

Chapter 6: Decision making and branching

What are we studying in this chapter?

- ◆ Introduction
- ◆ Single selection statement: if statement
- ◆ Two way selection
 - The if else statement
 - The ?: operator
 - Nested if..else statement
 - Simplifying if statements
 - Handling errors
 - Two-way selection Examples
- ◆ Multi-way selection
- ◆ The else..if ladder
- ◆ The switch statement
- ◆ The goto statement
- ◆ The break statement
- ◆ Software Engineering and programming style
- ◆ Tips and common programming errors

6.1 Introduction

By this time, we should know the definition of keywords, data types, how to declare the variables, how to assign the values to variables, various types of operators, how to write the expressions, how to evaluate the expressions etc. Also, we should know how to write simple programs that involves sequential control. Now, using these concepts let us proceed further and see how to write more complex programs. We start from simple programs and increase the complexity of programs stage by stage. By the time we read the complete text book, we will know how to write C programs for complex problems.

6.2 Selection/branching statements

Analogy: In real life, we may have some set of activities or jobs to be performed. All the activities are not performed all the time in sequence i.e., one after the other. Based on the time or based on the situation various activities are performed:

- ◆ Ex 1: If we are hungry, we take food
- ◆ Ex 2: If class test is announced, we study
- ◆ Ex 3: If it is 8.30 P.M, we watch news in T.V
- ◆ Ex 4: If it is 10P.M, we goto bed.

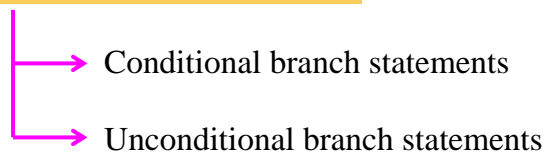
6. 2 Decision making and branching

On similar lines, in programming also, we may have to execute some statements when some conditions are met or we may skip execution of some statements based on other condition. These statements are called branch statements. Now, let us formally see:

Q:6.1: What are branching statements? Explain the different types of branching statements?

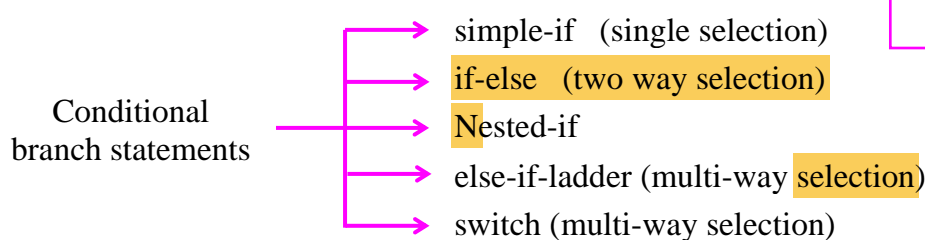
[03 marks] [BT-L1]

Definition: The statements that transfer the control from one place to other place in the program with or without any condition are called **branch statements**. The branching instructions are classified as shown below:



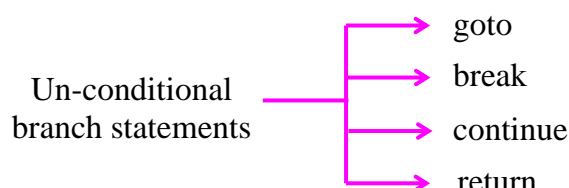
Marks:1

Conditional branch statements: The statements that transfer the control from one place to other place so as to execute a set of instructions if some condition is met or to skip the execution of some statements if the condition is not met are called **conditional branch statements**. They are also called **selection statements** or **decision statements**.



Marks:1

Definition: The statements that transfer the control from the one statement to other statement in the program without any condition are called **un-conditional branching statements** or **un-conditional control statements**. The unconditional statements are classified as shown below:



Marks:1

6.3 The if-statement (One-way selection/decision)

In this section, we discuss the syntax of if-statement, how it works and how it can be used in problem solving. Now, let us see:

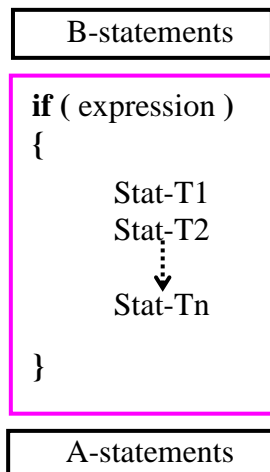
Q:6.2: What is an if-statement? Explain with syntax.
[04 marks] [BT-L1]

Definition: An *if-statement* is a single selection or decision statement. When a set of statements have to be executed when an expression is evaluated to **true** (non-zero value) or when a set of statements have to be skipped when an expression is evaluated to **false** (zero), then *if-statement* is used. It is used when we have only one alternative. Hence, it is also called *one-way decision/selection statement*.

Marks:1

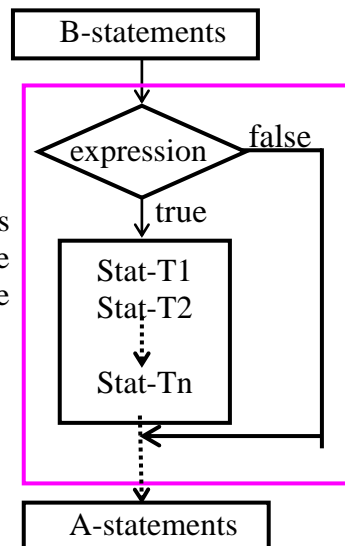
The syntax of *if statement* can be written as shown below:

C-SYNTAX



The statements T1 to Tn are executed if the condition is true

FLOW-CHART



The statements T1 to Tn are skipped if the condition is false

Marks:2

- ◆ The keyword **if** must be followed by an expression and the expression must be enclosed within parentheses.
- ◆ The statements T1, T2,...,Tn are enclosed within braces. If there is only one statement T1, then usage of braces is optional.

WORKING: The sequence of operations that are carried out when the if-statement is executed are shown below:

6. 4 Decision making and branching

- ◆ The B-statements that are present before the if-statement are executed one after the other.
- ◆ The if-statement is executed next. The expression inside the parentheses is evaluated.
- ◆ After evaluation, if the result is non-zero value, it is considered as **true** and the statements T1, T2,Tn are executed and finally control comes out of the if-statement
- ◆ After evaluation, if the result is zero value, it is considered as **false** and the statements T1, T2,...Tn are skipped and control comes out of the if statement.
- ◆ Once control comes out of if-statement, the A-statements present immediately after if-statement are executed.
- ◆ Observe that A-statements are executed whether the expression in the if-statement is evaluated to **true** or **false**.

Now, let us take some sample *if-statements* and see whether they are syntactically correct or not. Apart from that let us see any side effects or logical errors are present.

- 1) The keyword **if** must be followed by an expression and the expression must be enclosed within parentheses. For example,

Ex 1: Syntactically correct

```
if (a > b)      /* OK : expression a > b is within parentheses */
    printf("A is greater\n");
```

Ex 2: Syntax error

```
if a > b        /* Error : parentheses missing */
    printf("A is greater\n");
```

Ex 3: Syntax error

```
if (a > b       /* Error : right parentheses missing */
    printf("A is greater\n");
```

- 2) The expression may have side effect i.e., the values of a variables used in the expression may change during execution. For example,

```
if ( ++count > 10)    /* side effect: The value of count is incremented by 1 */
{
    count = 0;
}
```

Note: The incremented value of count is compared with 10.

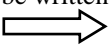
- 3) If multiple statements have to be executed when the expression is true, then all those statements must be enclosed within braces. If only one statement has to be executed, then usage of braces is optional.

Ex 1: Usage of braces is mandatory.

```
if (a > b)           // Line 1 : OK
{                   // Line 2 : OK
    printf("A is greater\n"); // Line 3 : OK
    exit(0);         // Line 4 : OK
}
```

Note: Both statements *printf()* and *exit()* must be executed whenever the value of *a* is greater than *b*. So, they must be enclosed within braces.

Ex 2: Usage of braces is optional in the following example

<pre>if (a > b) { printf("A is greater\n"); }</pre>	can be written as 	<pre>if (a > b) printf("A is greater\n");</pre>
--	--	--

/* Braces are optional */

Ex 3: The braces should match. Otherwise, it results in syntax error.

```
if (a > b)           // Line 1 : OK
{                   // Line 2 : OK
    printf("A is greater\n"); // Line 3 : OK
    exit(0);         // Line 4 : OK
] // Error: Replace the symbol '[' by the symbol '{'
```

- 4) If semicolon is present in the if-statement header, it is syntactically correct. For example, whenever the value of *a* is greater than *b*, we want the output “A is greater” Let us write the statement as shown below:

```
if (a > b) ;         // Line 1 : OK
    printf("A is greater\n"); // Line 2 : OK
```

Observe the following points in the above if-statement.

- ◆ There is no compilation error. The above statement is syntactically correct.
- ◆ The *semicolon* at the end of line 1 indicates a NULL statement. It means: whenever *a* is greater than *b* “do nothing”

6.6 Decision making and branching

- ◆ So, line 2 is not part of the **if**-statement and hence, it is executed immediately after if-statement is executed.
- ◆ By removing semicolon at the end of line 1, logical error (semantic error) can be corrected and can be written as shown below:

```
if (a > b)
    printf("A is greater\n");
```

6.3.1 Identify even number and odd number

Before designing an algorithm or a program, let us see *what is an even number and odd number*. A number which is divisible by 2 is an *even number* and a number which is not divisible by 2 is an odd number. The even numbers and odd numbers are always integers. For example,

- ◆ 2, 4, 6, etc. are even numbers
- ◆ 1, 3, 5 etc are odd numbers.

Q:6.3: Design and write a program to identify whether the number is even or odd
[04 marks] [BT-L5]

Statement of the problem: Given an integer n , we may have to output “Even” when n is divisible by 2 and output “Odd” when n is not divisible by 2.

Design: We know that n is the given input. After dividing n by 2, if the remainder is zero, then we display the message “Even” and terminate the function using **return** statement. So, the equivalent code can be written as:

```
if (n % 2 == 0)
{
    printf("Even");
    return;
}
printf("Odd");
```

Marks:2

Observe that when n is not divisible by 2, the control comes out of the if-statement and we display the message “Odd”.

Before executing the above statement, we have to input the value of n using `scanf()` function. Now, the complete program can be written as shown below:

Program 6.1: C program to check whether the given number is even or odd

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

Marks:2

```
void main()
{
```

```
    int    n;
```

```
    printf ("Enter the number\n");
    scanf ("%d",&n);
```

```
    if (n %2 == 0) // If remainder is zero
    {
```

```
        printf("Even");
        return;
```

```
    }
    printf("Odd");
}
```

TRACINGInput

```
Enter the number
10
```

Output

```
Even
```

Input

```
Enter the number
7
```

Output

```
Odd
```

6.3.2 Program to find maximum of 2 numbers

In this section, let us discuss the design methods and method of writing a program to find maximum of two numbers. Now, let us:

Q:6.4: Design and write a program to display maximum of two numbers.
[04 marks] [BT-L5]

Statement of the problem: Given two numbers say a and b , we find maximum of these two numbers and print the result.

Design: Given two numbers a and b , let us copy b to big using the statement:
 $big = b;$

Now, instead of comparing a and b , we can as well compare a and big and find the largest using the following statement:

```
if (a > big) big = a;
```

Observe the following points:

- 1) In the above if statement, if a is greater than big , a is the largest and it is copied into big and hence big has largest value.
- 2) If big is greater than a , the statement " $big = a;$ " is not executed and big itself is the largest.

6.8 Decision making and branching

Observe that when the above if-statement is executed, whether the condition is true or false, *big* has the largest value and we print the value of *big* using the statement:

```
printf("Largest = %d\n", big)
```

Marks: 2

Now, given *a* and *b*, the complete set of instructions to find the largest of *a* and *b* can be written as shown below:

```
big = b;  
if (a > big) big = a;  
printf("Largest = %d\n", big)
```

Given *a* and *b*, this portion is body of the function to find largest of two numbers

Before executing the above three statements, we should **input the values of *a* and *b* using the scanf() statement**. The complete program can be written as shown below:

Program 6.2: C program to print maximum of two numbers

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

Marks: 2

```
void main()
```

```
{
```

```
    int    a, b, big;
```

```
    printf ("Enter two numbers\n");
```

```
    scanf("%d %d",&a, &b);
```

```
    big = b;
```

```
    if (a > big) big = a;
```

```
    printf("Largest = %d\n", big)
```

```
}
```

TRACING1

Input

Enter two numbers

10 20

Computations

Largest = 20

TRACING 2

Input

Enter two nos.

20 10

Computations

Largest = 20

6.3.3 To find largest of 3 numbers

In this section, let us discuss the design methods and method of writing a program to find maximum of three numbers.

Q:6.5: Design and write a program to find maximum of three numbers
[04 marks] [BT-L5]

Statement of the problem: Given three numbers say a , b and c , we find maximum of these three numbers and print the result.

Design: Given three numbers a , b and c , let us copy b to big using the statement:
`big = b;`

Now, instead of comparing a and b , we can as well compare a and big and find the largest of a and b using the following statement:

`if (a > big) big = a;`

Now, big contains the largest of a and b . Now, let us compare c and big using the following statement:

`if (c > big) big = c;`

Observe that when the above two if-statements are executed, whether the conditions are **true** or **false**, the variable big has the largest value and we print the value of big using the statement:

`printf("Largest = %d\n", big)`

Now, given a , b and c , the complete set of instructions to find the largest of a , b and c can be written as shown below:

```
big = b;
if (a > big) big = a;
if (c > big) big = c;
printf("Largest = %d\n", big)
```

Given a , b and c , this portion is body of the function to find largest of three numbers

Marks: 2

Before executing the above statements, we should **input the values of a , b and c using the `scanf()` statement**. The complete program can be written as shown below:

Program 6.3: C program to print maximum of three numbers

`#include <stdio.h>`

Marks: 2

`void main()`

```
{
    int    a, b, c, big;
```

6. 10 Decision making and branching

<pre>printf ("Enter three numbers\n"); scanf("%d %d %d",&a, &b,&c); big = b; if (a > big) big = a; if (c > big) big = c; printf("Largest = %d\n", big) }</pre>	<pre>Enter 3 numbers 10 20 30 Computations Largest = 30</pre>	<pre>Enter 3 numbers 30 20 10 Largest = 30</pre>
--	---	---

6.3.4 To find whether given number is zero, +ve or -ve number

In this section, let us discuss the design methods and method of writing a program to find whether a given number is +ve or -ve or zero. Now, let us:

Q:6.6: Design and write a program to find whether a given number is +ve or -ve or zero.
[04 marks] [BT-L5]

Statement of the problem: Given a number n , we have to check whether the number is *positive* or *negative* or *zero*.

Design: Given a number n , if it is greater than zero it is positive and so display the message “Positive” and terminate the function using **return** statement as shown below:

```
if (n > 0)                                /* Check for positive number */
{
    printf("Positive");
    return;
}
```

If the given number is less than zero it is negative and so display the message “Negative” and terminate the function using **return** statement as shown below:

```
if (n < 0)                                /* Check for negative number */
{
    printf("Negative");
    return;
}
```

When control comes out of the two if-statements, we know that the given number has zero value and so we display the message “Zero” using the following statement:

```
printf("Zero");                          /* number is zero*/
```

Now, given n , the complete set of instructions to find whether a given number is positive, negative or zero, can be written as shown below:

```
if (n > 0)
{
    printf("Positive");
    return;
}
if (n < 0)
{
    printf("Negative");
    return;
}
printf("Zero");
```

Given n , this portion is body of the function to check whether the given number is positive, negative or zero.

Marks: 2

Before executing the above statements, we should **input the value of n using the scanf() statement**. The complete program can be written as shown below:

Program 6.4: C program to check whether a number is positive, negative or zero

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

```
void main()
```

```
{
```

```
    int    n;
```

```
    printf("Enter a number\n");
    scanf("%d",&n);
```

```
    if (n > 0)
```

```
    {
```

```
        printf("Positive\n");
        return;
```

```
    }
```

```
    if ( n < 0 )
```

```
    {
```

```
        printf("Negative\n");
        return
```

```
    }
```

```
    printf("Zero\n");
```

```
}
```

Input 1

Enter numbr
10

Output 1

Positive

Input 2 Input 3

Enter numbr Enter numb
-10 0

Output 2

Negative

Output 3

zero

Marks: 2

6. 12 Decision making and branching

6.3.5 Solving Quadratic equation

In this section, let us solve quadratic equations, discuss the design methods and write a C program. Now, let us:

Q:6.7: Design and write a program to find the root of the quadratic equation
[010 marks] [BT-L5]

Design: The design aspects are given one by one as shown below:

Statement of the problem: The general quadratic equation is of the form:

$$ax^2 + bx + c = 0.$$

Given the coefficients values a , b and c , we need to find the roots of quadratic equation. The above equation can be solved using the following two formulas:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Marks:1

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad \text{----- 6.1}$$

Case 1: ($b^2 - 4ac = 0$) Two equal roots. In the above two equations, if we substitute $b^2 - 4ac$ as 0, then

$$x_1 = x_2 = -\frac{b}{2a}$$

For example, in the equation $x^2 - 4x + 4 = 0$, $a = 1$, $b = -4$ and $c = 4$. Solving this equation, we get two equal roots $x_1 = x_2 = 2$.

\therefore , if ($b^2 - 4ac = 0$) then

Marks:1

$$x_1 = x_2 = -\frac{b}{2a}$$

print (x1, x2)

return

end if /* end of the if statement */

Case 2: ($b^2 - 4ac > 0$) Two distinct real roots if $b^2 - 4ac$ is greater than 0, we get two distinct roots and can be obtained using equations shown in 6.1.

For example, consider the equation $x^2 - 5x + 6 = 0$. Here, $a = 1$, $b = -5$ and $c = 6$. Solving this equation, we get $x_1 = 2$ and $x_2 = 3$

\therefore , if ($b^2 - 4ac > 0$) then

// find two real and distinct roots

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad \text{----- 6.2}$$

print (x1, x2)

return

end if /* end of the if statement */

Marks:1

Case 3: ($b^2 - 4ac < 0$) Two complex roots In equation 6.1, if $b^2 - 4ac$ is less than 0, we get two complex roots.

For example, consider the equation $x^2 - 4x + 13 = 0$. Solving this equation, we get two complex roots.

\therefore if ($b^2 - 4ac < 0$)

print "complex roots"

return

end if /* end of the if statement */

Marks:1

Now, the complete algorithm to find the roots of quadratic equation is shown below:

6. 14 Decision making and branching

Algorithm QUADRATIC

Marks:6

FLOWCHART

Step 1:[Input the coefficients]

read a, b, c

Step 1: [Find two equal roots]

if ($b^2 - 4ac == 0$) then

$x1 = x2 = -b / (2a)$

print ("Two equal roots", x1, x2)

return

end if

Step 2: [Find two distinct roots]

if ($b^2 - 4ac > 0$) then

$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$

print ("Two distinct roots", x1, x2)

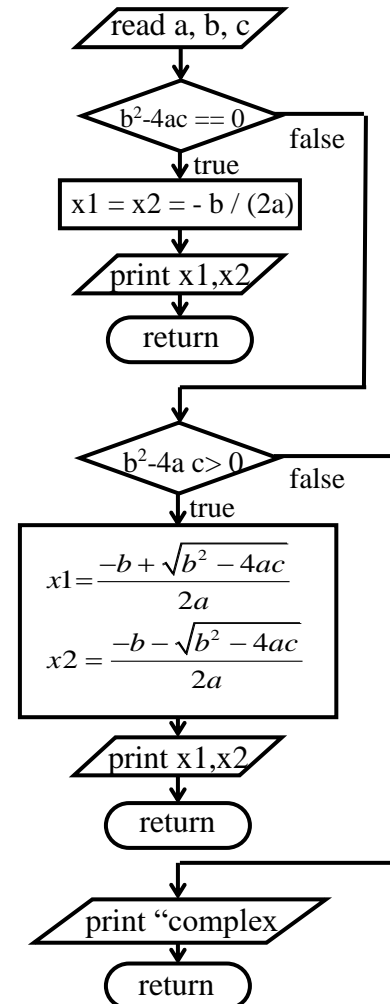
return

end if

Step 3: [Print complex roots]

print ("roots complex")

return



Now, complete C program can be written as shown below:

Program 6.5: C program to find roots of quadratic equation

```
void main()
{
    float a, b, c, x1, x2, disc;

    printf("Enter the co-efficients\n");
    scanf ("%f %f %f", &a, &b, &c);
```

Marks:6

<pre> /* Compute the discriminant */ disc = b*b - 4*a*c; /* Find the equal roots */ if (disc == 0) { x1 = x2 = -b/(2*a); printf("The roots are equal\n"); printf("X1 = %f\nX2 = %f\n",x1, x2); return; } /* Find the distinct roots */ if (disc > 0) { x1 = (-b + sqrt(disc)) / (2*a); x2 = (-b - sqrt(disc)) / (2*a); printf("The roots are distinct\n"); printf("X1 = %f\nX2 = %f\n",x1,x2); return; } /* Find the complex roots */ x1 = -b/(2*a); x2 = sqrt(fabs(disc))/(2*a); printf("The roots are complex\n"); printf("The first root = %f + i%f\n",x1,x2); printf("The second root = %f - i%f\n",x1,x2); </pre>		
		<p><u>Input1 TRACE 1</u></p> <p>a = 1, b = -4, c = 4</p> <p>x1 = x2 = 2.0</p> <p><u>Output1</u></p> <p>X1 = X2 = 2.00</p>
		<p><u>Input2 TRACE 2</u></p> <p>a = 1, b = -5, c = 6</p> <p>x1 = 3.0</p> <p>x2 = 2.0</p> <p><u>Output2</u></p> <p>X1 = 3.0 X2 = 2.0</p>
		<p><u>Input3 TRACE 3</u></p> <p>a = 6, b = 2, c = 4</p> <p><u>Output3</u></p> <p>First root = -0.167 + i0.8</p> <p>Second root = -0.167 - i0.8</p>

6. 16 Decision making and branching

BE CAREFUL: Using = operator in place of == can cause very serious problems. Using == operator in place of = operator also cause serious problems. For example, consider the statement:

```
area = length * breadth;      // compute the area of rectangle
```

The above statement evaluates the expression *length * breadth* and assigns the resultant value to *area*. Suppose, we type statement as shown below:

```
area == length * breadth;
```

The above statement compares the value of expression *length * breadth* with the value of *area*. The value of *area* will not change (since the operation performed is comparison). So, this results in logical error. Compiler will not display any error. But, we get the wrong output.

6.3.6 Advantage and disadvantages of if-statement

Now, let us see:

Q:6.8: What are the advantages and disadvantages of if-statement?
[BT-L1]

Advantages Since the output of if-statement is **TRUE** or **FALSE**, the if-statement is used as one-way decision/selection statement in the following situations:

- ◆ When a set of statements have to be executed when a condition is satisfied
- ◆ When a set of statements have to be skipped when a condition is satisfied

Disadvantages It is one-way decision/selection statement. So, if one action has to be performed when the condition is *true* and another action has to be performed when the condition is *false*, then if-statement is not recommended. This disadvantage we can overcome using *two-way decision/selection statement* called “*if-else statement*”.

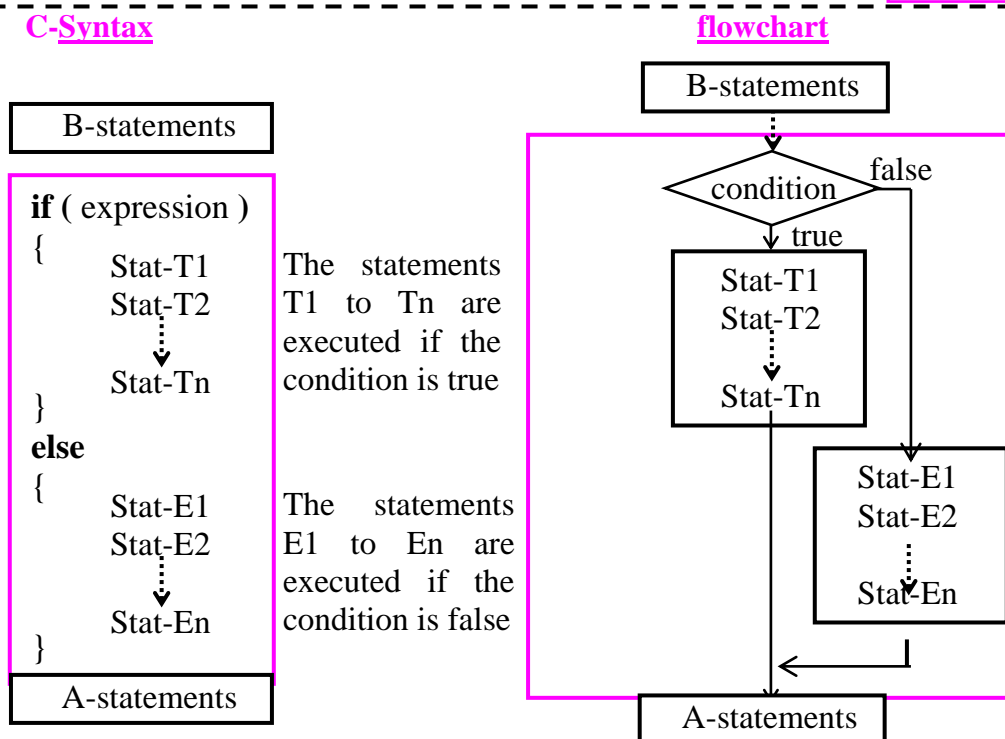
6.4 The if-else statement

In this section, we discuss about if-else, its syntax and how it can be used while writing a program. Now, let us see:

Q:6.9: What is if-else statement and when if-else-statement can be used? Explain with syntax.
[05 marks] [BT-L3]

Definition: If one set of activities have to be performed when an expression is evaluated to *true* and another set of activities have to be performed when an expression is evaluated to *false*, then *if-else-statement* is used. The *if-else-statement* is a simple selection/decision statement that is used when we must choose between two alternatives. Hence, it is also called *two-way decision/selection statement*. The syntax of *if-else statement* along with equivalent flowchart is shown below:

Marks:1



Note: In the syntax used, **if** and **else** are keywords in C language.

Marks:2

WORKING: The sequence of operations that are carried out when if-else statement is executed are shown below:

- ◆ The B-statements that appear before the if-else statement are executed one after the other.
- ◆ The expression immediately after the keyword **if**, is evaluated to **true** or **false**.
- ◆ If the expression is evaluated to non-zero value, it is considered as **true** and the statements T1, T2,Tn are executed. Then, the execution of statements E1 to En are skipped. But, the A-statements after if-else are executed.
- ◆ If the expression is evaluated to zero value, it is considered as **false** and the statements E1, E2,En are executed followed by A-statements. **Note:** Now, the statements T1, T2,...Tn will not be executed.

Marks:2

6. 18 Decision making and branching

Now, let us observe some important points and see what are the various types of errors that can be introduced by the user knowingly or unknowingly. These are:

- 1) The keyword **if** must be followed by an expression and the expression must be enclosed within parentheses. For example,

Ex 1: Syntactically correct

```
if (a > b)      /* OK : expression a > b is within parentheses */
    printf("A is greater\n");
else
    printf("B is greater\n");
```

Ex 2: Syntax error

```
if a > b      /* Error : parentheses missing */
    printf("A is greater\n");
else
    printf("B is greater\n");
```

Ex 3: Syntax error

```
if (a > b      /* Error : right parentheses missing */
    printf("A is greater\n");
else
    printf("B is greater\n");
```

- 2) If multiple statements have to be executed when the expression is true or false, then all those statements must be enclosed within braces. If only one statement has to be executed, then usage of braces is optional.

Ex 1: Usage of braces is mandatory.

```
if (a > b)
{
    printf("A is greater\n");
    exit(0);
}
else
{
    printf("B is greater\n");
    exit(0);
}
```

Note: Both statements *printf()* and *exit()* must be executed. So, they must be enclosed within braces.

Ex 2: Syntax error

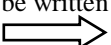
```

1. if (a > b)
2.     printf("A is greater\n");
3.     exit(0);
4. else      /* Error : Syntax error: misplaced else */
5. {
6.     printf("B is greater\n");
7.     exit(0);
8. }

```

Note: The above if-statement starts in line 1 and ends in line 2. So, in line 4 will get an error “Misplaced else” since the keyword **else** is not associated with **if**. To eliminate the syntax error, we must enclose lines 2 and 3 within braces.

Ex 3: Usage of braces is optional in the following example

<pre> if (a > b) { printf("A is greater\n"); } else { printf("B is greater\n"); } </pre>	<p>can be written as</p> 	<pre> if (a > b) printf("A is greater\n"); else printf("B is greater\n"); </pre>
---	--	---

/* Braces are optional */

/* Braces are optional */

3) No semicolon is required for an if-statement. For example, we want to print “A is greater” if value of *a* is greater than *b*; otherwise, print “B is greater”. Let us write the statement as shown below:

<pre> if (a > b) ; printf("A is greater\n"); else printf("B is greater\n"); </pre>	<pre> /* Line 1: OK: semicolon indicates NULL statement meaning “do-nothing” */ /* Line 2 : OK /* Line 3: Error : Misplaced else */ /* Line 4: OK */ </pre>
---	--

Observe the following points:

- ◆ The if-statement ends in line 1 because of NULL statement. No error in line 1
- ◆ No-error in line 2
- ◆ Since if-statement in line 1 ends in line 1 because of NULL statement, we encounter **else** without if-statement in line 3. So, it is an error.

6. 20 Decision making and branching

- ◆ By removing semicolon at the end of line 1, the above error can be corrected as shown below:

Correct:

```
if (a > b)
    printf("A is greater\n");
else
    printf("B is greater\n");
```

Syntax error:

```
if (a > b)   // Line 1 : OK
    printf("A is greater\n"); // Line 2 : OK
else // Line 3 : Error: else without if
    printf("B is greater\n"); // Line 4 : OK
```

Observe the following points:

- ◆ Observe *semicolon* at the end of line 1. The **if** statement in line 1 is terminated.
- ◆ Semicolon at the end of if-statement indicates NULL statement and if statement ends there only.
- ◆ Hence, in line 3 an **else** exists without an if-statement which results in compilation error. So, it can be corrected by removing semicolon in line 1 as shown below:

```
if (a > b) // Line 1 : OK
    printf("A is greater\n"); // Line 2 : OK
else // Line 3 : OK
    printf("B is greater\n"); // Line 4 : OK
```

Consider the following statements:

```
if (marks >= 35) // Line 1 : OK
    printf("Passed\n"); // Line 2 : OK
    printf("Failed\n"); // Line 3 : Logical error
```

There is no syntax error. Hence, compiler will not display any error. Let us execute the above program segment with value of *marks* is 10. The output is:

Failed

which is correct. But, if the value of *marks* is 60, the output is:

Passed
Failed

which is unsatisfactory answer. Observe the following facts:

- ◆ The if-statement controls the execution of only line 2
- ◆ The statement in line 3 is executed always since it is not part of the if statement.
- ◆ After executing line 2, line 3 should not be executed. So, the correct code can be written by inserting the keyword **else** as shown below:

```

if (marks >= 35)           // Line 1 : OK
    printf("Passed\n");    // Line 2 : OK
else
    printf("Failed\n");    // Line 3 : OK

```

6.4.1 Program to check whether the number is even or odd

In this section, let us write a program to check whether the given number is even or odd.

Q:6.10: Design and write a program to check whether the number is even or odd.
[03 marks] [BT-L1]

Design: Let n is the given number. After dividing n by 2, if the remainder is zero, then n is even; Otherwise, n is odd. The equivalent statement can be written as:

```

if ( n % 2 == 0 )
    printf ("Even");
else
    printf ("Odd");

```

Marks:1

Before executing the above statements, we should **input the value of n using the scanf() statement**. The complete program can be written as shown below:

Program 6.6: C program to print even or odd using if-else-statement

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

Marks:2

```
void main()
```

```
{
```

```
    int    n;
```

```
    printf ("Enter the number\n");
    scanf ("%d",&n);
```

```
    if ( n % 2 == 0 )
        printf ("Even");
```

```
    else
        printf ("Odd");
```

```
}
```

TRACING-1

Input

Enter the number 4

Output

Even

TRACING-2

Input

Enter the number 3

Output

Odd

6. 22 Decision making and branching

6.4.2 Program to find larger of 2 numbers

In this section, we design and write a program to find larger of two numbers. Now, let us:

Q:6.11: Design and write a program to find largest of two numbers
[03 marks] [BT-L5]

Statement of the problem: Given two numbers say a and b , we find maximum of these two numbers and print the result.

Design: Let a and b are two numbers. If a is greater than b , then a is bigger; otherwise b is bigger. The equivalent statement can be written as:

```
if (a > b)
    printf("Big = %d\n", a);
else
    printf("Big = %d\n", b);
```

Marks:1

Before executing the above statements, we should **input the value of n using the scanf() statement.**

Marks:2

The complete program can be written as shown below:

Program 6.7: C program to print maximum of two numbers

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

```
void main()
```

```
{
```

```
    int    a, b;
```

```
    printf ("Enter two number\n");
```

```
    scanf("%d %d",&a, &b);
```

```
    /* print larger of two */
```

```
    if (a > b)
```

```
        printf("Big = %d\n", a);
```

```
    else
```

```
        printf("Big = %d\n", b);
```

```
}
```

TRACING-1

Input

Enter 2 numbers

10 5

Output

Big = 10

TRACING-2

Input

Enter 2 numbers

3 25

Output

Big = 25

6.4.3 Check for leap year

In this section, we first see the definition of a leap year, design and write a C program to check for leap year.

Q:6.12: What is a leap year? Design and write a C program to read a year as an input and find whether it is Leap year or not
[04 marks] [BT-L4] [DEC-2016/JAN-2017]

Design: A year is leap year if one of the following two conditions are satisfied.

- ♦ The year is divisible by 4 and should not be divisibly by 100
- ♦ The year is divisible by 400

The code for this can be written as shown below:

```
if ( (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0) )
    printf("%d is a leap year\n", year);
else
    printf("%d is not a leap year\n", year);
```

Marks:2

The complete program can be written as shown below:

Program 6.8: C program to print whether year is leap year or non-leap year

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

Marks:2

```
void main()
{
    int year;

    printf("Enter the year\n");
    scanf("%d",&year);

    if ( (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0) )
        printf("%d is a leap year\n", year);
    else
        printf("%d is Non-leap year\n", year);
}
```

Output 1:

Enter the year
2000
Leap year

Output 2:

Enter the year
2004
Leap year

Output 3:

Enter the year
3000
Non-leap year

6. 24 Decision making and branching

The following table shows whether a year is a leap year or not along with the reasons.

Year	Case 1 satisfied (true/false)		Case 2 satisfied (true/false)	Remarks
	divisible by 4	not divisible by 100	Divisible by 400	
2000	TRUE	FALSE	TRUE	2 nd case satisfied. So, leap year
2004	TRUE	TRUE	–	1 st case satisfied So, leap year
3000	TRUE	FALSE	FALSE	Both the conditions are not satisfied. So, it is not leap year

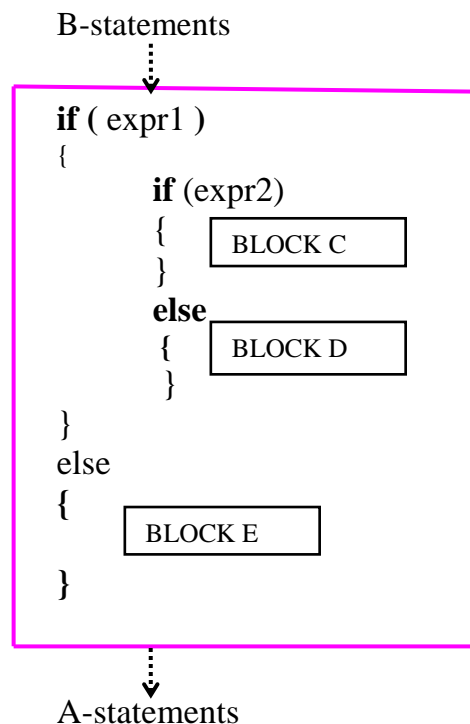
6.5 The nested if-statement

In this section, we discuss the syntax and programs using nested if-statement.

Q:6.13: What is a nested-if-statement? Explain with syntax.
[04 marks] [BT-L3]

Definition: An *if* or *if-else* statement within another *if* or *if-else* statement is called “*nested if statement*”. When an action has to be performed based on many decisions involving various types of expressions and variables, then this statement is used. The syntax of *nested-if* is shown on the right hand side where

- ◆ The B-statements before the if-statement are executed one after the other
- ◆ Then, the outermost if-statement is executed.
- ◆ If the *expr1* fails, the statements in BLOCK E are executed. After executing these statements, control comes out of the if-statement and A-statements are executed.
- ◆ If the *expr1* is evaluated to **true**, the next inner if-statement is executed. The Cond2 is checked for **true** or **false**. If it is evaluated to **true**, then the statements



in BLOCK C are executed otherwise, the statements in BLOCK D are executed. After executing the inner *if-else* (whether the condition is true or false), control comes out and A-statements are executed. **For example**, consider the statement:

```

if ( salary >= 10000)
{
    if ( sex == 'M')
        print "Pay Tax"
    else
        print "No tax"
}

```

In this program segment, the if-else statement is inside the if-statement and hence the above statement is called ***nested-if statement***. Here, if salary >= 10000 and sex is male then the output "Pay Tax" is obtained. If salary is >= 10000 and if sex is female, the output will be "No tax".

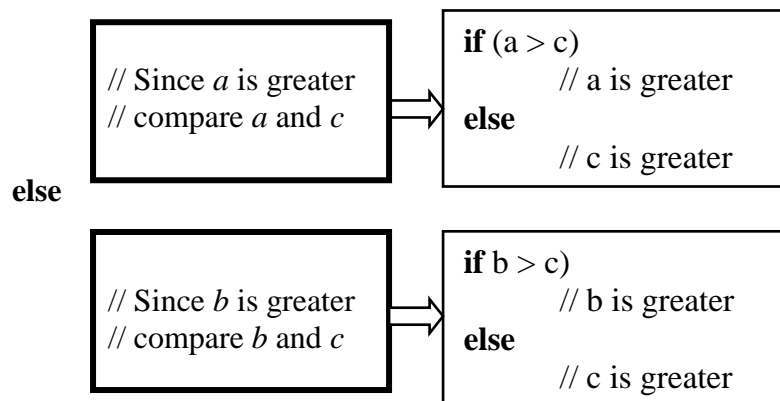
6.5.1 To find the largest of 3 numbers

In section, we discuss the method of finding the largest of 3 numbers using nested-if statement.

Q:6.14: Design and develop a program to find largest of 3 numbers using nested-if
[04 marks] [BT-L1]

Design: We can always compare two numbers. Now, let us compare two numbers at a time and find largest of three numbers. This can be done using nesting as shown below:

if (a > b)



6. 26 Decision making and branching

Using the above logic, the partial code can be written as shown below:

```
if (a > b)           // find maximum of a and c
{
    if (a > c)
        printf("Max = %d\n", a);
    else
        printf("Max = %d\n", c);
}
else                // find maximum of b and c
{
    if (b > c)
        printf("Max = %d\n", b);
    else
        printf("Max = %d\n", c);
}
```

Marks: 2

Before executing the above statements, we should input the values for *a*, *b* and *c* using the `scanf()` statement. The complete program can be written as shown below:

Program 6.9: C program to find maximum of 3 numbers using nested-if-statement

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

Marks: 2

```
void main()
{
```

```
    int    a, b, c;
```

```
    printf("Enter the value of a, b and c\n");
    scanf("%d %d %d",&a, &b, &c);
```

```
    if (a > b)
        if (a > c)
            printf("Max = %d\n", a);
        else
            printf("Max = %d\n", c);
```

```
    else
        if (b > c)
            printf("Max = %d\n", b);
        else
            printf("Max = %d\n", c);
```

```
}
```

TRACE1	TRACE2	TRACE3	TRACE4
<u>Input</u>	<u>Input</u>	<u>Input</u>	<u>Input</u>
a b c	a b c	a b c	a b c
40 20 30	4 2 10	3 6 5	3 5 9
<u>Output</u>	<u>Output</u>	<u>Output</u>	<u>Output</u>
Max = 40			
	Max=10		
		Max=6	
			Max=9

6.5.2 Advantages and disadvantages

In this section, let us see the advantages and disadvantages of nested-if statement.

Q:6.15: What are the advantages and disadvantages of nested-if statement?
[03 marks] [BT-L1]

Advantages

The nested-if is used in following situations:

- ◆ There are situations involving series of decisions where we are forced to use an *if* or *if-else* in another *if* or *if-else* statement. In such situations, *nested-if-statements* are used.
- ◆ When an action has to be performed based on many decisions involving various types of expressions and variables, then this statement is used

Disadvantages

- ◆ As depth of nesting increases, the readability of the program decreases.
- ◆ Hence, as depth increases, it is difficult to read and understand the logic of the program.

6.6 The else-if-ladder statement

If a selection is based on a range of values, then a slight variation of nested if called *else-if* is used. But, there is no C construct like *else-if*. But, it is only a style of writing *if-else* only in the *else* part of the *if-else* statement. This is termed as *else-if-ladder*.

Q:6.16: What is else-if ladder statement? Explain with syntax.
[04 marks] [BT-L3]

Definition: An *else-if ladder* is a special case of *nested-if* statement where nesting take place only in the else part. When an action has to be selected based on range of values, then this statement is used. So, it is called *multi-way decision/selection statement*. The orderly nesting of if-else-statement only in the else part is called *else-if-ladder*.

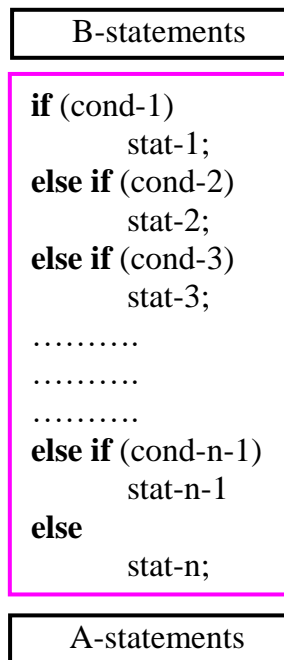
Marks:1

The syntax of *else-if-ladder* along with equivalent flowchart is shown below:

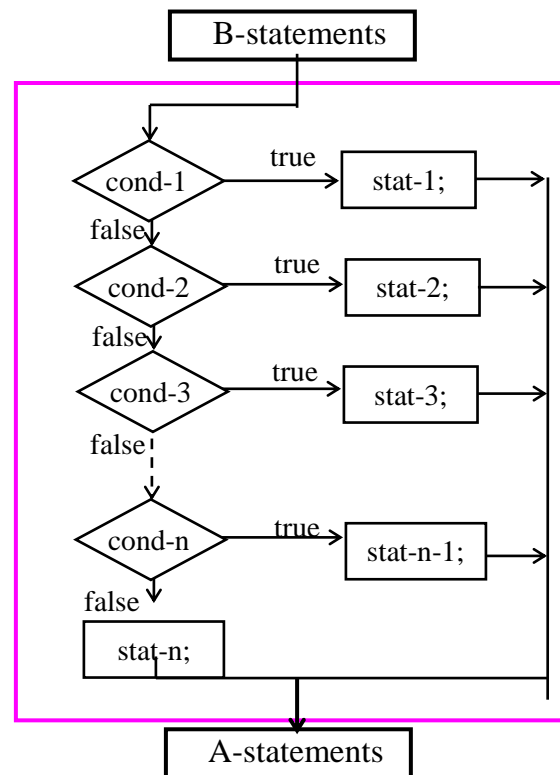
6. 28 Decision making and branching

Marks:2

C-Syntax



Flowchart



The following points are observed from the above flowchart:

Marks:1

- ◆ The B-statements before the else-if-ladder are executed one after the other. Then, the conditions *cond-1*, *cond-2* etc., of *else-if-ladder* are evaluated one after the other as long as the conditions are *false*. Once the condition is evaluated to *true*, the corresponding statement is executed and control comes out of the entire *else-if-ladder* and execute the A-statements which comes after the else-if-ladder.
- ◆ If all the conditions are evaluated to *false*, the last statement *stat-n* is executed. Then control comes out of else-if-ladder and A-statements are executed.

6.6.1 Grading based on the marks

Now, let us design and write a program using else-if-ladder. Now, the problem can be stated as shown below:

Q:6.17: Write a program to display the grades based on the marks scored by a student.

<u>Marks</u>	<u>Grades</u>
0 to 39	F (Fail)
40 to 49	E
50 to 59	D
60 to 69	C
70 to 79	B
80 to 89	A
90 to 100	O (Out standing)

[03 marks] [BT-L2]

It is very straight forward and the equivalent program is shown below:

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

    printf("Enter the marks\n");
    scanf("%d",&marks);

    if (marks <= 39)
        printf("Grade F\n");
    else if (marks <= 49)
        printf("Grade E\n");
    else if (marks <= 59)
        printf("Grade D\n");
    else if (marks <= 69)
        printf("Grade C\n");
    else if (marks <= 79)
        printf("Grade B\n");
    else if (marks <= 89)
        printf("Grade A\n");
    else
        printf("Grade 0 –Out standing");
}
```

6.6.2 To find roots of quadratic equation

The algorithm and flowchart is already discussed in section 6.3.5. The C program to find the roots of quadratic equation is shown below:

6. 30 Decision making and branching

Example 6.13: C program to find roots of quadratic equation using else-if-ladder

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

void main ()
{
    float a, b, c, x1, x2, disc;

    printf("Enter the co-efficients\n");
    scanf ("%f %f %f", &a, &b, &c);

    disc = b*b - 4*a*c;    /* find the discriminant */

    if (disc > 0)    /* Find the distinct roots */
    {
        x1 = (-b + sqrt(disc)) / (2*a);    3.0
        x2 = (-b - sqrt(disc)) / (2*a);    2.0
        printf("The roots are distinct\n");
        printf("X1 = %f\nX2 = %f\n",x1,x2);    X1 = 3.0 X2 = 2.0
    }
    else if (disc == 0)    /* Find the equal roots */
    {
        x1 = x2 = -b/(2*a);
        printf("The roots are equal\n");
        printf("X1 = %f\nX2 = %f\n",x1, x2);    X1 = X2 = 2.00
    }
    else
    {
        /* Find the complex roots */
        x1 = -b/(2*a);
        x2 = sqrt(fabs(disc))/(2*a);
        printf("The roots are complex\n");
        printf("first root = %f + i%f\n",x1,x2);    First root = -0.167 + i0.8
        printf("second root = %f - i%f\n",x1,x2);    Second root = -0.167 - i0.8
    }
}
```

6.6.3 Simulation of a simple calculator

Now, let us see how the else-if ladder can be used to perform various arithmetic operations.

Q:6.18: Write a C program that reads from the user an arithmetic operator and two operands, perform the corresponding arithmetic operation on the operands using else-if ladder. **[06 marks] [BT-L4] DEC-2016/JAN-2017**

	TRACE1	TRACE2	TRACE3	TRACE4
Input	5+6	6/0	5*6	6/4
Output	res = 11			
Input			5*6	
Output			res=30	
Input				6/4
Output				res=1
Input				
Output		div by 0		
Input				
Output				
Input				
Output				
Input				
Output	5+6=11		5*6=30	6/4 = 1

6. 32 Decision making and branching

Note: In the above function, we have used the function `exit(0)` to terminate the program. Its declaration is available in the header file “`process.h`”. Hence, it is included in the beginning of the program.

6.6.4 Finding the largest of 3 numbers using else-if-ladder

We have already seen how to write the program for the given problem in earlier sections 6.3.3 and 6.5.1. Now, let us:

Q:6.19: Design and develop a program to find largest of three numbers using else-if ladder [04 marks] [BT-L5]

Design: Given a , b and c , if a is greater than b and c , then we say a is maximum. If b is greater than a and c , then we say b is maximum. Otherwise, c is maximum. This can be written using C statement as shown below:

```
if (a > b && a > c)
    printf("Max = %d\n", a);
else if (b > a && b > c)
    printf("Max = %d\n", b);
else
    printf("Max = %d\n", c);
```

Marks:2

The program can be written as shown below:

Program 6.10: C program to find maximum of 3 numbers using else-if-ladder

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

```
void main()
{
```

```
    int    a, b, c;

    printf("Enter the value of a, b and c\n");
    scanf("%d %d %d",&a, &b, &c);

    if (a > b && a > c)
        printf("Max = %d\n", a);
    else if (b > a && b > c)
        printf("Max = %d\n", b);
    else
        printf("Max = %d\n", c);
}
```

Marks:2

<u>Input</u>	<u>Input</u>	<u>Input</u>	<u>Input</u>
a b c	a b c	a b c	a b c
40 20 30	4 2 10	3 6 5	3 5 9
Max=40		Max=6	
	Max = 10		

6.6.5 Check for the type of triangle (else-if-ladder)

Statement of the problem: Given 3-sides of a triangle, check whether a triangle can be formed or not? If a triangle can be formed, check what type of triangle it is? It can be an equilateral triangle or an isosceles triangle or scalene triangle? The given triangle can be right-angled triangle also. If so print the appropriate messages.

Q:6.20: Design and develop a program to check for the type of triangle
[08 marks] [BT-L5]

Design: Let a , b and c are the 3 sides of the triangle where a has largest length, b and c as second largest and third largest. Now, we will have the following conditions:

- | | |
|----------------------------|----------------------|
| 1. No triangle | if $a \geq b + c$ |
| 2. A right angled triangle | if $a^2 = b^2 + c^2$ |
| 3. An obtuse triangle | if $a^2 > b^2 + c^2$ |
| 4. An acute triangle | if $a^2 < b^2 + c^2$ |

The code for the above four cases can be written as shown below:

```
if ( a >= b + c )
    printf("No triangle can be formed");
else if (a*a == b*b + c*c)
    printf("Right angled triangle");
else if (a*a > b*b + c*c)
    printf("Obtuse triangle");
else
    printf("Acute triangle");
```

At the same time we have

- | | |
|-------------------------|---------------------------------------|
| 5. Equilateral triangle | if all sides of the triangle are same |
| 6. Isosceles triangle | if two sides of triangle are same |
| 7. Scalene triangle | Otherwise |

The code for the above four cases can be written as shown below:

```
if ( a == b && b == c)
    printf("Equilateral triangle");
else if ( a == b || b == c || c == a)
    printf("Isosceles triangle");
else
    printf("Scalene triangle");
```

Marks:4

The complete program can be written as shown below:

6. 34 Decision making and branching

Program 6.11: C program to find the type of the triangle

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

Marks:4

```
void main()
{
```

```
    float a, b, c;
```

```
    printf("Enter 3-sides of triangle with a as highest, b & c as next highest");
    scanf("%f %f %f", &a, &b, &c);
```

```
    /* Identify for right angle, obtuse and acute trinagle */
```

```
    if ( a >= b + c )
        printf("No triangle can be formed");
    else if (a*a == b*b + c*c)
        printf("Right angled \n");
    else if (a*a > b*b + c*c)
        printf("Obtuse \n");
    else
        printf("Acute \n");
```

```
    /* Identify for equilateral, isosceles or scalene*/
```

```
    if ( a == b && b == c)
        printf("Equilateral");
    else if ( a == b || b == c || c == a)
        printf("Isosceles \n");
    else
        printf("Scalene\n");
```

```
}
```

Output1

Enter 3-sides
10 10 10
Acute
Equilateral

Output2

Enter 3-sides
4 3 5
Right angled
scalene

Output3

Enter 3-sides
8 10 8
Acute
Isosceles

Output4

Enter 3-sides
1 20 5
No triangle

6.7 The switch statement

In this section, let us see another important branching statement called *switch statement* along with its syntax. Now, the question is:

Q:6.21: What is a switch statement? Explain with syntax.

[05 marks] [BT-L3]

Definition: The *switch* statement is a control statement used to make a selection between many alternatives. When we use switch statement, the selection condition must reduce to an integer value i.e., the choice can either be any integer value or a character. The choice can also be an expression which results in an integer value. Based on this integer value, the control is transferred to a particular *case value* where necessary statements are executed.

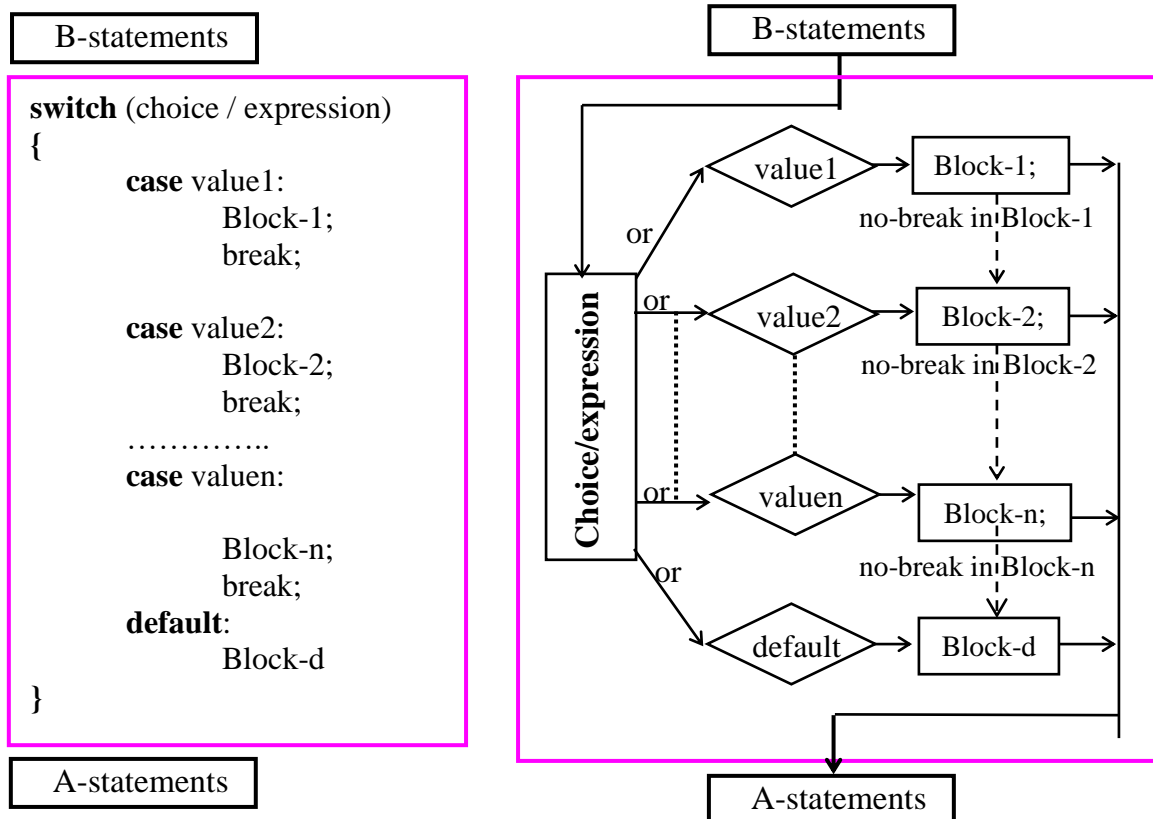
Marks:1

The syntax of *switch statement* is shown below:

Marks:2

C-Syntax

Flowchart



The switch statement works as shown below:

- ◆ After executing the B-statements sequentially, the expression in *switch* is evaluated to an integer value.

6. 36 Decision making and branching

- ◆ The integer value thus obtained if it matches with any of the values *value1*, *value2*,*valuen*, the control is transferred to the appropriate block.
- ◆ During execution of a block of statements, if *break* statement is executed, then the control comes out of the *switch* statement and the A-statements which comes after the *switch* statement are executed.
- ◆ During execution of a particular *case*, if *break* is not encountered, then control goes to the subsequent *cases* and the statements under those cases will be executed till the *break statement* is encountered.
- ◆ If the value of the expression does not match with any of the case values *value1*, *value2*,*valuen* then, control goes to *default* label.
- ◆ If the value of the expression does not match with any of the case values *value1*, *value2*,*valuen* and *default* label is not there, then, control comes out of the switch statement and A- statements following switch will be executed.

Marks:2

Note: The switch statement is used when a decision has to be made between many alternatives and when the selection condition reduces to fixed integer value. The switch statement cannot be used when a series of decision/condition involve a logical or relational expressions.

Q:6.22: What are the rules to be followed while using switch-statement?
[03 marks] [BT-L1]

Now, let us see The various rules are shown below:

- ◆ The expression that follows the keyword *switch* must be evaluated to an integer value.
 - Ex 1: `switch(choice)` // valid if *choice* is an integer variable
 - Ex 2: `switch (i+2)` // valid if *i* is an integer variable
 - Ex 3: `switch (i+2.5)` // invalid even if *i* is an integer variable. Because,
// the expression will not be evaluated to integer
- ◆ The expression that follows the keyword *case* should not contain any variables: For example,
 - `case 10+2:` // valid
 - `case i+2:` // invalid: Since variable *i* is present
- ◆ Two or more case labels with same value are not allowed. For example,
 - `case 1: printf("Hello");`
 `break;`
 - `case 1: printf("computer");` // Invalid: Case 1 is already defined
 `break;`

- ◆ Two or more case labels can be associated with same statements. For example,
- ```

case 1:
case 2:
case 3: printf("Hello"); // valid

```

After evaluation, if the expression in *switch* is reduced to either 1 or 2 or 3, the statement:  
 printf("Hello");

is executed. Some valid and invalid ways of writing switch expressions and case constants are shown below by assuming the following declarations:

```

int i;
char c;
float f;
double d;

```

| Program segment                                | Valid/<br>invalid | Reasons for invalidity                                                                                                                                          |
|------------------------------------------------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| switch(i)                                      | valid             |                                                                                                                                                                 |
| switch(i + 10)                                 | valid             |                                                                                                                                                                 |
| switch(i + 5.5)                                | invalid           | Floating point numbers not allowed                                                                                                                              |
| switch (c + 10)                                | valid             |                                                                                                                                                                 |
| switch (c + d)                                 | invalid           | d is double. double/float variables are not allowed                                                                                                             |
| case 4:                                        | valid             |                                                                                                                                                                 |
| case 4                                         | invalid           | Colon missing after 4                                                                                                                                           |
| case4:                                         | invalid           | Space must be present between <i>case</i> and 4. It is not a syntax error (Compiler will not treat it as error). It is a logical error and hence it is invalid. |
| Case 4:                                        | invalid           | Case is not a keyword ( <b>case</b> is a keyword)                                                                                                               |
| case +:                                        | invalid           | + should be enclosed between two single quotes                                                                                                                  |
| case "+":                                      | invalid           | Strings are not allowed.                                                                                                                                        |
| case 'A':                                      | valid             |                                                                                                                                                                 |
| case 'A'-'B':                                  | valid             | Integer expressions with constants are allowed.                                                                                                                 |
| case i + 2:                                    | invalid           | Variables are not allowed in case                                                                                                                               |
| case "A":                                      | invalid           | Strings are not allowed in case                                                                                                                                 |
| case 'choice':                                 | invalid           | one character is allowed within two single quotes.                                                                                                              |
| case 1:2:3:                                    | invalid           | There should be separate case for each of 1, 2, 3                                                                                                               |
| case 1:<br>case '2':<br>case 3: printf("mon"); | valid             | Two or more case labels can be associated with same statements                                                                                                  |

## 6. 38 Decision making and branching

### 6.7.1 Simulation of simple calculator

The simulation of calculator discussed in section 6.6.3 can be written using switch statement as shown below:

**Q:6.23:** Write a program to simulate the operations of a calculator using switch statement. [05 marks] [BT-L5]

#### PROGRAM

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

void main()
{
 float a, b, res;
 char op;

 printf("Enter the expression.\n");
 scanf("%f %c %f",&a, &op, &b);

 switch (op)
 {
 case '+':
 res = a + b;
 break;
 case '-':
 res = a - b;
 break;
 case '*':
 res = a * b;
 break;
 case '/':
 res = a / b;
 }
 printf("%f %c %f = %f\n",a, op, b, res);
}
```

#### TRACE1 TRACE2 TRACE3 TRACE4

| <u>Input</u> | <u>Input</u> | <u>Input</u> | <u>Input</u> |
|--------------|--------------|--------------|--------------|
| 5+6          | 6/2          | 5*6          | 6/4          |
| res = 11     |              |              |              |
|              |              | res=30       |              |
|              | res=3        |              | res=1.5      |
| 5+6=11       |              | 5*6=30       | 6/4 = 1.5    |

#### Advantages

- ◆ Improves readability of the program
- ◆ More structured way of writing the program

**Disadvantages**

- ♦ Used only if the expressions used for checking the conditions results in integer value (char is also allowed). If the result of expression is not integer value, *switch* statement cannot be used.
- ♦ Cannot be used when a selection is based on a range of values.

**Note:** When **break** statement is executed, the control comes out of the switch statement. The detailed explanation is given in section 7.6.2.

**6.8 Un-conditional branch statements**

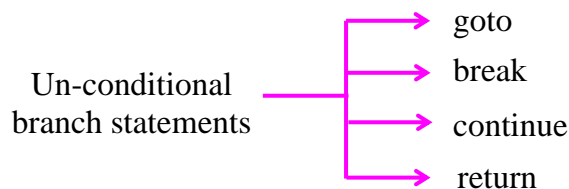
In this section, we discuss about another set of branch statements that transfer the control to different part of the program without any conditions. They are called unconditional branch statements. Now, let us see:

---

**Q:6.24:** What are unconditional branch statements? Explain different types of unconditional branch statements.  
[02 marks] [BT-L1]

---

**Definition:** The sequential statements are executed one after the other. However, we can give instructions to transfer the control from one statement to some other statement during execution of the program. These statements that transfer the control from the one statement to other statement in the program without any condition are called *un-conditional branching statements* or *un-conditional control statements*. The unconditional statements are classified as shown below:

**6.8.1 The goto statement**

In this section, we discuss about goto statement and see how to write programs using goto statement.

---

**Q:6.25:** What is goto? Give the syntax. Write a program to find sum of  $n$  natural numbers using goto statement.  
[05 marks] [BT-L2]

---

## 6. 40 Decision making and branching

---

The sequential statements are executed one after the other. But, if the programmer wants to transfer the control from one point to some other point in the program, the *goto* statement can be used. Using this statement, control can be transferred from one statement to the specified statement without any condition (unconditionally).

**Syntax:** The syntax of the *goto* is  
*goto label;*

where *label* is any identifier. The rules that are applicable to form a variable name are applicable for formation of label also. But, the labels need not be declared. The label should be used along with a statement to which the control is transferred as shown below:

*label: statement;*

The program to find sum of first 10 natural numbers can be written as shown below:

```
void main()
{
 int sum = 0, i = 0;

 top: if (i > 10) goto end;

 sum = sum + i;
 i++;

 goto top;

 end: printf("Sum = %d\n", sum);
}
```

In the above program, the statements  
sum = sum + i;  
i++;

will be executed as long as the value of *i* is less than or equal to 10. Once the value of *i* is greater than 10, control goes to the statement labeled *end* and the sum of first 10 integers will be displayed.

### Disadvantages

Many programmers avoid the usage of *goto* statement because of following reasons

- ◆ Using *goto*, the code is difficult to read and understand.
- ◆ The usage of *goto* results in *unstructured programming*.
- ◆ It is not good programming style

**Note:** Let us avoid *goto* statements in all our subsequent programs.



**6.8.2 The break statement**

[Discussed in next chapter – section 7.6.2]

**6.8.3 The continue statement**

[Discussed in section 7.6.3]

**6.8.4 The return statement**

[Discussed in detail in chapter on functions]

**6.9 Software engineering and programming style**

All programs must be indented for better readability and understanding. It is a good programming style. Now, let us see:

**Q:6.26: What is indentation? [BT-L1]**

**Definition:** Indentation is nothing but formatting the source program by inserting appropriate white space characters such as spaces, tabs and newline characters in the program so as to improve the readability of the program. It is one of the techniques of good programming style. The various advantages of using indentation are shown below:

- ◆ If the program is properly indented, it is easy for the programmer to read and understand the program.
- ◆ It helps us to understand the flow of control.
- ◆ It helps us to identify the mistakes in the program.

For example, consider the program to find largest of two numbers. The first program is without indentation and second program with good indentation.

**Poor programming style****Good programming style**

|                                                                                                                                                                                                               |                                                                                                                                                                                                                                       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>#include &lt;stdio.h&gt;  /* function to find largest */ void largest(int a, int b) {     if (a &gt; b)         printf("A is greater\n");     else         printf("B is greater\n");     return; }</pre> | <pre>#include &lt;stdio.h&gt;  /* function to find the largest of 2 no.s */ void largest (int a, int b) {     if (a &gt; b)         printf("%d is greater\n", a);     else         printf("%d is greater\n", b);      return; }</pre> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 6. 42 Decision making and branching

|                                                                                                                                    |                                                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>void main() { int m,n;  printf("Enter two numbers\n");      scanf("%d %d",&amp;m, &amp;n);     largest(m,n);  return; }</pre> | <pre><b>void</b> main() {     <b>int</b> m, n;      printf("Enter two numbers\n");     scanf("%d %d",&amp;m, &amp;n);      largest(m,n);      <b>return</b>; }</pre> |
|------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Observe the following points from above two programs:

- ◆ Type the above two programs. Both give the same result.
- ◆ The first program looks awkward and very difficult to read and understand whereas the second program looks neat and we can easily read and understand the program.

Now, let us see “What are indentation rules to be followed while writing the program?”

- 1) Insert a space before the operator and after the operator. For example,

**Poor programming style**

c=a+b;

**Good programming style**

c = a + b;

**Note:** In poor programming style there is no space before and after the operator whereas in good programming style, a space is inserted before and after the operator.

- 2) Align **else** with corresponding **if** i.e., the keyword **if** and corresponding **else** should be in same column. For example,

**if** ( a > b )

.....

**else**

.....

**Exercises**

- 1) What are branching statements? Explain the different types of branching statements? [03 marks] [BT-L1]
- 2) What is an if-statement? Explain with syntax. [04 marks] [BT-L1]
- 3) Design and write a program to identify whether the number is even or odd [04 marks] [BT-L5]
- 4) Design and write a program to display maximum of two numbers. [04 marks] [BT-L5]
- 5) Design and write a program to find maximum of three numbers [04 marks] [BT-L5]
- 6) Design and write a program to find whether a given number is +ve or -ve or zero. [04 marks] [BT-L5]
- 7) Design and write a program to find the root of the quadratic equation [10 marks] [BT-L5]
- 8) What are the advantages and disadvantages of if-statement? [BT-L1]
- 9) What is if-else statement and when if-else-statement can be used? Explain with syntax. [05 marks] [BT-L3]
- 10) Design and write a program to check whether the number is even or odd. [03 marks] [BT-L1]
- 11) Design and write a program to find largest of two numbers [03 marks] [BT-L5]
- 12) What is a leap year? Design and write a C program to read a year as an input and find whether it is Leap year or not [04 marks] [BT-L4] [DEC-2016/JAN-2017]
- 13) What is a nested-if-statement? Explain with syntax. [04 marks] [BT-L3]
- 14) Design and develop a program to find largest of 3 numbers using nested-if [04 marks] [BT-L1]
- 15) What are the advantages and disadvantages of nested-if statement? [03 marks] [BT-L1]
- 16) What is else-if ladder statement? Explain with syntax. [04 marks] [BT-L3]
- 17) Write a program to display the grades based on the marks scored by a student.
 

| <u>Marks</u> | <u>Grades</u>    |
|--------------|------------------|
| b. 0 to 39   | F (Fail)         |
| c. 40 to 49  | E                |
| d. 50 to 59  | D                |
| e. 60 to 69  | C                |
| f. 70 to 79  | B                |
| g. 80 to 89  | A                |
| h. 90 to 100 | O (Out standing) |
- 18) Write a C program that reads from the user an arithmetic operator and two operands, perform the corresponding arithmetic operation on the operands using switch statement. [06 marks] [BT-L4] DEC-2016/JAN-2017

#### 6. 44 Decision making and branching

---

- 19) Design and develop a program to find largest of three numbers using else-if ladder [04 marks] [BT-L5]
- 20) Design and develop a program to check for the type of triangle [08 marks] [BT-L5]
- 21) What is a switch statement? Explain with syntax. [05 marks] [BT-L3]
- 22) What are the rules to be followed while using switch-statement? [03 marks] [BT-L1]
- 23) Write a program to simulate the operations of a calculator using switch statement. [05 marks] [BT-L5]
- 24) What are unconditional branch statements? Explain different types of unconditional branch statements. [02 marks] [BT-L1]
- 25) What is goto? Give the syntax. Write a program to find sum of  $n$  natural numbers using goto statement. [05 marks] [BT-L2]
- 26) What is indentation? [BT-L1]