CONTROL STATEMENTS IN C

by
Dr. Sumit Srivastava
Dept. of Computer Science & Engineering

What do they do?

- Allow different sets of instructions to be executed depending on the outcome of a logical test.
  - Whether TRUE or FALSE.
  - This is called *branching*.
- Some applications may also require that a set of instructions be executed repeatedly, possibly again based on some condition.
  - This is called *looping*.

How do we specify the conditions?

- Using relational operators.

– Four relation operators:	<, <=, >, >=
– Two equality operators:	==, !=

- Using logical operators / connectives.

– Two logical connectives:	&&, ||
– Unary negation operator:	!

The conditions evaluate to …

- Zero
  - Indicates *FALSE*.
- Non-zero
  - Indicates *TRUE*.
  - Typically, the condition TRUE is represented by the value ‘1’.

Control Statements

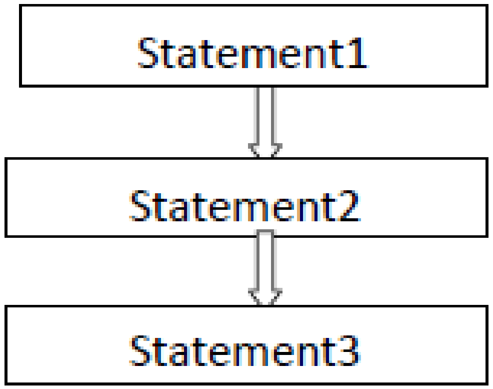
- Control statements determine the “flow of control” in a program.
- It control the flow of execution of the statements in a program.
- It specify the order in which the various instructions in a program are to be executed.
- There are three basic control statements:
    - Sequence
    - Selection/decision
    - Repetition or Loop

Control Structures

- All programs can be written in terms of three control structures (like building blocks)
  - Sequence
    - 'Built-in' to C
      - Unless otherwise directed, one statement after the next is executed
  - Selection (three types)
    - Depending on a ***condition***, *select* between one statement or another
      - If var1 is greater than 10, do *this*…, else do *that*…
      - (if, if/else, switch)
  - Repetition (three types)
    - Depending on a ***condition***, execute one or more statements *repeatedly*
    - *(while, do/while, for)*

Sequence Instruction (Sequential control)

- Executing one instruction after another, in the order in which they occur in the source file.
- This is usually built into the language as a default action.



Conditional Control (Selection Control or Decision Control)

- Executing different sections of code depending on a specific condition or the value of a variable.

- The execution of statements depends upon the condition-test. If the condition evaluates to true, then a set of statements is executed otherwise another set of statements is followed.

- This control is also called Decision Control because it helps in making decision about which set of statements is to be executed.

Conditional Control (Selection Control or Decision Control)

- Decision control structure in C can be implemented by using:-

  1. If statement
  2. If-else statement
  3. Nested if else statement
  4. else-if ladder
  5. case control structure
  6. conditional operator

Iteration Control (Loops)

- Executing the same section of code more than once.
- A section of code may either be executed a fixed number of times, or while some condition is true.
- C provides three looping statements:

  1. While loop
  2. Do-while loop
  3. For loop

SELECTION CONTROL

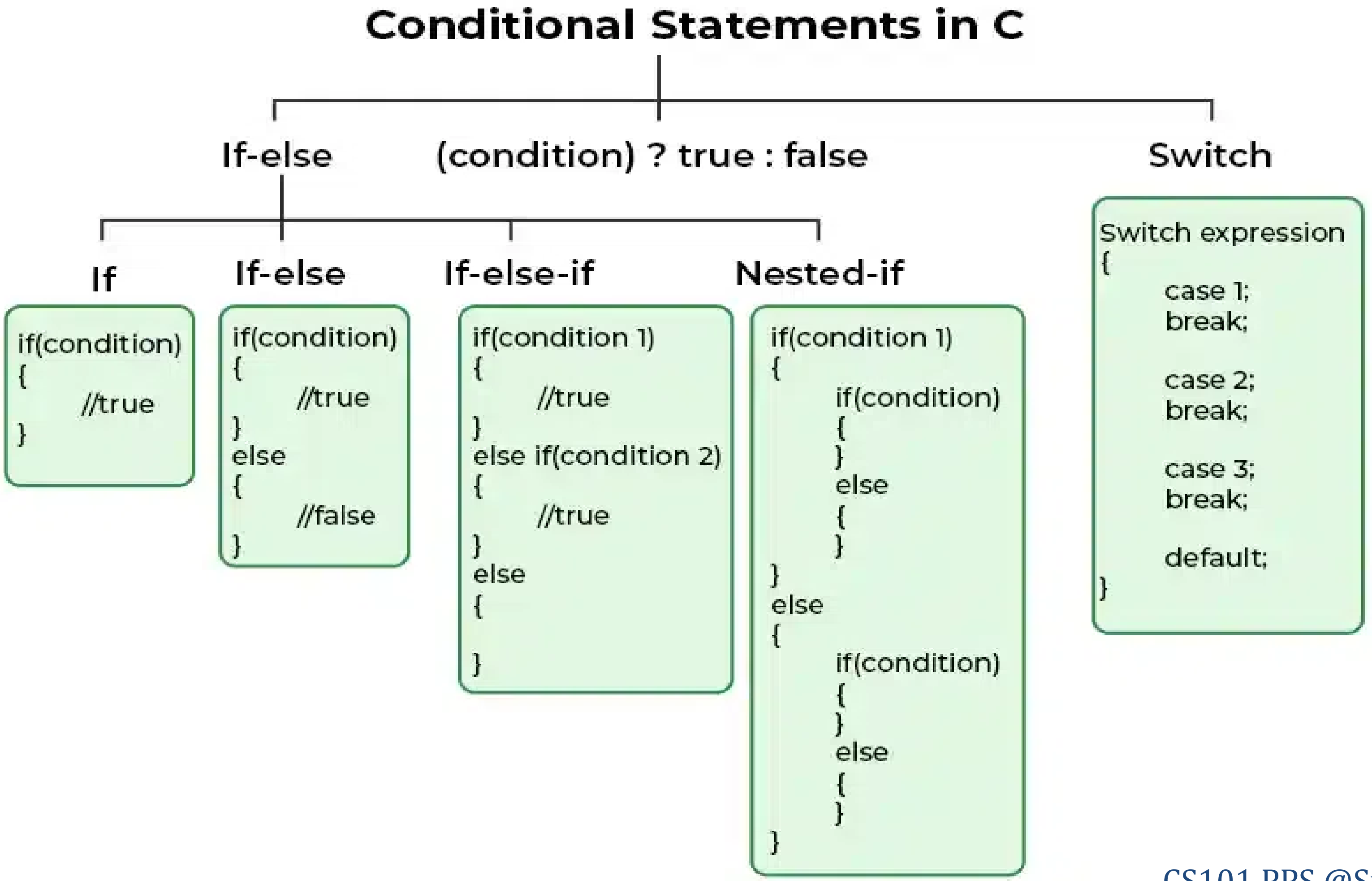
Selection Statements

- One-way decisions using if statement
- Two-way decisions using if-else statement
- Multi-way decisions
- Dangling else Problem

Selection Structure Overview

- Three kinds of selections structures
  - **if** (also called, ‘single-selection’)
    - if ***condition*** is true
      Perform action
    - if ***condition*** is false, action is skipped, program continues
  - **if/else** (also called, ‘double-selection’)
    - if ***condition*** is true
      Perform action
    - else (if ***condition*** is false)
      Perform a *different* action (this will be skipped if condition is true)
  - **switch** (also called ‘multiple-selection’)
    - Allows selection among many actions depending on the integral value of a variable or expression

Conditional Statement



```
if(condition)
{
        //true
}
```

```
if(condition)
{
        //true
}
else
{
        //false
}
```

```
if(condition 1)
{
        //true
}
else if(condition 2)
{
        //true
}
else
{

}
```

```
if(condition 1)
{
        if(condition)
        {
        }
        else
        {
        }
}
else
{
        if(condition)
        {
        }
        else
        {
        }
}
```

```
Switch expression
{
        case 1;
        break;

        case 2;
        break;

        case 3;
        break;

        default;
}
```

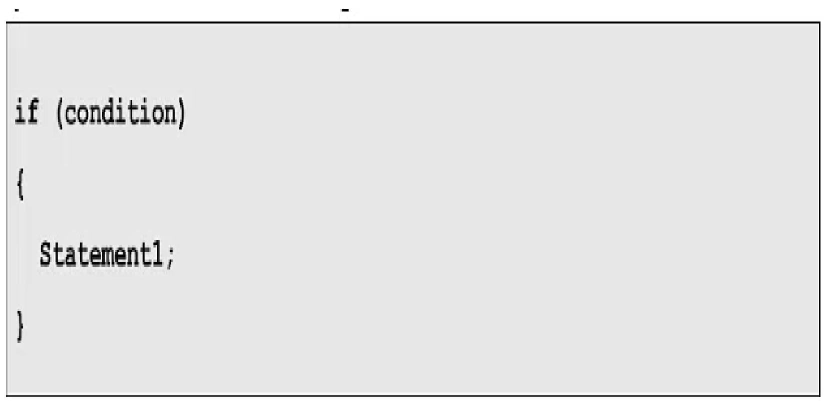
if statement

- If the test expression is true, the statement-block will be executed; otherwise the statement-block will be skipped and the execution will jump to the statement-x.

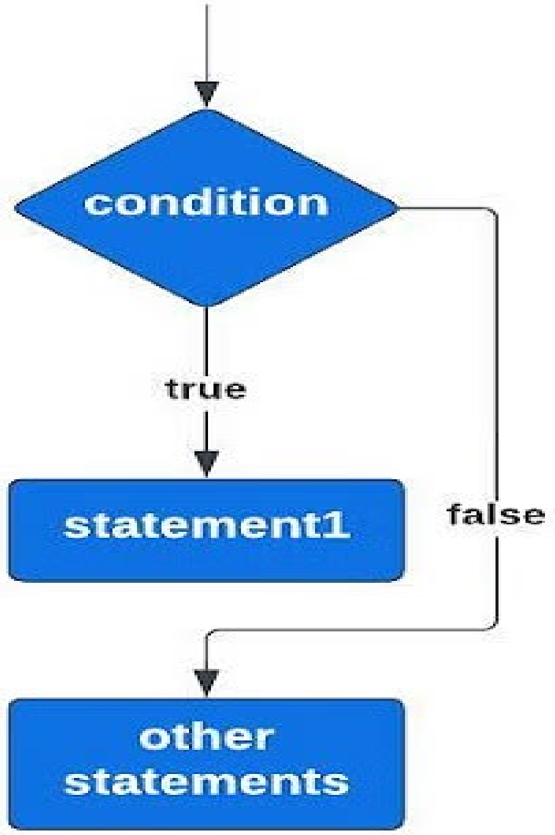
```
if (test expression)
{
        Statement-block;
 }
Statement-x;
```

- *The 'statement-block' may be a single statement or a group of statements.*

if statement



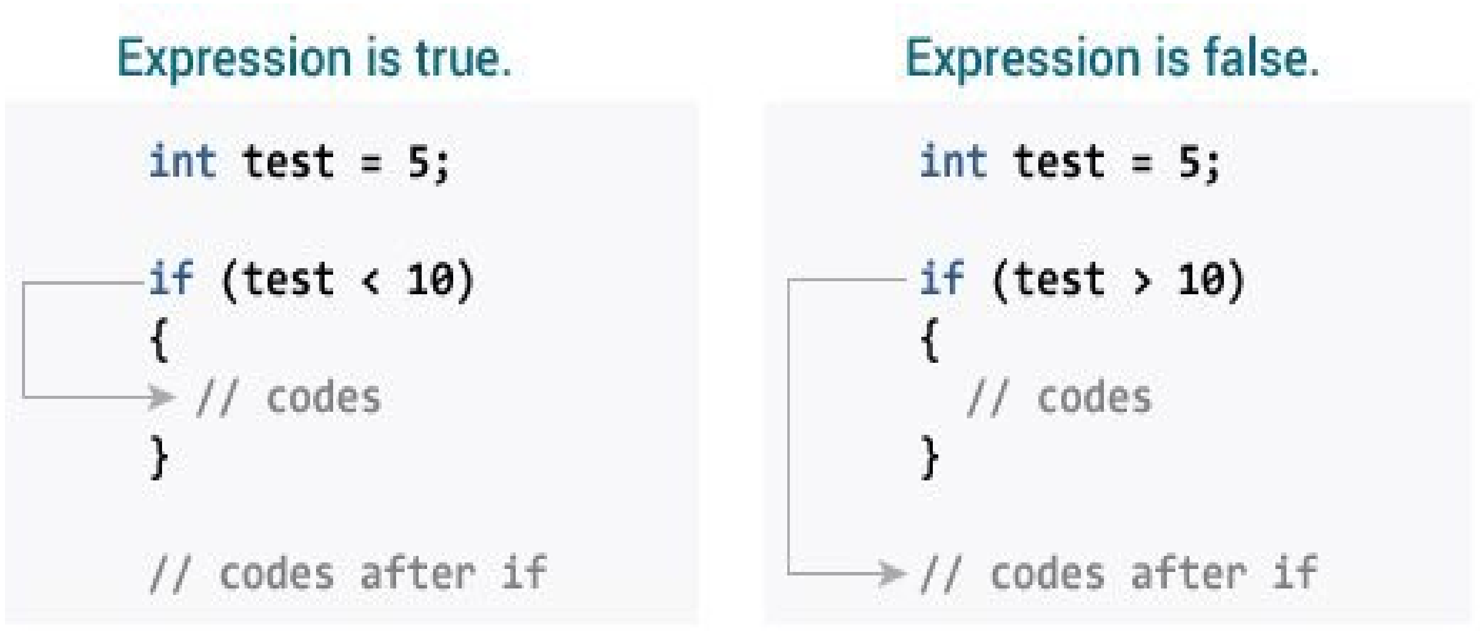
```
if (condition)
{
  Statement1;
}
```



How if statement works?

- The if statement evaluates the test expression inside the parenthesis ().
  - If the test expression is evaluated to true, statements inside the body of if are executed.
  - If the test expression is evaluated to false, statements inside the body of if are not executed.

Working of if Statement



```
int test = 5;

if (test < 10)
{
  // codes
}

// codes after if
```

```
int test = 5;

if (test > 10)
{
  // codes
}

// codes after if
```

if statement: Example

```
 // Program to display a number if it is negative
#include <stdio.h>
int main()
{
     int number;

      printf("Enter an integer: ");
      scanf("%d", &number);

      // true if number is less than 0
      if (number < 0)
      {
            printf("You entered %d.\n", number);
        }

        printf("The if statement is easy.");

       return 0;
   }
```

if statement: Example

- **Output 1:**

```
Enter an integer: -2
You entered -2.
The if statement is easy
```

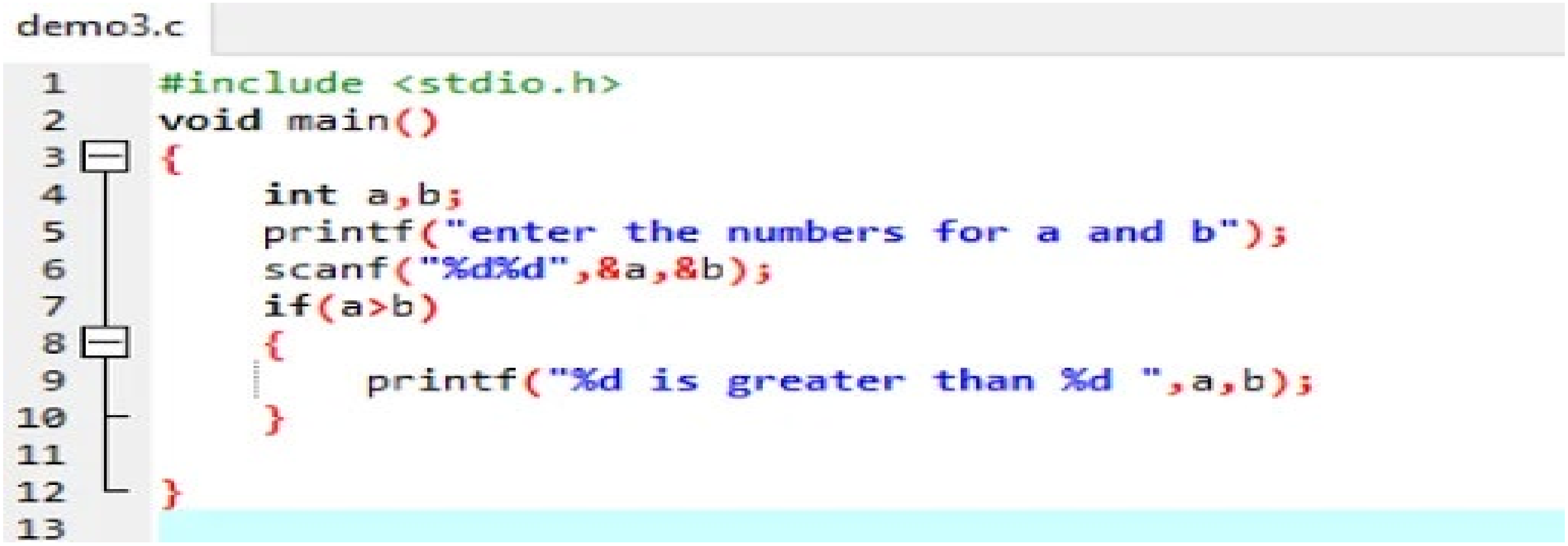
When the user enters -2, the test expression number<0 is evaluated to true. Hence, You entered -2 is displayed on the screen.

- **Output 2:**

```
Enter an integer: 5
The if statement is easy.
```

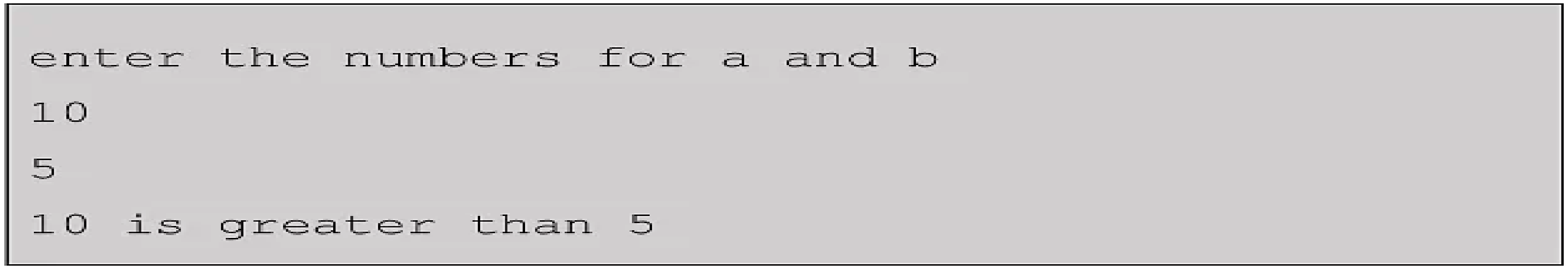
When the user enters 5, the test expression number<0 is evaluated to false and the statement inside the body of if is not executed

if statement: Example



demo3.c

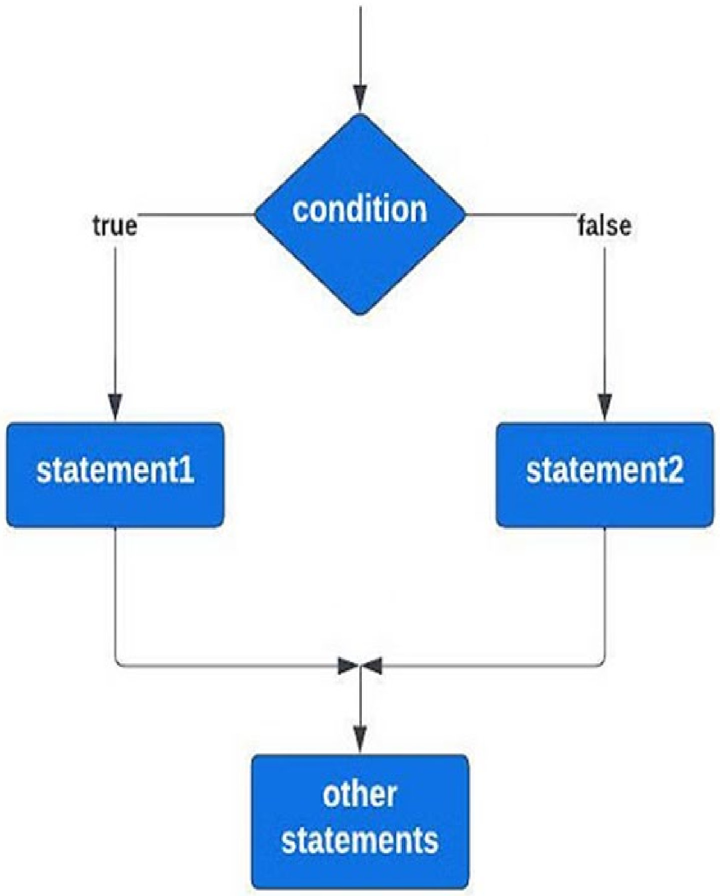
```c
#include <stdio.h>
void main()
{
        int a,b;
        printf("enter the numbers for a and b");
        scanf("%d%d",&a,&b);
        if(a>b)
        {
                printf("%d is greater than %d ",a,b);
        }

}
```



```
enter the numbers for a and b
10
5
10 is greater than 5
```

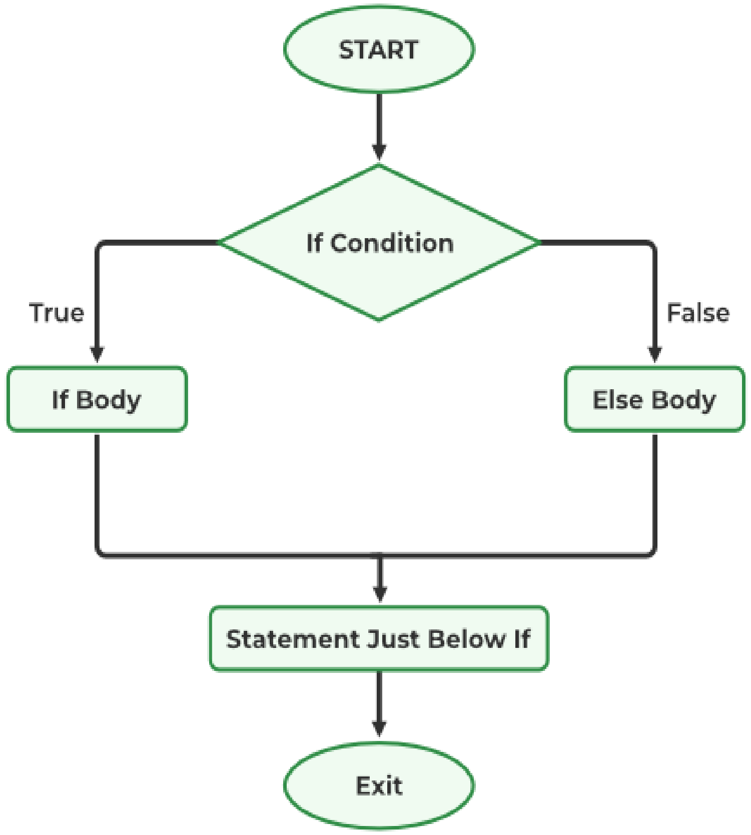
If-else Statement

```
If(condition)
  {
   // Executes this block if condition is true
   Statement 1(s);
  }
else
  {
    // Executes this block if condition is False
    Statement 2(s)
  }
Statement
```



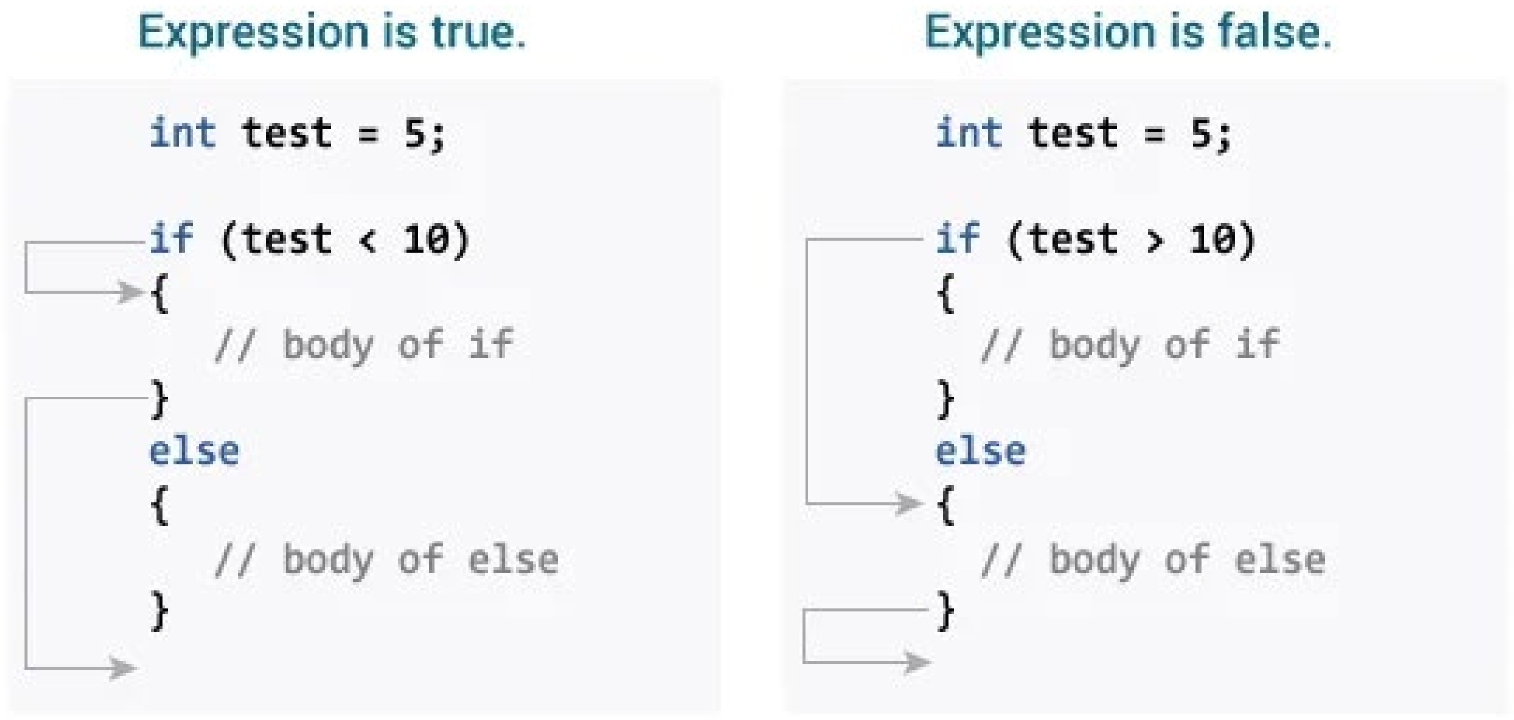
If-else Statement

```
if (Condition)
{
    True block of statements
}
Else
{
    False block of statements
}
```



How if...else statement works?

- If the test expression is evaluated to true,
  - statements inside the body of **if** are executed.
  - statements inside the body of **else** are skipped from execution.
- If the test expression is evaluated to false,
  - statements inside the body of **else** are executed
  - statements inside the body of **if** are skipped from execution.



Expression is true.

```
int test = 5;

if (test < 10)
{
   // body of if
}
else
{
   // body of else
}
```

Expression is false.

```
int test = 5;

if (test > 10)
{
   // body of if
}
else
{
   // body of else
}
```

If-else Statement: Example

```
// C program to illustrate If statement
#include <stdio.h>

int main()
{
    int i = 20;

    if (i < 15) {

        printf("i is smaller than 15");
    }
    else {

        printf("i is greater than 15");
    }
    return 0;
}
```

Output:

```
i is greater than 15
```

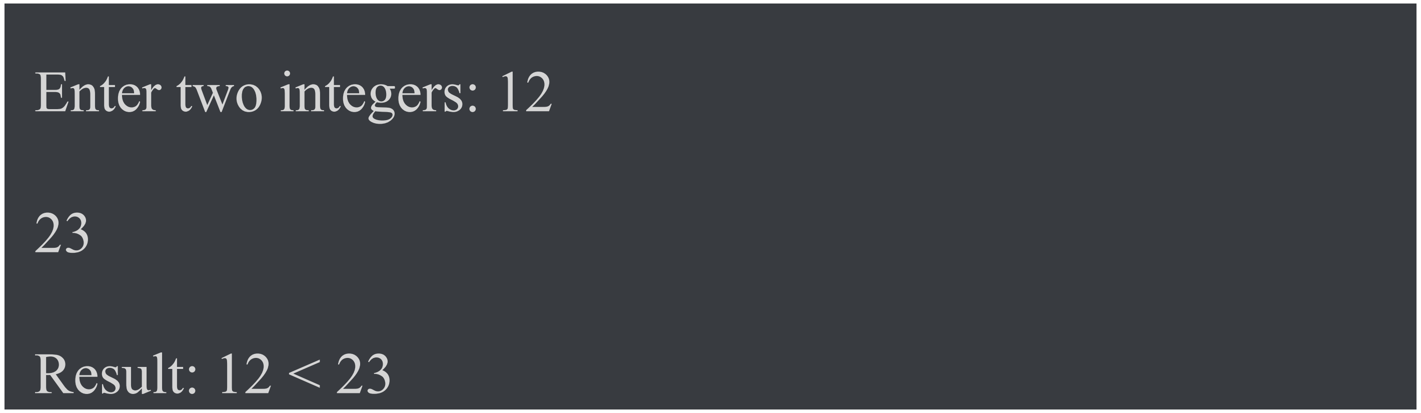
if...else Ladder

```
if (test expression1)
{
// statement(s)
}
else if(test expression2)
{
 // statement(s)
}
else if (test expression3)
{
// statement(s)
}
.
.
else
{
// statement(s)
}
```

```
        // Program to relate two integers using =, > or < symbol
 #include <stdio.h>
 int main() {
      int number1, number2;
      printf("Enter two integers: ");
      scanf("%d %d", &number1, &number2);

      //checks if the two integers are equal.
      if(number1 == number2)
      {
          printf("Result: %d = %d",number1,number2);
       }

      //checks if number1 is greater than number2.
     else if (number1 > number2)
     {
         printf("Result: %d > %d", number1, number2);
     }

     //checks if both test expressions are false
    else
    {
        printf("Result: %d < %d",number1, number2);
   }
return 0; }
```

if...else Ladder: Example

- Output:



```
Enter two integers: 12

23

Result: 12 < 23
```

- This code

```
if (a > b)
{
 printf("Hello");
}
 printf("Hi");
```

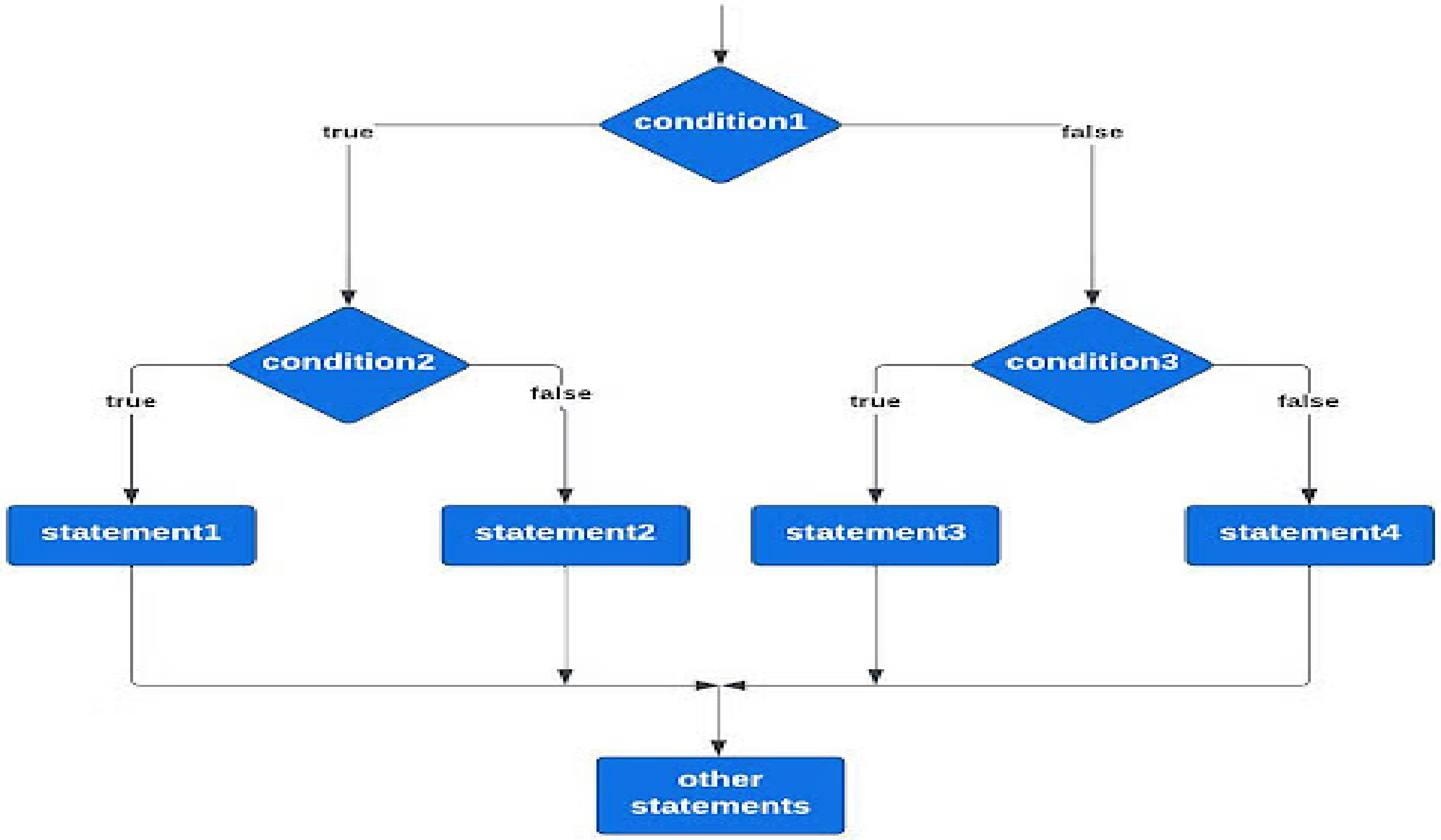
- Is equivalent to

```
if (a > b)
printf("Hello");
printf("Hi");
```

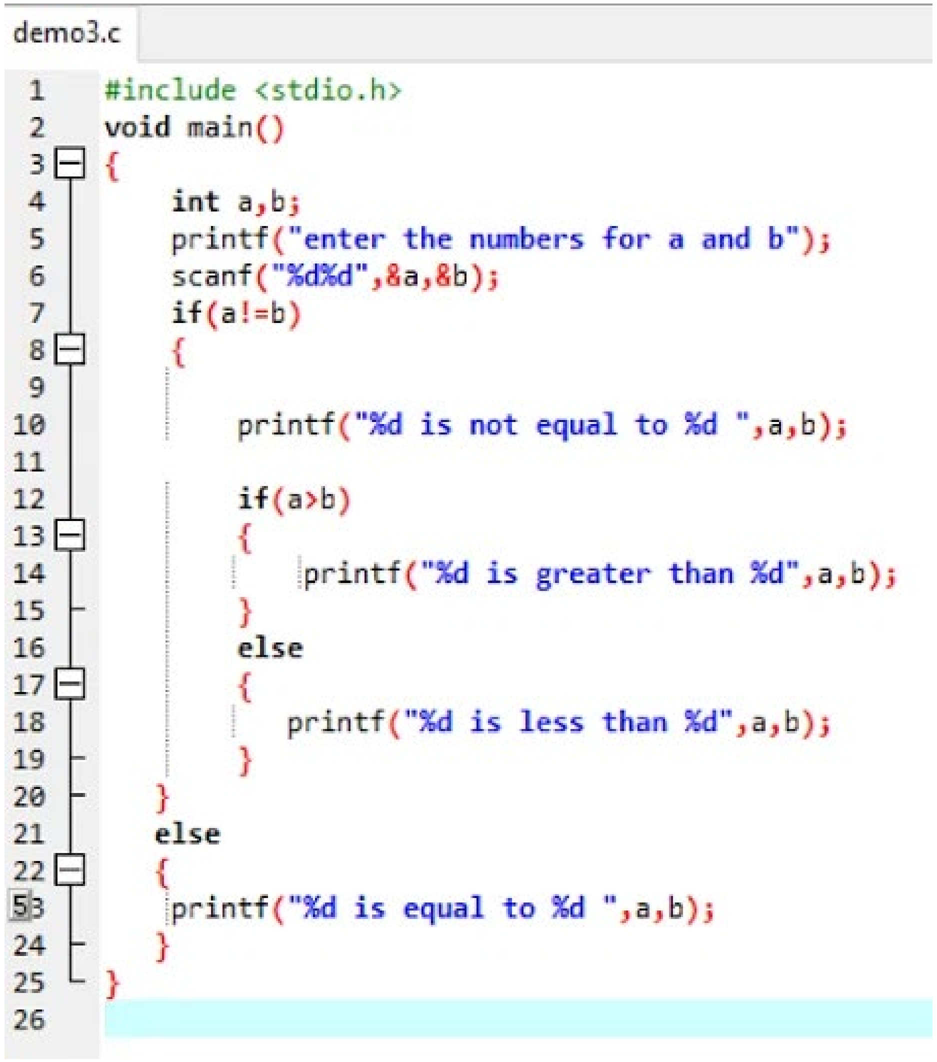
Nested if-else Statements

```
if (condition1)
{
   // Executes when condition1 is true
   if (condition2)
   {
      // Executes when condition2 is true
   }
   else
   {
         // Executes when condition2 is false
}
```

Nested if-else Statements

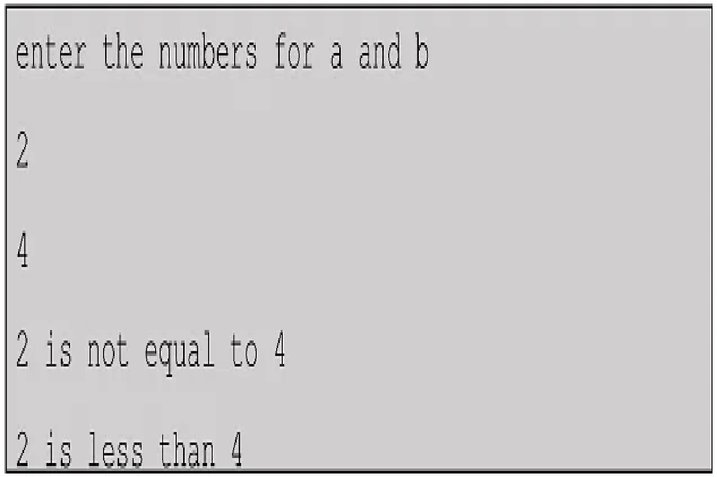


Nested if-else Statements



demo3.c

```c
#include <stdio.h>
void main()
{
    int a,b;
    printf("enter the numbers for a and b");
    scanf("%d%d",&a,&b);
    if(a!=b)
    {

        printf("%d is not equal to %d ",a,b);

        if(a>b)
        {
            printf("%d is greater than %d",a,b);
        }
        else
        {
           printf("%d is less than %d",a,b);
        }
    }
   else
   {
    printf("%d is equal to %d ",a,b);
   }
}
```



```
enter the numbers for a and b

2

4

2 is not equal to 4

2 is less than 4
```

Nested if-else Statements

```
// C program to illustrate nested-if statement
#include <stdio.h>
int main()
{
   int i = 10;

   if (i == 10) {
      // First if statement
      if (i < 15)
         printf("i is smaller than 15\n");

      // Nested - if statement
      // Will only be executed if statement above
      // is true
      if (i < 12)
         printf("i is smaller than 12 too\n");
      else
         printf("i is greater than 15");
   }

   return 0; }
```

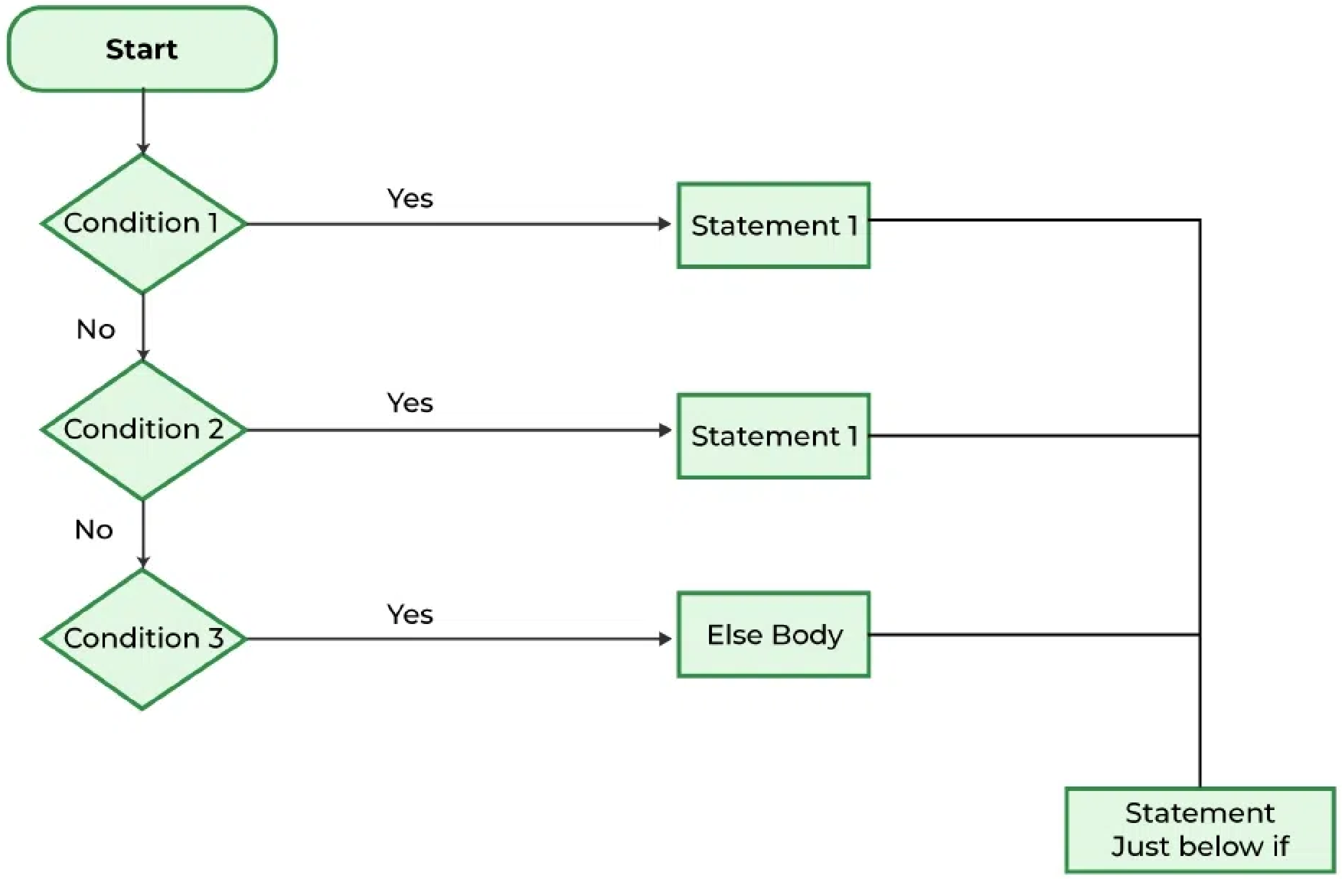
Output

```
i is smaller than 15
i is smaller than 12 too
```

if-else-if Ladder

```
if (condition)
    statement;
else if (condition)
    statement;
.
.

else
    statement;
```

if-else-if Ladder



if-else-if Ladder: Example

```
// C program to illustrate nested-if statement
#include <stdio.h>

int main()
{
    int i = 20;

    if (i == 10)
        printf("i is 10");
    else if (i == 15)
        printf("i is 15");
    else if (i == 20)
        printf("i is 20");
    else
        printf("i is not present");
}
```

Output:

```
i is 20
```

THANK YOU