Al-Mustaqbal University

College of Engineering & Technology
Biomedical Engineering Department

Computer

Lecture 5 & 6
Conditions & Loops statements

Dr. Ahmed Hasan Janabi
PhD in Cybersecurity
Email: Ahmed.Janabi@uomus.edu.iq

Conditions

- ❖ Conditions are very important to understand the flow-control of the code being executed.
- ❖ Conditions are used to make a decision (choose between two or more alternatives based on the condition)

  - ➢ For example, conditions become very essential to use when facing different directions, so based on the condition should choose the thing to be executed.

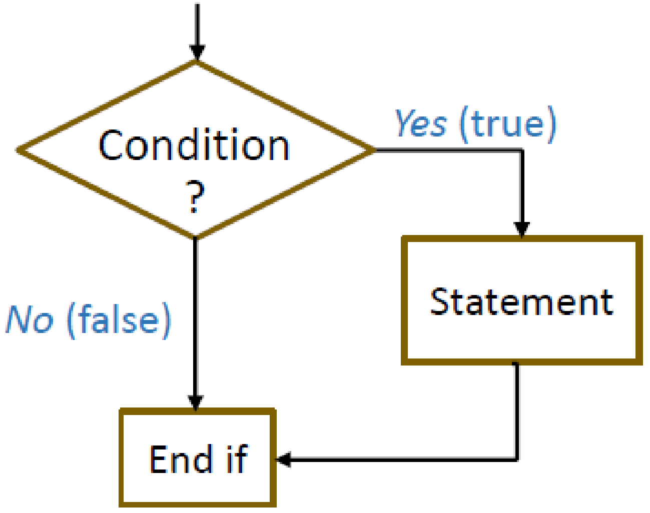
Use the *if* statement

```
If (condition) then
 run this statement;
Else
run the other statement;
```

Conditions

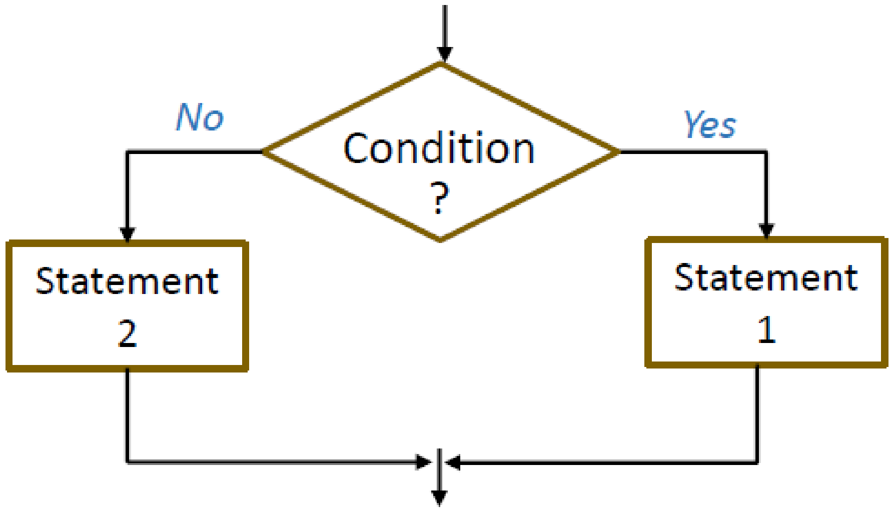
1 *if .... then*

```
If (condition) then
 statement(s)

End if
```



2 *if .... then .... else*

```
If (condition) then
 statement 1
Else
 statement 2

End if
```



Nested if structure

3 ***Nested structure of***
if .... then

```
If (condition 1) then
 statement 1

If (condition 2) then
 statement 2

If (condition 3) then
 statement 3

Else
   statement 4

End if
End if
End if
```

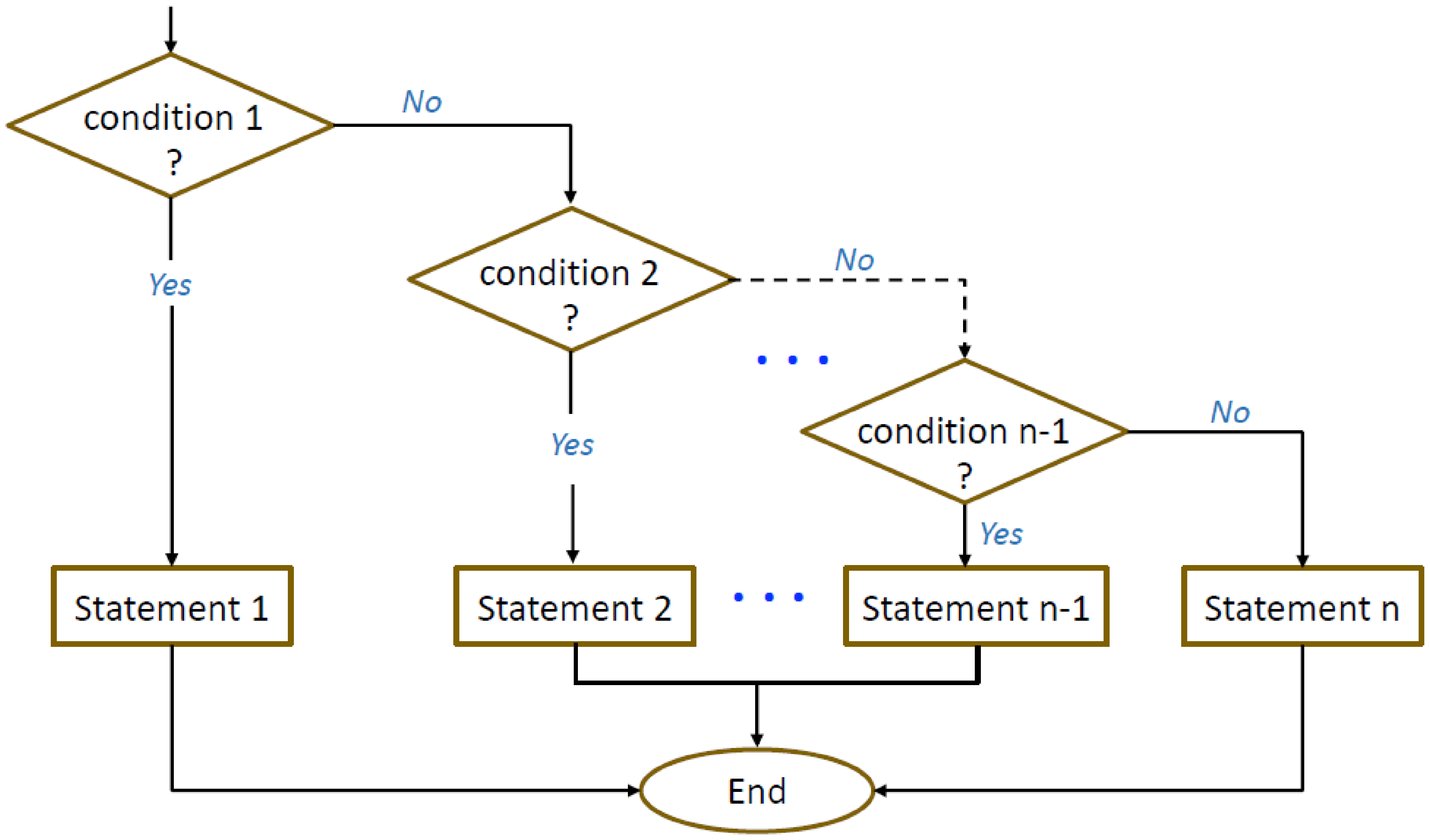
⟺

if .... then .... elseif

```
If (condition 1) then
 statement 1

Elseif (condition 2) then
 statement 2

Elseif (condition 3) then
 statement 3

Else
   statement 4

End if
```

if … then … elseif – Flowchart

- *if …. then …. elseif* Flowchart



Example

```
#include <iostream>
using namespace std;
int main()
{
 int number;
cout << "Enter an integer: ";
cin >> number;
if ( number > 0)
{
cout << "You entered a positive integer: " << number ;
 }
return 0;

}
```

Example

```
#include <iostream>
using namespace std;
int main()
{
 int number;
 cout << "Enter an integer: ";
 cin >> number;
if ( number >= 0) {
 cout << "You entered a positive integer: " << number; }
 else {
cout << "You entered a negative integer: " << number; }
}
```

LOOPS :

- ***<u>WHY DO WE NEED LOOPS ???</u>***

- There may be a situation, when you need to execute a block of code several number of times.
- In general statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.
- A loop statement allows us to execute a statement or group of statements multiple times

LOOPS :

- ***TYPES OF LOOPS :***

- WHILE LOOP
- FOR LOOP
- DO-WHILE LOOP
- NESTED LOOP

- ***LETS HAVE A CLOSER LOOK***

LOOPS => WHILE LOOP

A **while** loop statement repeatedly executes a target statement as long as a given condition is true.

Syntax:

The syntax of a while loop in C is:

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

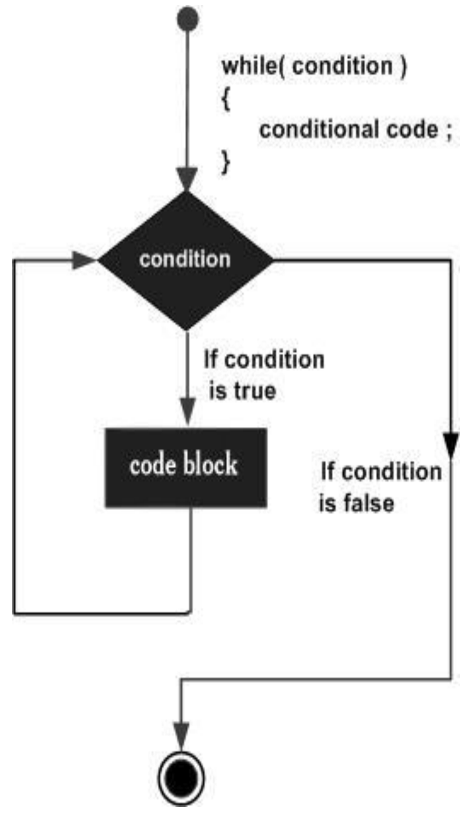
LOOPS => WHILE LOOP

- Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is 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

LOOPS => WHILE LOOP

FLOW DAIGRAM



LOOPS => WHILE LOOP

- EXAMPLE :

```
#include <iostream>
#include<stdlib.h>
int main ()
{
int a = 10;
while( a < 20 )
 {
      printf(“value of a:%d \n”, a);
       a++;
  }
}
```

LOOPS => WHILE LOOP

- When the above code is compiled and executed, it produces the following result:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

LOOPS :

<u>FOR LOOP:</u>

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.

<u>Syntax:</u>

The syntax of a for loop in C is:

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

LOOPS => FOR LOOP

- 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.

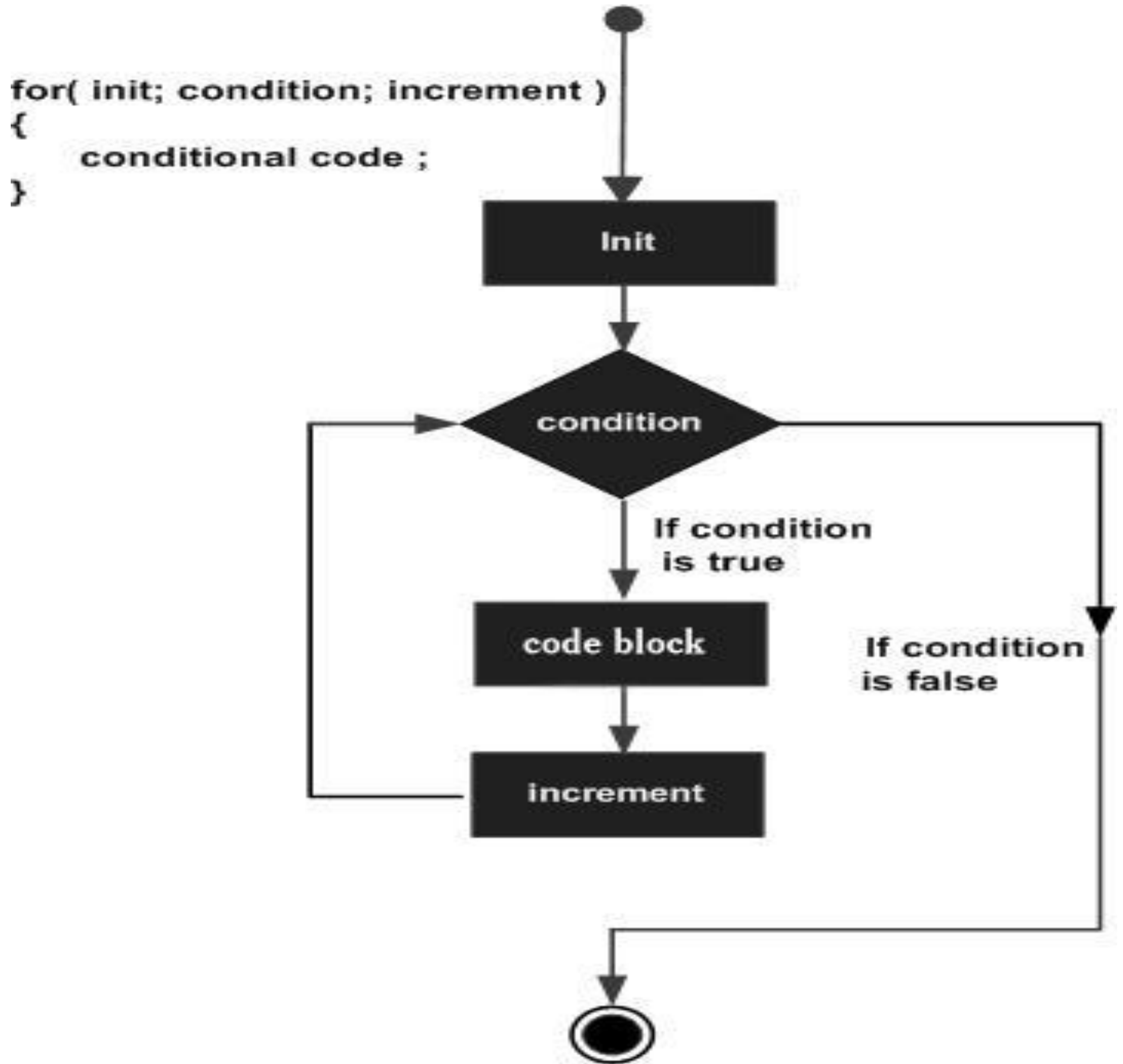
- Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the for loop.

LOOPS => FOR LOOP

- 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 loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.
- 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.

LOOPS=> FOR LOOP

- **Flow Diagram:**



LOOPS => FOR LOOP

Example:

```
#include<stdlib.h>
#include <iostream>
int main ()
{
for( int a = 10; a < 20; a = a + 1 )
  {
    printf("value of a: %d \n”, a);
  }
}
```

LOOPS => FOR LOOP

- When the above code is compiled and executed, it produces the following result:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

LOOPS :

- ***<u>DO-WHILE LOOP:</u>***

- Unlike **for** and **while** loops, which test the loop condition at the top of the loop, the **do...while** loop checks its condition at the bottom of the loop.

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

LOOPS => DO-WHILE LOOP

Syntax:

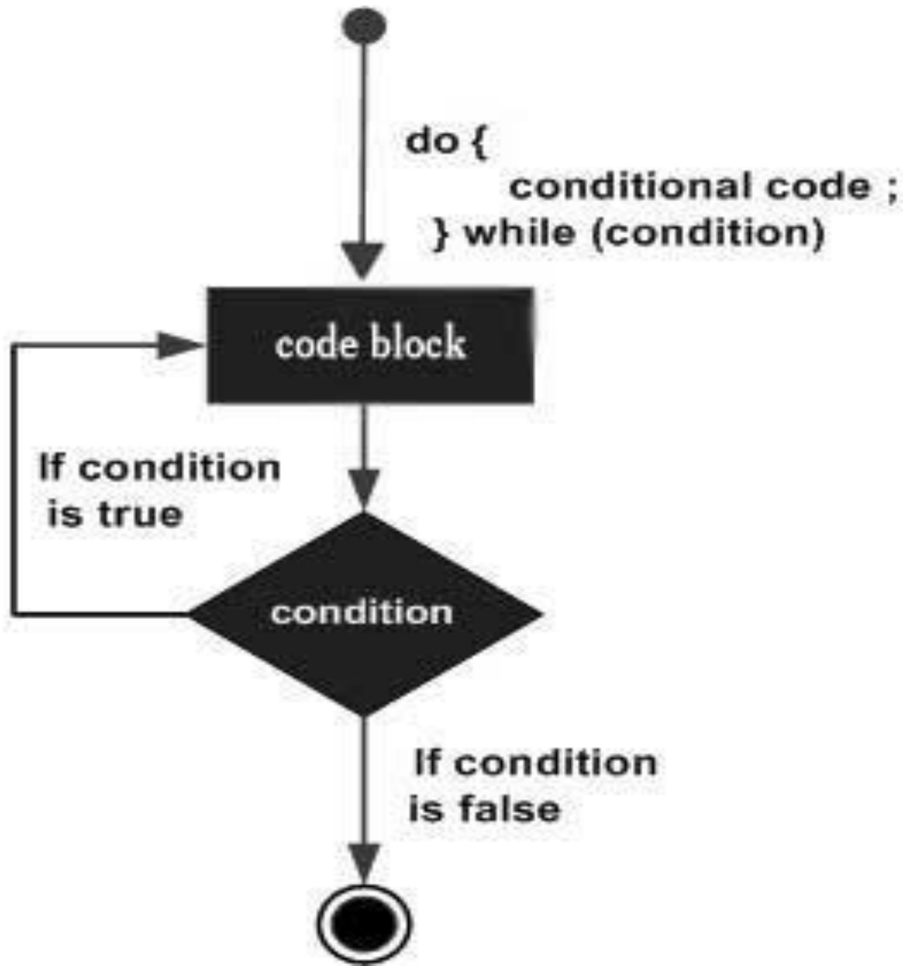
The syntax of a do...while loop in C is:

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

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop execute once before the condition is tested.

LOOPS => DO-WHILE LOOP

- **Flow Diagram:**



LOOPS => DO-WHILE LOOP

- **Example:**

```
#include<stdlib.h>
#include <iostream>
   int main ()
 {
      int a = 10;
do
      {
        printf( "value of a: %d\n “ ,a);
        a = a + 1;
       } while( a < 20 );
}
```

LOOPS => DO-WHILE LOOP

- When the above code is compiled and executed, it produces the following result:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

LOOPS :

- ***NESTED LOOPS :***

- A loop can be nested inside of another loop.

- **Syntax:**

The syntax for a **nested for loop** statement in C is as follows:

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

LOOPS => NESTED LOOP

- ***EXAMPLE :***

```
  #include<stdlib.h>
  int main ()
  {
     int a=1,b;
     while(a<=3)
      {
         for(b=1;b<=3;b++)
          {
            printf("a = %d , b = %d\n",a,b);
          }
         printf("\n");
         a++;
      }
}
```

LOOPS => NESTED LOOP

- When the above code is compiled and executed, it produces the following result:

```
a = 1 , b = 1
a = 1 , b = 2
a = 1 , b = 3

a = 2 , b = 1
a = 2 , b = 2
a = 2 , b = 3

a = 3 , b = 1
a = 3 , b = 2
a = 3 , b = 3
```

Exercises

Exercise 1: Write a program to check if a number is even or odd.

```
#include <stdio.h>
#include <iostream>
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);

   if (num % 2 == 0) {
      printf("%d is an even number.\n", num);
   } else {
      printf("%d is an odd number.\n", num);
   }
}
```

Exercises

Exercise 2: Write a program that takes an integer input N and prints all even numbers from 1 to N using a for loop.

```
#include <iostream>
using namespace std;
int main() {
   int N;
   cout << "Enter the value of N: ";
   cin >> N;
   cout << "Even numbers from 1 to " << N << " are: ";
   for (int i = 1; i <= N; i++) {
      if (i % 2 == 0) {
         cout << i << " ";
      }
   }
   cout << endl;
}
```

Exercises

Exercise 3: Write a program that calculates the <u>sum</u> of the first N natural numbers using a <u>while</u> loop.

```
#include <iostream>
using namespace std;
int main() {
   int N, sum = 0, i = 1;
   cout << "Enter the value of N: ";
   cin >> N;
   while (i <= N) {
      sum += i;
      i++;
   }
   cout << "The sum of the first " << N << " natural numbers is: " << sum << endl;

   return 0;
}
```

Exercises

Exercise 4: Write a program that prints the multiplication table of a number using a for loop.

```
#include <stdio.h>
int main() {
   int num;
   printf("Enter a number to print its multiplication table: ");
   scanf("%d", &num);

   printf("Multiplication table of %d:\n", num);
   for (int i = 1; i <= 10; i++) {
      printf("%d * %d = %d\n", num, i, num * i);
   }

   return 0;
}
```

Exercises

Exercise 5: Write a program to find the largest of three numbers using if-else

```
#include <stdio.h>
int main() {
  int num1, num2, num3;
  printf("Enter three numbers: ");
  scanf("%d %d %d", &num1, &num2, &num3);

  if (num1 >= num2 && num1 >= num3) {
    printf("The largest number is %d\n", num1);
  } else if (num2 >= num1 && num2 >= num3) {
    printf("The largest number is %d\n", num2);
  } else {
    printf("The largest number is %d\n", num3);
  }
}
```

THANK YOU