WEEK – 4	
1	Control Statements (Conditional): If and its Variants
2	Switch (Break)
3	Sample C Programs

Control Statements (Conditional – Decision Making)

We have a number of situations where we may have to change the order of execution of statements based on certain conditions, or repeat a group of statements until certain specified conditions are met. This involves a kind of decision making to see whether a particular condition has occurred or not and then direct the computer to execute certain statements accordingly.

C language possesses such decision making capabilities and supports the following statements known as control or decision making statements.

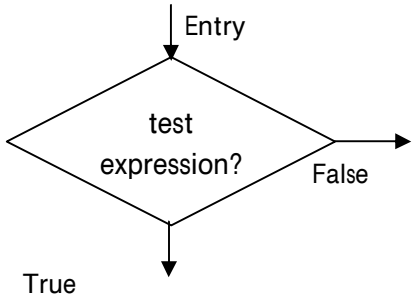
1. **if** statement
2. **switch** statement
3. **conditional operator** statement
4. **goto** statement

Decision making with 'if' statement

The **if** statement is a powerful decision making statement and is used to control the flow of execution of statements. It is basically a two-way decision statement and is used in conjunction with an expression. It takes the following form:

$$if\,(test\ expression)$$

It allows the computer to evaluate the expression first and then depending on whether the value of expression (or condition) is true (1) or false (0), it transfers the control to a particular statement. This point of program has two paths to follow, one for the true condition and the other for the false condition.



Two-way Branching

Examples of decision making, using if statement are

1. if (bank balance is zero) borrow money
2. if(age is more than 60) person retires

The **if** statement may be implemented in different forms depending on the complexity of conditions to be tested.

1. Simple **if** statement
2. **if…else** statement
3. Nested **if…else** statement
4. **else if** ladder

Simple 'if' statement

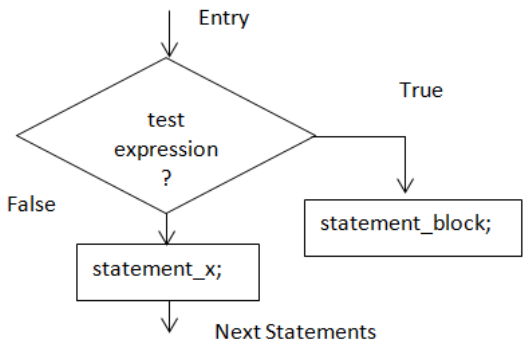
The general form of a 'simple if' statement is

Syntax:

```
if(test_expression)
{
    statement_block;
}
statement_x;
```

'statement_block' may be a single statement or a group of statements. 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 'statement_x'.

Flowchart for Simple If



Example: To check whether student is passed or failed.

```
/*Program to check whether student is passed or failed*/
#include<stdio.h>
#include<conio.h>
main()
{
    int marks;
    clrscr();
```

```
        printf("Enter student marks: ");
        scanf("%d",&marks);
        if(marks>50)
          printf("Student Passed");
    if(marks<50)
          printf("Student Failed");
    getch();
}
```

Output:

(1) Enter student marks: 55
Student Passed

(2) Enter student marks: 40
Student Failed

If-Else Statement

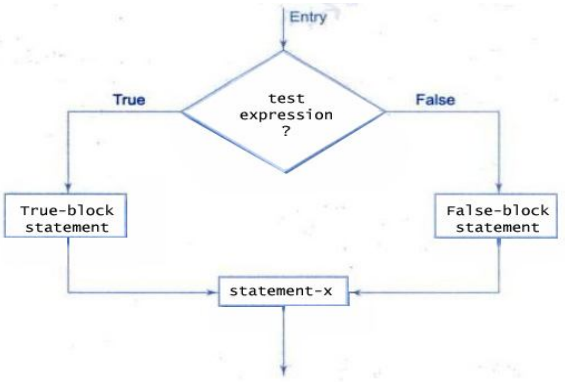
The if-else statement is an extension of the 'simple if' statement. The general form is

Syntax:

```
if(test_expression)
{
      true-block-statements;
}
else
{
      false-block-statements;
}
statement_x;
```

If the test_expression is true, then the true-block-statement(s), immediately following the if statement are executed; otherwise the false-block-statement(s) are executed. In either case, either true-block-statements or false-block-statements will be executed, not both.

Flowchart for If Else



Example: Program to check whether given number is even or odd

```
/*Program to check whether given number is even or odd*/
#include<stdio.h>
#include<conio.h>
main()
{
        int num;
        clrscr();
        printf("Enter number: ");
        scanf("%d",&num);
        if(num%2 == 0)
          printf("%d is even number",num);
    else
          printf("%d is odd number",num);
    getch();
}
```

Output:

(1) Enter number: 53
53 is odd number
(2) Enter student marks: 42
42 is even number

Nested If… Else Statement

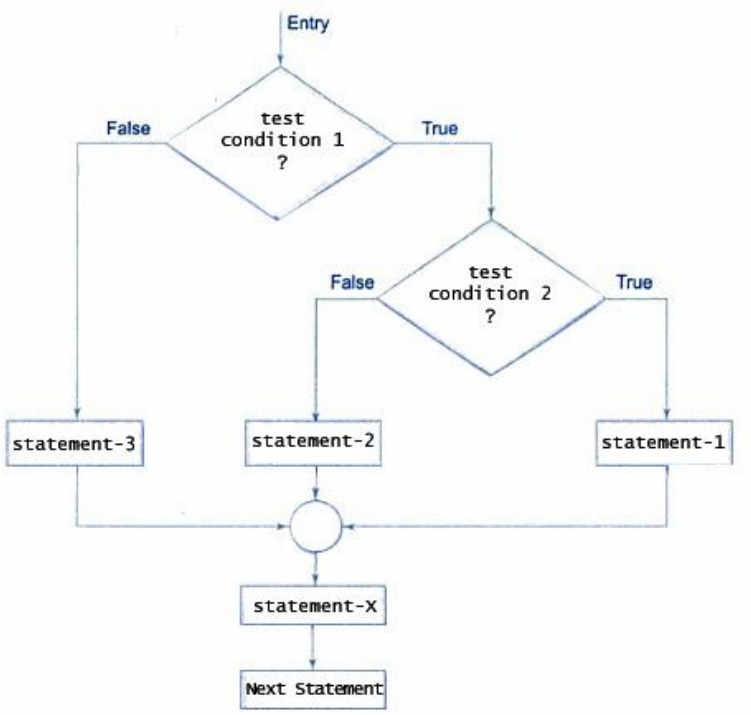
When a series of decisions are involved, we may have to use more than one if…else statement in nested form as follows:

Syntax:

```
if(test_condition1)
{
   if(test_condition2)
   {
      statement-1;
   }
   else
   {
      statement-2;
   }
}
else
{
   statement-3;
}
statement-x;
```

If the test_condition1 is false, the statement-3 will be executed; otherwise it continues to perform the second test. If the test_condition2 is true, the statement-1 will be executed otherwise statement-2 will be evaluated and then the control is transferred to the statement-x;

Flowchart for Nested If...Else



Example: Program to find the largest of three numbers

```
/*Program to find the largest of three numbers*/
#include<stdio.h>
#include<conio.h>
main()
{
        int a,b,c;
        clrscr();
        printf("Enter the three values: ");
        scanf("%d %d %d",&a,&b,&c);
        if(a>b && a>c)
          printf("%d is largest",a);
    else
          if(b>a && b>c)
            printf("%d is largest",b);
          else
            printf("%d is largest",c);
```

```
    getch();
}
```

Output:

```
Enter number: 5 6 7
7 is largest
```

Else If Ladder

There is another way of putting if's together when multipath decisions are involved. A multipath decision is a chain of if's in which the statement associated with each else is an if. It takes the following general form:

Syntax:

```
if(condition1)
        statement-1;
else if(condition-2)
        statement-2;
else if(condition-3)
        statement-3;
...
...
...
else if(condition-n)
        statement-n;
else
        default-statement;
statement-x;
```

This construct is known as else if ladder. The conditions are evaluated from the top downwards. As soon as a true condition is found, the statement associated with it is executed and the control is transferred to statement-x. When all the n conditions become false, then the final else containing the default-statement will be executed.

The logic of execution for 'else if ladder statements' is shown in the flowchart below.