Decision Making and Branching

By SAHIL



INTRODUCTION

1. We note that in aprogram the statements are executed sequentially in the order they appear.
2. Sometimes it is required to divert the execution based on result of certain condition.
3. The diversion of execution of a program on the basis of results of a certain condition is called branching.
4. The branching is achieved with the help of decision making statements.

DECISION MAKING STATEMENTS

C supports the following decision making statements.

1. if statement
2. switch statement
3. Conditional operator statement
4. goto statement

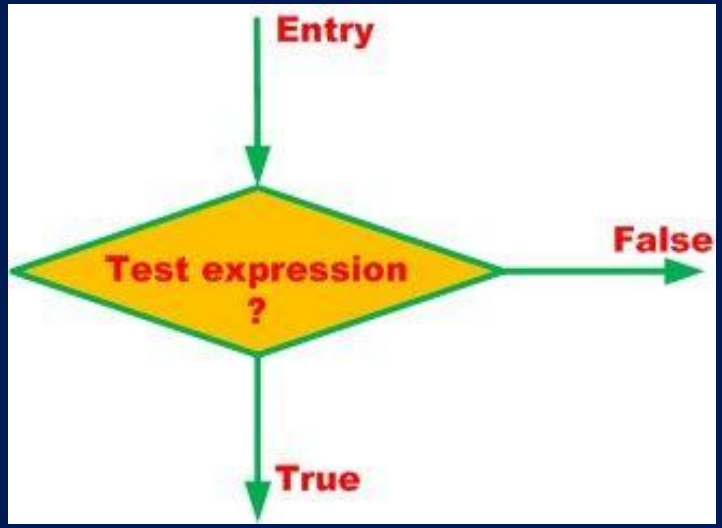
If statements

1. It is a two way decision making statement and is used to control the flow of execution of statements.
2. It is used in conjunction with an expression.

3. It takes the general form as:

```
if (test expression)
```

4. The associated expression evaluated first and then, depending upon the value of expression 'true' (or non-zero) or 'false' (zero) the control is transferred to a particular statement.



5. Following are the different forms of if statement which are used depending upon the complexity of the condition to be tested.

a. Simple if statement
b. if...else statement
c. Nested if...else statement
d. else if ladder

Simple if Statement

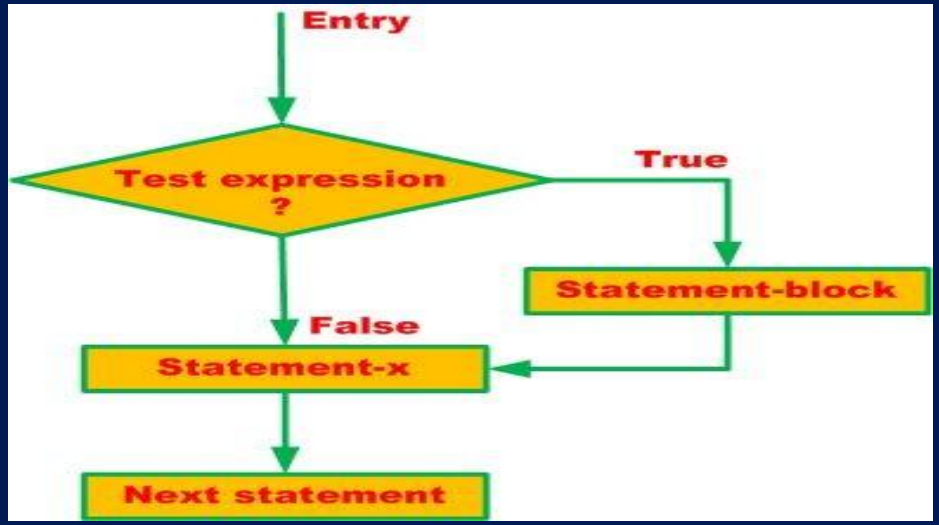
1. It takes the general form as:

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

2. The statement block may have a single statement or group of statements.

Note: If the statement block contains only one statement then pair of braces is optional.

3. If the test condition is true, the control will enter into the body of the if statement and statement-block inside the body is executed and after that execution comes to the statement-x.
4. Where as if the condition is false the body of if statement is by-passed and the execution will jump to the statement-x.

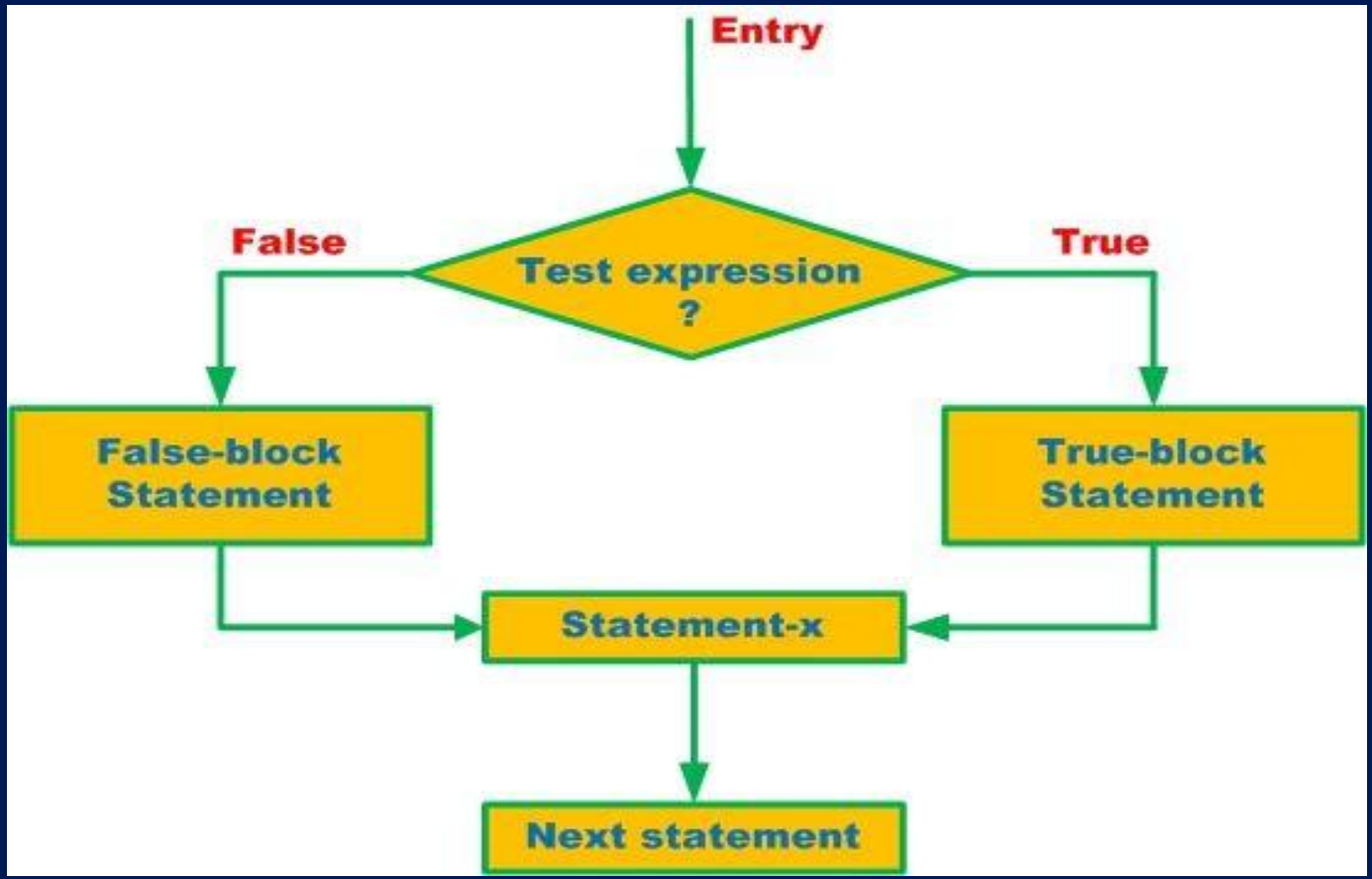


```
……..
……..
if (category == SPORTS )
{
    marks     =   marks + bonus_marks;
}
printf("%d", marks);
……..
……..
```

The if else Statement

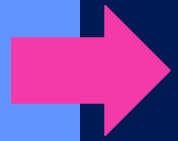
1. It is the extension of simple if statement.
2. It takes the general form as:

```
if (test expression )
{
    True-block statements;
}
else
{
    False-block statements;
}
statement-x;
```


Entry
False
Test expression
?
True
False-block
Statement
True-block
Statement
Statement-x
Next statement

Example: Consider the following example of counting number boys and girls in a class

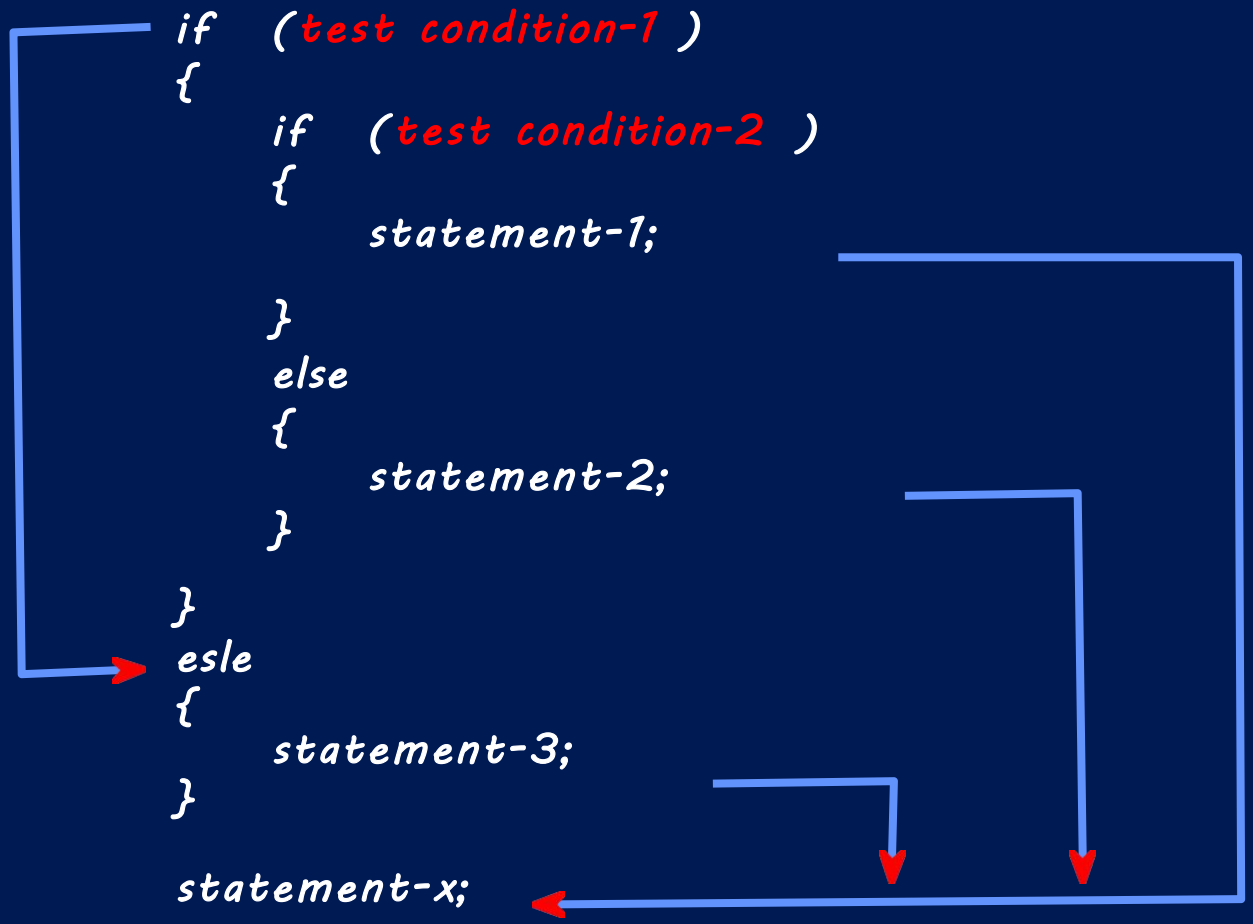
```
.....
.....
if (code == 1 )
    boy = boy +1;
if ( code == 2)
    girl = girl +1;
xxxxxxxxxx
.....
.....
```



```
.....
.....
if (code == 1 )
    boy = boy +1;
else
    girl = girl +1;
xxxxxxxxxx
.....
.....
```

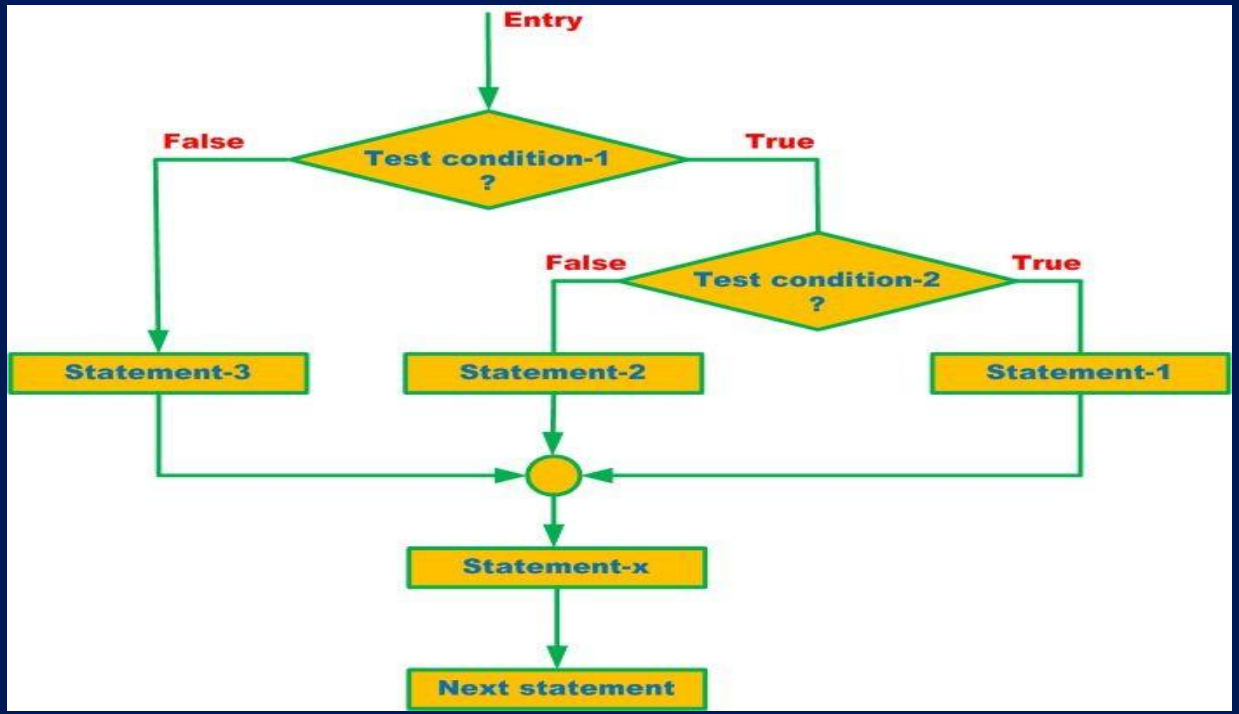
Nested if else Statement

1. A simple if...else statement is not sufficient when a series of decisions are involved.
2. In such cases more than one if...else statements are used in nested form.
3. It takes the general form as:


if (test condition-1)
{
if (test condition-2)
{
statement-1;
}
else
{
statement-2;
}
}
esle
{
statement-3;
}
statement-x;

Note:

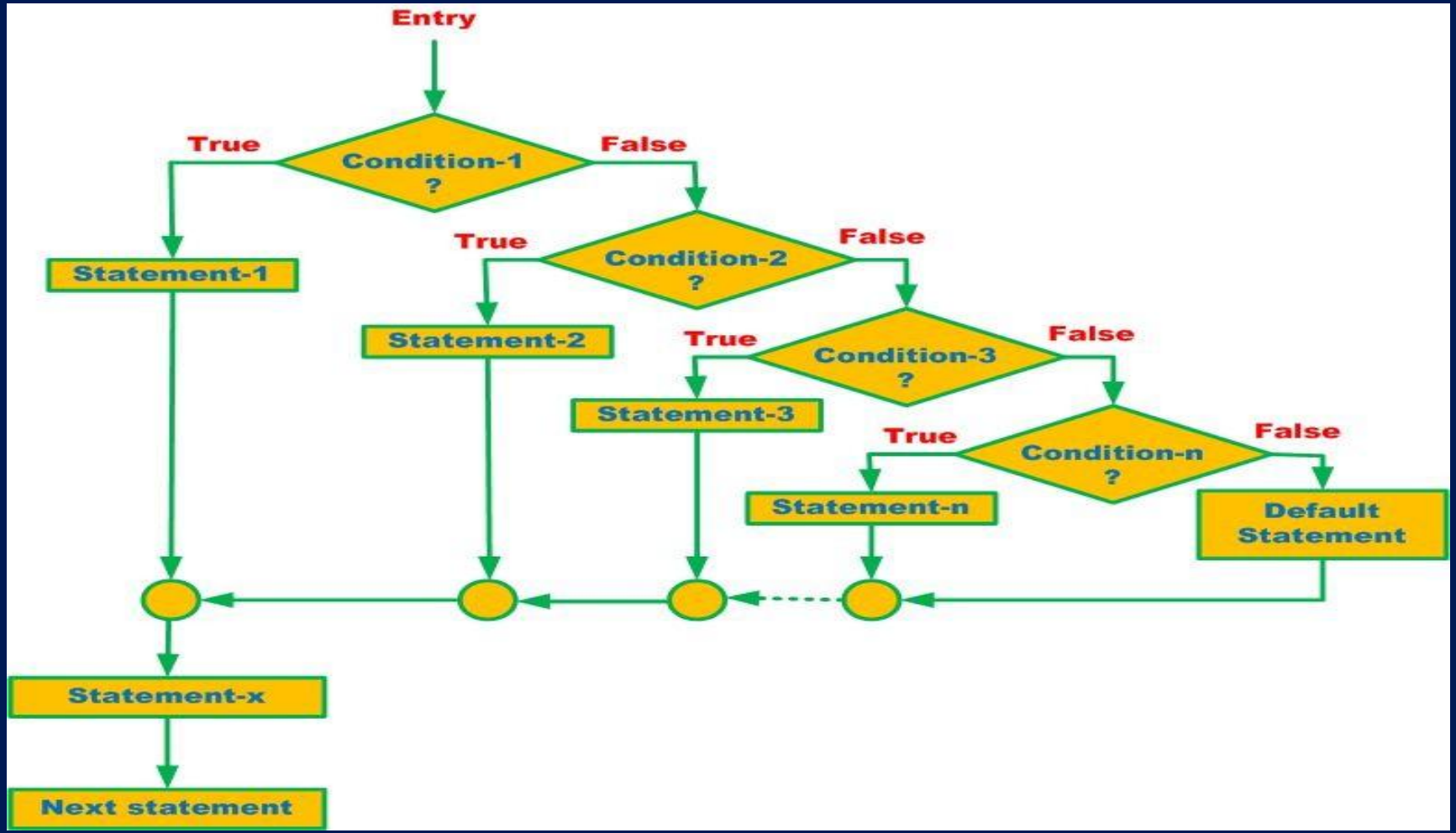
1. In some cases when false condition is not required, else statement may be omitted.
2. else is always paired with most recent unpaired if.



else if Ladder

1. When multipath decisions are involved, the else if ladder is used.
2. The else if ladder uses a chain of if 's in which the statement associated with each else is an if.
3. It takes the general form as:

```
if (test-condition1){
    statement -1
    }
else if(test-condition2){
    statement -2
    }
else if(test-condition3){
    statement -3
    }
else if(test-condition4){
    statement -4
    }
else {
    statement N
    }
```


Entry
Condition-1 ?
True
False
Statement-1
Condition-2 ?
True
False
Statement-2
Condition-3 ?
True
False
Statement-3
Condition-n ?
True
False
Statement-n
Default Statement
Statement-x
Next statement

Switch Statement

1. Multiway decision making using if statement makes a program complex..
2. As a solution to this C has a built-in multiway decision statement known as switch.
3. The variable (or the expression) associated with switch statement is tested against a list of case values.
4. When a match is found, a block of statements associated with case is executed.
5. It takes the general form as:

```
switch (expression )
{
    case value-1:
                    block-1
                    break;
    case value-2:
                    block-2
                    break;
    ..................
    ..................
    default:
                    default-block
                     break;
}
statement-x;
```

6. The expression is and integer expression or character.
7. value1, value-2... are the constants and are known as case labels.
8. Each of the labels must be unique within a switch statement.
9. block-1, block-2...··are the set of statements and may contain zero or more statements.
10. The case label end with a colon (:).
11. The break statement at the end of each block mark the end of a particular case and cause exit from switch statement and transfers the control to the statement following the switch.

12. The *default* is an optional case and is executed if *no match* of expression is found with any of the case values.

Conditional Operator

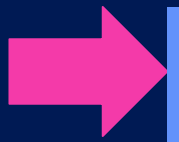
1. It is also a two-way decision making statement.
2. It is known as conditional operator or ternary operator.
3. It takes the general form as:

```
conditional expression ? expression 1 : expression 2 ;
```

4. The conditional expression is evaluated first.
5. If the result is nonzero, expression 1 is evaluated and is returned as the value of conditional expression.
6. Otherwise expression 2 is evaluated and its value is returned as the value of conditional expression.

Example:

```
if (x < 0)
    fiag = 0;
else
    fiag = 1;
```



```
x < 0  ? fiag = 0 : fiag = 1;
```

```
(age >= 18)  ? printf("Can Vote") : printf("Cannot Vote");
```

Thank You

By SAHIL

