SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

COURSE NAME : 19CST101 PROGRAMMING FOR PROBLEM SOLVING

I YEAR/ I SEMESTER

UNIT-II C PROGRAMMING BASICS

Topic: Decision Making Statements

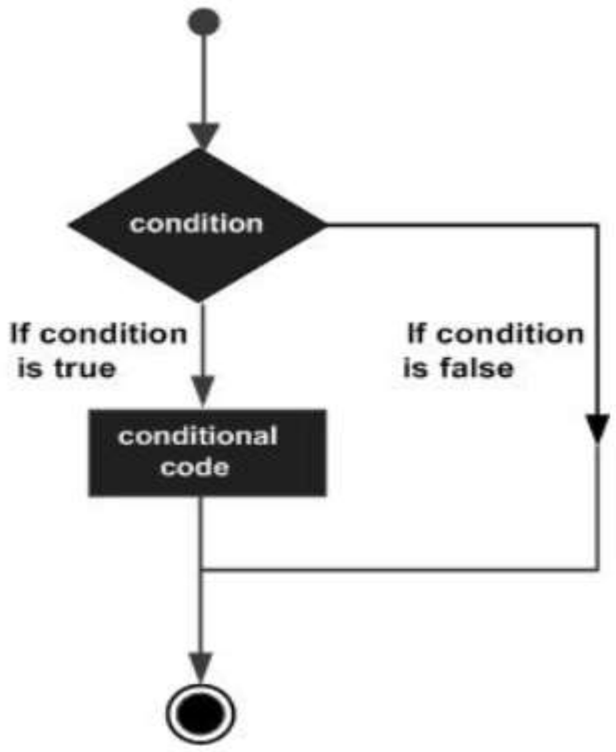
Mrs. Papithasri K
Assistant Professor
Department of Computer Science and Engineering

C Decision Making

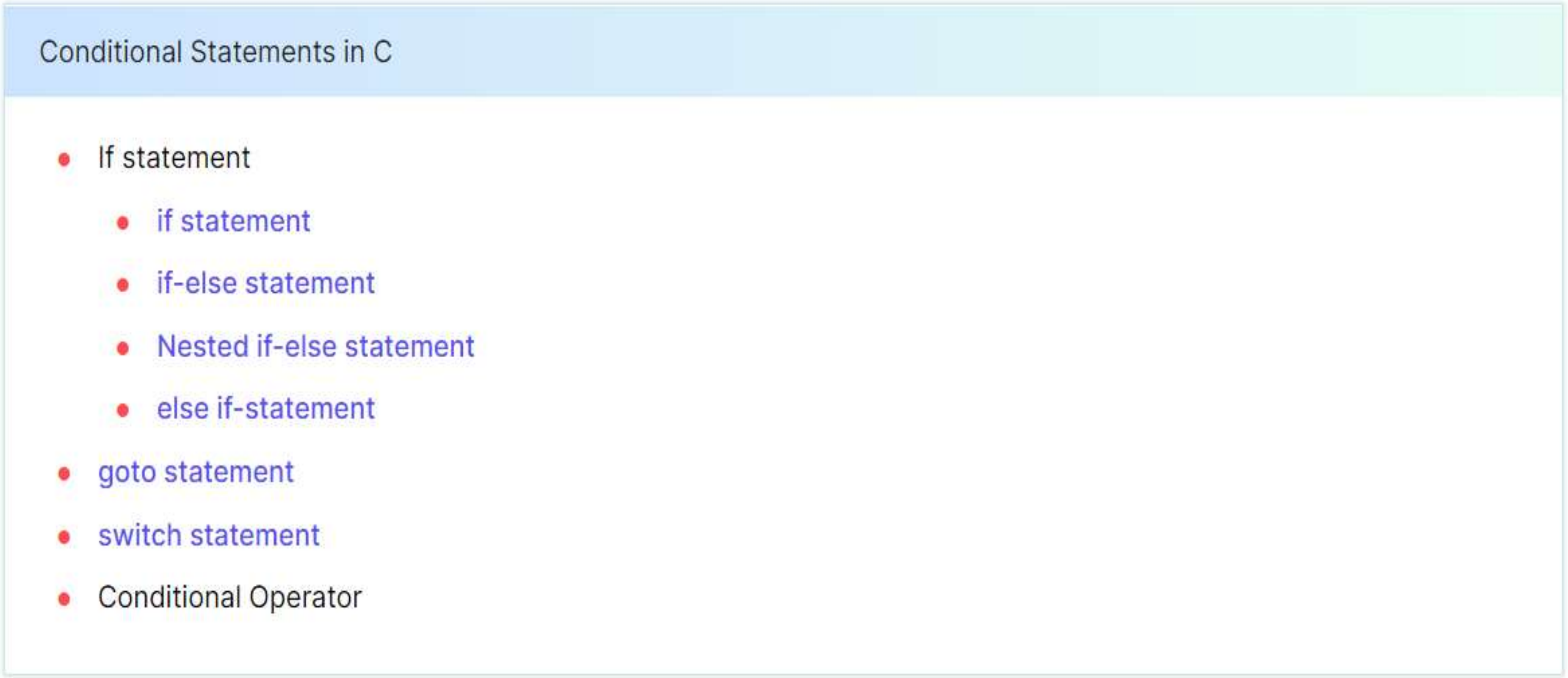
C conditional statements allow you to make a decision, based upon the result of a condition. These statements are called **Decision Making Statements** or **Conditional Statements**.

Decision-making statements are the statements that are used to verify a given condition and decide whether a block of statements gets executed or not based on the condition result.

The flowchart of the Decision-making technique in C can be expressed as:



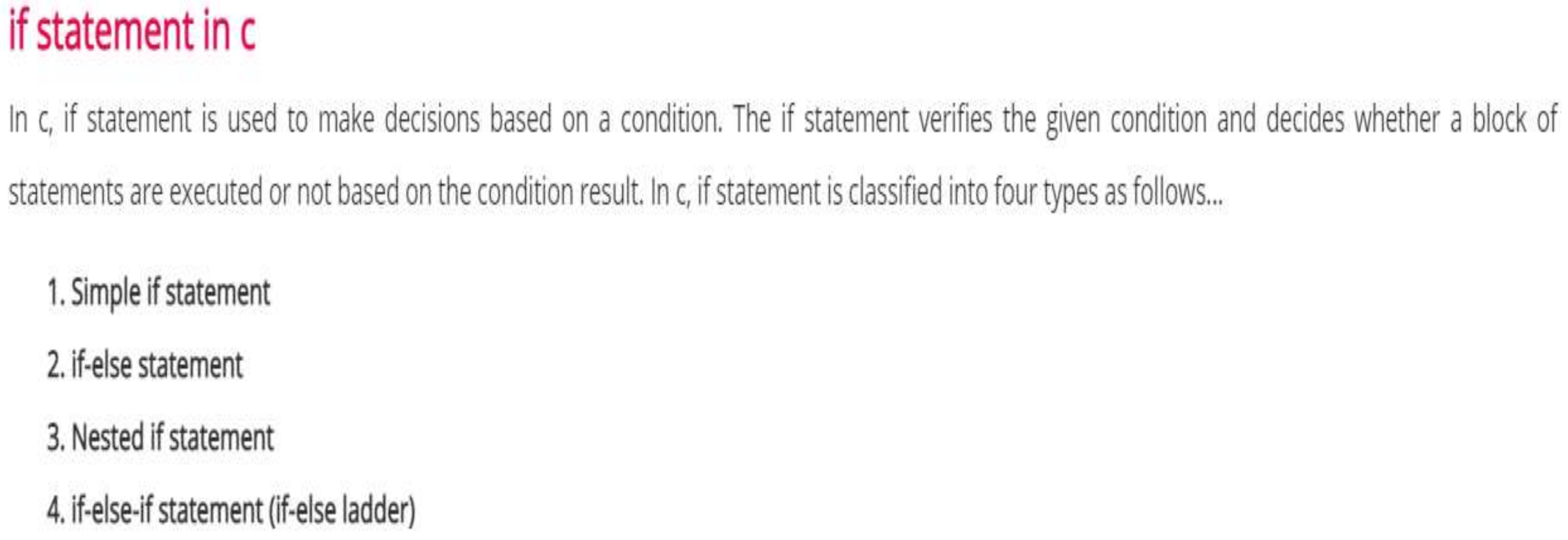
C Decision Making



Conditional Statements in C

- If statement
    - if statement
    - if-else statement
    - Nested if-else statement
    - else if-statement
- goto statement
- switch statement
- Conditional Operator

If Statements



if statement in c

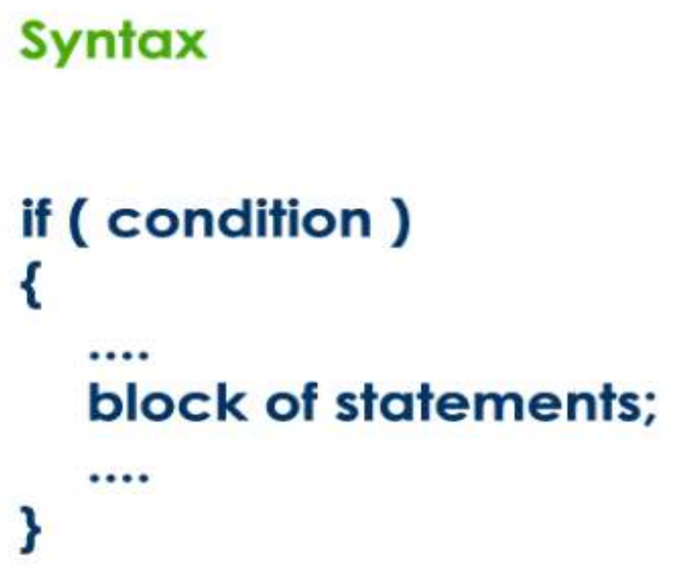
In c, if statement is used to make decisions based on a condition. The if statement verifies the given condition and decides whether a block of statements are executed or not based on the condition result. In c, if statement is classified into four types as follows...

1. Simple if statement
2. if-else statement
3. Nested if statement
4. if-else-if statement (if-else ladder)

If Statements

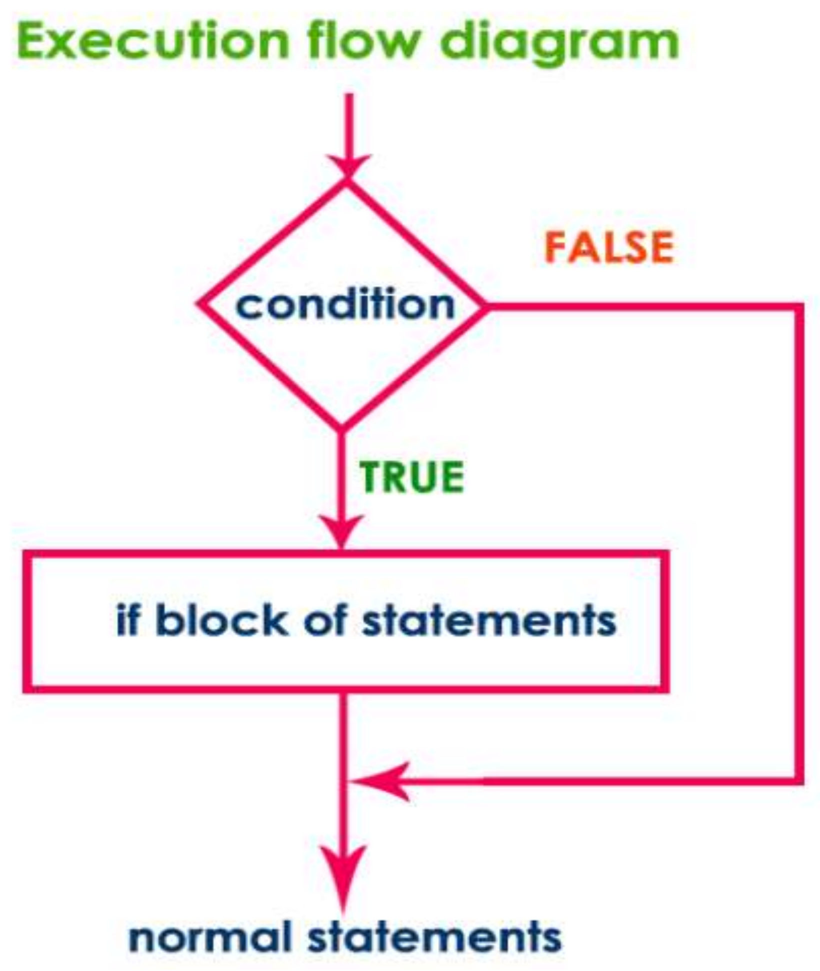
Simple if statement:

Simple if statement is used when we have only one option that is executed or skipped based on a condition.

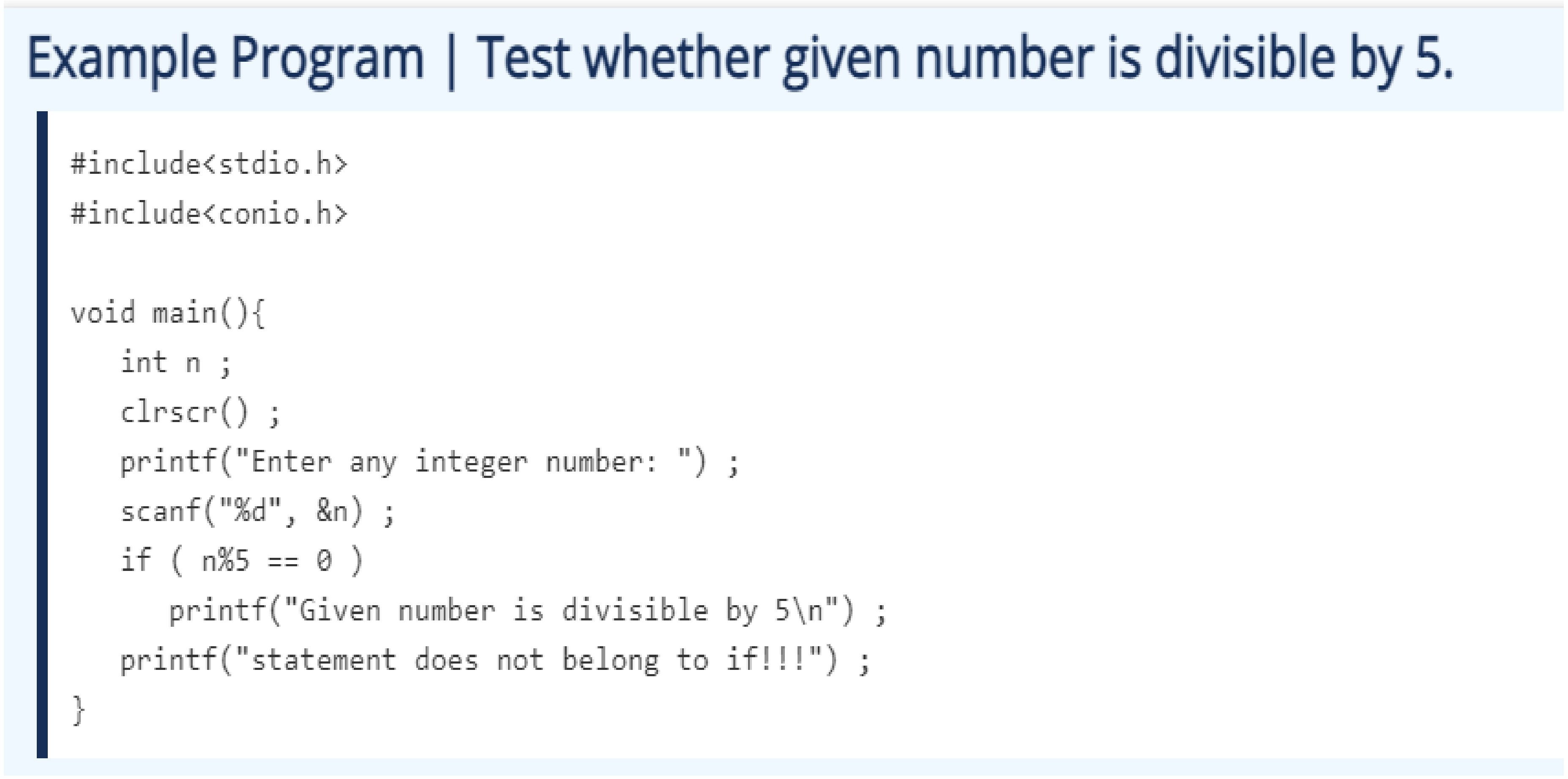


Syntax

```
if ( condition )
{
    ....
    block of statements;
    ....
}
```



If Statements

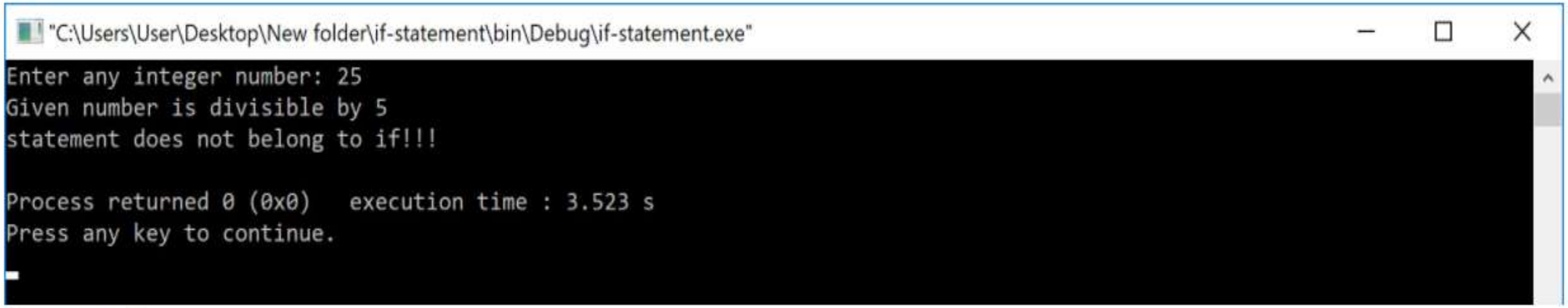


Example Program | Test whether given number is divisible by 5.

```
#include<stdio.h>
#include<conio.h>

void main(){
   int n ;
   clrscr() ;
   printf("Enter any integer number: ") ;
   scanf("%d", &n) ;
   if ( n%5 == 0 )
      printf("Given number is divisible by 5\n") ;
   printf("statement does not belong to if!!!") ;
}
```

If Statements

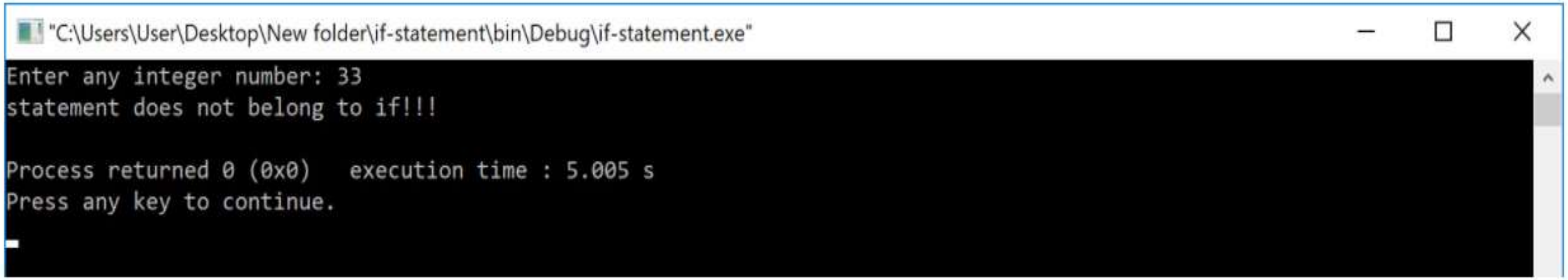
Output 1:



"C:\Users\User\Desktop\New folder\if-statement\bin\Debug\if-statement.exe"

```
Enter any integer number: 25
Given number is divisible by 5
statement does not belong to if!!!

Process returned 0 (0x0)   execution time : 3.523 s
Press any key to continue.
```

Output 2:



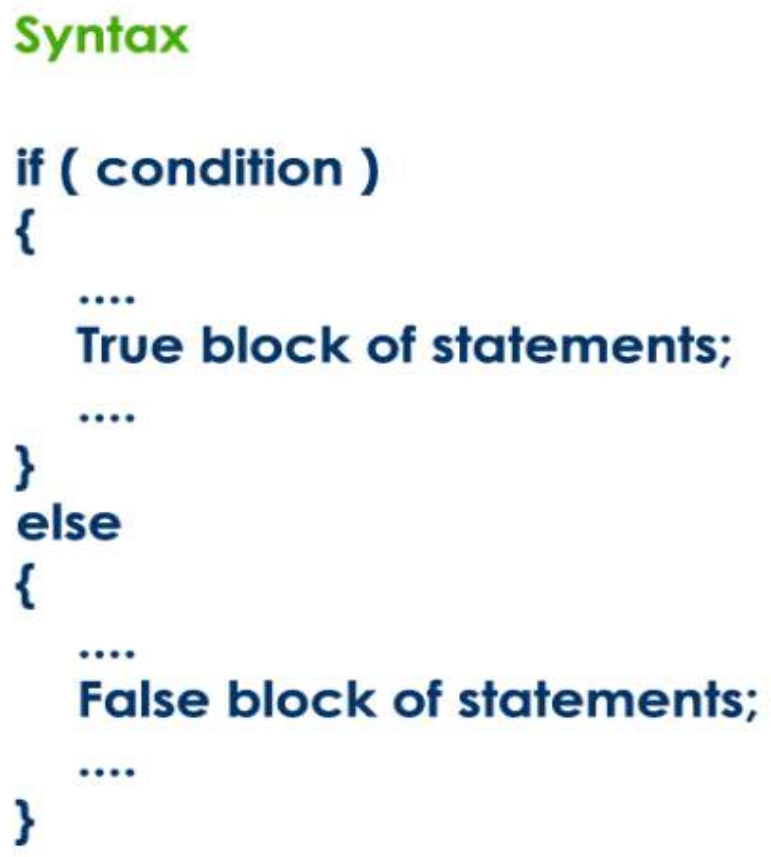
"C:\Users\User\Desktop\New folder\if-statement\bin\Debug\if-statement.exe"

```
Enter any integer number: 33
statement does not belong to if!!!

Process returned 0 (0x0)   execution time : 5.005 s
Press any key to continue.
```

If Statements

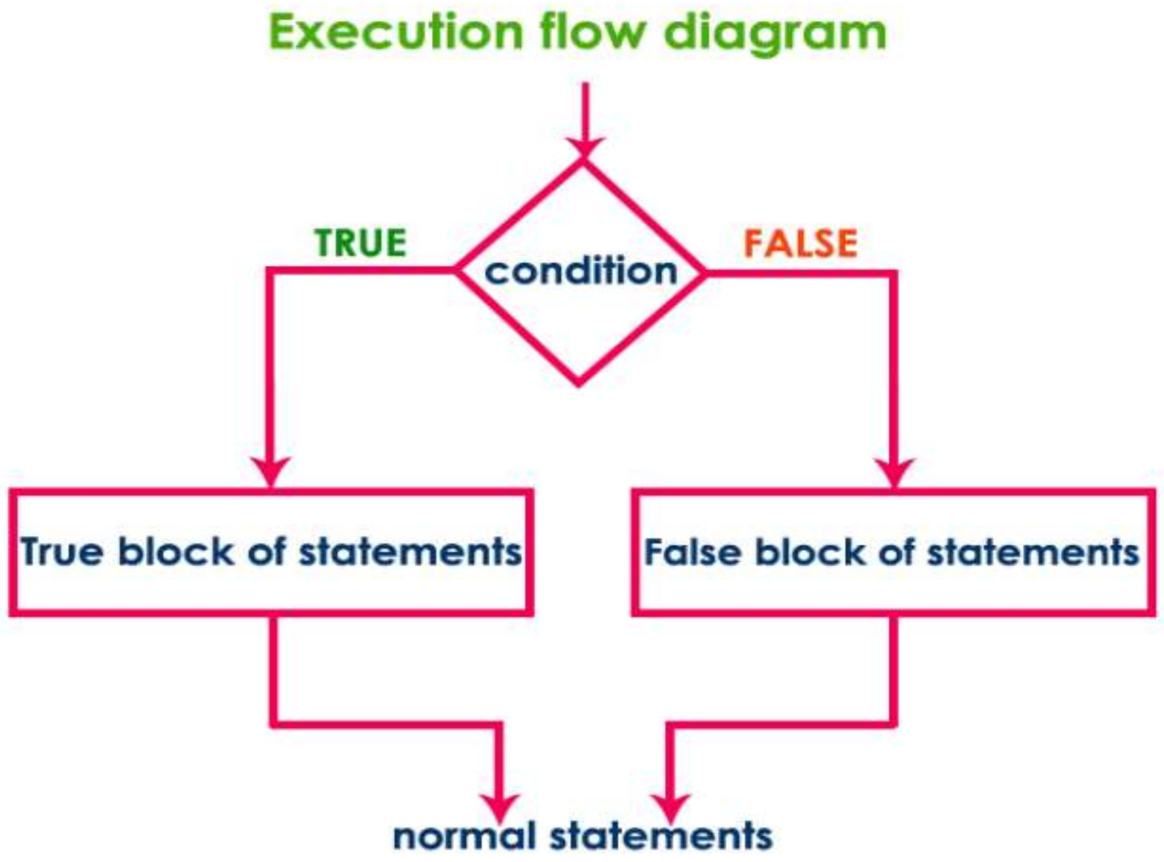
if-else statement:

The if-else statement is used when we have two options and only one option has to be executed based on a condition result (TRUE or FALSE).

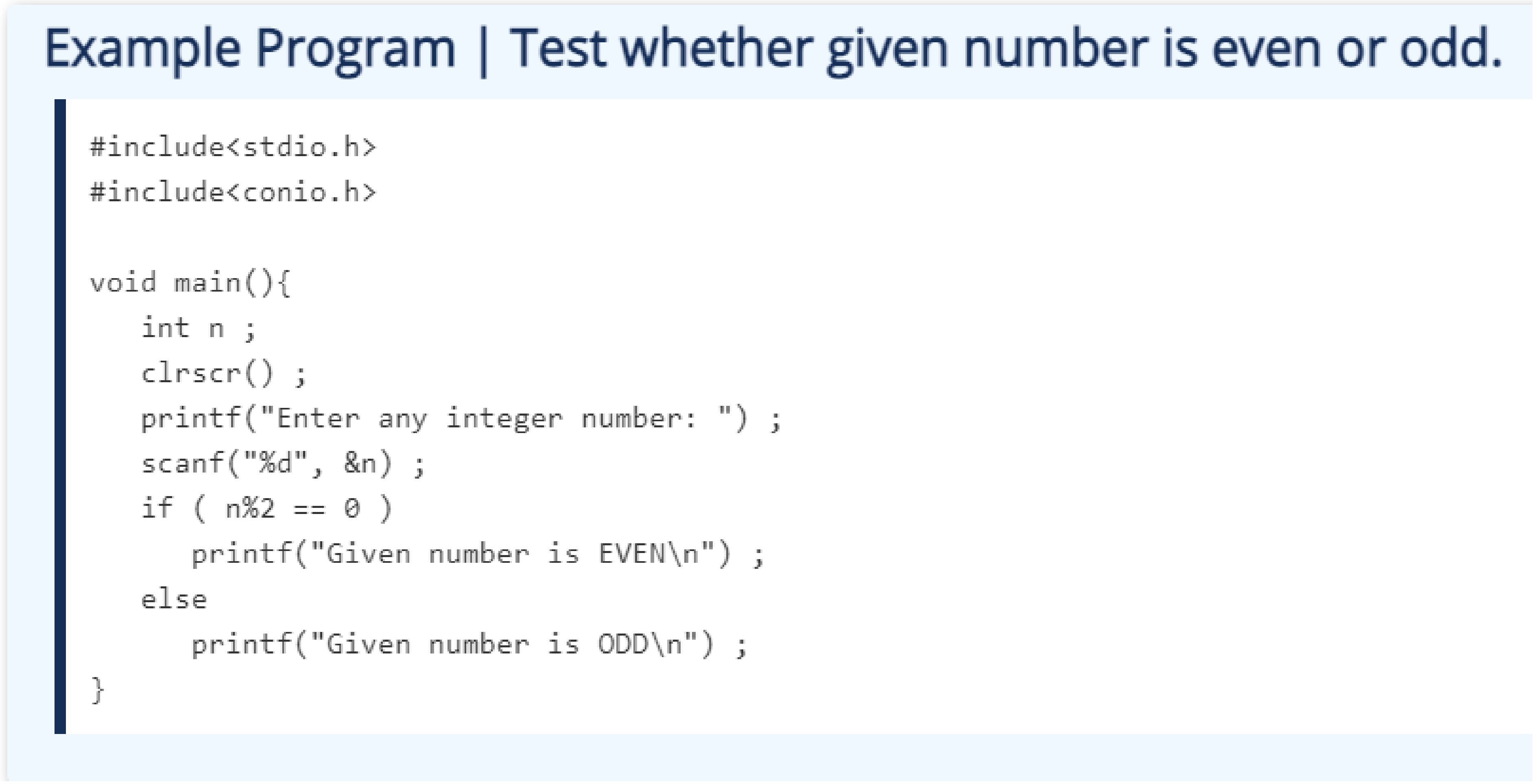


Syntax

```
if ( condition )
{
    ....
    True block of statements;
    ....
}
else
{
    ....
    False block of statements;
    ....
}
```



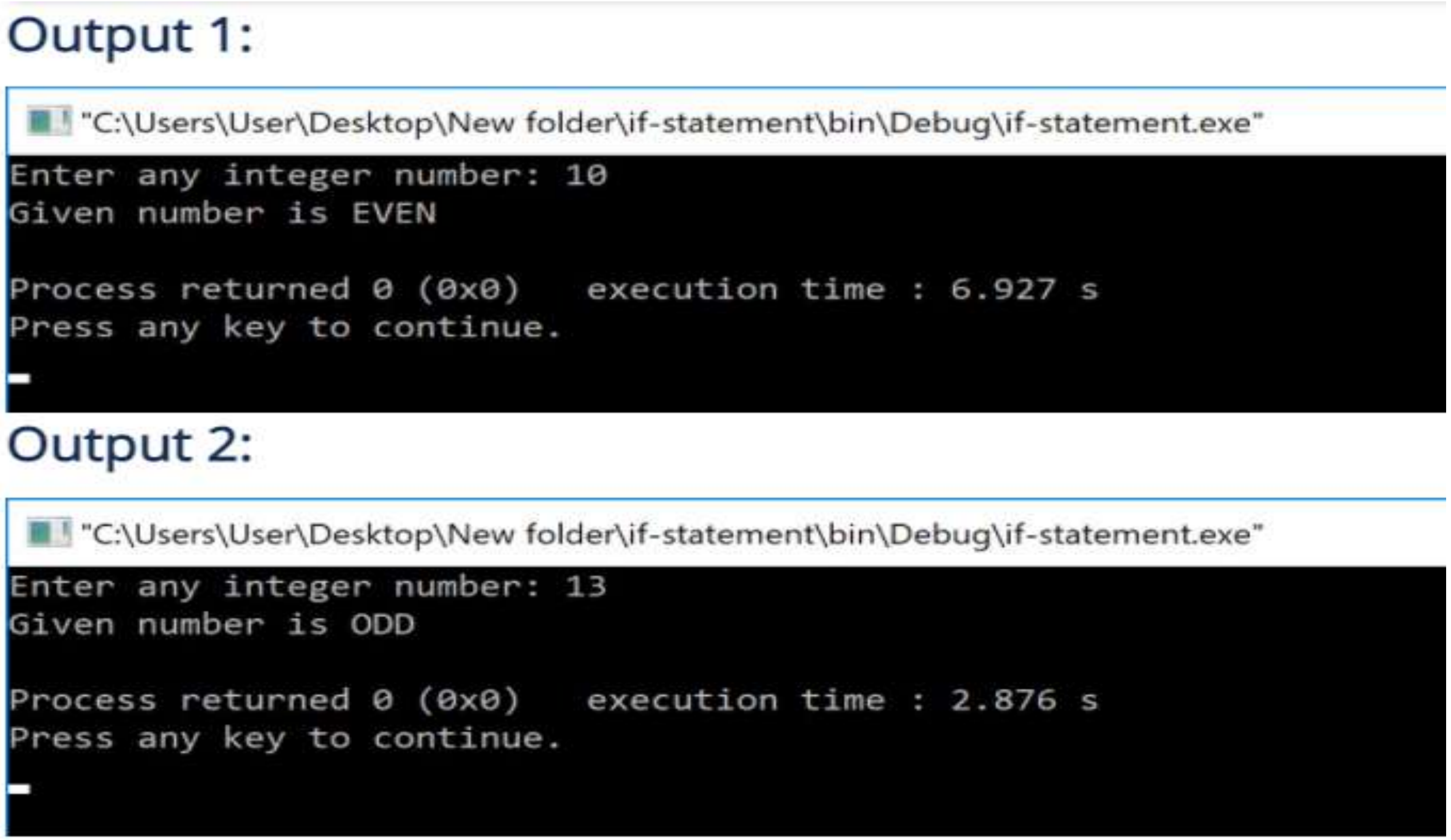
If Statements



Example Program | Test whether given number is even or odd.

```
#include<stdio.h>
#include<conio.h>

void main(){
   int n ;
   clrscr() ;
   printf("Enter any integer number: ") ;
   scanf("%d", &n) ;
   if ( n%2 == 0 )
      printf("Given number is EVEN\n") ;
   else
      printf("Given number is ODD\n") ;
}
```

If Statements


Output 1:

```
"C:\Users\User\Desktop\New folder\if-statement\bin\Debug\if-statement.exe"
Enter any integer number: 10
Given number is EVEN

Process returned 0 (0x0)   execution time : 6.927 s
Press any key to continue.
```

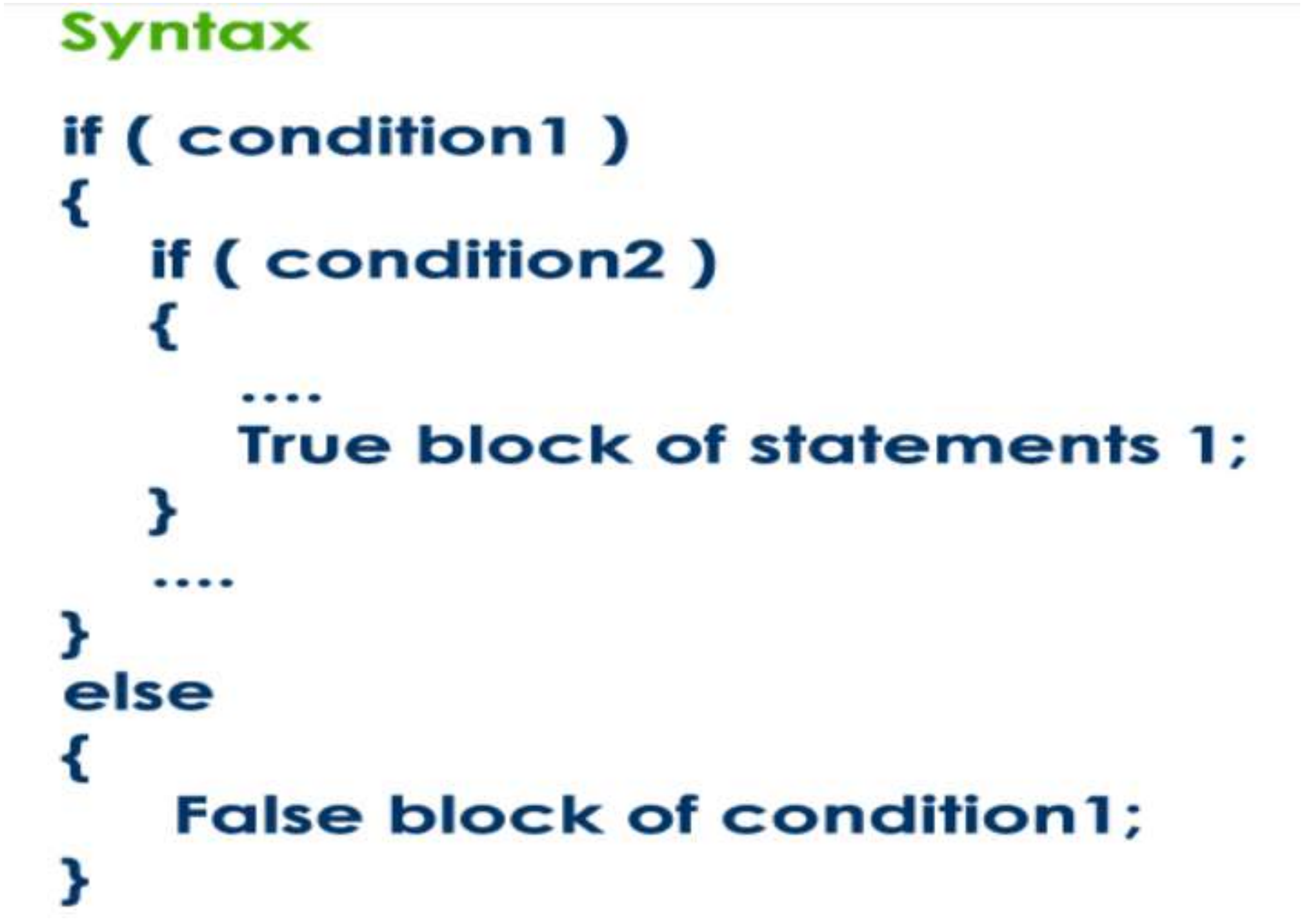
Output 2:

```
"C:\Users\User\Desktop\New folder\if-statement\bin\Debug\if-statement.exe"
Enter any integer number: 13
Given number is ODD

Process returned 0 (0x0)   execution time : 2.876 s
Press any key to continue.
```

If Statements

Nested if statement:

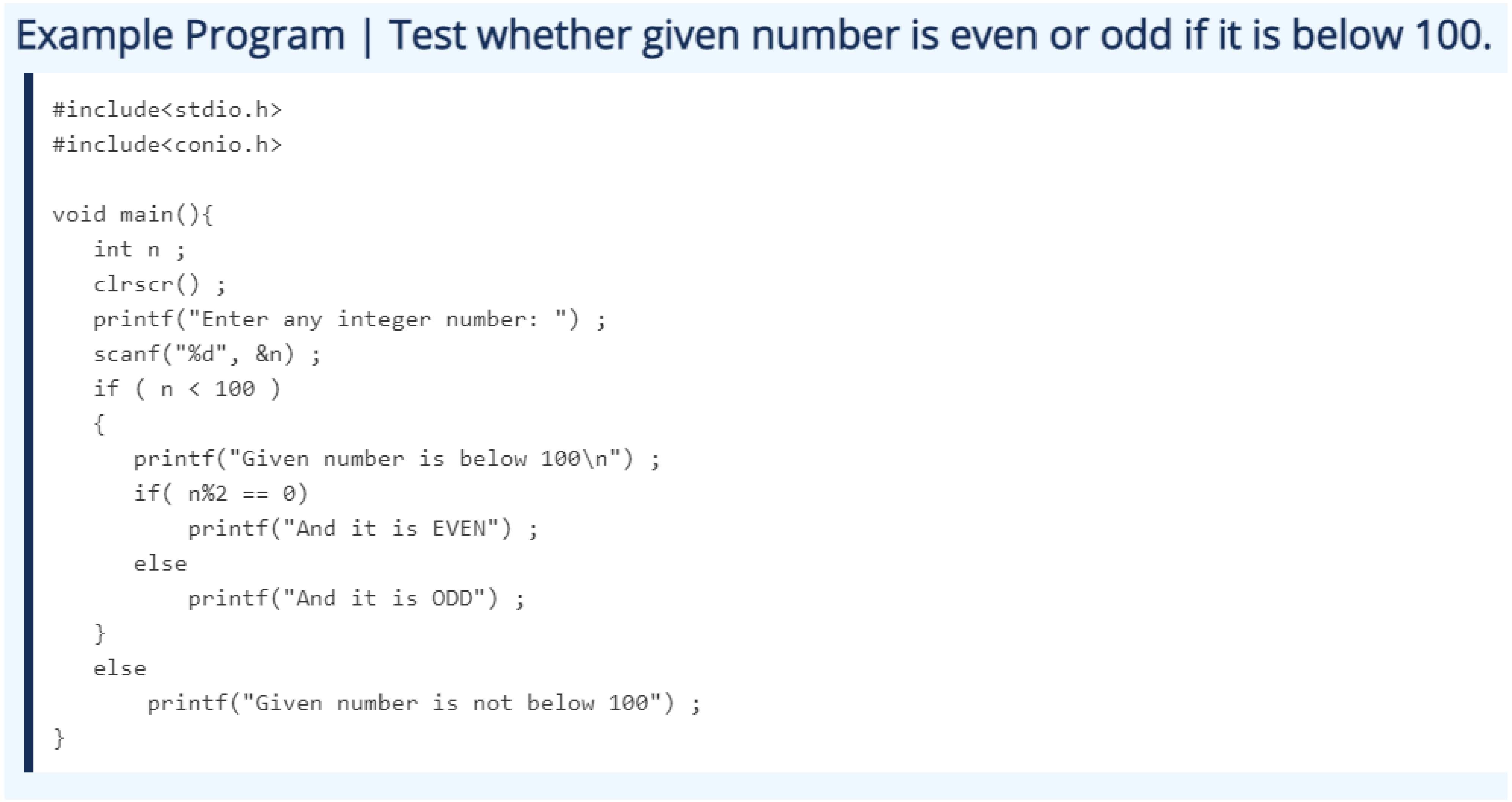
Writing a if statement inside another if statement is called nested if statement. The nested if statement can be defined using any combination of simple if & if-else statements.



Syntax

```
if ( condition1 )
{
    if ( condition2 )
    {
        ....
        True block of statements 1;
    }
    ....
}
else
{
     False block of condition1;
}
```

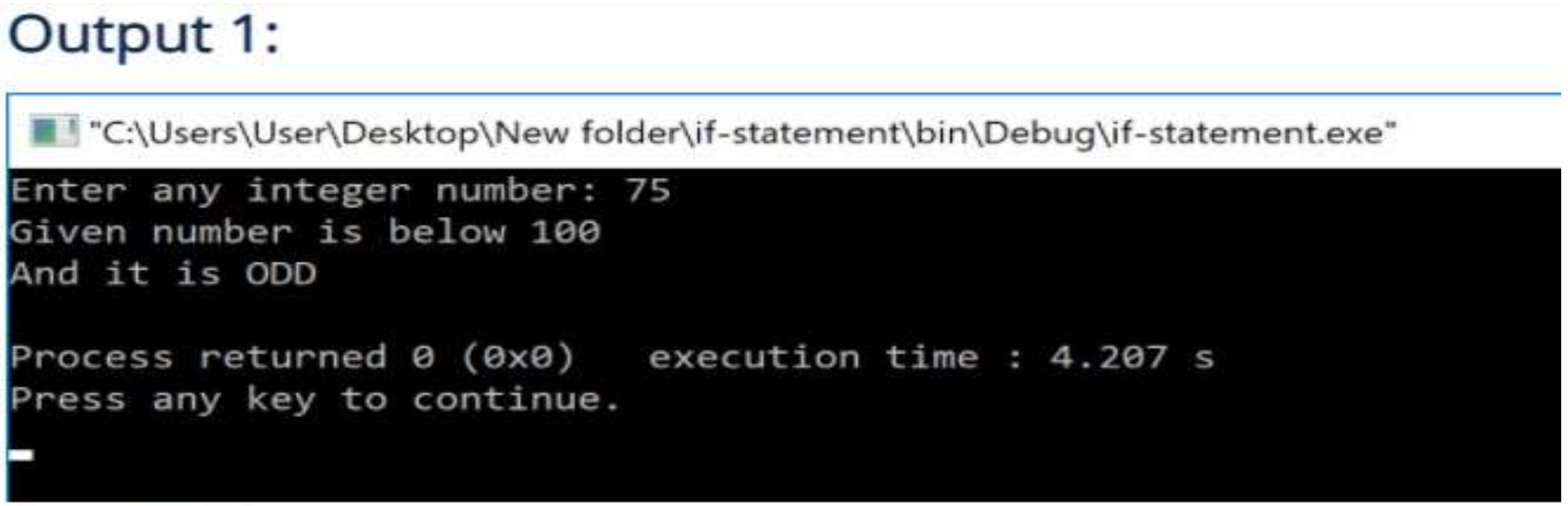
If Statements



Example Program | Test whether given number is even or odd if it is below 100.

```
#include<stdio.h>
#include<conio.h>

void main(){
   int n ;
   clrscr() ;
   printf("Enter any integer number: ") ;
   scanf("%d", &n) ;
   if ( n < 100 )
   {
      printf("Given number is below 100\n") ;
      if( n%2 == 0)
          printf("And it is EVEN") ;
      else
          printf("And it is ODD") ;
   }
   else
       printf("Given number is not below 100") ;
}
```

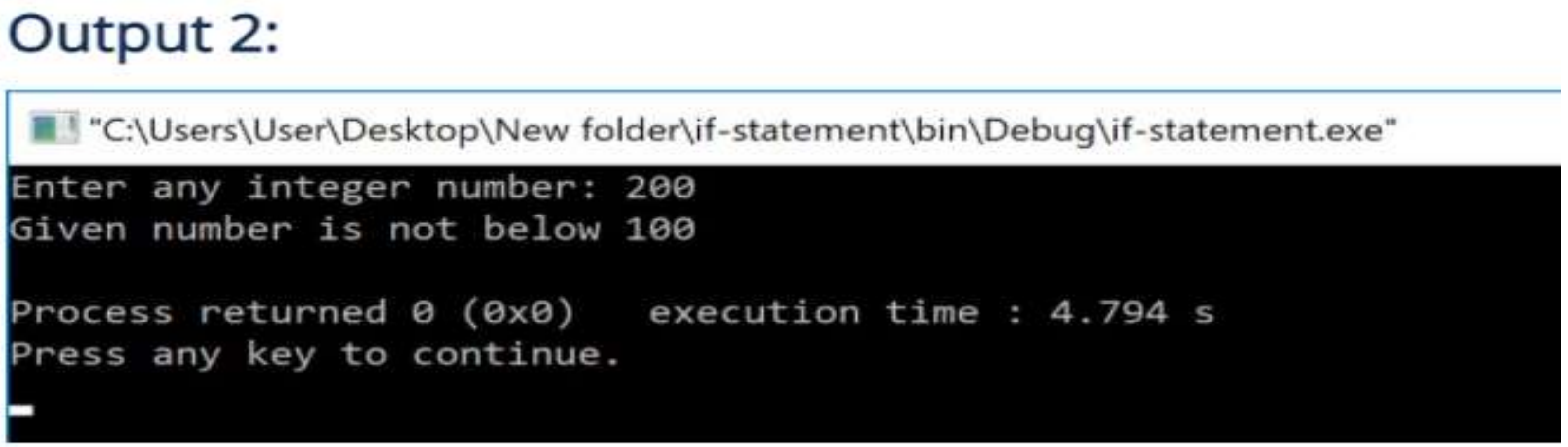
If Statements



Output 1:

"C:\Users\User\Desktop\New folder\if-statement\bin\Debug\if-statement.exe"

```
Enter any integer number: 75
Given number is below 100
And it is ODD

Process returned 0 (0x0)   execution time : 4.207 s
Press any key to continue.
```



Output 2:

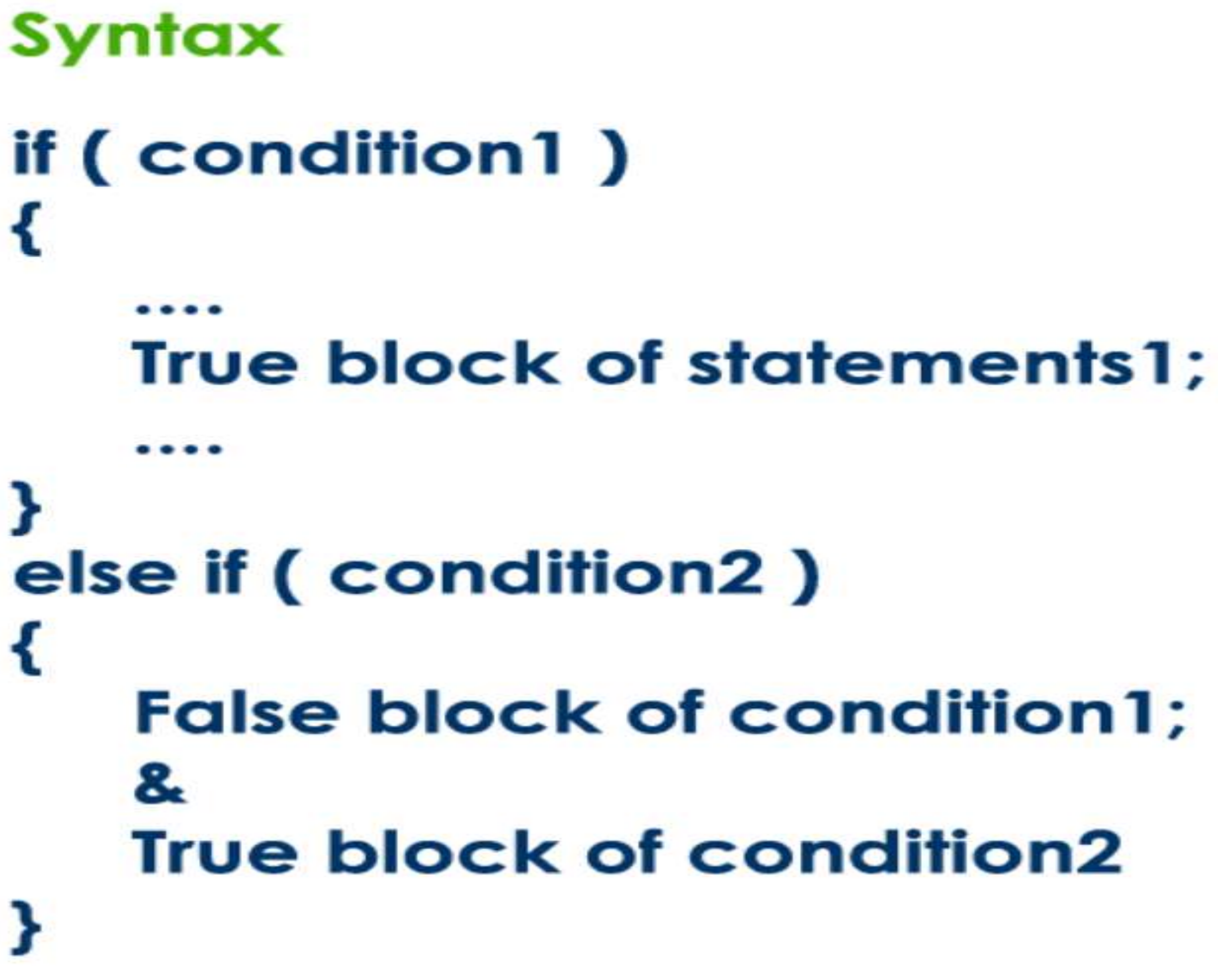
"C:\Users\User\Desktop\New folder\if-statement\bin\Debug\if-statement.exe"

```
Enter any integer number: 200
Given number is not below 100

Process returned 0 (0x0)   execution time : 4.794 s
Press any key to continue.
```

If Statements

if-else-if statement (if-else ladder):

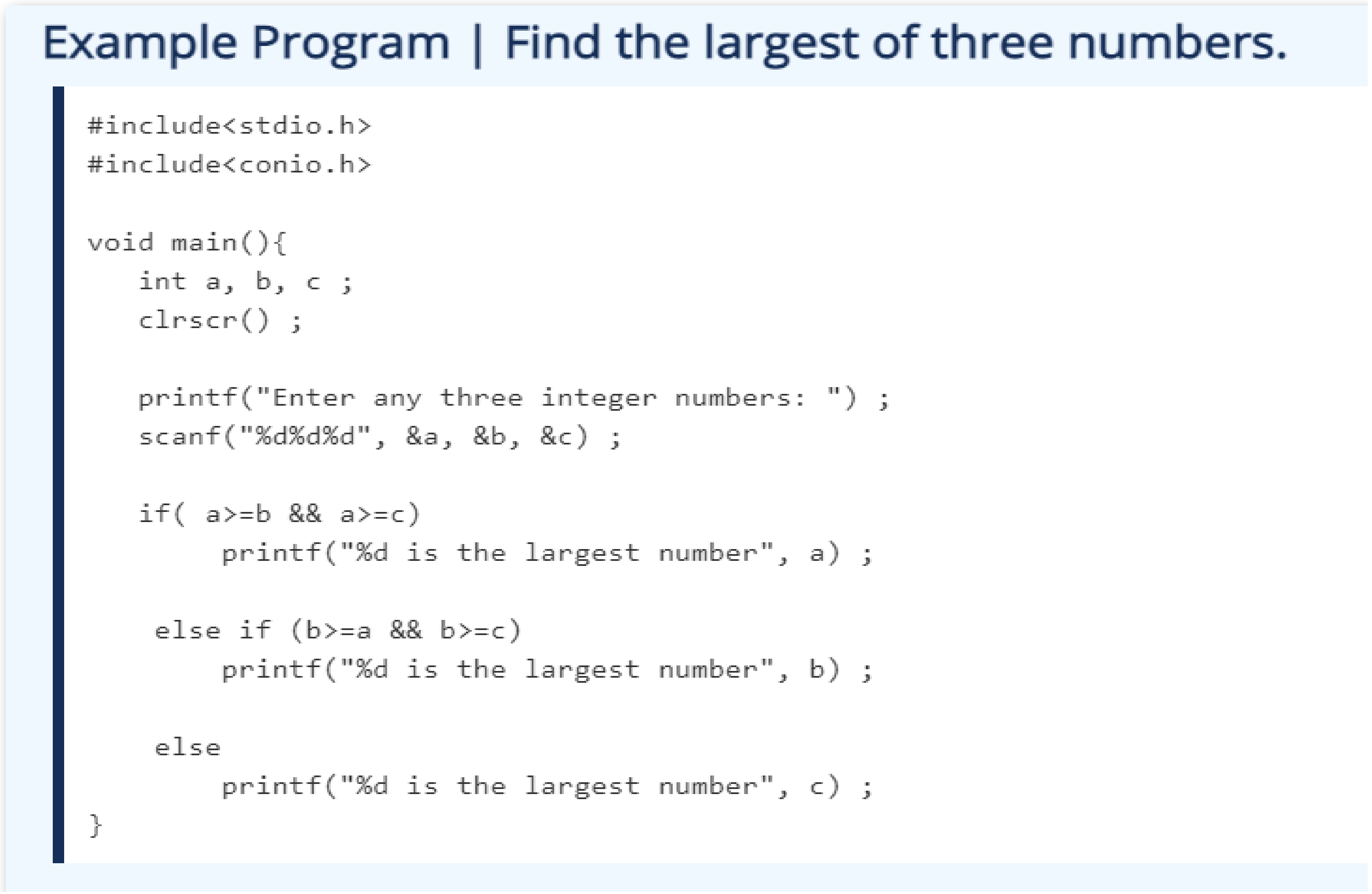
Writing a if statement inside else of an if statement is called if-else-if statement. The if-else-if statement can be defined using any combination of simple if & if-else statements.



Syntax

```
if ( condition1 )
{
     ....
     True block of statements1;
     ....
}
else if ( condition2 )
{
     False block of condition1;
     &
     True block of condition2
}
```

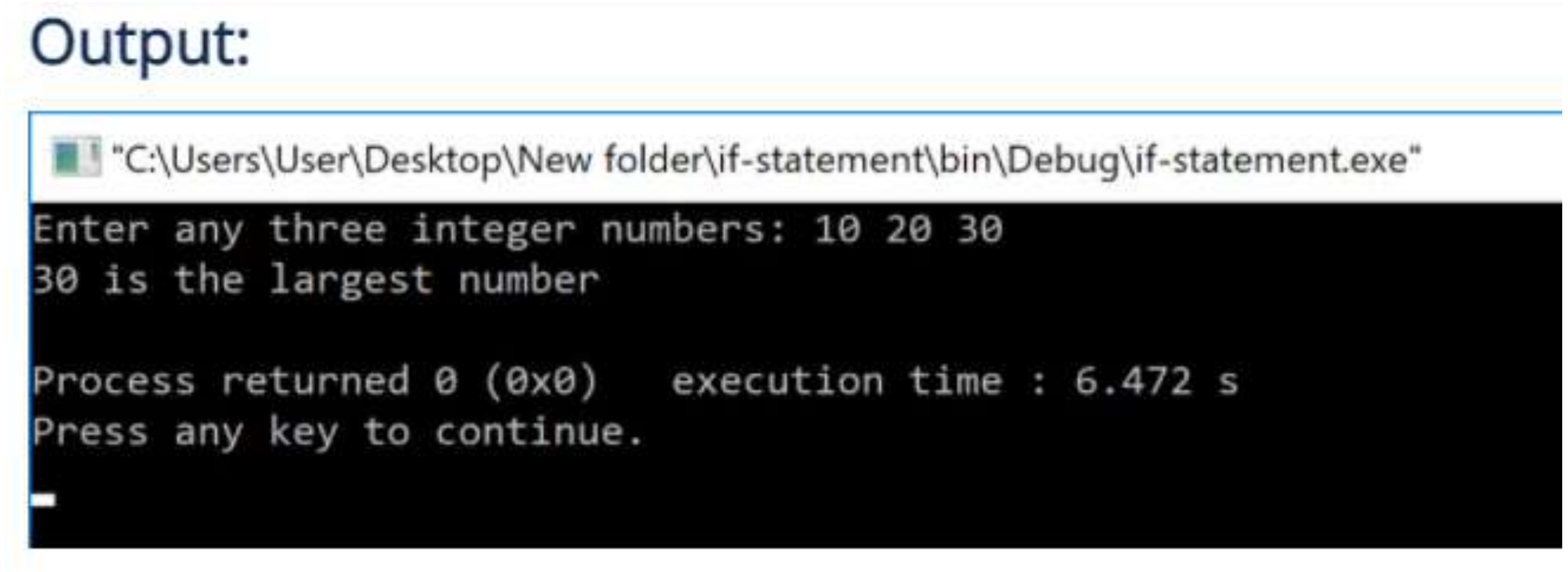
If Statements



Example Program | Find the largest of three numbers.

```
#include<stdio.h>
#include<conio.h>

void main(){
   int a, b, c ;
   clrscr() ;

   printf("Enter any three integer numbers: ") ;
   scanf("%d%d%d", &a, &b, &c) ;

   if( a>=b && a>=c)
        printf("%d is the largest number", a) ;

    else if (b>=a && b>=c)
        printf("%d is the largest number", b) ;

    else
        printf("%d is the largest number", c) ;
}
```

If Statements

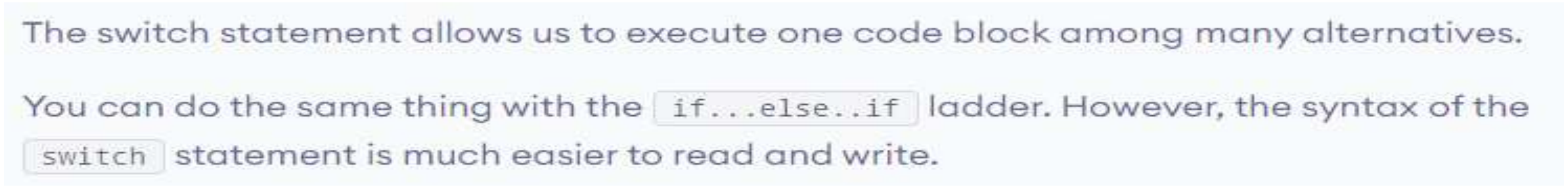


Output:

"C:\Users\User\Desktop\New folder\if-statement\bin\Debug\if-statement.exe"

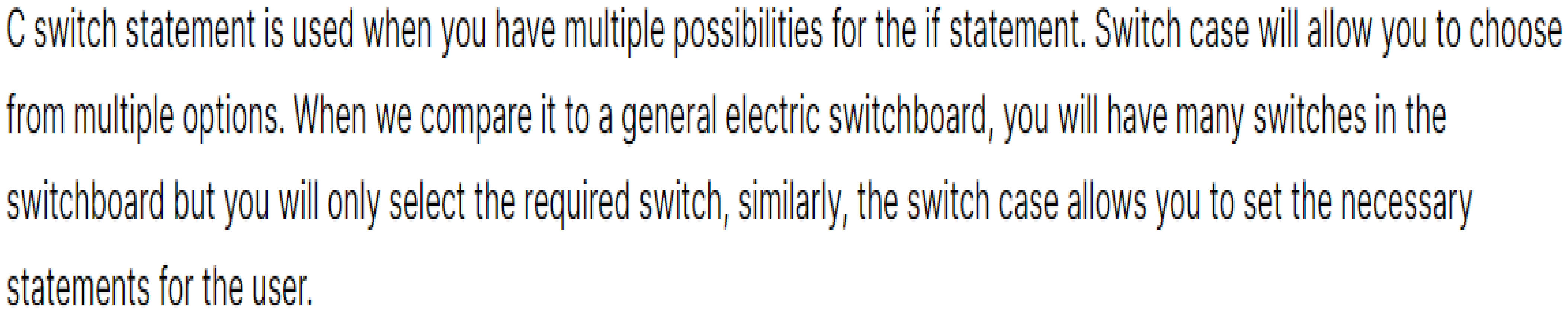
```
Enter any three integer numbers: 10 20 30
30 is the largest number

Process returned 0 (0x0)   execution time : 6.472 s
Press any key to continue.
```

'switch' statement in C



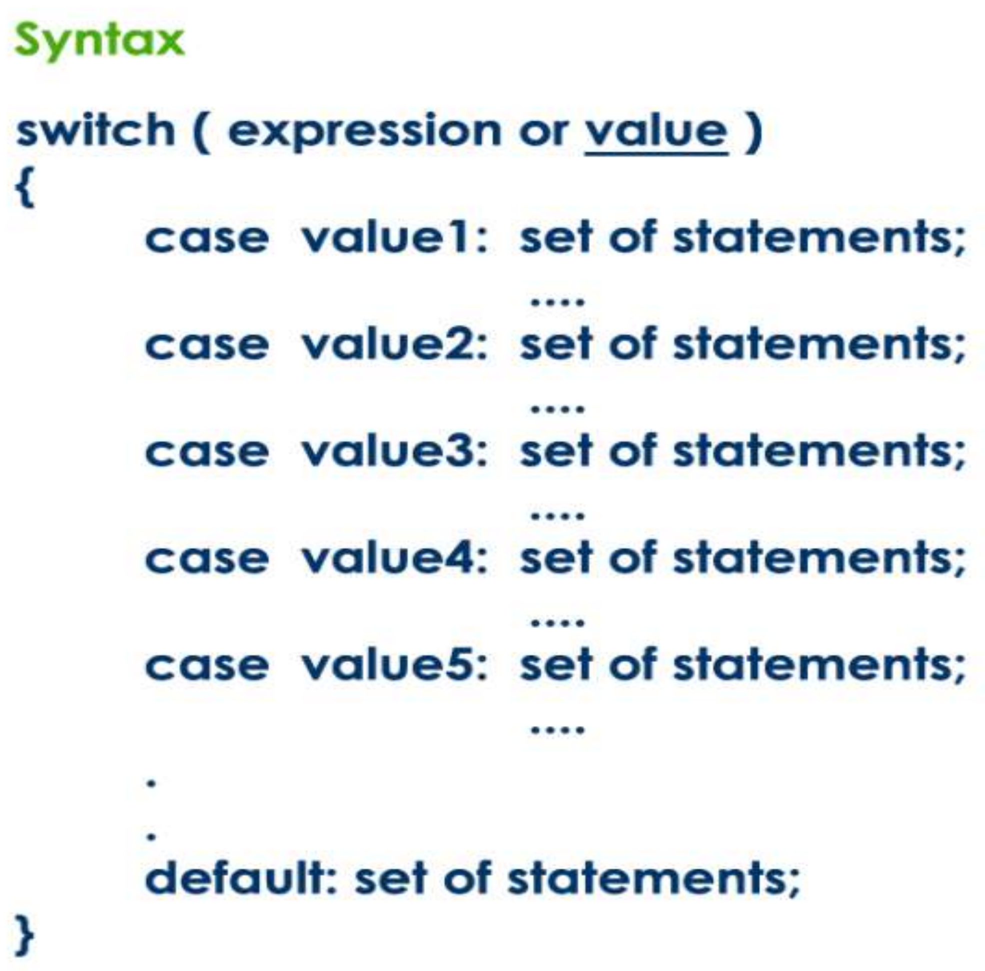
The switch statement allows us to execute one code block among many alternatives.

You can do the same thing with the `if...else..if` ladder. However, the syntax of the `switch` statement is much easier to read and write.



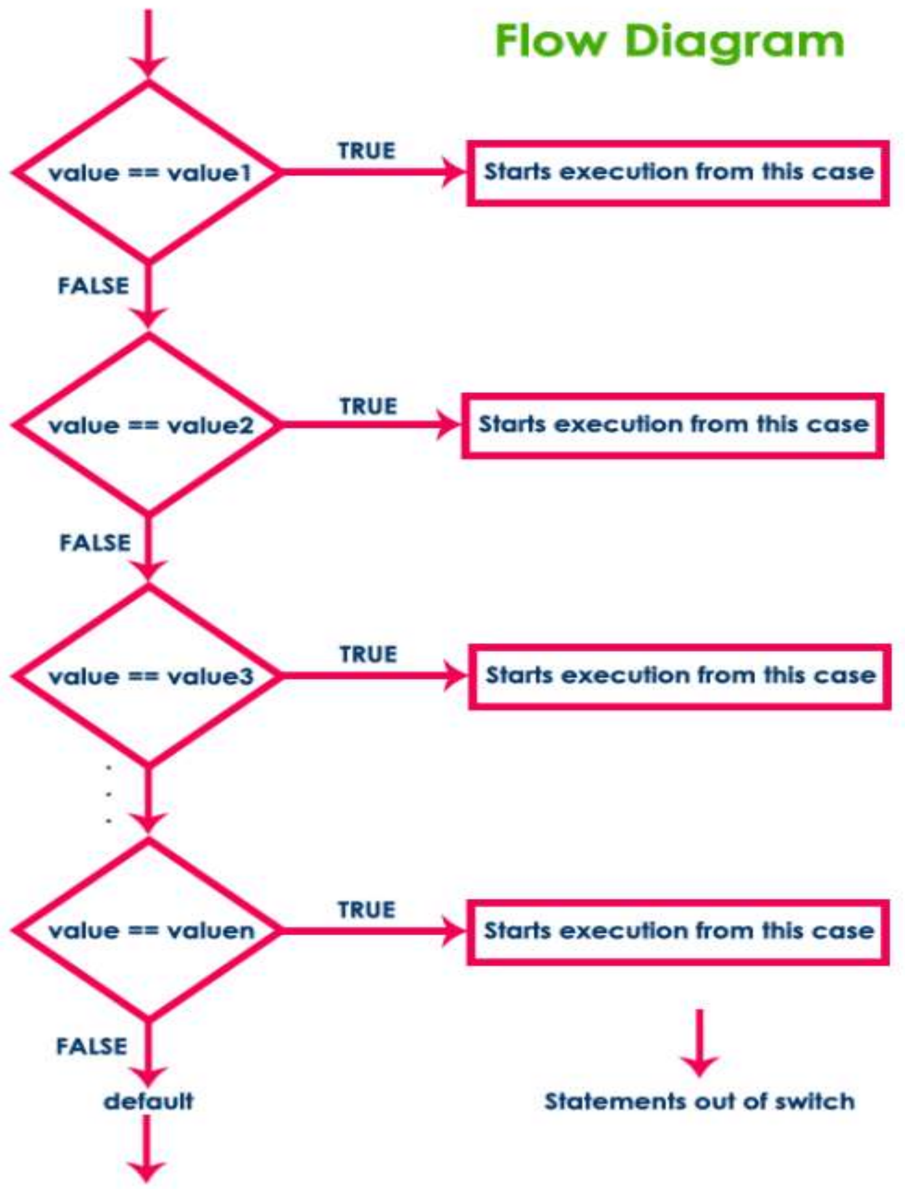
C switch statement is used when you have multiple possibilities for the if statement. Switch case will allow you to choose from multiple options. When we compare it to a general electric switchboard, you will have many switches in the switchboard but you will only select the required switch, similarly, the switch case allows you to set the necessary statements for the user.

'switch' statement in C

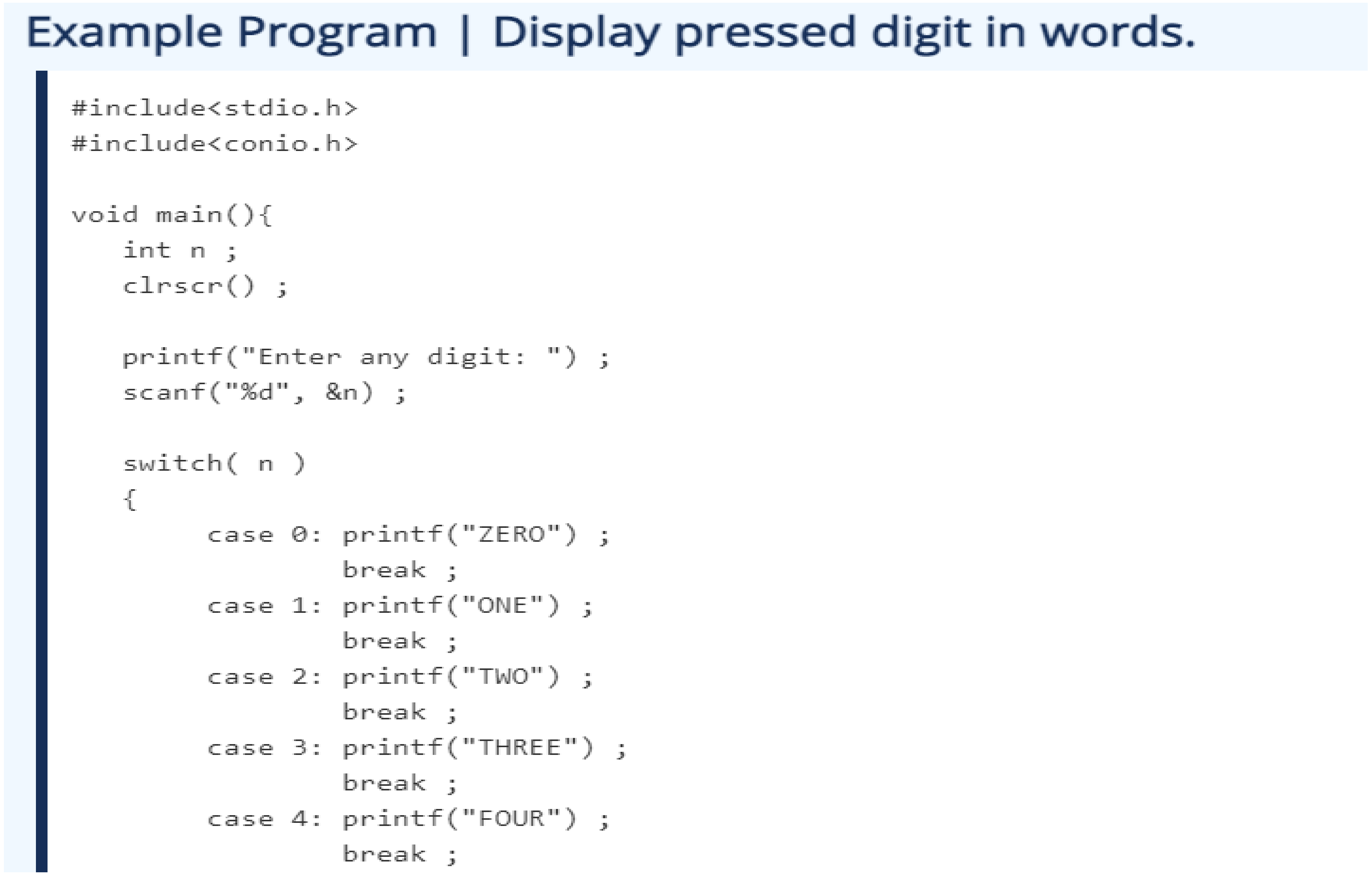


Syntax

```
switch ( expression or value )
{
        case  value1:  set of statements;
                       ....
        case  value2:  set of statements;
                       ....
        case  value3:  set of statements;
                       ....
        case  value4:  set of statements;
                       ....
        case  value5:  set of statements;
                       ....
        .
        .
        default: set of statements;
}
```



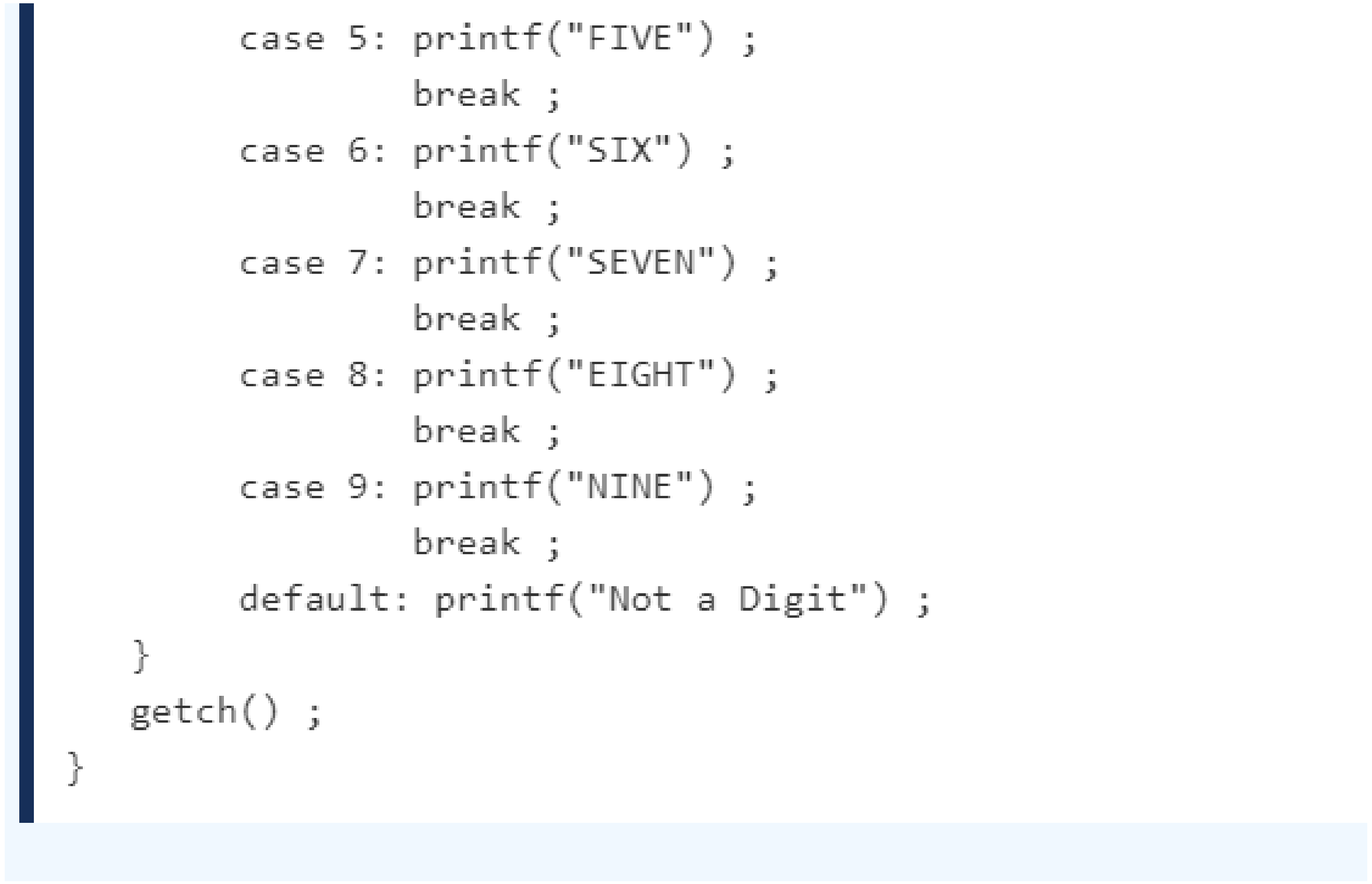
'switch' statement in C



Example Program | Display pressed digit in words.

```
#include<stdio.h>
#include<conio.h>

void main(){
   int n ;
   clrscr() ;

   printf("Enter any digit: ") ;
   scanf("%d", &n) ;

   switch( n )
   {
        case 0: printf("ZERO") ;
                break ;
        case 1: printf("ONE") ;
                break ;
        case 2: printf("TWO") ;
                break ;
        case 3: printf("THREE") ;
                break ;
        case 4: printf("FOUR") ;
                break ;
```

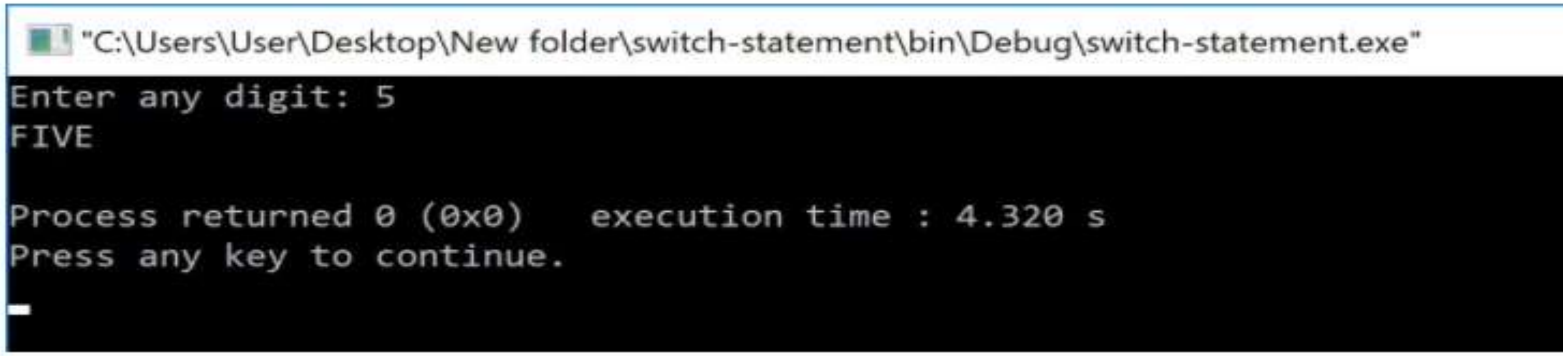
'switch' statement in C



```
        case 5: printf("FIVE") ;
                break ;
        case 6: printf("SIX") ;
                break ;
        case 7: printf("SEVEN") ;
                break ;
        case 8: printf("EIGHT") ;
                break ;
        case 9: printf("NINE") ;
                break ;
        default: printf("Not a Digit") ;
    }
    getch() ;
}
```

'switch' statement in C

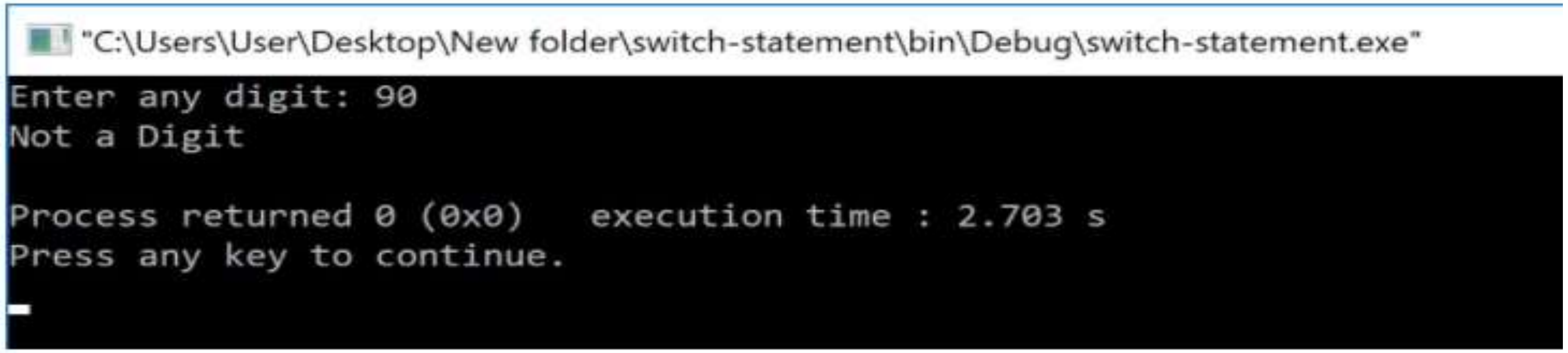
Output 1:



"C:\Users\User\Desktop\New folder\switch-statement\bin\Debug\switch-statement.exe"

```
Enter any digit: 5
FIVE

Process returned 0 (0x0)   execution time : 4.320 s
Press any key to continue.
```

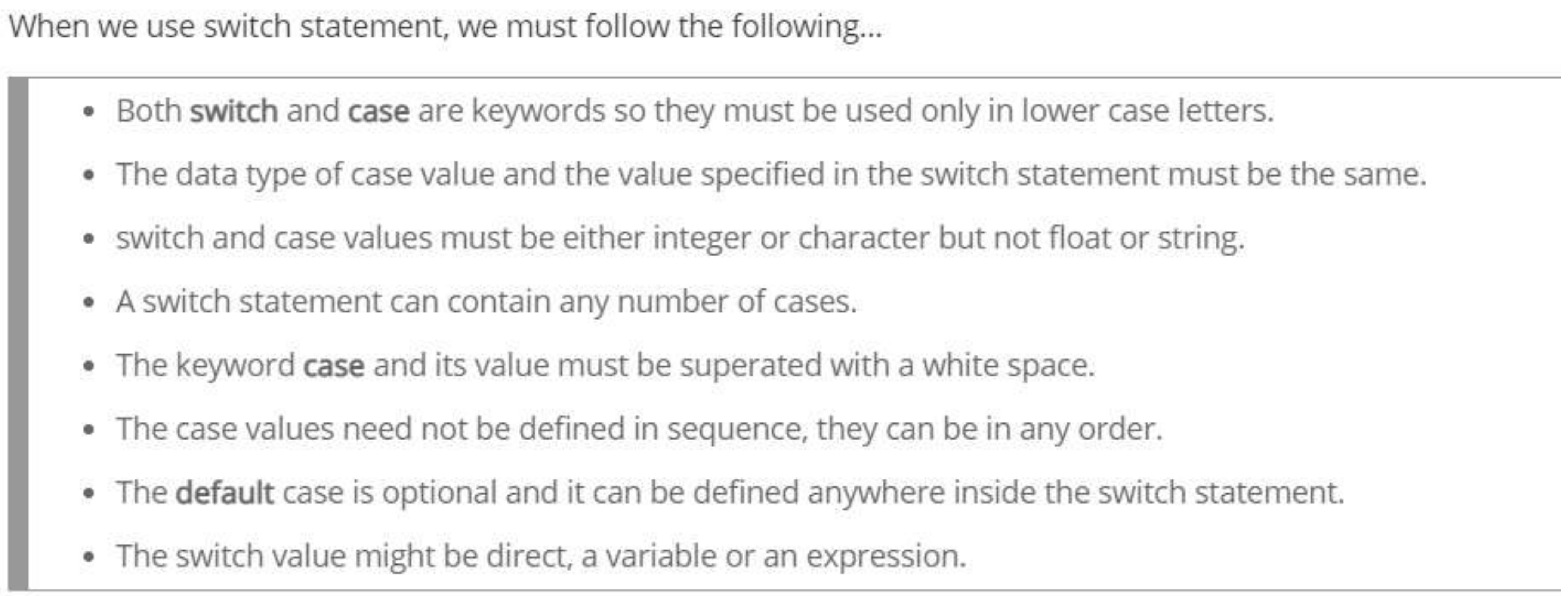
Output 2:



"C:\Users\User\Desktop\New folder\switch-statement\bin\Debug\switch-statement.exe"

```
Enter any digit: 90
Not a Digit

Process returned 0 (0x0)   execution time : 2.703 s
Press any key to continue.
```

'switch' statement in C



When we use switch statement, we must follow the following...

- Both **switch** and **case** are keywords so they must be used only in lower case letters.
- The data type of case value and the value specified in the switch statement must be the same.
- switch and case values must be either integer or character but not float or string.
- A switch statement can contain any number of cases.
- The keyword **case** and its value must be superated with a white space.
- The case values need not be defined in sequence, they can be in any order.
- The **default** case is optional and it can be defined anywhere inside the switch statement.
- The switch value might be direct, a variable or an expression.

'goto' statement in C

C supports a unique form of a statement that is the `goto` Statement which is used to branch unconditionally within a program from one point to another. Although it is not a good habit to use the goto statement in C, there may be some situations where the use of the goto statement might be desirable.

The goto statement is used by programmers to change the sequence of execution of a C program by shifting the control to a different part of the same program.

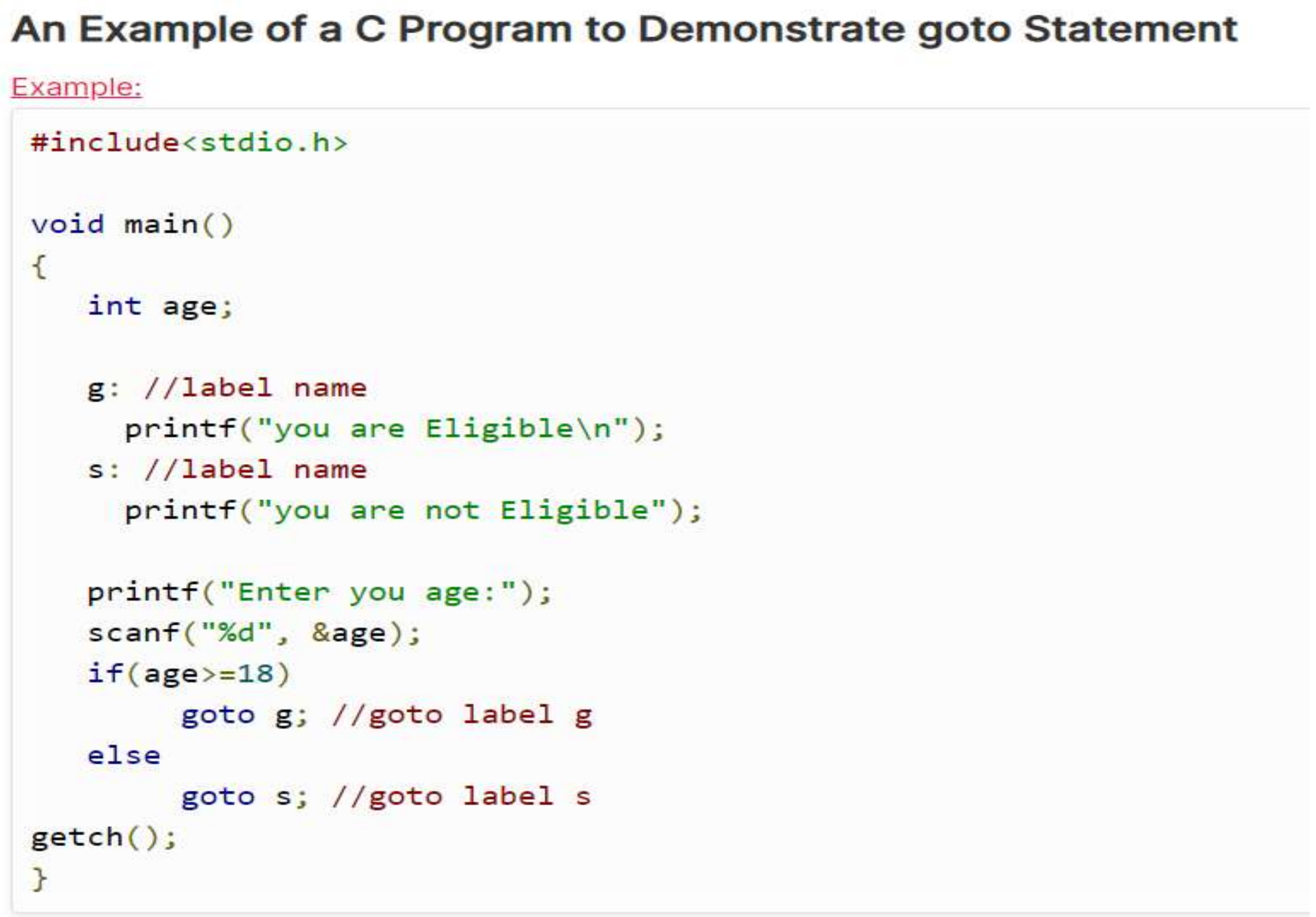
Syntax:

```
goto label;

 - - -- -   -
 - - - - - - - -

label:

statement - X;
/* This the forward jump of goto statement */
```

or

```
label:

 - - -- -   -
 - - - - - - - -

goto label;

/*This is the backward jump of goto statement */
```

'goto' statement in C



An Example of a C Program to Demonstrate goto Statement

Example:

```
#include<stdio.h>

void main()
{
   int age;

   g: //label name
     printf("you are Eligible\n");
   s: //label name
     printf("you are not Eligible");

   printf("Enter you age:");
   scanf("%d", &age);
   if(age>=18)
        goto g; //goto label g
   else
        goto s; //goto label s
getch();
}
```

Thank You!