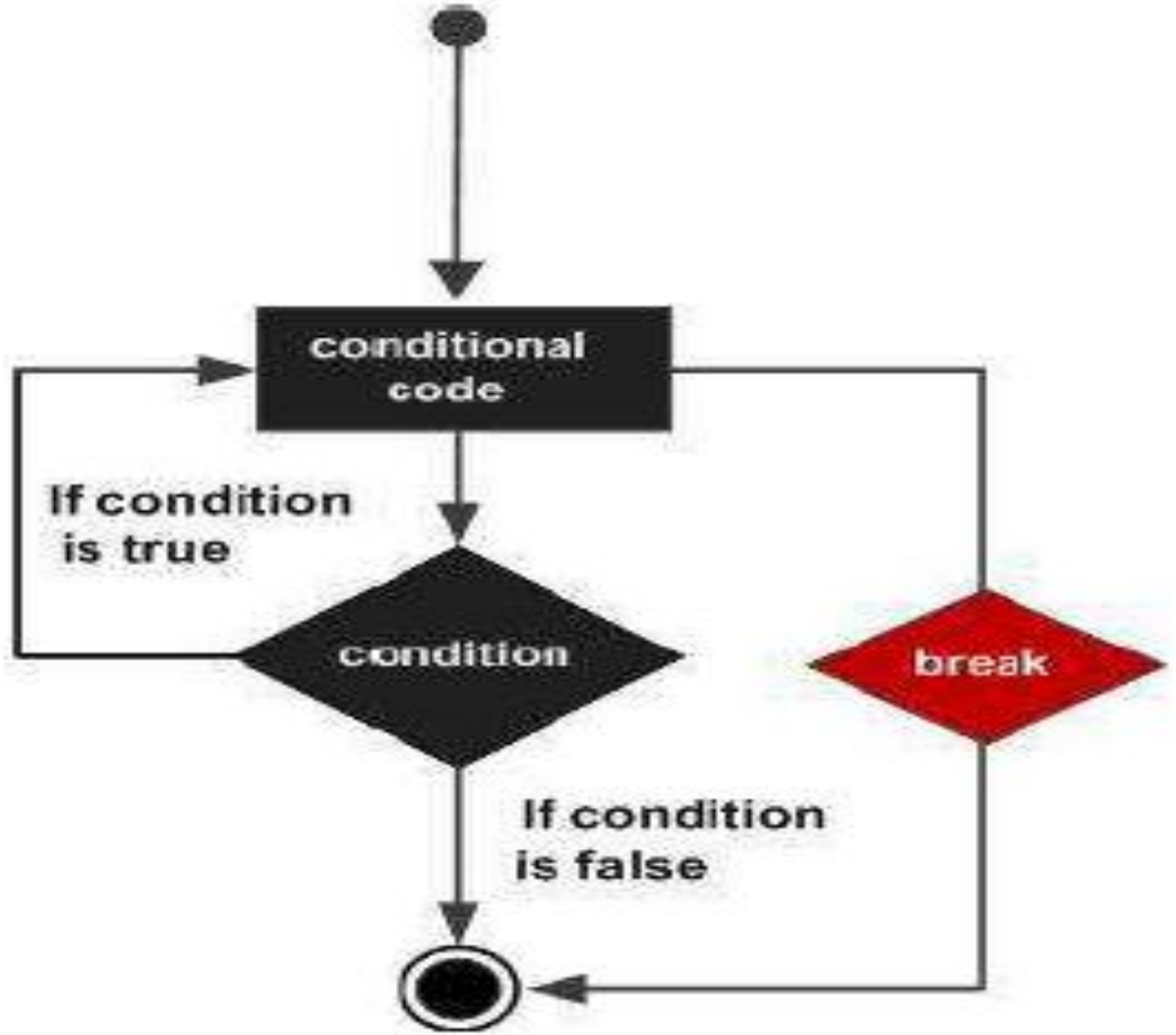
- The Loop control statement change the execution from its normal sequence. When the execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Control Statement	Description
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

break statement

- The break statement is used for premature termination of the current loop, After abandoning the loop, execution at the next statement is resumed.
- The most common use of break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.
- If you are using nested loops, the break statement stops execution of the innermost loop and starts executing the next line of the code after the block.

break statement



break statement Example

- Source code: https://repl.it/repls/ToughQuarterlyFlies

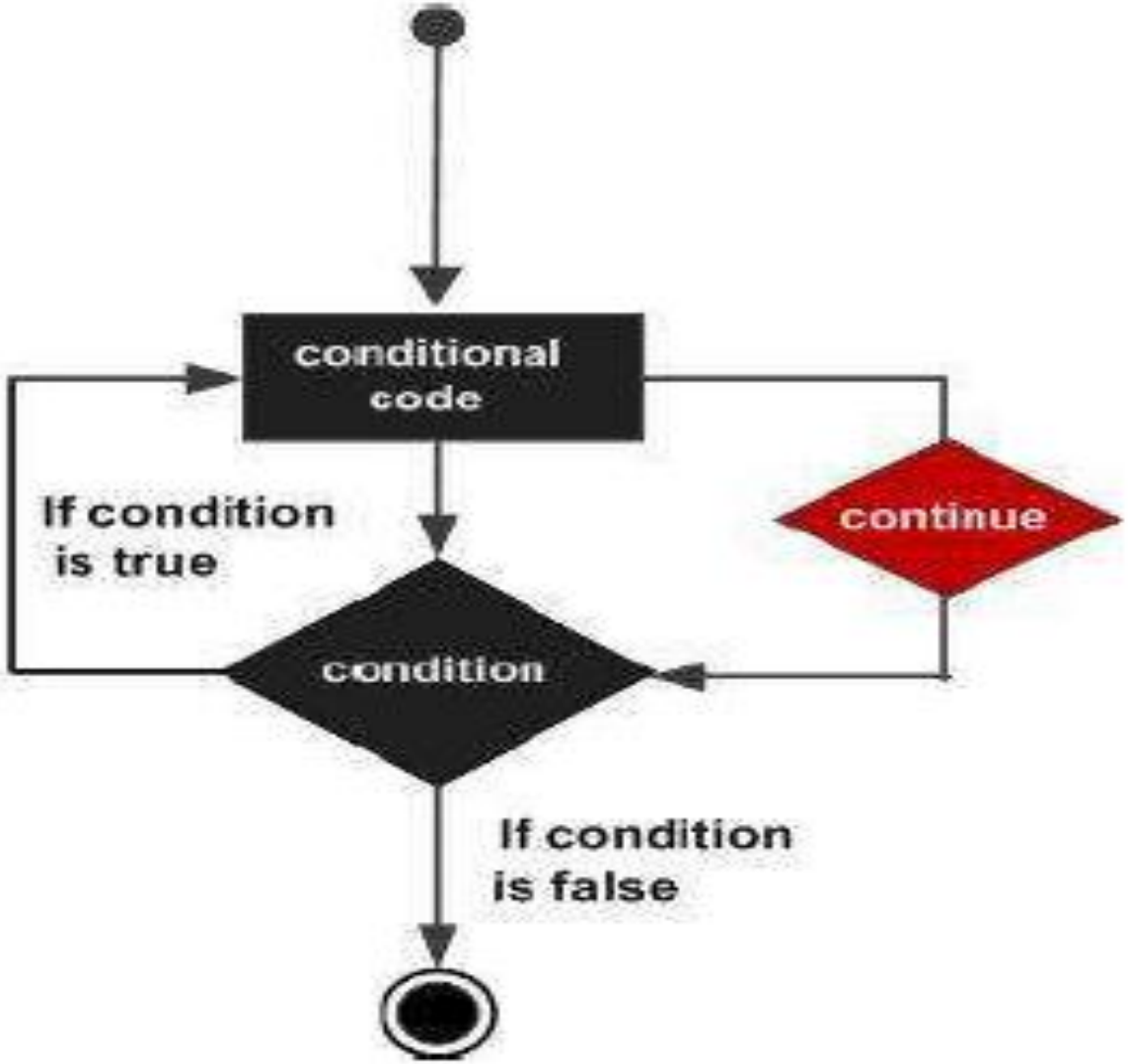
```cpp
#include <iostream>
using namespace std;

int main()
{
    for (int i = 0; i < 7; i++)
    {
        if ( i == 4)
        {
            break;
        }
        cout << i << '\n';
    }
}
```

```
0
1
2
3
```

continue Statement

- The continue statement in C++ returns the control to the beginning of the current loop. When encountered, the loop starts next iteration without executing the remaining statements in the current iteration.

continue Statement



continue Statement Example

- Source code: https://repl.it/repls/ChillyRoyalLizard

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

int main()
{
    for (int i = 0; i < 7; i++)
    {
        if ( i == 4)
        {
            continue;
        }
        cout << i << '\n';
    }
}
```

```
0
1
2
3
5
6
```

Practice

- Take as an input n and print summation of its digits using while loop such that n >= 0 and integer
- Test Cases:

```
954
18
```

```
684
18
```

```
1286
17
```

```
57968
35
```

```
3492
18
```

```
607935
30
```

```
12493
19
```

```
2745650
29
```

Practice Solution

- Source code: https://repl.it/repls/DamagedMammothShearwater

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

int main()
{
    int n;
    cin >> n;
    int sum = 0;

    while(n > 0)
    {
        sum += n % 10;
        n /= 10;
    }

    cout << sum;
}
```

Practice

- Take as an input n and print summation of its digits using for loop such that n >= 0 and integer
- Test Cases:

```
954
18
```

```
684
18
```

```
1286
17
```

```
57968
35
```

```
3492
18
```

```
607935
30
```

```
12493
19
```

```
2745650
29
```

Practice Solution

- Source code: https://repl.it/repls/MustyFrequentChinchilla

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

int main()
{
    int n;
    cin >> n;
    int sum = 0;

    for ( ; n > 0; n /= 10)
    {
        sum += n % 10;
    }

    cout << sum;
}
```

Practice

- Take as an input n and check if n is prime or not prime using for loop such that n >= 2 and integer
- Test Cases:

```
 2
2 is prime
```

```
 3
3 is prime
```

```
 4
4 is not prime
```

```
 5
5 is prime
```

```
 6
6 is not prime
```

```
 7
7 is prime
```

```
 8
8 is not prime
```

```
 9
9 is not prime
```

```
 10
10 is not prime
```

```
 11
11 is prime
```

Practice Solution

- Source code: https://repl.it/repls/DarkkhakiDownrightArthropods

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

int main()
{
    int n;
    cin >> n;
    bool isPrime = true;

    for (int i = 2; i < n; i++)
    {
        if(n % i == 0)
        {
            isPrime = false;
            break;
        }
    }

    cout << ((isPrime)? "Prime":"Not Prime");
}
```

Practice

- Take as an input n and check if n is prime or not prime using while loop such that n >= 2 and integer
- Test Cases:

```
 2
2 is prime
```

```
 3
3 is prime
```

```
 4
4 is not prime
```

```
 5
5 is prime
```

```
 6
6 is not prime
```

```
 7
7 is prime
```

```
 8
8 is not prime
```

```
 9
9 is not prime
```

```
 10
10 is not prime
```

```
 11
11 is prime
```

Practice Solution

- Source code: https://repl.it/repls/EarnestNarrowTaruca

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

int main()
{
    int n;
    cin >> n;
    bool isPrime = true;

    int i = 2;
    while ( i < n )
    {
        if(n % i == 0)
        {
            isPrime = false;
            break;
        }
        i++;
    }

    cout << ((isPrime)? "Prime":"Not Prime");
}
```

Questions ?

References

Online Courses YouTube playlists:	http://bit.ly/2kAPL5K
C++ Documentation	http://bit.ly/1flmcHO
CPP For School	http://bit.ly/2kifMdj
C++ Language Tutorial	http://bit.ly/1kyBMdz
C++ Language Tutorial	http://bit.ly/2rzE4hQ
C++ Tutorial Point	http://bit.ly/2BGFeO0
Fundamentals of C++ Programming	http://bit.ly/2rJHhyI
Teach Yourself C++ in 21 Days	http://bit.ly/1JXhDtL
A Complete Guide to Programming in C++	http://bit.ly/2dVkGY9