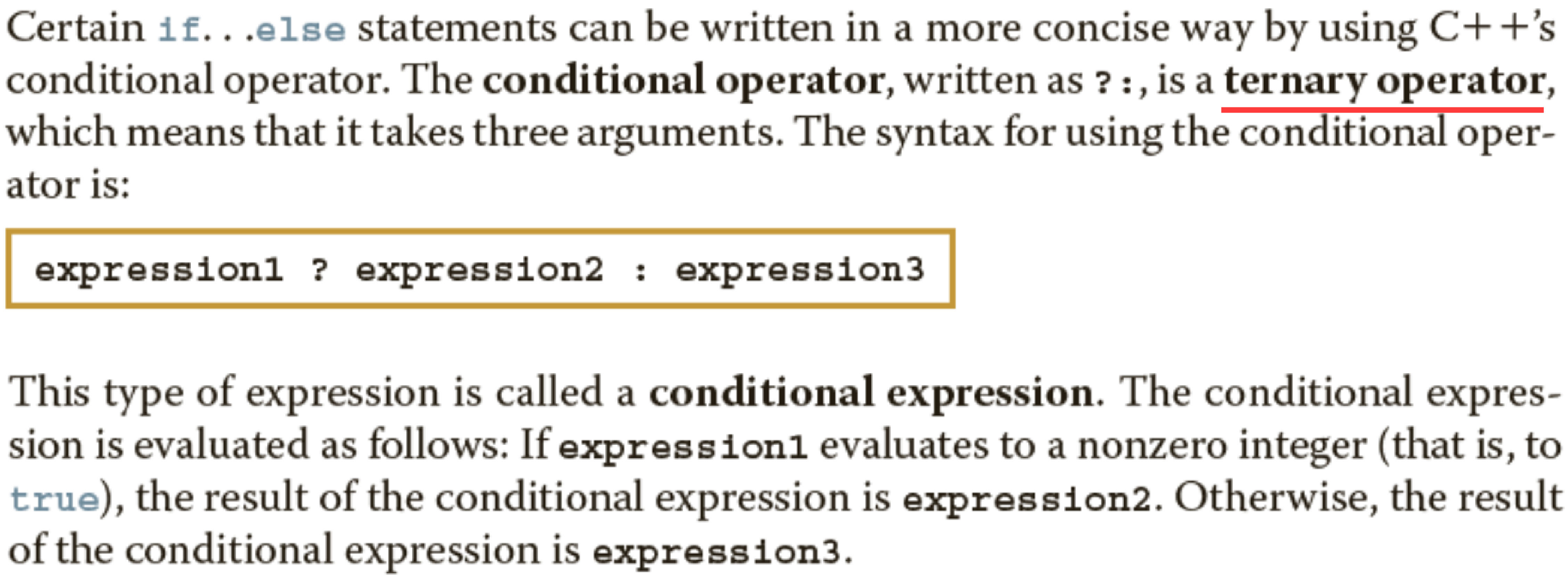
Programming Principles (MT162)

Lecture 7

Dr. Ahmed Fathalla

Conditional operator (?:)



Certain `if...else` statements can be written in a more concise way by using C++'s conditional operator. The **conditional operator**, written as `?:`, is a **ternary operator**, which means that it takes three arguments. The syntax for using the conditional operator is:

```
expression1 ? expression2 : expression3
```

This type of expression is called a **conditional expression**. The conditional expression is evaluated as follows: If **`expression1`** evaluates to a nonzero integer (that is, to `true`), the result of the conditional expression is **`expression2`**. Otherwise, the result of the conditional expression is **`expression3`**.

Conditional operator (?:)

Consider the following statements:

```
if (a >= b)
    max = a;
else
    max = b;
```

You can use the conditional operator to simplify the writing of this `if`. . .`else` statement as follows:

```
max = (a >= b) ? a : b;
```

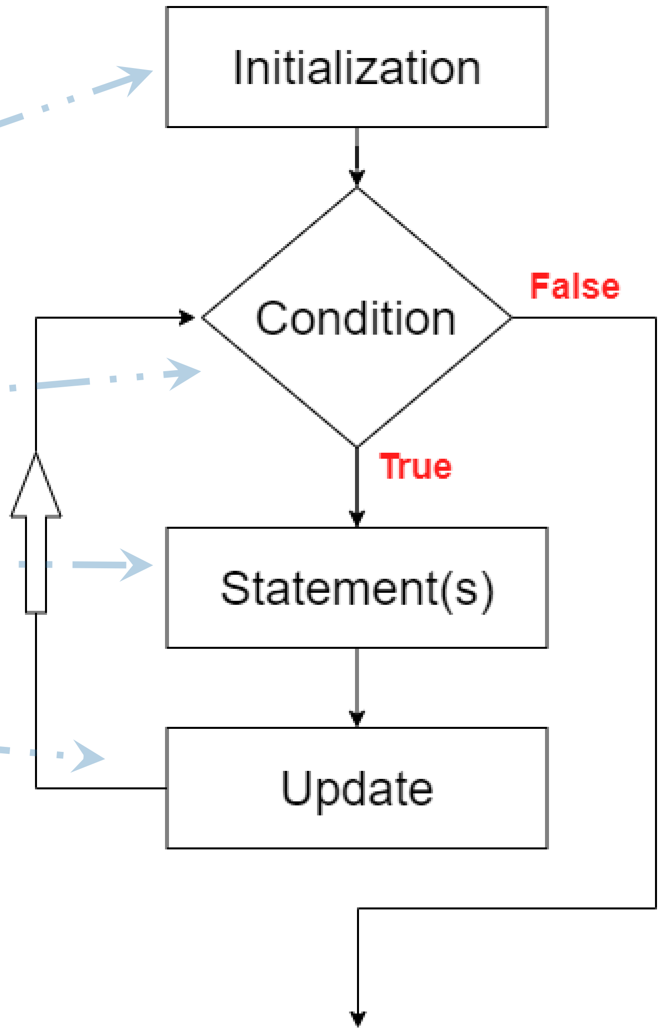
Answer of the bounce exercise

Overflow https://www.cplusplus.com/articles/DE18T05o/

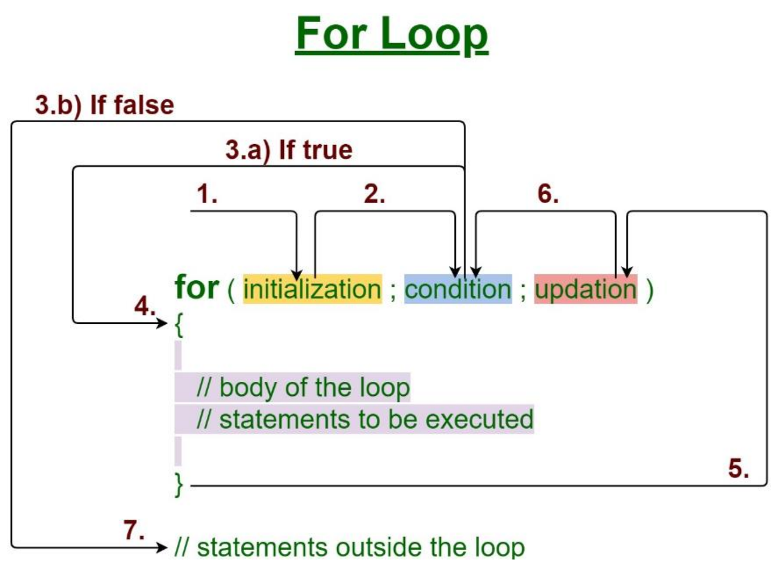
Control Structures II (Repetition)

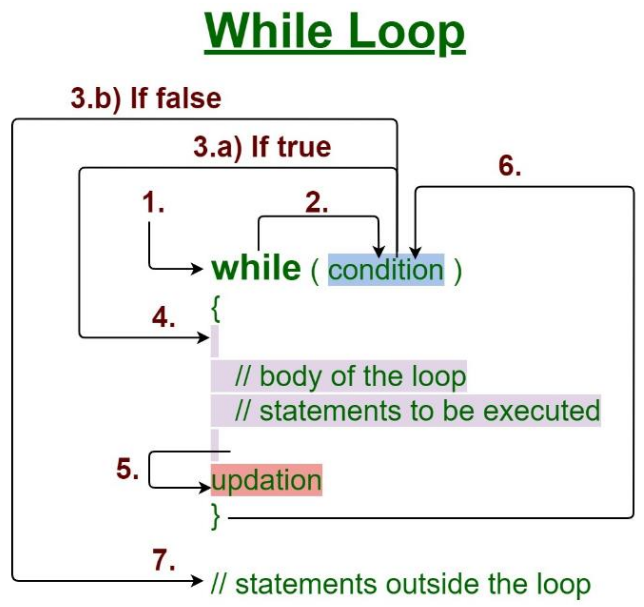
Loops

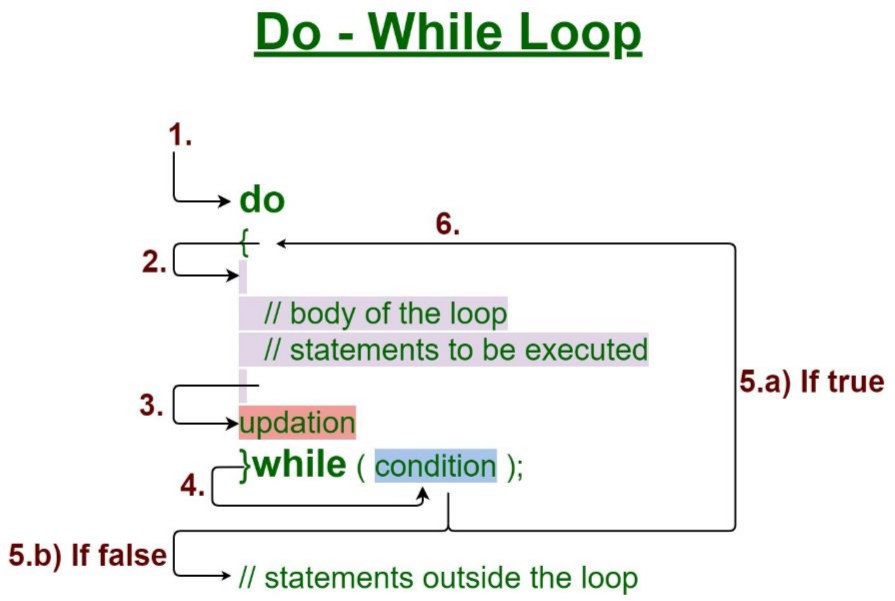
- Main components
  - Initialization.
  - Condition.
  - Statement(s) (What to do)
  - Update.



Loop types


For Loop
3.b) If false
3.a) If true
1.
2.
6.
for (initialization ; condition ; updation)
4.
{
// body of the loop
// statements to be executed
}
5.
7.
// statements outside the loop

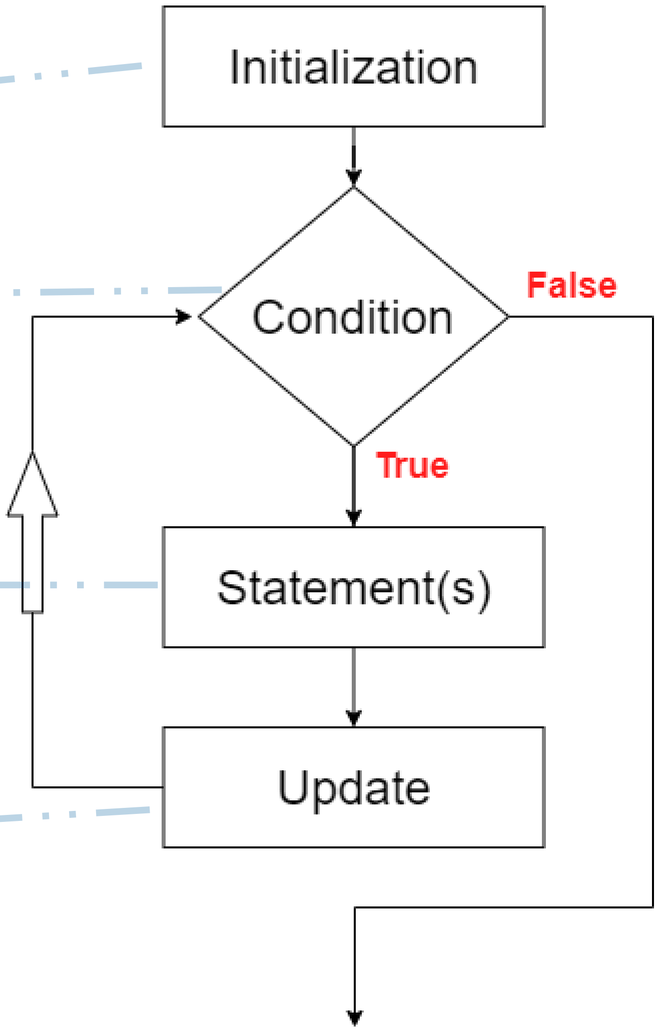

While Loop
3.b) If false
3.a) If true
6.
1.
2.
while (condition)
4.
{
// body of the loop
// statements to be executed
5.
updation
}
7.
// statements outside the loop


Do - While Loop
1.
do
6.
2.
{
// body of the loop
// statements to be executed
5.a) If true
3.
updation
4.
}while (condition);
5.b) If false
// statements outside the loop

While loop

```
Initialization;
while (condition)
{
        statement_1;
        statement_2;
        statement_n;

        update;
}
```

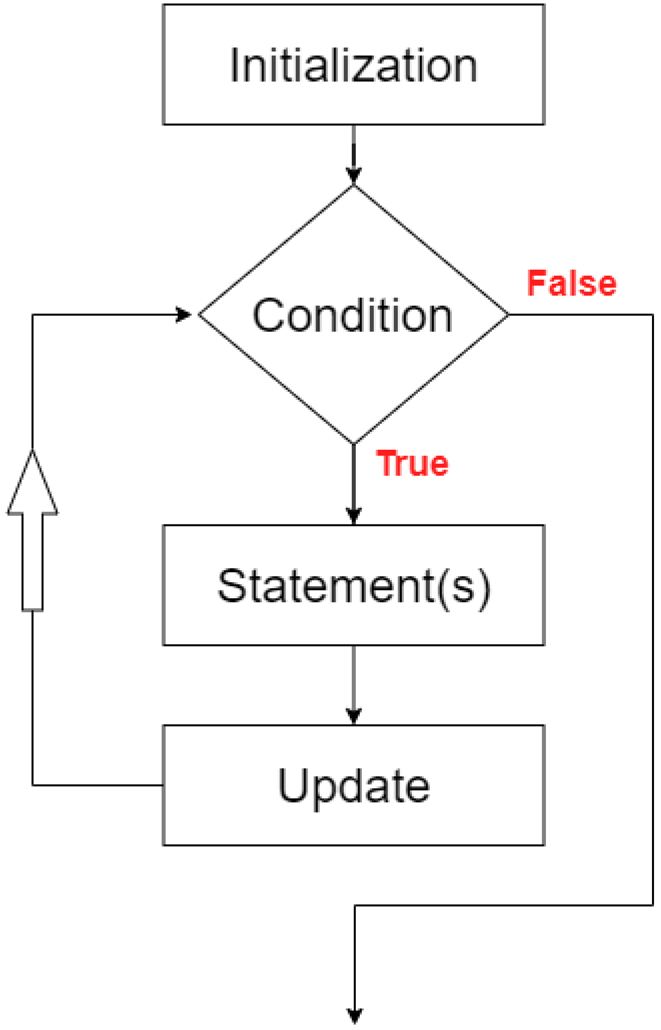
body of the loop



for loop

```
for ( Initialization; condition; update )
{
        statement_1;
        statement_2;

        statement_n;
}
```

body of the loop



Exercise: Print “Hello World” 100 times.

```
int main()
{
  int i = 1;
  while (i<=100)
  {
     cout<<"Hello world"<<endl;
     i += 1;
  }
}
```

```
int i = 0;
while( i++<100)
{
   cout<<i<<“\n";
}
```

```
int main()
{
   for (int i=1;i<=100;i++)
   {
      cout<<"Hello world"<<endl;
   }
}
```

Exercise: Print even and odd numbers between [1, 100].

```
int main()
{
    int i = 1;
    while (i<=100)
    {
        if (i%2==0)
            cout<<i++<<" even "<<endl;
        else
            cout<<i++<<" odd "<<endl;
    }
}
```

```
int main()
{
   for (int i=1;i<=100;i++)
    {
        if (i%2==0)
            cout<<i<<" even "<<endl;
        else
            cout<<i<<" odd "<<endl;
    }
}
```

Exercise: Print even and odd numbers between [1, 100]. *Using conditional operator*

```
int main()
{
    int i = 1;
    while (i<=100)
    {
        cout<<((i%2)?“odd":“even");
        i++;
     }
}
```

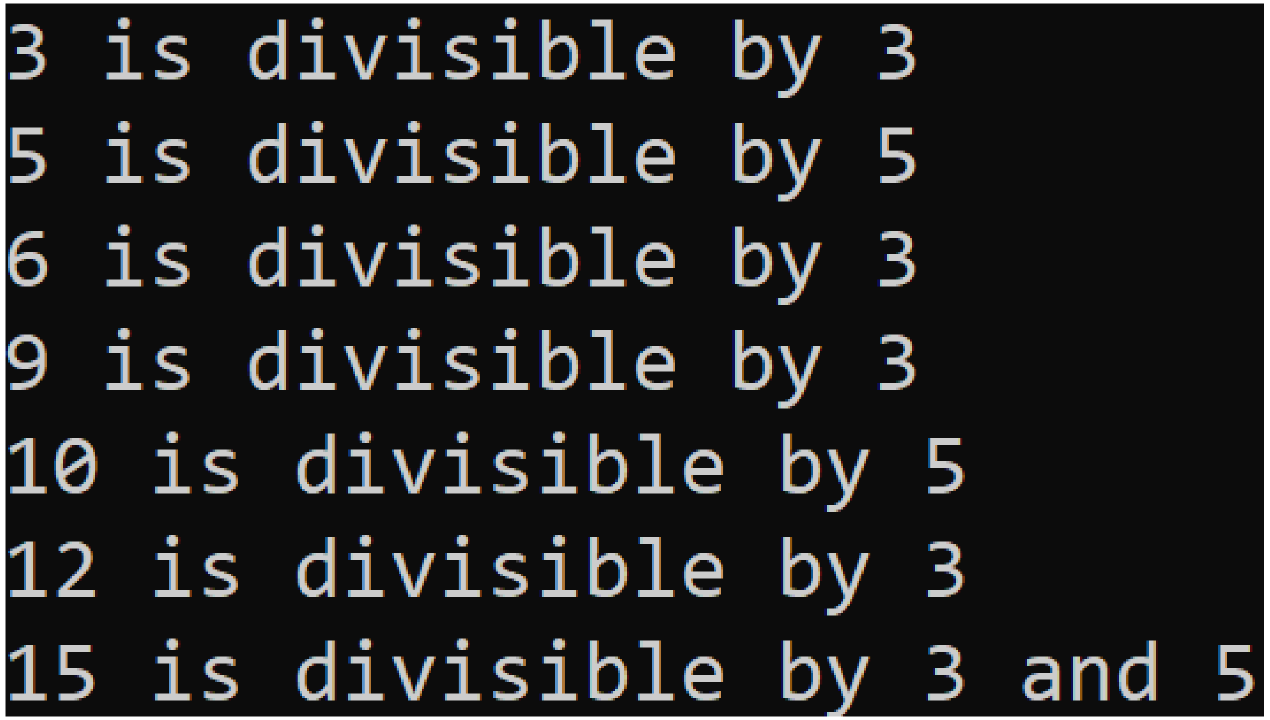
```
If (i%2) // means if (i%2!=0)
{
   do something;
}
```

```
int main()
{
   for (int i=1;i<=100;i++)
    {
          cout<<((i%2)?“odd":“even");
    }
}
```

Previous Quiz

- Print numbers which are divisible by 3, 5, or both in the interval between 1 and an input number.

The program output should be as follows



```
3 is divisible by 3
5 is divisible by 5
6 is divisible by 3
9 is divisible by 3
10 is divisible by 5
12 is divisible by 3
15 is divisible by 3 and 5
```

Answer

```
int main()
{
   int i=1;
   while (i<=100)
   {
      if (i%3 == 0 && i%5 == 0)
         cout<<i<<" is divisible by 3 and 5\n";
      else if (i%5==0)
         cout<<i<<" is divisible by 5\n";
      else if (i%3==0)
         cout<<i<<" is divisible by 3\n";
      i++;
   }
   return 0;
}
```

```
int main()
{
   for(int i =1;i<=100;i++)
   {
      if (i%3 == 0 && i%5 == 0)
         cout<<i<<" is divisible by 3 and 5\n";
      else if (i%5==0)
         cout<<i<<" is divisible by 5\n";
      else if (i%3==0)
         cout<<i<<" is divisible by 3\n";
   }
   return 0;
}
```

Exercise_1: Write a C++ program to calculate the sum of numbers from 1 to 100

```
int main()
{
  int i = 1, sum=0;
  while (i <= 100)
  {
    sum += i;
    i++;
  }
  cout << "\n The sum of numbers
    from 1 to 100 is: "<<sum << endl;
  return 0;
}
```

```
int main()
{
  int sum=0;
  for (int i = 1 ;i <= 100; i++;)
  {
    sum += i;
  }
  cout << "\n The sum of numbers
    from 1 to 100 is: "<<sum << endl;
  return 0;
}
```

Exercise_2: Write a C++ program to find Factorial of a given number

```
int main()
{
  int i = 1, factorial=1, n;
  cin>>n;
  while (i <= n)
  {
    factorial *=  i;
    i++;
  }
  cout << n <<“! = "<<factorial <<endl;
  return 0;
}
```

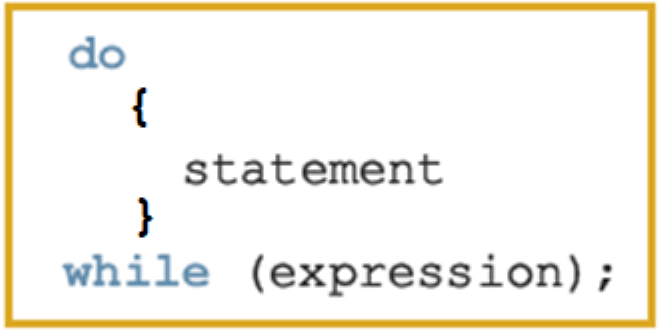
```
int main()
{
  int factorial = 1, n;
  cin>>n;
  for (int i = 1 ;i <= n; i++;)
  {
    factorial *=  i;
  }
   cout << n <<“! = "<<factorial<<endl;
  return 0;
}
```

Exercise_3: Write a program to calculate x^y where x and y are given numbers.

```
int main()
{
    int x, y, power=1,i;
    cout << " Input values of x and y: ";
    cin >> x >> y;
    for (int i = 1; i <=y; i++)
    {
        power = power *x;
    }
    cout <<x<<" ^ "<<y<<" = "<<power<<endl ;
}
```

do…while Looping (Repetition) Structure

- General form of a do…while:



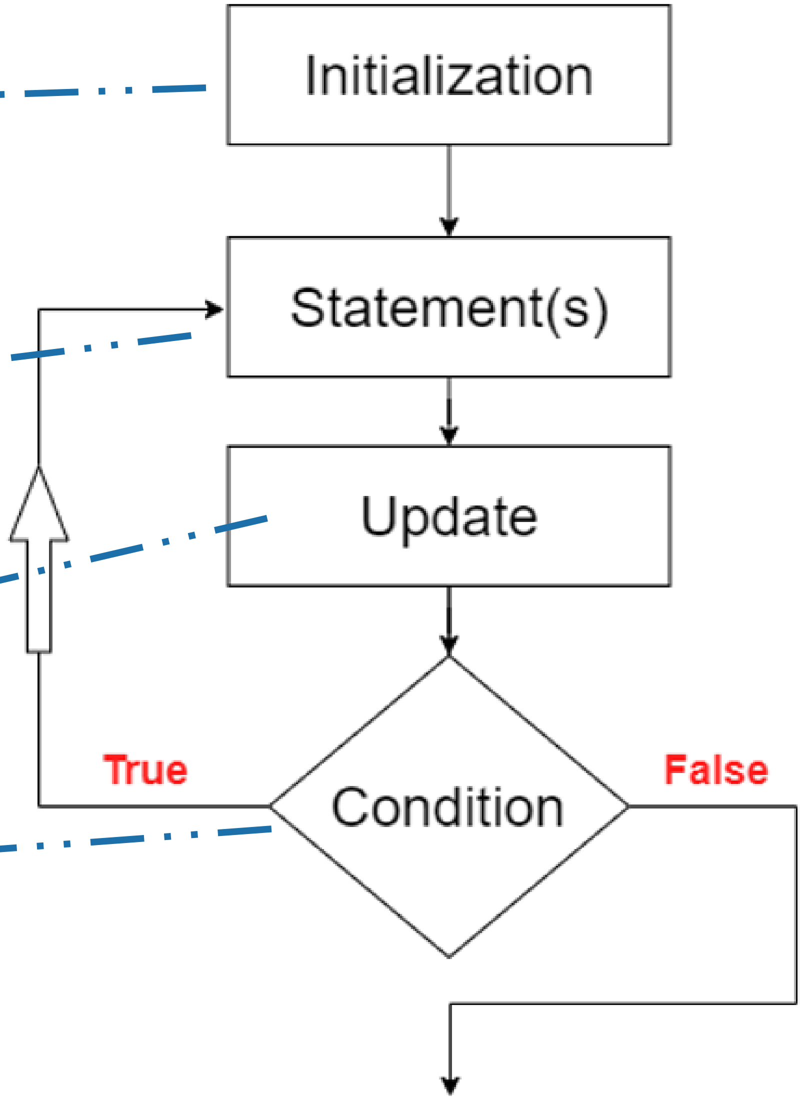
```
do
   {
      statement
   }
while (expression);
```

- The statement executes first, and then the expression is evaluated
- To avoid an infinite loop, body must contain a statement that makes the expression false
- The statement can be simple or compound
- Loop always iterates at least once

do-while loop

```
Initialization;
do
{
        statement_1;
        statement_2;
        statement_n;

        update;
} while (condition);
```

body of the loop



Exercise_4: write a program to sum input numbers until the user enters zero.

```
int main()
 {
    double number, sum = 0;
    // the body of the loop is executed at least once
    do
    {
        cout<<"Enter a number: ";
        cin>>number;
        sum += number;
     } while(number != 0.0);
    cout<<"Sum = "<<sum;
    return 0;
}
```

Exercise: Extend previous exercise to get average, min and max of the input numbers.

Infinite loop

- **Infinite loop**: continues to execute endlessly
  - Avoided by including statements in loop body that assure exit condition is eventually `false`
- Example of infinite loops:

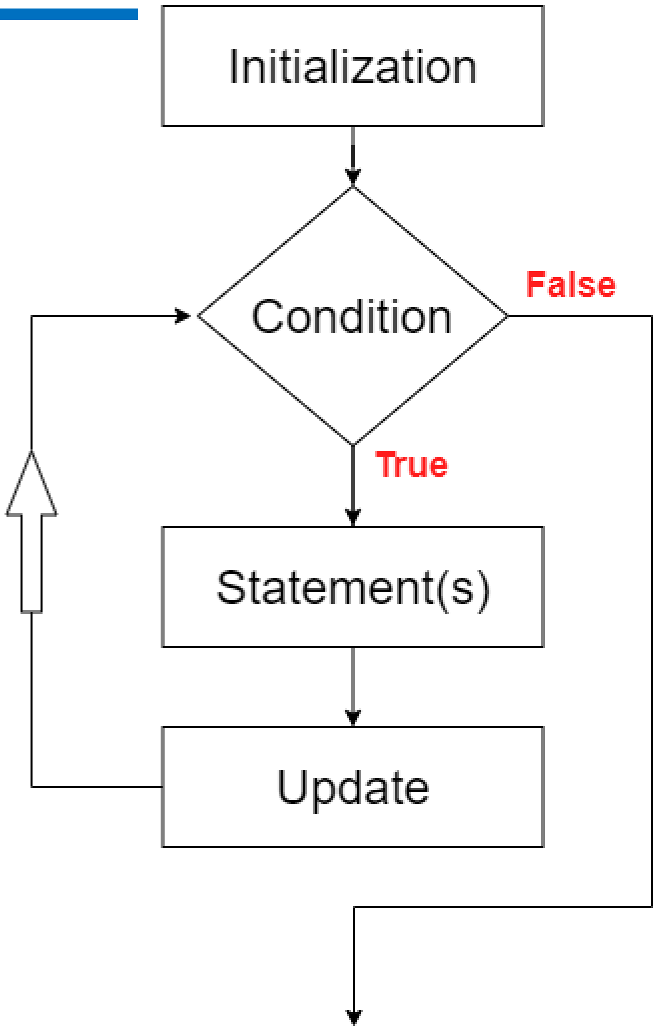
```
for( ; ; ) {
   cout<<"This loop will run forever.\n";
}

while(true)
{
    cout<<"This loop will run forever.\n";
}
```

Choosing the correct looping

- Number of repetitions is known —> **for loop**.
- Number of repetitions unknown + could be zero —> **while loop**.
- Number of repetitions unknown + at least 1 —> **do...while loop**.

Control Statement: break and continue

- The break statement, when executed it provides an immediate exit from the loop structure.
  - The break statement is typically used to exit early from a loop.
  - After the break statement executes, the program continues to execute with the first statement after the structure.
- The continue statement is used in while, for, and do.. while structures. When the continue statement is executed in a loop, it skips the remaining iteration/statements in the loop and proceeds with the next iteration of the loop.



Exercise_3: Print **even** numbers between [1, 100]. (using continue statement)

```
for(int i=1; i<=100; i++)
   {
      if (i%2!=0)
         continue;
      cout<<i<<"\n";
   }
```

Exercise_4: Print numbers between [1, 100]. (using break statement)

```
for(int i=1; ; i++)
  {
    if (i>100)
      break;
    cout<<i<<"\n";
  }
```

```
int i=1;
while (true)
{
      if (i>100)
         break;
      cout<<i++<<"\n";
   }
```

Nested Control Structures

1. **Write a c++ program to find the multiplication table of a given number**
2. **Modify the program to find the multiplication table of for all numbers between 1:10**

Part_1

```
int n;
cin>>n;
for(int i=1; i<=n; i++)
{
   cout<<n*i<<"\t";
}
cout<<"\n";
```

Part_2

```
int n;
for(n=1;n<=10;n++)
   {
      for(int i=1; i<=n; i++)
      {
         cout<<n*i<<"\t";
      }
      cout<<"\n";
   }
```

Nested Control Structures

Write a program to create the following pattern:

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

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

Solving *'s and numbers pattern problems

- https://www.youtube.com/playlist?list=PLwCMLs3sjOY4viWniHr0oMn0nyRU_G2dz