


Decision Making and Branching

Unit 4

INTRODUCTION

In the previous lessons we have learned about the programming structure, data types, declaration of variables, tokens, constants, keywords and operators as well as how to write program in C and how to execute program. In this unit we will teach about another feature of C. In this unit we will also learn how to make a decision in program and how to execute this concept.

<p>Timeframe</p>  <p>How long ?</p>	<p>We expect that this unit will take maximum 10 hours to complete.</p>
--	---

<p>Unit Structure</p>	
<p>Lesson- 1 : Basic concept on decision making statements</p> <p>Lesson- 2 : Decision making using if and if-else statements</p> <p>Lesson- 3 : Decision making using nesting of if-else statements</p> <p>Lesson- 4 : Decision making using else-if ladder statements</p> <p>Lesson- 5 : Decision making using switch case statements</p>	

Lesson-1

Basic Concept on Decision Making Statements

Learning Outcomes



Outcomes

Upon completion of this lesson you will be able to

- Understand basic idea about decision making statements.

	Keywords	Decision making, Statements, Control, Execution
--	-----------------	--

PRELIMINARIES

We have already seen that, basically a C program is a set of statements which are usually executed successively. But in practice, we have a number of situations where we may have to change the order of execution of statements based on certain conditions, or replicate a group of statements until certain specified circumstances are met.

The general form of a standard decision making structure found in most of the programming languages like C is as follows:

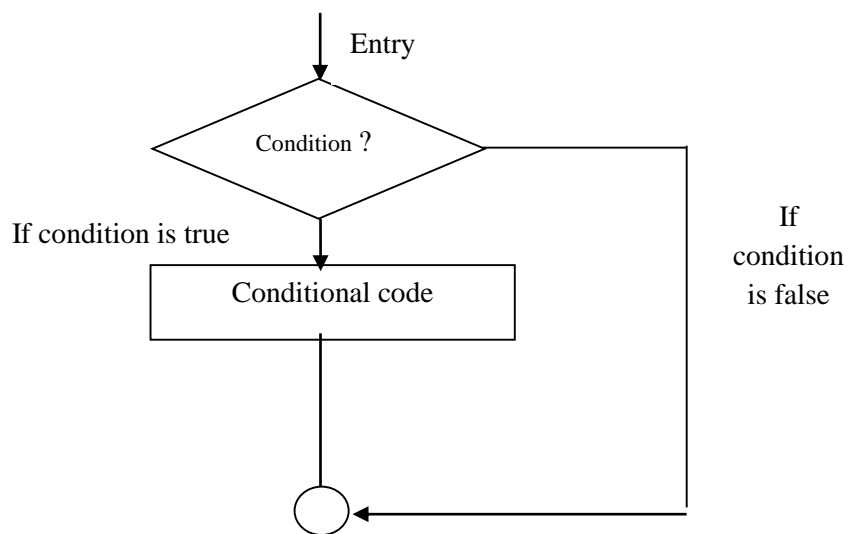


Figure 4.1.1 General form of decision making structure

C language handles decision-making by supporting the following statements:

- i. if statement
- ii. if-else statement
- iii. nesting if-else statement
- iv. else-if ladder statement
- v. switch case statement




Note it!

We have a number of situations where we may have to change the order of execution of statements based on certain conditions

The above decision making statements are shortly described here, and described in details in next lessons.

- **if statement:** *if* statement is used to control the flow of execution of statements. It allows evaluate the condition or expression first and then, depending on whether the value of the expression is true or false.
- **if-else statement:** *if-else* statement is used to carry out a logical test and then take one of two possible actions depending on the result of the test.
- **Nesting if-else statement:** *nesting if-else* statements are used, when a series of decisions are concerned, we may have to employ more than one if-else statement.
- **else-if ladder statements:** *else-if ladder statements* are used, when multipath decisions are concerned. A multipath decision is a chain of ifs in which the statement associated with each *else* is an *if*.
- **switch case statement:** *switch* statement checks the value of a given expression against a list of *case* values and when a match is found a block of statements associated with the *case* is executed.

<p>Summary</p>  <p>Summary</p>	
<p>In this lesson</p> <ul style="list-style-type: none"> ▪ We have understood about C language decision making statements. 	

Assessment



Assessment

Fill in the blanks

1. *if* statement is used to of execution of statements.
2. A multipath decision is a chain of.....
3. *if-else* statement is used to carry out a test.

Multiple Choice Questions (MCQ)

1. C language handles decision-making by supporting-
 - a) 3 statements b) 4 statements c) 5 statements d) 6 statements
2. *Nesting if-else* statements are used-

<ol style="list-style-type: none"> a) When a series of decisions are concerned c) No decision are allowed 	<ol style="list-style-type: none"> b) When a single of decision is concerned d) None of these
---	---

Exercises

1. What do you mean by control statement?
2. Explain shortly various decision making statements.

Lesson-2 Decision Making Using if and if-else Statements


Learning Outcomes



Outcomes

Upon completion of this lesson you will be able to

- Understand about the if statement.
- Understand about the if-else statement.

	Keywords	Statement, If, If-else, Execution, Expression, Condition
---	-----------------	---

SIMPLE IF STATEMENT

The simple *if* statement is an influential decision making statement. It is used to control the flow of execution of statements. On other hand, it is also a two-way decision making statement.

It is also used to evaluate the expression or condition first and then, depending on whether the value of the expression is true or false.

The general structure of a simple *if* statement is as follows:

```

if ( Condition/expression )
{
    Statement-block;
    .....
}
Statement-n;
```

Here, *statement-block* can be single or a group of statements. If condition/expression is true then, the *statement-block* will be executed, otherwise the *statement-block* will be omitted and the execution will jump to the *statement-n*.

In this situation, it is remembered that, when the condition is true both the *statement-block* and *statement-n* are executed in sequence.

The general flow chart of *if* statement is shown in below:

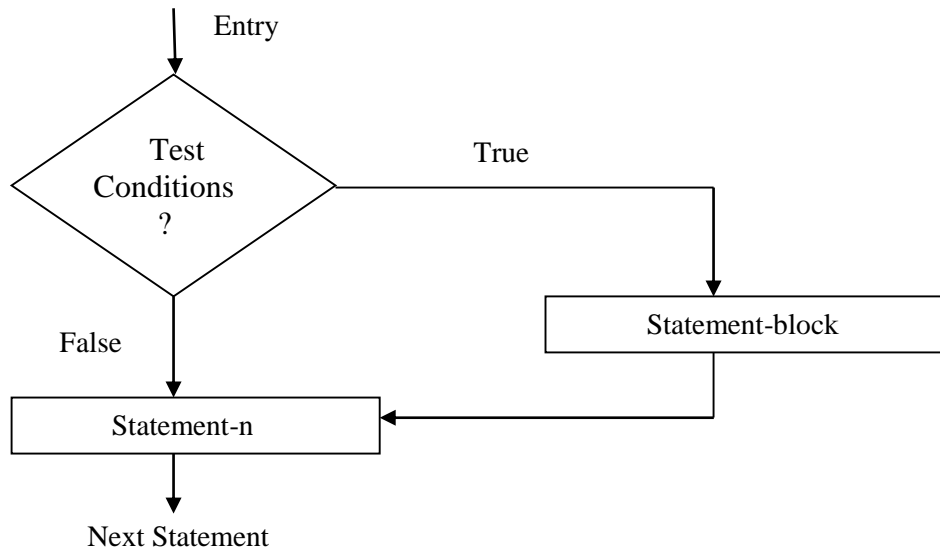


Figure 4.2.1: Flowchart of simple *if* control statement

Consider the following program segment:

```

void Main()
{
    .....
    .....
    if( N= =100)
    {
        N=N+50;
    }
    printf(“%d”, N);
    .....
    .....
}
    
```

In the above program segment, the program checks the value of *N*. If the value of *N* is 100 then extra 50 are added with *N* before printed, otherwise the value of *N* will be printed.



Note it!

The if statement is used to evaluate the expression or condition first and then, depending on whether the value of the expression is true or false

Program 4.2.1 Write a program to display a number if user enters negative number. If user enters positive number, that number won't be displayed.

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int num;
    printf("Enter Integer Number:");
    scanf("%d",&num);
}
    
```

```

if(num < 0)
{
printf("You entered %d. \n", num);
}

printf("Have a nice Day!!");
getch();
}
.....
Output
Enter Integer Number:-10
You entered -10.
Have a nice Day!!
Enter Integer Number: 50
Have a nice Day!!

```

Program 4.2.2 Write a program to display salary of an employee, where if salary is less than 20000 taka then add home allowance 5000 taka with basic salary, otherwise display salary without home allowance.

```

#include<stdio.h>
#include<conio.h>
void main()
{
long int salary;
printf("Enter Basic salary of an employee:");
scanf("%ld",&salary);

if(salary < 20000)
{
salary=salary+5000;
}

printf("Employee Salary is:%ld\n",salary);
getch();
}
.....
Output
Enter Basic salary of an employee: 10000
Employee Salary is:15000

Enter Basic salary of an employee: 30000
Employee Salary is:30000

```

SIMPLE IF-ELSE STATEMENT

The *if-else* statement is an extension of the simple *if* statement. This type of statement is used to carry out a logical test and then take one of the two possible actions depending on the result of the test. The general structure of *if-else* statement is as follows:

```

if (test condition/expression)
{

```

```

        True statements-block;
    }
    else
    {
        False statements-block;
    }
    statement- n;

```

The *if-else* statement executes some code if the test condition or expression is true and some other code if the test expression or condition is false.

Here, if the test condition or expression is true (non-zero) then true statements-block is executed otherwise false (zero) statements-block is executed. It is remembered that, in *if-else* statement, either true statements-block or false statements-block will be executed, not both. In both cases the control is transferred subsequently to statement-n.

The general flow chart of *if-else* statement is shown in below:

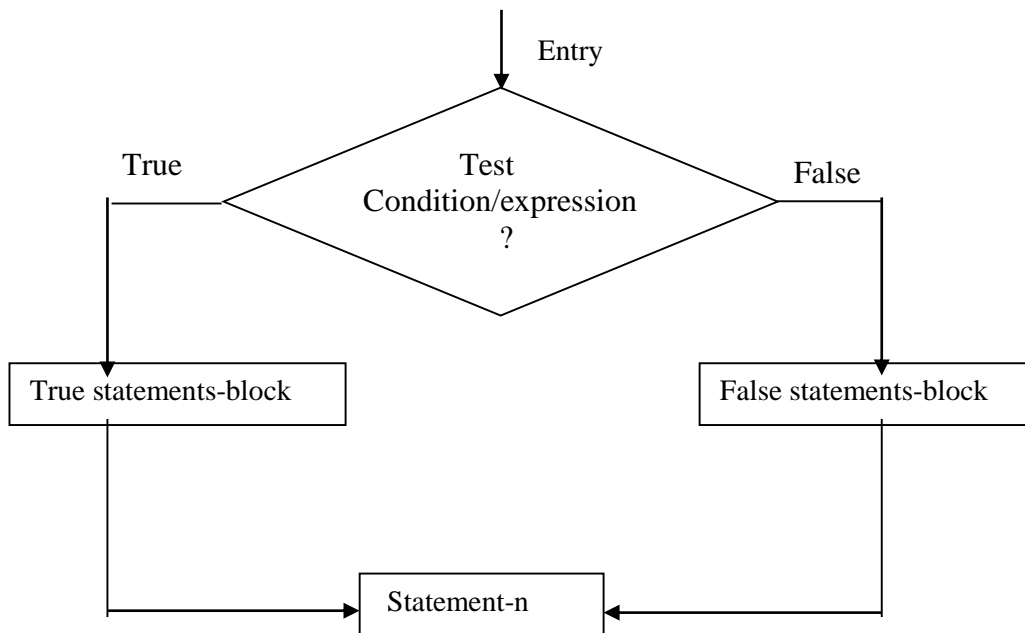


Figure 4.2.2: flowchart of *if-else* control statement

Consider the following program segment:

```

void main()
{
    .....
    .....
    if(test= = 1)
    {
        scanf("%f",&radius);
        Area=3.14159*radius*radius;
        printf("Area of Circle is:= %f\n",Area);
    }
    else

```

```

        {
            scanf("%f %f",&base,&height);
            Area=0.5*base*height;
            printf("Area of traingle is:= %f\n",Area);
        }
    }

```

In the above example, if the variable *test* value is only 1 then true statement block of *if* will be executed and print the area of circle, otherwise false statement block of *else* will be executed and print the area of triangle.

Consider another programming example is as follows:

```

void main()
{
    int age;

    printf(" Enter your real age:");
    scanf("%d",&age);

    if(age >= 18)
    {
        printf("you are eligible for voting.");
    }
    else
    {
        printf("you are not eligible for voting.");
    }
}

```

In the above example, user enter his or her age from keyboard, if the variable *age* value is greater than or equal 18 years then, true statement block of *if* will be executed and print ‘**you are eligible for voting**’, otherwise false statement block of *else* will be executed and print ‘**you are not eligible for voting**’.



Note it!

The if-else statement executes some code if the test condition or expression is true and some other code if the test expression or condition is false.

Program 4.2.3 write a program that checks a number is either EVEN or ODD.

```

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

void main()
{
    clrscr();
    int number;

    printf("Enter a integer number:");
    scanf("%d",&number);
}

```



```
if(number%2==0)
{
    printf("Given Number is EVEN.\n");
}
else
{
    printf("Given Number is ODD.\n");
}
getch();
}
```

.....

Output

Enter a integer number: 100
Given Number is EVEN.
Enter a integer number: 15
Given Number is ODD

Program 4.2.4 Write a program to print employee monthly salary, where, if basic salary is grater that 20000 taka then 40% home allowance of basic salary and 1500 taka medical allowance are added with basic salary, otherwise 55% home allowance and 1000 taka medical allowance are added with basic salary.

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

void main()
{
    clrscr();
    long int bsalary,msalary;

    printf("Enter Employee Basic salary: ");
    scanf("%ld",&bsalary);

    if(bsalary > 20000)
    {
        msalary=bsalary+bsalary*0.4+1500;
        printf("Employee Monthly Salary is: %ld",msalary);
    }
    else
    {
        msalary=bsalary+bsalary*0.55+1000;
        printf("Employee Monthly Salary is: %ld",msalary);
    }

    getch();
}
```

.....

Output

Enter Employee Basic salary: 30000
Employee Monthly Salary is: 43500
Enter Employee Basic salary: 20000
Employee Monthly Salary is: 32000



Study skills

1. Determine the output of the following program segment

```
void main( )
{
.....
.....
Test= 50;
if( Test>=10)
{
    printf(“%d”,Test);
}
Test= Test+100;
printf(“%d”,Test);
}
```

2. Determine the output of the following program segment

```
void main( )
{
.....
.....
Test= 50;
if( Test>=10)
{
    printf(“%d”,Test);
}
else
{
    Test= Test+100;
    printf(“%d”,Test);
}
}
```

3. Find out the errors from the following program segments:

a.

 If(x+y=p && z=1)
 Printf(“ ”);

b.

 if(code>1)
 {
 a=b+c
 else
 a=0
 }

4. The following is a segment of a program:

```
X=1;
Y=1;
if(n>0){
    X=X+1;
    Y=Y-1;
}
printf(“%d %d”,X,Y);
```


What will be the values of X and Y if n=1 and n=0.



Activity

1. Write a program to check you are either YOUNG or OLD.

2. Write a program to find the number of and sum of all integers greater than 100 and less than 200 that are divisible by 7.

Summary  Summary	
In this lesson <ul style="list-style-type: none"> ▪ We have gathered knowledge about C control statements. ▪ Moreover we have learned that how <i>if</i> statement works in program. ▪ We also have understood that how <i>if-else</i> statement works in program. 	

ASSIGNMENT



Assignment

1. Mention the main differences between *if* and *if-else* statements.

2. Write a program to check whether a year is leap year or not.

Assessment



Assessment

Write “T” for true and “F” for false for the following sentences

1. The *if-else* statement is an extension of the simple *if* statement.
2. *if* statement is a single-way decision making statement.
3. The *if-else* statement executes some code if the test condition is true and some other code if the test condition is false.

Multiple Choice Questions (MCQ)

1. In the *if-else* statement, if the test condition or expression is true then-

a) true statements-block is executed	b) false statements-block is executed
c) both true and false statements are executed	d) none of these
2. *if* statement is-

a) two-way decision making	b) single way decision making statement
c) no way decision making statement	d) none of these

Exercises

1. What do you mean by control statement?
2. Differentiate between *if* and *if-else* conditional statements with an example.
3. Explain *if* control statement with flowchart and an example.
4. Explain *if-else* control statement with flowchart and an example.

Lesson-3**Decision Making Using Nesting of if-else Statements****Learning Outcomes****Outcomes**

Upon completion of this lesson you will be able to

- Explain nesting if-else statements.

**Keywords****Statement, If-else, Execution, Expression, Nesting****NESTING OF IF-ELSE STATEMENTS**

In the previous lesson, we have seen that, the *if-else* statement executes only two different codes depending upon whether the test expression/condition is true or false. However, in practical, sometimes, a choice has to be made from more than two possibilities.

When sequences of decisions are concerned, we may have to apply more than one if-else statement in nested form. Nested *if-else* statement is similar to if-else statement, where new block of *if-else* statement is defined in existing *if* or *else* block statement. The nested *if-else* statement allows us to verify for multiple test expressions and execute different codes for more than two conditions.

The general structure of nesting of *if-else* statement is as follows:

```

if(test expression-1)
{
    if(test expression-2)
    {
        block of statement-1;
    }
    else
    {
        block of statement-2;
    }
else
{
    block of statement-3;
}
statement-n;

```

The logic execution flow chart is demonstrated in figure 4.3.1.

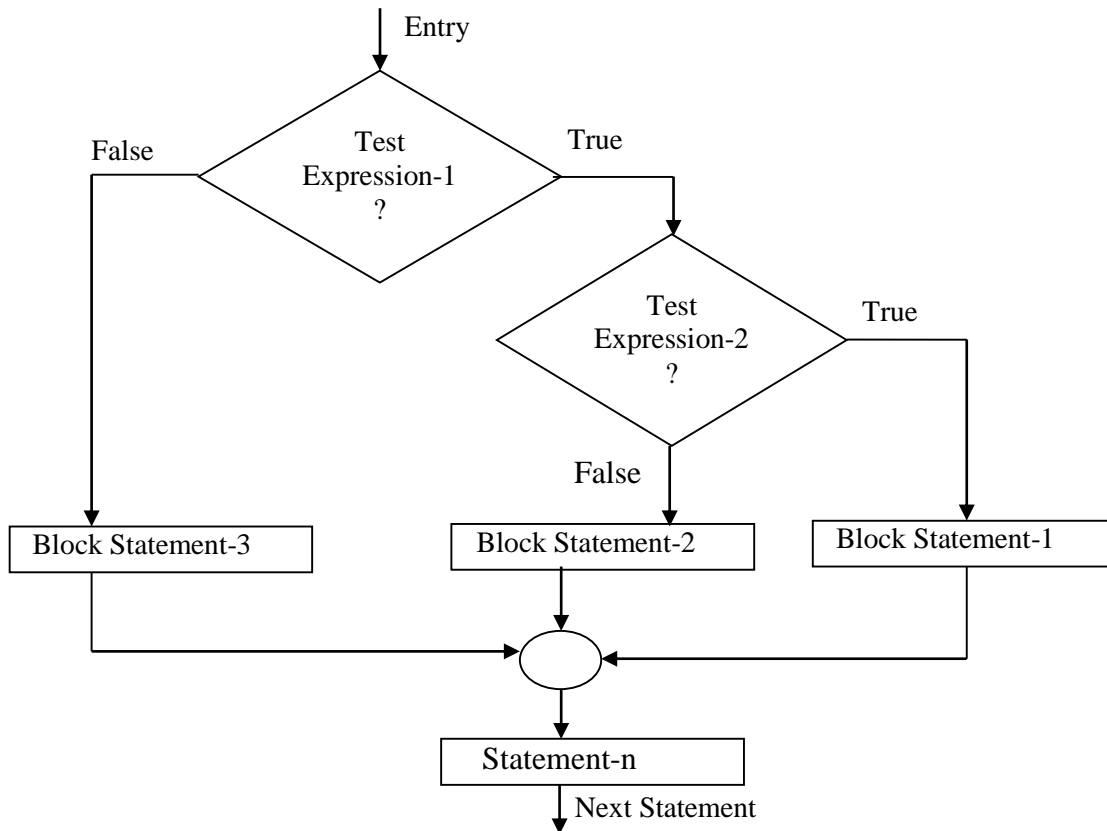


Figure 4.3.1: flowchart of nested *if-else* statements

From the flowchart, we can say that, if the *test expression-1* is false the block statement-3 will be executed; otherwise it continues to perform the second test expression. If the *test expression-2* is true the block statement-1 will be executed; otherwise the block statement-2 will be executed and then the control goes to the statement-n and so on.



Note it!

Nested if-else statement is similar to if-else statement, where new block of if-else statement is defined in existing if or else block statement.



Activity

1. Mention the significance of using nesting if-else by your own concept.

2. Write the nested if-else statements syntax with five if and three else.

Program 4.3.1 Write a program to display largest number from three integer numbers using nested if-else statements.

```

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

void main()
{
    clrscr();
}
    
```

```
int num1,num2,num3;

printf("Enter three integer numbers:\n ");
scanf("%d %d %d",&num1,&num2,&num3);

if(num1 > num2)
{
    if(num1 > num3)
    {
        printf("Largest Number is: %d\n", num1);
    }
    else
    {
        printf("Largest number is: %d\n", num3);
    }
}
else
{
    if(num3 > num2)
    {
        printf("Largest Number is: %d\n",num3);
    }
    else
    {
        printf("Largest Number is: %d\n",num2);
    }
}

getch();
}
.....
Output
Enter three integer numbers:
20
400
100
Largest Number is: 400
```

Program 4.3.2 Write a program to check username and password for login using nested if-else statements.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char username;
    int password;
    clrscr();
    printf("Enter Your Username:");
    scanf("%c",&username);
    printf("Enter Your Password:");
    scanf("%d",&password);
```

```
if(username == 'a')
{
    if(password == 12345)
    {
        printf("Login successful");
    }
    else
    {
        printf("Password is incorrect, Try again!!!");
    }
}
else
{
    printf("Username is incorrect, Try again!!!");
}
getch();
}
```

Output

```
Enter Your Username: am
Enter Your Password:1234
Username is incorrect, Try again!!!"
```

```
Enter Your Username: a
Enter Your Password:12345
Login successful
```

Program 4.3.3 Write a program to check monthly balance of a customer of a bank, where if customer sex is female and current balance is greater than 10000 taka, then bonus will be 10% of current balance, and if current balance is less than 10000 taka bonus will be 5%, otherwise bonus will be 2%.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char sex;
    long int balance,bonus,mbalance;
    clrscr();
    printf("Enter your sex:");
    scanf("%c",&sex);
    printf("Enter your balance:");
    scanf("%ld",&balance);
    if(sex=='F')
    {
        if(balance>10000)
        {
            bonus=balance*.10;
        }
        else
        {
            bonus=balance*.05;
        }
    }
}
```

```

    }
    else
    {
        bonus=balance*.02;
    }

    mbalance=balance+bonus;
    printf("Monthly Balance will be: %ld",mbalance);
    getch();
}
.....
Output
Enter your sex: M
Enter your balance: 12000
Monthly Balance will be: 12240

Enter your sex: F
Enter your balance: 15000
Monthly Balance will be: 16500
    
```



Study skills

1. Identify the errors form the following program segments and correct them :

```

void main()
{
    int x,y,z;
    x=10;y=5,z=6;

    if(x<=10);
    {
        if(y>5)
            y=x++;
        else
            if(z = =6 && x=10)
                z=x+y;
            else
                x=x++;
    }
    printf("%d\n",&x)
    printf("%d\n",&y);
}

    printf("%d\n",&z);
}
    
```

2. After correction the above program what will be the output?

Summary



Summary

In this lesson

- We have gathered knowledge about C control statements.
- Moreover we have learned that how nesting *if-else* statement works in program.

ASSIGNMENT



Assignment

1. Write down the benefits of nesting if-else in C program.
.....
.....

2. Determine the roots of the quadratic equation $ax^2 + bx + c = 0$ using the well known quadratic formula:

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

Assessment



Assessment

Write “T” for true and “F” for false for the following sentences

1. When sequences of decisions are concerned, we may have to apply more than one if-else statement in nested form.
2. Nested *if-else* statement allows us to verify for multiple test expressions.

Exercises

1. Define nesting if-else statements.
2. What are the benefits of using nesting if-else statements?
3. Draw and explain the flow graph of nesting if-else statements.
4. Write a C program to determine maximum and minimum number from three integer numbers using nesting if-else statements.

Lesson-4**Decision Making Using else-if Ladder Statements****Learning Outcomes****Outcomes**

Upon completion of this lesson you will be able to

- Explain else-if ladder statements.
- Explain else-if ladder general syntax.
- Explain else-if ladder flowchart.

**Keywords**

Statement, If-else ladder, Execution, Flowchart

ELSE-IF LADDER STATEMENTS

In practical, when we need to test different conditions with different statements, where multipath decisions are involved, this is called else-if ladder. Actually multipath decision is a sequence of *ifs* in which the statements related with each *else* is an *if*. **If-else ladder** statement is used to test set of conditions in sequence that is if we have different test conditions with different statements then we need else-if ladder statements. The general syntax of if-else ladder is as follows:

```

if (test_condition-1)
{
    statement-1;
}
else if(test_condition-2)
{
    statement-2;
}
else if(test_condition-3)
{
    statement-3;
}
else if(test_condition-4)
{
    statement-4;
}
.....
else if(test_condition-n)
{
    statement-n;
}
else
{
    default-statement;
}
Statement-X;

```

The above construction is known as if-else ladder. Here, the conditions are evaluated from top to bottom. As soon as when a test condition is true, then the statement(s) associated with it is executed and the programming control is transferred to the *statement-X*. When all conditions are failed or false then the final *else* containing the *default-statement* will be executed. The flowchart of if-else ladder is shown in below:

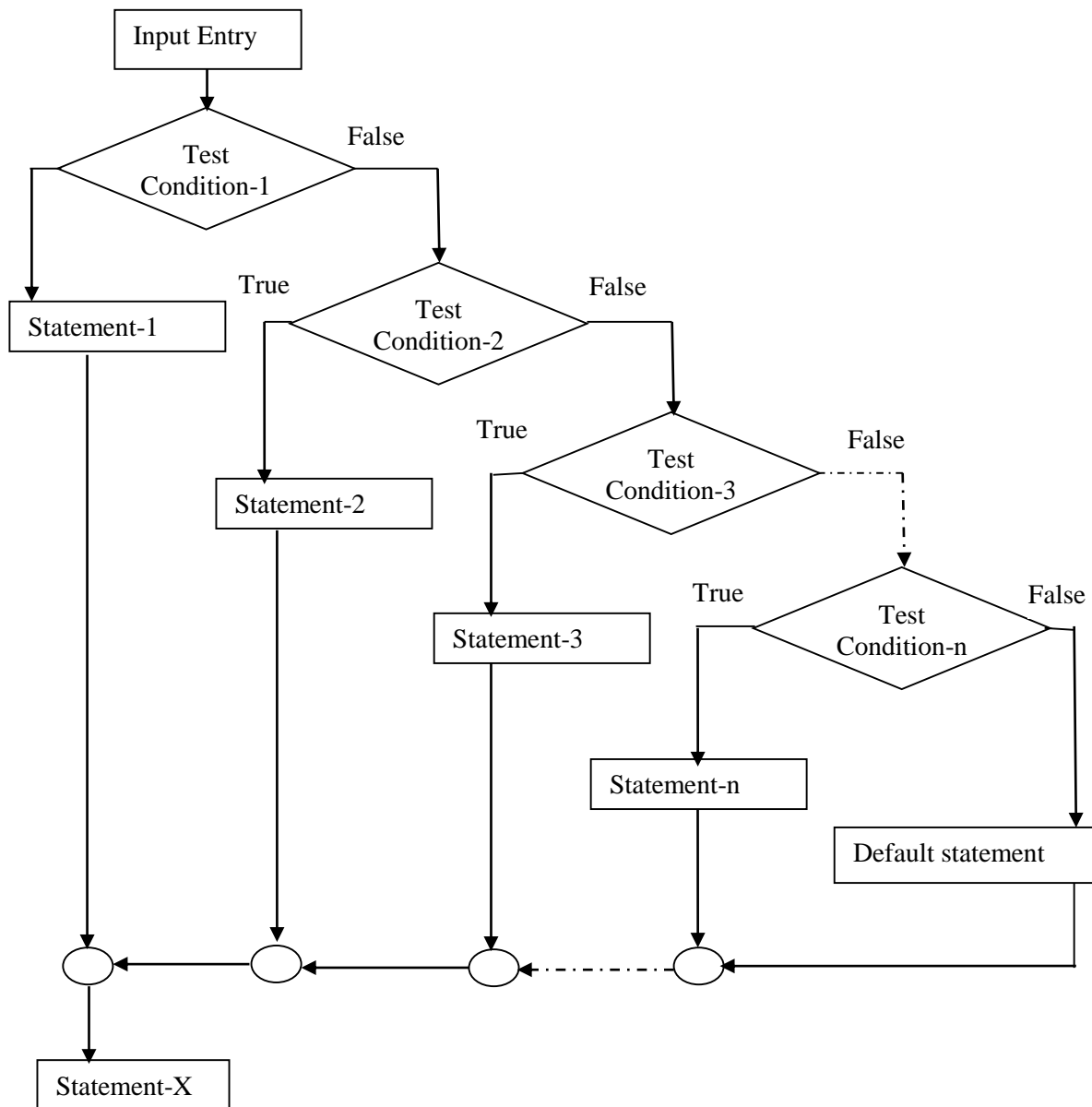


Figure 4.4.1: Flowchart of if-else ladder



Note it!

If-else ladder statement is used to test set of conditions in sequence that is if we have different test conditions with different it statements then we need else-if ladder statements

Program 4.4.1 Assume an example of grading system of the students in an institution. The grading is done according to the following rules:

Obtained marks	Grade	Obtained marks	Grade
80 to 100	A+	55 to less than 60	B
75 to less than 80	A-	50 to less than 55	C+
70 to less than 75	A	45 to less than 50	C
65 to less than 70	B+	40 to less than 45	D
60 to less than 65	B-	Less than 40	F

Now write a C program to Calculate students' grade using if-else ladder concept.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int marks;
    printf("Enter your Marks:");
    scanf("%d",&marks);
    if(marks>100)
    {
        printf("\nInvalid Marks. Marks must be less than 100");
    }
    else
    {
        if(marks<=100 && marks>=80)
        {
            printf("Your Grade is A+");
        }
        else if(marks<80 && marks>=75)
        {
            printf("Your Grade is A-");
        }
        else if(marks<75 && marks>=70)
        {
            printf("Your Grade is A");
        }
        else if(marks<70 && marks>=65)
        {
            printf("Your Grade is B+");
        }
        else if(marks<65 && marks>=60)
        {
            printf("Your Grade is B-");
        }
        else if(marks<60 && marks>=55)
        {
            printf("Your Grade is B");
        }
        else if(marks<55 && marks>=50)
        {
            printf("Your Grade is C+");
        }
        else if(marks<50 && marks>=45)
        {
```

```

    printf("Your Grade is C");
}
else if(marks<45 && marks>=40)
{
    printf("Your Grade is D");
}
else
{
    printf("Your Grade is F");
}
}
getch();
}

```

Output

Enter your Marks: 78

Your Grade is A-

Enter your Marks: 120

Invalid Marks. Marks must be less than or equal 100

Program 4.4.2 Assume that DESCO power distribution company charges its domestic customers as follows:

Consumption Units	Rate of charge	Consumption Units	Rate of charge
0-100	1.00 taka per unit	401-500	3.50 taka per unit
101-200	1.50 taka per unit	501-600	4.70 taka per unit
201-300	2.00 taka per unit	601-700	5.00 taka per unit
301-400	2.75 taka per unit	701 or above	8.50 taka per unit

Now write a C program that reads power consumed unit and prints the amount to be paid by the customer.

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int units;
    float pa_amount;
    printf("Enter Consumed Units:");
    scanf("%d",&units);
    if(units>701)
    {
        pa_amount=units*8.50;
        printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
    }
    else
    {
        if(units<=100 && units>=0)
        {
            pa_amount=units*1.00;
            printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
        }
        else if(units<=200 && units>=101)
        {

```

```

    pa_amount=units*1.50;
    printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
}
else if(units<=300 && units>=201)
{
    pa_amount=units*2.00;
    printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
}
else if(units<=400 && units>=301)
{
    pa_amount=units*2.75;
    printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
}
else if(units<=500 && units>=401)
{
    pa_amount=units*3.50;
    printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
}
else if(units<=600 && units>=501)
{
    pa_amount=units*4.70;
    printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
}
else if(units<=700 && units>=601)
{
    pa_amount=units*5.00;
    printf("\n Customer Total Payable amount is: %f Taka",pa_amount);
}
else
{
    printf("Illigible Customer!!!!");
}
}
getch();
}
.....
Output
Enter Consumed Units:800
Customer Total Payable amount is: 6800.00000 Taka
Enter Consumed Units:300
Customer Total Payable amount is: 600.00000 Taka


```



Activity

1. Mention the significance of using if-else ladder statement by your own concept.

2. Write the if-else ladder statement syntax with 10 conditions.

<p>Summary</p>  <p>Summary</p>	
<p>In this lesson</p> <ul style="list-style-type: none"> ▪ We have gathered knowledge about C control statements. ▪ Moreover we have learned that how <i>if-else ladder</i> statement works in program. ▪ We also have understood that how <i>if-else ladder flowchart</i> works in program. 	

ASSIGNMENT



Assignment

1. Suppose A Firm has 100 employee. The basic salary range and house rent and bonus of different categories employees are given bellow:

Basic salary	House rent	Bonus
Greater than 40000	35% of basic salary	20% of basic salary
30000 to less than 40000	40% of basic salary	15% of basic salary
20000 to less than 30000	45% of basic salary	12% of basic salary
10000 to less than 20000	50% of basic salary	10% of basic salary
Less than 10000	55% of basic salary	8% of basic salary

Now write a C program to calculate total salary of an employee, where total salary is the sum of basic salary, house rent and bonus.

Assessment



Assessment

Write “T” for true and “F” for false for the following sentences

1. If-else ladder statement is used to test set of conditions in sequence.
2. When all conditions are failed or false then the final *else* containing the *default-statement* will not be executed.

Exercises

1. What are the benefits of using nesting if-else ladder statements?
2. Draw and explain the flow graph of if-else ladder statements.

Lesson-5**Decision Making Using Switch Case Statement****Learning Outcomes****Outcomes**

Upon completion of this lesson you will be able to

- Explain switch case statement.
- Explain switch case general syntax.
- Explain switch case flowchart.

**Keywords****Statement, Switch, Case, Execution, Expression, Flowchart****SWITCH CASE STATEMENT**

In previous lesson we have seen that the nested if-else statement allows us to execute a block of statements or code among many conditions or alternatives. If we are checking on the value of a single variable in nested if –else statement, in this situation it is better to use switch case statement. It is frequently faster than nested if-else but not always. Fortunately, C has built-in multi-way decision statement known as switch statement. A switch statement tests the value of a given *variable or expression* against a list of case values. In this statement, when case is matched or found then block of statements associated with that case is executed. This type of statement is mostly used when we have number of options or choices or cases and we may need to perform a different task for each choice or case. The general syntax of switch case statement is as follows:

```

switch (variable or expression)
{
    case Constant-1:
        block-code-1;
        break;
    case Constant-2:
        block-code-2;
        break;
    case Constant-3:
        block-code-3;
        break;
    .....
    .....
    case Constant-n:
        block-code-n;
        break;

    default:
        default block;
        break;
}

```


Statement-X;

In the above syntax, the **expression** is an integer expression or characters. Constant-1, Constant-2, Constant-3.... Constant-n are known as **case labels**. Here, each of these values must be unique within switch statement and bock-code-1, bock-code-2, bock-code-3... bock-code-n are statement lists. When **switch** is executed the variable or value of the expression is successively compared against the cases. If a **case** is found or matched with the value of the expression, then the associated block code or statement(s) are executed.

The **break** statement at the end of the each block code means the end of specific case and causes an exit from the **switch** statement, and for that reason, transfers program control to statement-X. If the **break** statement is missed from any case, then program executes all statements until get break statement from another case. The default is optional. When variable or value of expression does not match with any of the case values then default statement(s) are executed.

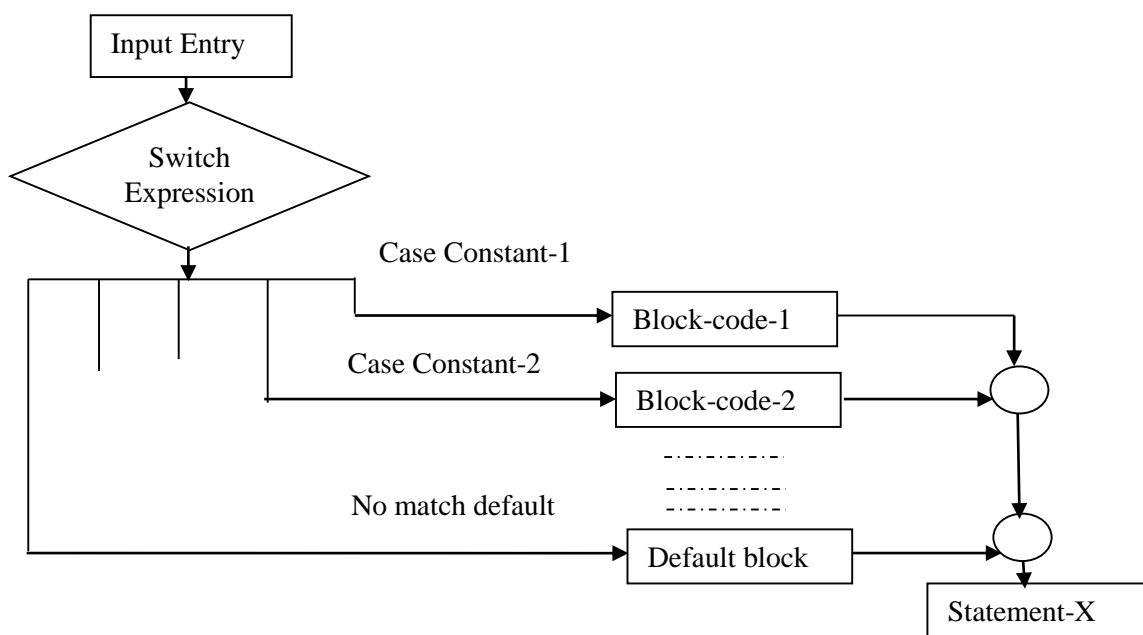


Figure 4.5.1: Flowchart of switch case statement



A switch statement tests the value of a given variable or expression against a list of case values. When case is matched or found then block of statements associated with that case is executed.



1. Mention the significance of using switch case statement by your own concept.

2. Write the switch case syntax with 10 cases.

Program 4.5.1 Assume an example of grading system of the students in an institution. The grading is done according to the following rules:

Obtained marks	Grade	Obtained marks	Grade
80 to 100	A+	50 to less than 60	C
70 to less than 80	A-	40 to less than 50	D
60 to less than 70	B	Less than 40	F

Now write a C program to Calculate students' grade using Switch case concept.

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

    int marks, value;

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


    value =marks/10;
    switch(value)
    {

        case 10:
        case 9:
        case 8:
            printf("Well done Your grade is A+");break;
        case 7:
            printf("Well done Your grade is A-");break;
        case 6:
            printf("Well done Your grade is B");break;
        case 5:
            printf("Well done Your grade is C"); break;
        case 4:
            printf("Well done Your grade is D");break;
        default:
            printf("Sorry Your grade is F!!");
    }
    getch();
}
```

.....

Output

```
Enter the marks: 95
Well done Your grade is A+
Enter the marks: 66
Well done Your grade is B
Enter the marks: 33
Sorry Your grade is F!!"
```

 Summary	
In this lesson <ul style="list-style-type: none"> ▪ We have gathered knowledge about basic idea about switc case statement. ▪ Moreover we have learned that how <i>switch case</i> statement works in program. 	

ASSIGNMENT



Assignment

Suppose A Firm has 100 employee. The basic salary range and house rent and bonus of different categories employees are given bellow:

Basic salary	House rent	Bonus
Greater than 40000	35% of basic salary	20% of basic salary
30000 to less than 40000	40% of basic salary	15% of basic salary
20000 to less than 30000	45% of basic salary	12% of basic salary
10000 to less than 20000	50% of basic salary	10% of basic salary
Less than 10000	55% of basic salary	8% of basic salary

Now write a C program to calculate total salary of an employee, where total salary is the sum of basic salary, house rent and bonus using switch case statement.

Assessment



Assessment

Fill in the blanks with appropriate word

1. C has built-in decision statement known as..... statement.
2. When *switch* is executed the is successively compared against the cases.
3. In switch case statement the is optional.
4. Astatement tests the value of a given.....against a list of case values.

Exercises

1. What is switch in C program? Why is switch case statement used in C program?
2. Draw and explain the flowchart of switch case statement.
3. Briefly explain the general syntax of Switch case statement.
4. In what ways does a switch statement differ from an if statement?