**What are branching statements in C?**

**Branching statements** give us code which is optionally executable, depending on the outcome of certain tests or you can say certain cases which we can define.

Branching is deciding what actions to take

Branching is so called because the program chooses to follow one branch or another.

Branching statements are classified into two types

1. Conditional branching
2. Unconditional branching

**Conditional Branching:**

It includes the statements which works on checking the condition of a given expression to execute the code.

They are:

1. If
2. If-else
3. Nested if
4. Switch
5. Else-if ladder

1] **if statement**

This is the most simple form of the branching statements. It takes an expression in parenthesis and a statement or block of statements. if the expression is true then the statement or block of statements gets executed otherwise these statements are skipped.

It is one of the powerful conditional statement. If statement is responsible for modifying the flow of execution of a program. If statement is always used with a condition. The condition is evaluated first before executing any statement inside the body of If.

if (condition)

instruction;

The condition evaluates to either true or false. True is always a non-zero value, and false is a value that contains zero. Instructions can be a single instruction or a code block enclosed by curly braces { }.

Following program illustrates the use of if construct in 'C' programming:

#include<stdio.h>

int main()

{

int num1=1;

int num2=2;

if(num1<num2) //test-condition

{

printf("num1 is smaller than num2");

}

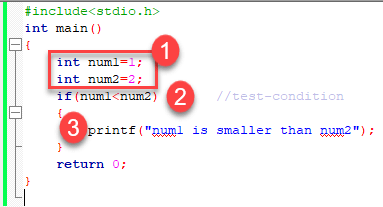
return 0;

}

Output:

num1 is smaller than num2

The above program illustrates the use of if construct to check equality of two numbers.

[](https://www.guru99.com/images/1/020819_0436_CConditiona1.png)

1. In the above program, we have initialized two variables with num1, num2 with value as 1, 2 respectively.
2. Then, we have used if with a test-expression to check which number is the smallest and which number is the largest. We have used a relational expression in if construct. Since the value of num1 is smaller than num2, the condition will evaluate to true.
3. Thus it will print the statement inside the block of If. After that, the control will go outside of the block and program will be terminated with a successful result.

## Relational Operators

C has six relational operators that can be used to formulate a Boolean expression for making a decision and testing conditions, which returns true or false :

< less than

<= less than or equal to

> greater than

>= greater than or equal to

== equal to

!= not equal to

Notice that the equal test (==) is different from the assignment operator (=) because it is one of the most common problems that a programmer faces by mixing them up.

For example:

int x = 41;

x =x+ 1;

if (x == 42) {

printf("You succeed!");}

Output :

You succeed

Keep in mind that a condition that evaluates to a non-zero value is considered as true.

For example:

int present = 1;

if (present)

printf("There is someone present in the classroom \n");

Output :

There is someone present in the classroom

### Example of multiple if statements

We can use multiple if statements to check more than one conditions.

#include <stdio.h>

int main()

{

int x, y;

printf("enter the value of x:");

scanf("%d", &x);

printf("enter the value of y:");

scanf("%d", &y);

if (x>y)

{

printf("x is greater than y\n");

}

if (x<y)

{

printf("x is less than y\n");

}

if (x==y)

{

printf("x is equal to y\n");

}

printf("End of Program");

return 0;

}

In the above example the output depends on the user input.

Output:

enter the value of x:20

enter the value of y:20

x is equal to y

End of Program

1] **if Else statement**

It is used to run a set of statements given some condition is true

Following is the program flow of the If-else statement.

Condition is a boolean statement, it is either true(anything other than 0) or false(0). First the compiler solves the condition given inside the if parenthesis, if it is true, value 1 is returned and 0 otherwise.

1. **int** i = 3<2; //i will have a value 0
2. **int** j = 5>3; //j will have a value 1

The set of conditions given inside the if block will run iff condition returns true (1). The set of conditions given inside the else block will run if condition return false (0).

1. **if**('a'=='g'){
2. printf("if block runs");
3. }
4. **else**{
5. printf("Else block runs");
6. }

Note that the condition (‘a’==’g’) return 0. So compiler skips the instructions in the if part and executes the one listed in the else part.

Now consider this :

1. **if**('a') printf("if runs"); //no need of curly braces if only one statement follows

The above condition is true and the print statement runs because ‘a’ has some value, i.e, NOT ZERO value. So if condition is true. Similarly if you pass 1, -1, 352, array pointer, condition is true.

Here’s a tricky one :

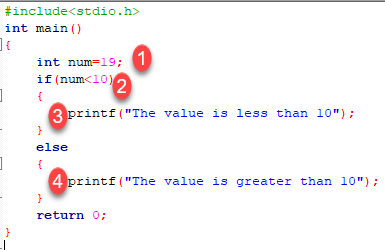
1. **int** \*x;
2. **if**(x)printf("If block runs");

will the print statement run ? NO.

Because we haven’t allocated a memory to this pointer so it holds a NULL value. And NULL value is 0 numerically. So If part won’t run.

1. x = (**int** \*)malloc(1\***sizeof**(**int**));
2. **if**(x)printf("If block runs");

But now it will run because we have allocated some memory address to the x pointer, so it holds some NOT NULL or NOT ZERO value.

[](https://www.guru99.com/images/1/020819_0436_CConditiona2.png)

1. We have initialized a variable with value 19. We have to find out whether the number is bigger or smaller than 10 using a 'C' program. To do this, we have used the if-else construct.
2. Here we have provided a condition num<10 because we have to compare our value with 10.
3. As you can see the first block is always a true block which means, if the value of test-expression is true then the first block which is If, will be executed.
4. The second block is an else block. This block contains the statements which will be executed if the value of the test-expression becomes false. In our program, the value of num is greater than ten hence the test-condition becomes false and else block is executed. Thus, our output will be from an else block which is "The value is greater than 10". After the if-else, the program will terminate with a successful result.

Let's try to understand with this code ,

I have write a code to check whether the number is divisible by 2 or not .

**Coding part :**-

#include<studio.h>

#include<conio.h>

void main()

{

int a;

printf(“\n enter a number”);

scanf(“%d”,a);

If(a%2==o)

{

Printf(“\n %d number is divisible by 2”,a);

}

else

{

printf(“\n %d number is not divisible by 2”, a);

}

getch();

}

In 'C' programming we can use multiple if-else constructs within each other which are referred to as nesting of if-else statements.

## Conditional Expressions

There is another way to express an if-else statement is by introducing the **?:** operator. In a conditional expression the **?:** operator has only one statement associated with the if and the else.

For example:

#include <stdio.h>

int main() {

int y;

int x = 2;

y = (x >= 6) ? 6 : x;/\* This is equivalent to: if (x >= 5) y = 5; else y = x; \*/

printf("y =%d ",y);

return 0;}

Output :

y =2

## Nested If-else Statements

When a series of decision is required, nested if-else is used. Nesting means using one if-else construct within another one.

Let's write a program to illustrate the use of nested if-else.

#include<stdio.h>

int main()

{

int num=1;

if(num<10)

{

if(num==1)

{

printf("The value is:%d\n",num);

}

else

{

printf("The value is greater than 1");

}

}

else

{

printf("The value is greater than 10");

}

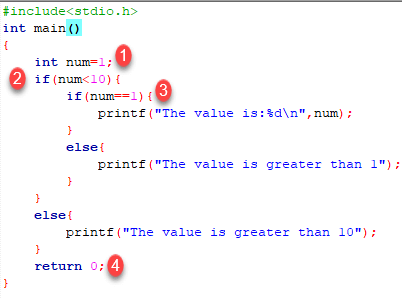
return 0;

}

Output:

The value is:1

The above program checks if a number is less or greater than 10 and prints the result using nested if-else construct.

[](https://www.guru99.com/images/1/020819_0436_CConditiona3.png)

1. Firstly, we have declared a variable num with value as 1. Then we have used if-else construct.
2. In the outer if-else, the condition provided checks if a number is less than 10. If the condition is true then and only then it will execute the inner loop. In this case, the condition is true hence the inner block is processed.
3. In the inner block, we again have a condition that checks if our variable contains the value 1 or not. When a condition is true, then it will process the If block otherwise it will process an else block. In this case, the condition is true hence the If a block is executed and the value is printed on the output screen.
4. The above program will print the value of a variable and exit with success.

Try changing the value of variable see how the program behaves.

**NOTE:** In nested if-else, we have to be careful with the indentation because multiple if-else constructs are involved in this process, so it becomes difficult to figure out individual constructs. Proper indentation makes it easy to read the program.

## Nested Else-if statements

Nested else-if is used when multipath decisions are required.

The general syntax of how else-if ladders are constructed in 'C' programming is as follows:

if (test - expression 1) {

statement1;

} else if (test - expression 2) {

Statement2;

} else if (test - expression 3) {

Statement3;

} else if (test - expression n) {

Statement n;

} else {

default;

}

Statement x;

This type of structure is known as the else-if ladder. This chain generally looks like a ladder hence it is also called as an else-if ladder. The test-expressions are evaluated from top to bottom. Whenever a true test-expression if found, statement associated with it is executed. When all the n test-expressions becomes false, then the default else statement is executed.

Let us see the actual working with the help of a program.

#include<stdio.h>

int main()

{

int marks=83;

if(marks>75){

printf("First class");

}

else if(marks>65){

printf("Second class");

}

else if(marks>55){

printf("Third class");

}

else{

printf("Fourth class");

}

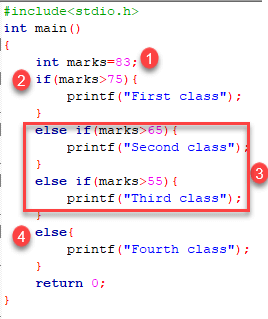
return 0;

}

Output:

First class

The above program prints the grade as per the marks scored in a test. We have used the else-if ladder construct in the above program.

[](https://www.guru99.com/images/1/020819_0436_CConditiona4.png)

1. We have initialized a variable with marks. In the else-if ladder structure, we have provided various conditions.
2. The value from the variable marks will be compared with the first condition since it is true the statement associated with it will be printed on the output screen.
3. If the first test condition turns out false, then it is compared with the second condition.
4. This process will go on until the all expression is evaluated otherwise control will go out of the else-if ladder, and default statement will be printed.

Try modifying the value and notice the change in the output.

**Summary**

* Decision making or branching statements are used to select one path based on the result of the evaluated expression.
* It is also called as control statements because it controls the flow of execution of a program.
* 'C' provides if, if-else constructs for decision-making statements.
* We can also nest if-else within one another when multiple paths have to be tested.
* The else-if ladder is used when we have to check various ways based upon the result of the expression.

**Unconditional Branching:**

These statements doesn't check any condition to execute the code. These statements are pretty much useful to transfer the control from one block to another block over the program.

They are:

1. Goto
2. Return
3. Continue
4. Break