• Chapter 11 : Decision Constructs

11.1 Overview

Q : 11-01-01 : Describe the need and organization of Control Structures ?

Answer :

Control Structures : [Control structures are statements used to control the flow of execution in a program or function. The C control structures enable us to group individual instructions into a single logical unit with one entry point and one exit point]. Program instructions can be organized into three kinds of control structures to control execution flow i.e. sequence, selection and repetition. All programs, whether simple or complex, use these control structures to implement the program logic. Only sequential flow, is also called default flow. In case of sequence structure, instructions are executed in the same order in which they are specified in the program. A compound statement refers to a group of statements enclosed in opening and closing braces :

```
{
        Statement1 ;
        Statement2 ;
        .
        .
        Statementn ;
}
```

Control flows from statement1 to $statement_n$, in a logical sequence. Solutions of some problems require steps with two or more alternative course of action. A selection structure chooses which statement or a block of statements is to execute. In C, there are two basic selection statements :

if – else
switch

There are some variations of if - else structure. The third control structure is repetition, which is also called iteration or loop. It is a control structure, which repeats a statement or a group of statements in a program. C provides different types of loop structures to be used in various situations. These include for loop, while loop, and do-while loop.

11.2 IF Statement

Q : 11-02-01 : Describe the IF Statement ? Explain test condition with an example ?

Answer :

IF Statement : ”if” is one of the keywords in **C** language. It is used to select a path flow in a program based on a condition. A condition is an expression that either evaluates to true (usually represented by 1) or false (represented by 0). The result of this evaluation can be assigned to a variable.

Example :

```
#include<stdio.h>
void main (void)
{
        int age, status ;
        printf("Enter the age : ") ;
        scanf("%d", &age) ;
        status = (age > 60) ;
        printf("Status = %d", status) ;
}
```

Note : Value of the variable status will be 1 if the age is greater than 60, otherwise status will be 0.

Q : 11-02-02 : Describe Simple IF Statement ? Explain it with diagram and an example ?

Answer :

if **Statement** : If statement is the simplest form of decision constructs. It allows a statement or a set of statements to be executed conditionally. The general form of simple if statement is :

```
if (condition)
{
        Statement1 ;
        Statement2 ;
        .
        .
        .
        Statementn ;
}
```

The statement(s) in the block of if statement are executed if the condition is **true;** otherwise these are skipped. If there are more statements in if block then these should be enclosed in braces as a compound statement. However, in case of a single statement the braces are optional. This flow chart describes a situation where a simple if statement can be used.

www.pakcity.org
EDUCATION
pakcity.org

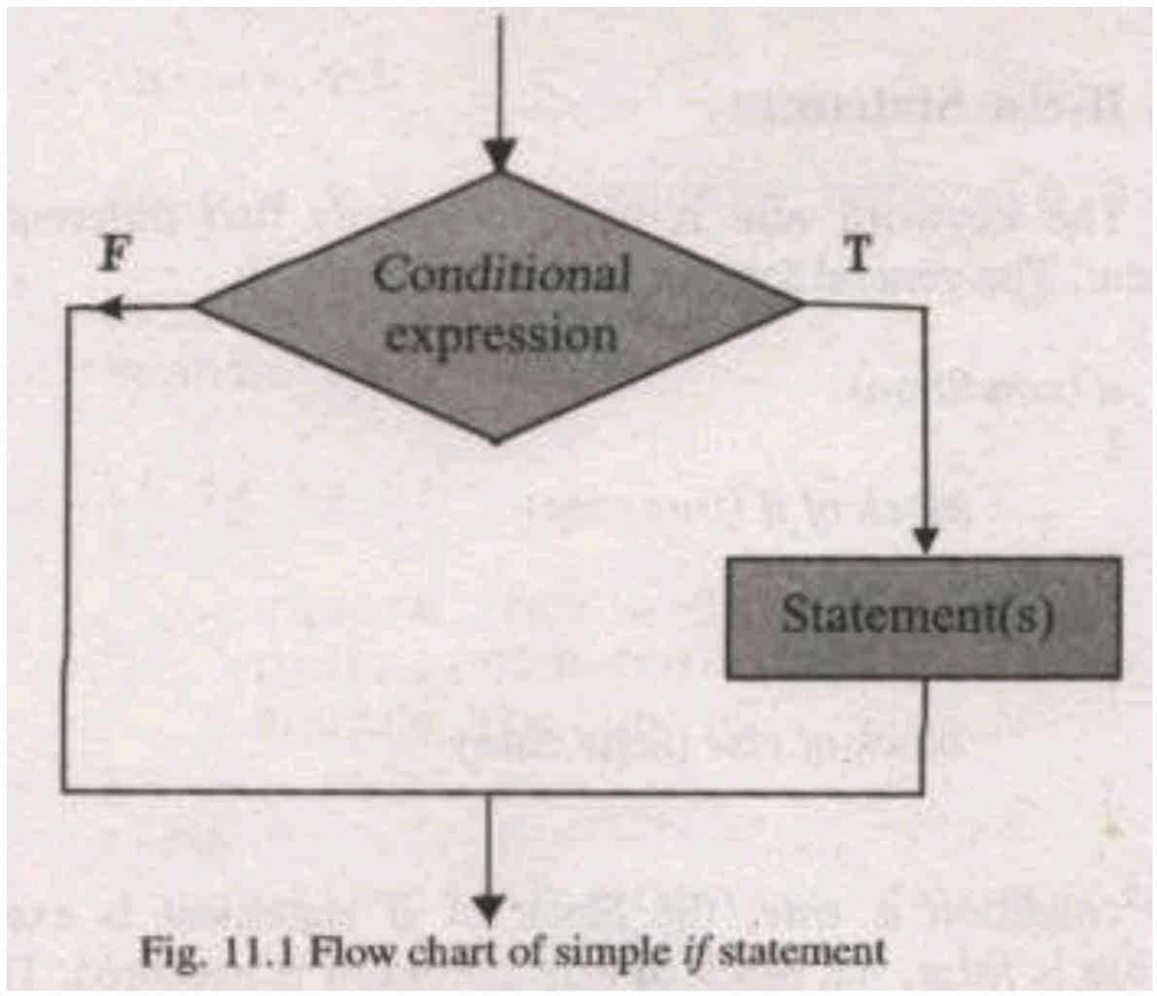

Fig. 11.1 Flow chart of simple *if* statement

Example : A program that calculates the square root of a number input by user.

```
#include<stdio.h>
#include<math.h>
void main(void)
{
        double x = 0.0, square_root = 0.0 ;
        printf("Enter a number => ") ;
        scanf("%lf", &x) ;
        if (x > 0)
        {
                square_root = sqrt(x) ;
                printf("Square Root of %f = %f", x, square_root) ;
        }
}
```

As the square root of negative numbers is imaginary, therefore the program finds the square root of positive numbers only. If a negative number is input, the program does nothing and terminates. We use a function sqrt( ) from math library. This includes the definition of sqrt( ) and many other mathematical functions.

Important Note : A flow chart is the pictorial representation of a program and it logic.

Q : 11-02-03 : Describe IF-ELSE Statement ? Explain it with diagram and an example ?

pakcity.org

Answer :

IF-ELSE Statement : If statement is the simplest form of decision constructs. It allows a statement or a set of statements to be executed conditionally. The keyword else is used to specify two different choices with if statement. The general form of if-else statement is :

```
if (condition)
{
        Block of if (TRUE CASE)
}
```

```
else
{
        Block of else (FALSE CASE)
}
```

If the condition is TRUE, the block of if statement is executed and if the condition is FALSE, the block of else statement is executed. This flow chart explains the idea :

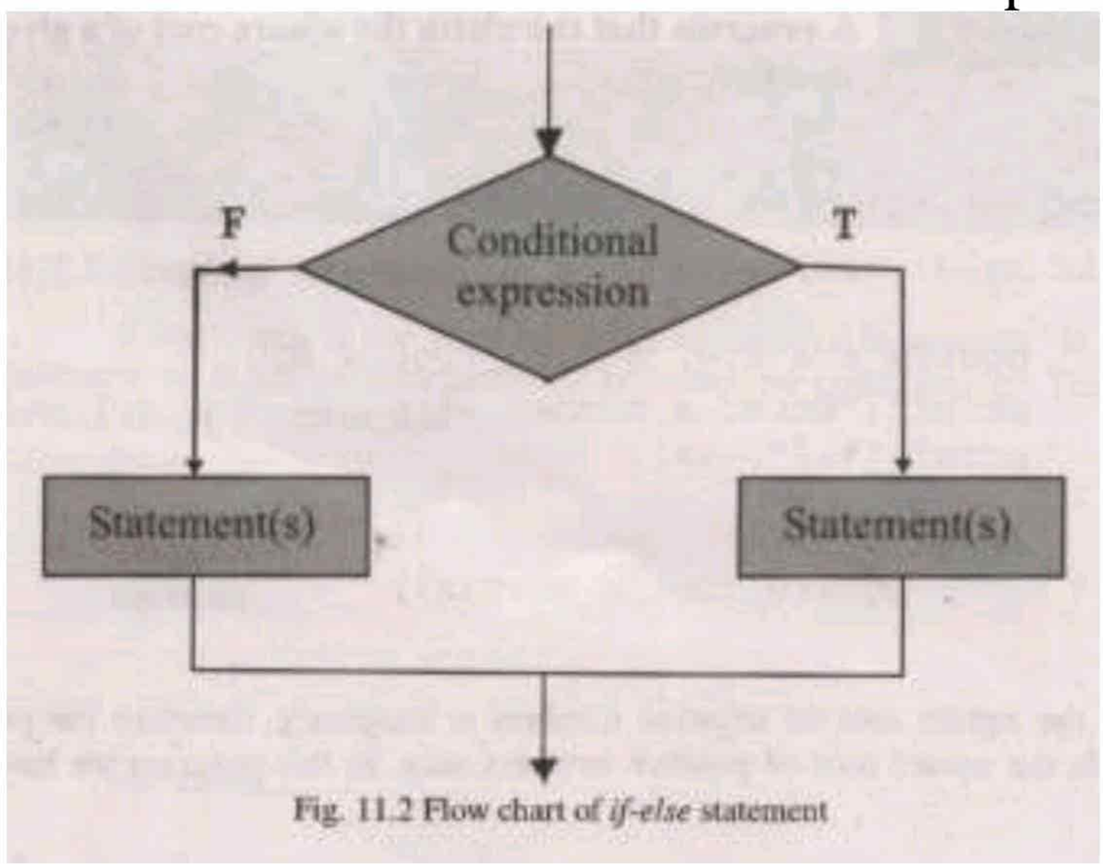

Fig. 11.2 Flow chart of *if-else* statement

Example : A program that calculates the square root of a number input by user using if-else.

```
#include<stdio.h>
#include<math.h>
void main(void)
{
        double x = 0.0, square_root = 0.0 ;
        printf("Enter a number => ") ;
        scanf("%lf", &x) ;
        if (x >= 0)
        {
                square_root = sqrt(x) ;
                printf("Square Root of %f = %f", x, square_root) ;
        }
        else
                printf("Square Root of a Negative number can't be
        calculated") ;
}
```

There are two blocks of statements in this program, either of which is conditionally executed. If the condition evaluates to true the square root of x will be calculated and the output will be shown on the screen and in case the condition evaluates to false, the message "Square Root of a Negative number can't be calculated" will be displayed.

Important Note : The block of if is enclosed in braces, and no bracket has been used in the body of else statement. The reason is that we want to execute multiple statements in case of true condition, so these must be represented as a compound statement. If we omit the braces from the body of if statement, the error message "Illegal else without matching if" will appear. So, to avoid this message, one should always enclose a compound statement in braces. If there is a compound statement in the body of else, omitting braces will not cause the above-mentioned error, rather only

one statement after the else will be treated as the body of else and the rest of the statements will always be executed sequentially.

Q : 11-02-04 : Describe Nested IF Statement ? Explain it with diagram and an example ?

Answer :

Nested IF Statement : Nested if statement means an if statement inside another if statement. Nesting can be done up to any level. The programmer may use as many (statements inside another if statement as (s)he wants. However, the increase in the level of nesting also increases the complexity of the nested if statement. The general form of nested if statement is :

```
if (condition1)
{
        Block of Outer if Begins
        if (condition2)
        {
                Block of INNER if
        }
        Block of Outer if Ends
}
```

Important Note : The else statement is optional, it may or may not be used with outer or inner if statement. However, if used, its block can also contain other if statements.

Flowchart : This flow chart of nested if statement explains the logic.

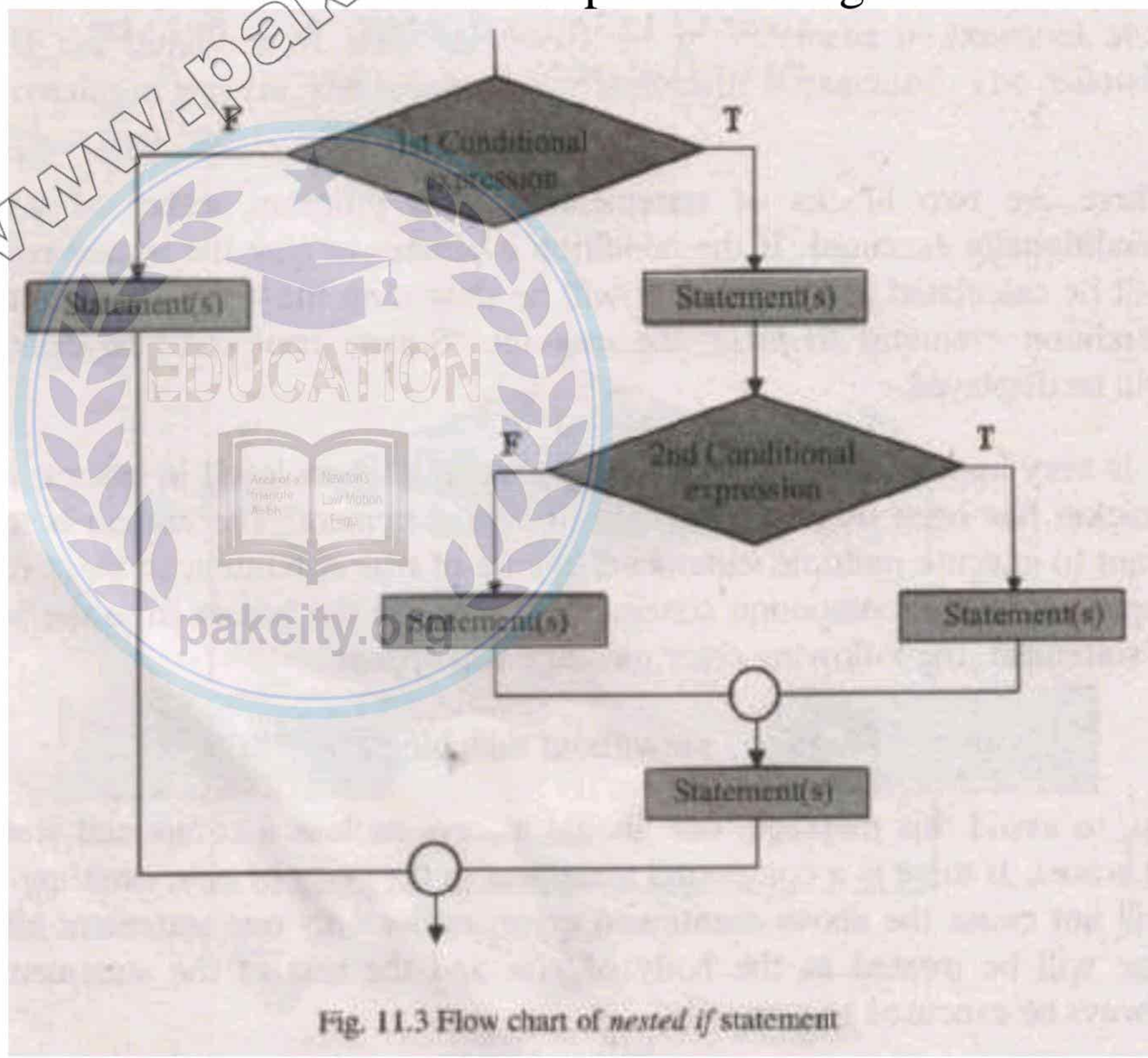

Fig. 11.3 Flow chart of *nested if* statement

Example : A program that accepts three numbers from the user and displays the largest number.

```
#include<stdio.h>
void main(void)
{
        int     a,      b,      c ;
```

```
printf("Enter THREE numbers => ") ;
scanf("%d %d %d", &a, &b, &c) ;
if (a > b)
{
        if (a > c)
                printf("%d is the largest", a) ;
        else
                printf("%d is the largest", c) ;
}
else
{
        if (b > c)
                printf("%d is the largest", b) ;
        else
                printf("%d is the largest", c) ;
        }
}
```

This is a simple program that compares three numbers to find the largest one. The body of if and the body of else, both contain other if statements (nested ifs are boxed). This sort of arrangement of multiple if statements is called nested-if statements.

Execution Sequence : The first condition (a > b) is tested, if it is true then the second condition (a > c) is tested, if it is also true, the rest of the conditions will be skipped and we conclude that a is the largest number. If the second condition is false, this but smaller than c. Therefore c is the largest number. If the first condition (a > b) is false, the body of if is skipped and the flow control is transferred to the body of else where the condition (b > c) is tested. If it is true then the second condition (b > c) is tested, if it is also true, means b is greater than a (from the first condition) and b is greater than c (from the second condition), therefore we conclude that b is the 1argest number. If the second condition in the body of else is false, it means b greater than a (from the first condition), but b is smaller than c, therefore we conclude that c is the largest number. We can implement decision making in the program using statement.

Q : 11-02-05 : Compare Nested if and Sequence of ifs and explain with example ?

Answer :

Comparison of Nested if and Sequence of ifs : Due to the complexity of nested if statement, beginners usually prefer to use a sequence of if statements rather than a single nested statement. Sometimes, it is useful to use a nested if instead of sequence of ifs. In case of nested if statement, when the control flow reaches a logical decision, the rest of the conditions are skipped. Whereas in a sequence of if statements, all conditions are tested in any case.

Example : A program that inputs a number from the user and determines whether it is positive, negative or zero.

```
#include<stdio.h>
void main(void)
{
```

```
                int     num ;
                printf("Enter a number => ") ;
                scanf("%d", &num) ;
                if (num > 0)
                        printf("The number is Positive") ;
                if (num < 0)
                        printf("The number is Negative") ;
                if (num == 0)
                        printf("The number is Zero") ;
        }
```

This program implements the solution using a sequence of if statements. Suppose, the user enters a positive number, the answer is decided in the first comparison i.e. the number is positive. So there is no need- to check the number for its being negative or zero as it is impossible. But this solution suggests doing the last two comparisons unnecessarily, wasting the CPU time. This situation may be avoided by using some other form of if statement such as if-else statement.

Q : 11-02-06 : Describe if-else-if Statement with data flow diagram and example ?

Answer :

if-else-if Statement : Nested if statements can become quite complex, if there are more than three alternatives and indentation is not consistent, it may be difficult to determine the logical structure of the if statement. In such situations, if statement with multiple alternatives (if-else-if) can be a good option. The general form of if-else-if statement is :

```
if (condition1)
        statement1 ;
else if (condition2)
        statement2 ;
.
.
.
else if (conditionn)
        statementn ;
else
        statementk ;
```

The test conditions in if statement with multiple alternatives are executed in sequence until a true condition is reached. If a condition is TRUE, the statement(s) following it is executed, and the rest of the alternatives skipped. If a condition is FALSE, the statement following it is skipped and the next condition is tested. If all conditions are FALSE, then $statement_k$ following the last else is executed. **Flowchart** : It shows the execution flow of the program through if-else-if statement.

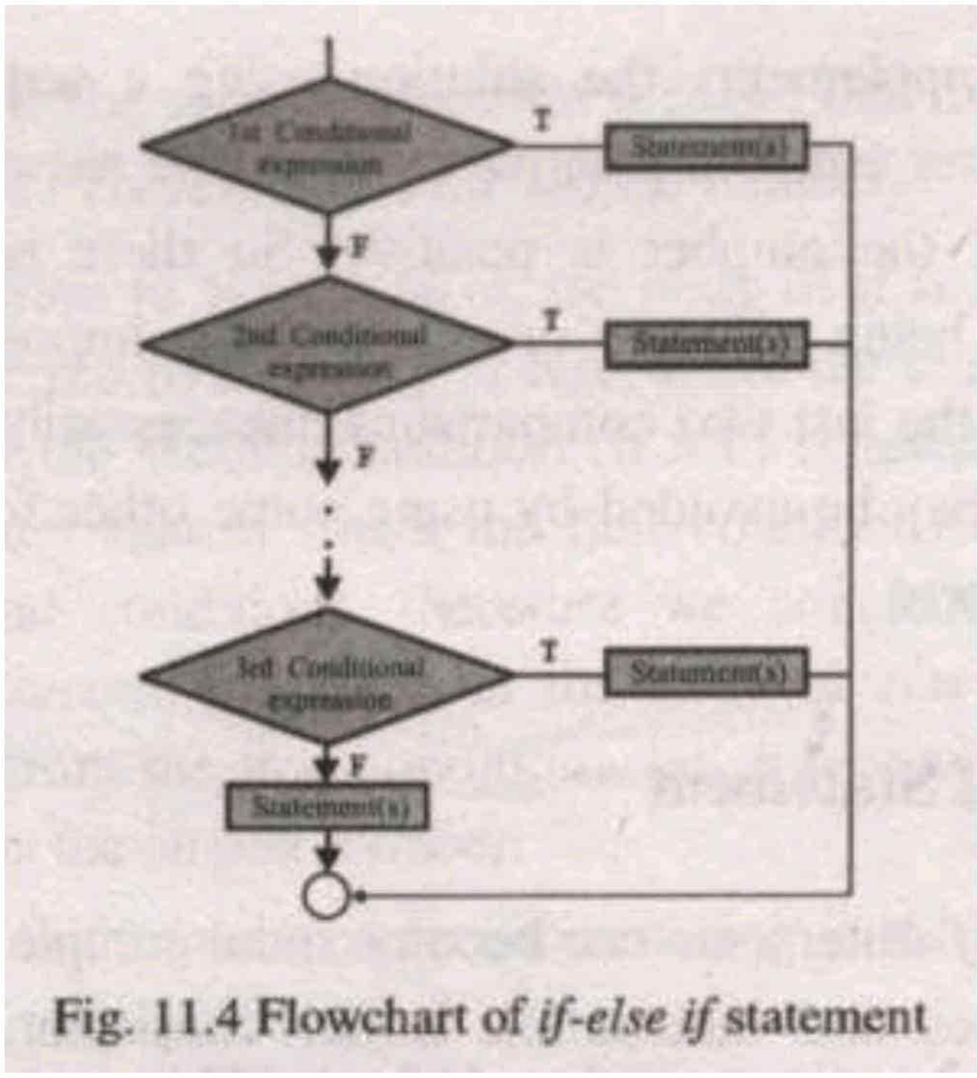

Fig. 11.4 Flowchart of *if-else if* statement

Example : A program that inputs a number from the user and determines whether it is positive, negative or zero. The program is re-written using if-else-if statement :

```
#include<stdio.h>
void main(void)
{
        int     num ;
        printf("Enter a number =>") ;
        scanf("%d", &num) ;
        if (num > 0)
                printf("The number is Positive") ;
        else if (num < 0)
                printf("The number is Negative") ;
        else
                printf("The number is Zero") ;
}
```

It can be seen that the if-else-if construct has greatly simplified the program while preserving the efficiency as well. If a positive number is entered, we will reach the answer in the first comparison and rest of the conditions will be skipped.

11.3 Use of Logical Operators

Q : 11-03-01 : Illustrate the use of Logical Operators giving example ?

pakcity.org

Answer :

Use of Logical Operators : Three logical operators AND, OR, and NOT play an important role in constructing certain compound conditions to be used with if statement. Now, we'll see how complex program logic can be simplified using logical operators :

Example : A program that accepts three numbers from the user and displays the largest number.

```
#include<stdio.h>
void main(void)
{
        int     a,      b,      c ;
        printf("Enter THREE numbers => ") ;
        scanf("%d %d %d", &a, &b, &c) ;
        if (a > b && a > c)
                printf("%d is the largest", a) ;
        else if (b > a && b > c)
                printf("%d is the largest", b) ;
        else
                printf("%d is the largest", c) ;
}
```

There may be situations where the use of logical operators may simplify the program logic. We just need to concentrate on the underlying problem.

11.4 Switch Statement

Q : 11-04-01 : Illustrate the use of Switch Statement giving example ?

Answer :

Switch Statement : The switch statement can also be used in C to select one of many alternatives. Like if statement, it is also a conditional statement that compares the value of an expression against a list of cases. The case label can be an integral or character constant. Similarly, the value of the expression must be an integer or a character; it must not be a double or float value. Here's the syntax of switch statement.

```
switch(expression)
{
        case val1 :
                statements1 ;
                break;
        case val2 :
                statements2 ;
                break;
        .
        .
        .
        case valn :
                statementsn ;
                break;
        default :
                statementsk ;
}
```

The value of expression is compared to each case label. The statement following the first label with a matching evaluated value are executed until a break statement is encountered. The break causes the rest of the switch statement to skipped. If all break

statements are omitted from the switch statement, the code from the first TRUE case down to the end of the switch statement will execute sequentially. A default label may be used to provide code to be executed if none of the case label matched. However the position of default label is not fixed. It may be placed before the first case statement or after the last case. The switch statement is especially useful when the selection is based on the value.

Example : Write a program that inputs a character from the user and checks whether it is a vowel or a consonant.

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    char   ch ;
    printf("Enter an ALPHABET => ") ;
    ch     =     getche( ) ;
    switch (ch)
    {
        case 'a' :
        case 'A' :
                printf("It's a Vowel") ;
                break ;
        case 'e' :
        case 'E' :
                printf("It's a Vowel") ;
                break ;
        case 'i' :
        case 'I' :
                printf("It's a Vowel") ;
                break ;
        case 'o' :
        case 'O' :
                printf("It's a Vowel") ;
                break ;
        case 'u' :
        case 'U' :
                printf("It's a Vowel") ;
                break ;
        default :
                printf("It's NOT a Vowel, It's a
            CONSONANT") ;
                break ;
        }
    }
```

In this program, we have used two case labels, one after the other, without having any other statement between them. This sort of arrangement of case labels works equivalent to the logical OR operator i.e., whether the value of the variable ch is 'a' or 'A', the code following the labels case 'a and case 'A' will execute.

Important Note 1 : The value of expression in switch statement must be of type int or char, but not of type float or double.
Important Note 2 : If the value of expression in switch statement is of float or double, the compiler will generate error message : "Switch selection expression must be of integral type".

11.5 Conditional Operator

Q : 11-05-01 : Illustrate the use of Conditional Operator / Statement (ternary operator) giving example ?

Answer :
Conditional Operator / Statement : Conditional operator is used as an alternative to the if-else statement. It is a ternary operator (requires three operand). Its general form is; conditional expression ? true-case statement false-case statement ; The expression may be a relational or logical expression. If the expression is true then the true-case statement will be executed otherwise the false-case statement is executed. e.g.,

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
        int     a,      b ;
        printf("Enter TWO numbers => ") ;
        scanf("%d %d", &a, &b) ;
        a > b ? printf("%d is larger", a) : printf("%d is larger", b) ;
}
```

11.6 Locating a Point in The Coordinate Plane

Case Study : Write a program that input x- and y-coordinates of a point in the coordinate plane. Based on these values, it then determines where it lies, on x- or y-axis, or in any of the four quadrants. The first step towards the solution of any problem (simple or complex) is to understand it. Think on the problem from all aspects; identify input and output requirements at different types of restrictions on the data. After analyzing the problem, try to develop a simple solution. Don't indulge yourself in unnecessary details. You can trace the so1utio on the paper in the form of an algorithm. [An algorithm is' step-by-step procedure to solve any problem in finite number of steps].It is clear from the figure 11.1 that each point will lay on either of the two axes or in any of the four quadrants.
Input Values : The program will input the values of x-coordinate and y-coordinate. These values will be of type int.
Output : The program output describing the position of the point in the coordinate plane.
What Do We Know ?

If both x- and y-coordinates are zero, the point will be at the origin.
If x-coordinate is zero and y-coordinate is non-zero, then the point will be on y-axis.

If y-coordinate is zero and x-coordinate is non-zero, then the point will be on x-axis.
If both x- and y-coordinates are positive (x > 0 && y > 0), the point will lie in the 1st quadrant.
If x-coordinate is negative (x < 0) and y-coordinate is positive (y > 0), the point will lie in the 2nd quadrant.
If both x- and y-coordinates are negative (x < 0 && y < 0), the point will lie in the 3rd quadrant.
If x-coordinate is positive (x > 0) and y-coordinate is negative (y < 0) the point will lie in the 4th quadrant.

Program : A program that accepts TWO numbers x and y from the user and determines / displays the quadrant on which the point (x, y) lies.

```
#include<stdio.h>
void main(void)
{
        int     x,      y ;
        printf("Enter values of x and y coordinates => ") ;
        scanf("%d %d", &x, &y) ;
        if (x == 0)
        {
                if (y == 0)
                        printf("The point lies on THE ORIGIN") ;
                else
                        printf("The point lies on Y-AXIS") ;
        }
        else if (x > 0)
        {
                if (y == 0)
                        printf("The point lies on X-AXIS") ;
                else if (y > 0)
                        printf("The point lies in 1st Quadrant") ;
                else
                        printf("The point lies in 4th Quadrant") ;
        }
        else
        {
                if (y == 0)
                        printf("The point lies on X-AXIS") ;
                else if (y > 0)
                        printf("The point lies in 2nd Quadrant") ;
                else
                        printf("The point lies in 3rd Quadrant") ;
        }
}
```

Exercise 11

Q-9. A year is a leap year if it is divisible by four, except that any year divisible by 100 is a leap year only if it is divisible by 400. Write a program that inputs a year such as 1996, 1800, and 2010, and displays "Leap Year" if it is a leap year, otherwise displays "Not a Leap Year".

Answer :

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
        unsigned int year = 0 ;
printf("Enter The Year => ") ;
scanf("%d", &year) ;
if (year % 4 == 0)
{
        if (year % 100 == 0)
        {
                if (year % 400 == 0)
                {
                        printf("%d is a Leap Year", year) ;
                }
                else
                {
                        printf("%d is a NOT Leap Year", year) ;
                }
        }
        else
        {
                printf("%d is a Leap Year", year) ;
        }
}
else
{
        printf("%d is Not a Leap Year", year) ;
}
}
```

Q-10. Write a program that inputs temperature and displays a message according to the following data :

Temperature	Message
Greater Than 35	Hot Day
Between 25 and 35 (Both Inclusive)	Pleasant Day
Less Than 25	Cool Day

Answer :

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

```
void main(void)
{
        int     temerature ;
        printf("Enter Temperature in Celsius => ") ;
        scanf("%d", &temerature) ;
        if (temperature > 35)
        {
                printf("%d Degrees Celsius ! Its a HOT Day", temperature) ;
        }
        else if (temperature >= 25)
        {
                printf("%d Degrees Celsius ! Its a PLEASANT Day",
                temperature) ;
        }
        else
        {
                printf("%d Degrees Celsius ! Its a COOL Day", temperature)
                ;
        }
}
```

Q-11. Write a program that inputs obtained marks of a student, calculates percentage (assume total marks are 1100 or may get input from user), and displays his / her grade. The grade should be calculated according to the following rules :

Percentage	**Grade**
More than or equal to 80	A+
Between 70 (inclusive) and 80	A
Between 60 (inclusive) and 70	B
Between 50 (inclusive) and 60	C
Between 40 (inclusive) and 50	D
Between 33 (inclusive) and 40	E
Less than 33	F

Answer :

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
        int     marks ;
        float percent ;
        printf("Enter Obtained Marks => ") ;
        scanf("%d", &marks) ;
        percent = (marks / 1100) * 100 ; // Assume Total Marks = 1100
        if (percent >= 80)
        {
                printf("A+") ;
        }
```

```
else if (percent >= 70)
{
        printf("A") ;
}
else if (percent >= 60)
{
        printf("B") ;
}
else if (percent >= 50)
{
        printf("C") ;
}
else if (percent >= 40)
{
        printf("D") ;
}
else if (percent >= 33)
{
        printf("E") ;
}
else
{
        printf("F") ;
}
}
```

Q-12. Write a program that inputs two numbers and asks for the choice of the user, if user enters 1 then displays the sum of numbers, if user enters 2 then displays result of subtraction of the numbers, if user enters 3 then displays the result of the multiplication of the numbers and if user enters 4 then displays result of the division of the numbers (divide the larger number by the smaller number), otherwise display the message "Wrong Choice". [Use switch statement to implement the solution].

Answer :

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
        float   num1, num2, result ;
        float result2 ;
        char choice ;
        printf("Enter The Tow Numbers => ") ;
        scanf("%d %d", &num1, &num2) ;
        printf("1.      Add\n") ;
        printf("2.      Subtract\n") ;
        printf("3.      Multiply\n") ;
        printf("4.      Divide\n") ;
```

www.pakcity.org

```
printf("Please Enter Your Choice From 1 To 4 ==> ") ;
choice = getche() ;
switch(choice)
{
        case '1':       result = num1 + num2 ;
                        printf("\nAddition %d + %d = %d", num1,
                        num2, result) ;
                        break ;
        case '2':       result = num1 – num2 ;
                        printf("\nSubtraction %d - %d = %d" , num1,
                        num2, result) ;
                        break ;
        case '3':       result = num1 * num2 ;
                        printf("\nMultiplication %d * %d = %d" ,
                        num1, num2, result) ;
                        break ;
        case '4':       if (num1 > num2)
                        {
                                Result2 = num1 / num2 ;
                                printf("\nDivsion %d / %d = %f" , num1,
                                num2, result2) ;
                        }
                        else
                        {
                                Result2 = num2 / num1 ;
                                printf("\nDivsion %d / %d = %d" , num2,
                                num1, result2) ;
                        }
                        break ;
      default :         printf("\nIn-Valid Choice : Try 1 To 4") ;
    } // end of switch block
} // end of main block
```

www.pakcity.org
pakcity.org