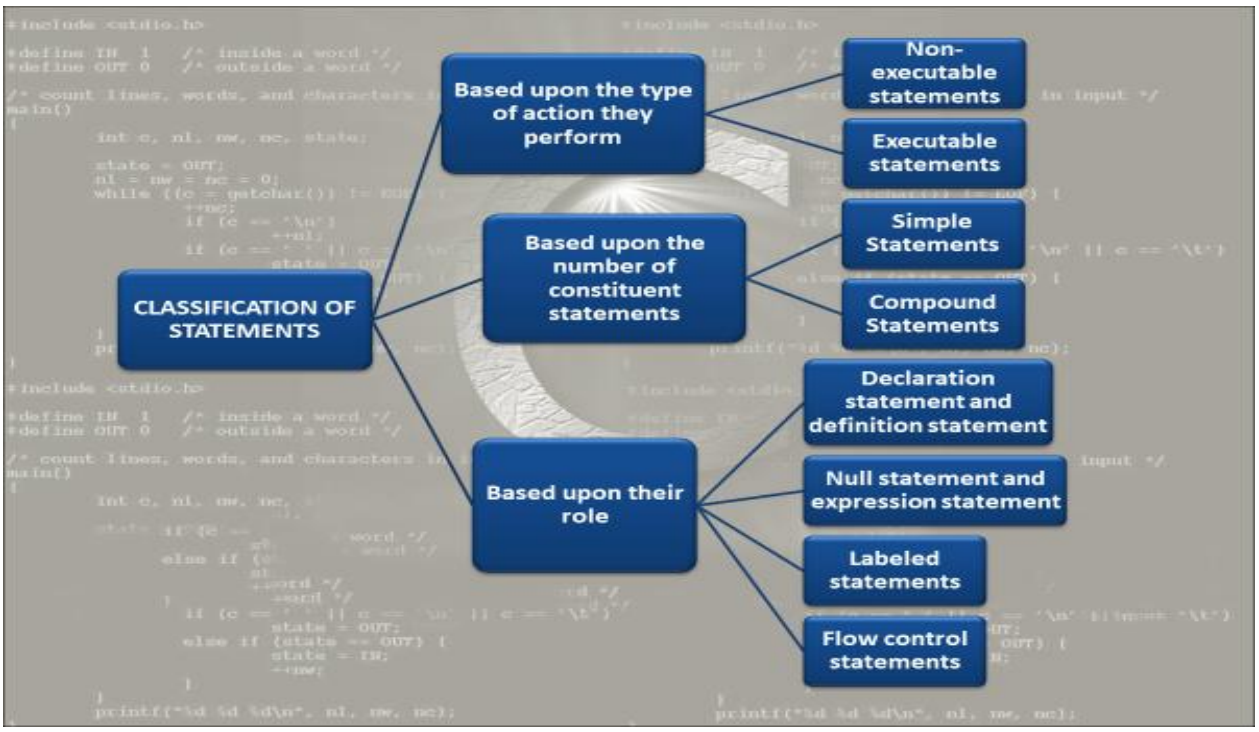
STATEMENTS

A statement is the smallest logical entity that can independently exist in a C program. No entity smaller than statement i.e. expressions, variable names, constants etc. can independently exist in a C program unless and until they are converted into statements.

For example, a=2+3 is an expression. When it is terminated with a semicolon, it forms an expression statement i.e. a=2+3;.



Non-Executable statements:

- These statements help the compiler to determine how to allocate memory, interpret and compile other statements in a program.
- These statements are intended mainly for compiler and no machine code is generated for them. Only executable statements play a role during the execution of a program.
- The order in which non-executable statements appear in a program is important. When a compiler compiles a program, it scans all the statements from top to bottom. A non-executable statement can only affect the statements which appear below it. Thus, ***a non-executable statement should appear only before executable statements within a block.***
- Only non-executable statements can appear outside the body of a function .
- Examples of non-executable statements are: function prototypes, global variable declarations, constant declarations and preprocessor directive statements.

Executable statements:

- Executable statements represent the instructions which are to be performed when the program is executed. The following are the important points about executable statements:
- For an executable statement, some machine code is generated by the compiler.
- Executable statement can appear only inside the body of a function.
- The examples of executable statements are: assignment statements, branching statements, looping statements, function call statements etc.

Simple statements:

- **Simple statement** consists of a single statement. Simple statement terminates with a semicolon. Following are the examples of simple statements:
- int variable=10; //← definition statement
- variable+5; //← expression statement
- variable=variable+10; //← assignment statement

Compound statements:

- Compound statement consists of a sequence of simple statements enclosed within a pair of braces . The following is an example of a compound statement:

```
{                          //← a compound statement consisting of three simple statements
int variable=10;
variable=variable*2;
variable+=5;
}
```

- ✓ A compound statement is also known as **block**.
- ✓ A compound statement need not be terminated with semicolon.
- ✓ A compound statement is treated as a single unit. Either all the constituent simple statements will be executed or none will be executed.
- ✓ A compound statement can appear at any point in a program wherever a simple statement can appear.
- ✓ In a block, non-executable statements (e.g. declaration statements) should come before executable statements.
- ✓ **A compound statement can be empty i.e. there is no simple statement present inside the pair of braces like {}. An empty compound statement is equivalent to a null†† statement but it cannot act as a terminator for a statement.**

if(a==b) { }		**if(a==b) ; //←null statement**	**Valid as {} is equivalent to null statement (i.e. ;)**
printf(“Hello”){}		**printf(“Hello”);**	**Invalid as {} cannot act as a terminator**

Flow Control Statements

- By default, statements in a C program are executed in a sequential order. The order in which the program statements are executed is known as "flow of program control" or just "flow of control". By default, the program control flows sequentially from top to bottom.
- Practical situations like decision making, repetitive execution of certain task etc. require deviation or alteration from the default flow of program control.
- The default flow of control can be altered by using control flow statements. Control flow statements are of two types:

1. Branching statements
   - Selection statements
   - Jump statements
2. Iteration statements

Branching Statements:

Branching statements are used to transfer the program control from one point to another. They are categorized as:

Conditional branching:	**In conditional branching, also known as selection, program control is transferred from one point to another based upon the outcome of a certain condition. Following selection statements are available in C: if, if-else and switch statement.**
Unconditional branching:	In **unconditional branching**, also known as **jumping**, program control is transferred from one point to another without checking any condition. Following j**ump statements** are available in C: goto, break, continue and return statement.

Selection Statements:

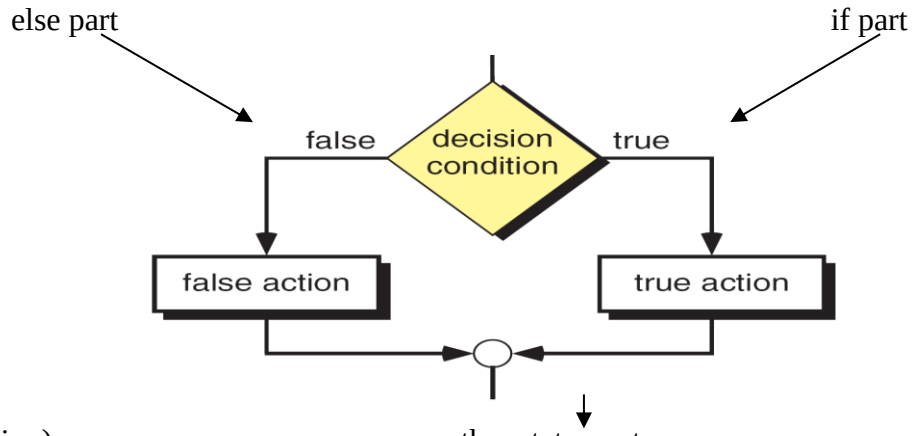
- Based upon the outcome of a particular condition, selection statements transfer control from one point to another. Selection statements select a statement to be executed amongst a set of various statements.

→The selection statements available in C are:

1. if statement
2. if-else statement
3. switch statement

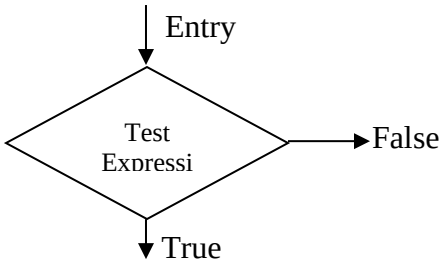
If Statements

Two – way Decision logic (conditional statement)




```
if (expression)
{ statement 1;
- - - -- - - ---}
else
{statement 2;
- - -- - -}
```

other statements……

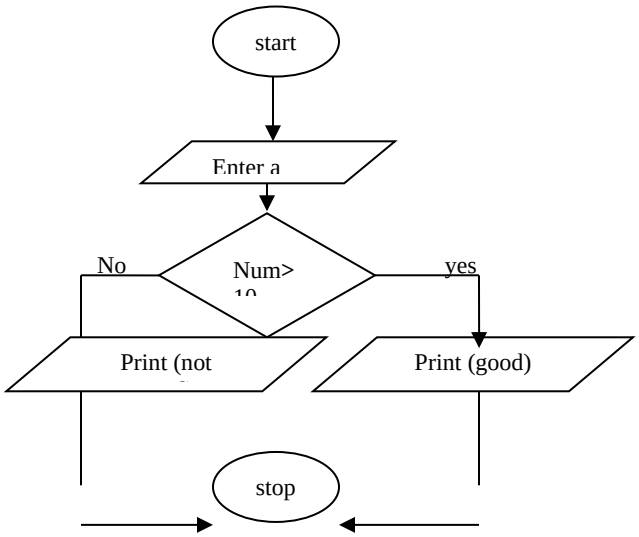


Forms of IF statement:

(i) Simple if statement.
(ii) If ….. else statement.
(iii) Nested if …. Else statement.
(iv) If Else if ladder.

(i) *Simple if statement:*

Program-statement: Write A Program in which- enter a no. from keyboard if this no. is bigger than 10 then print you are good .



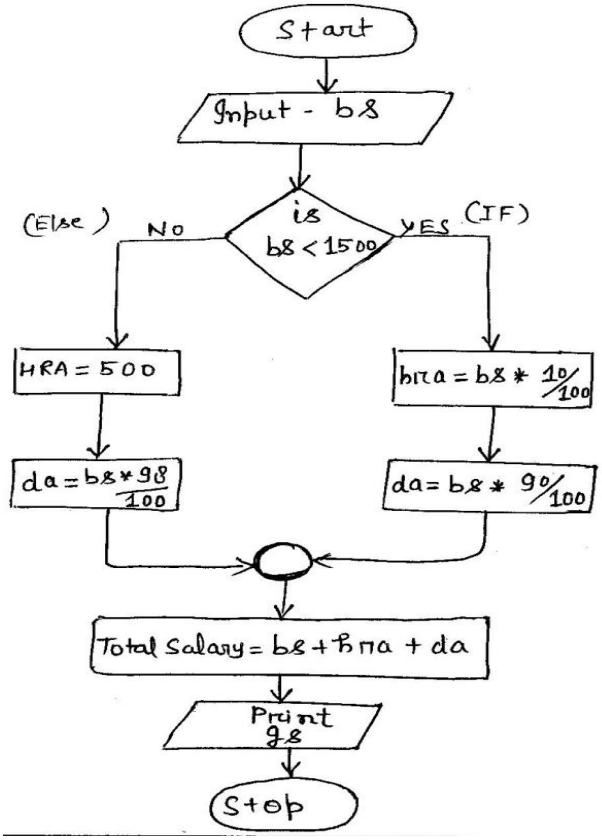

```
main ()
{
        int num;
        printf("enter your no.");
        scanf("%d",&num);
        if(num >15)
        {
                Printf("you are good ");
        }
        else{
                Printf("you are not good ");
        }
}
```

(ii) if-else statement.

```
if (test expression)
{
True-block statement(s)
}
else
{
False-block statement(s)
}
::
Other statements in program
::
```

Program stmt: WAP to calculate gross salary of an employee as-if basic salary is less than Rs. 1500, then HRA = 10% of basic, DA=90% of basic
If salary is ≥ 1500, then HRA = Rs. 500,
DA = 98% of basic. Input basic salary by keyboard.

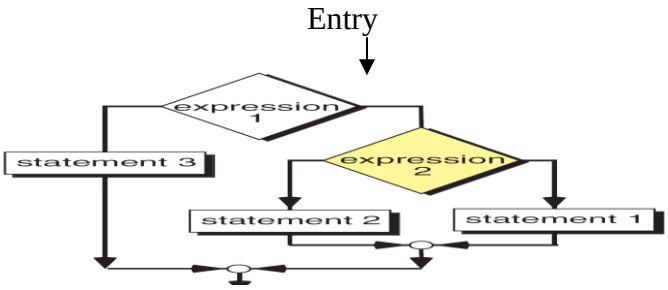
```
main()
{
float bs, gs, da, hra
printf("Enter basic salary");
scanf("%f",&bs);
if(bs<1500)
{
        hra=bs*10/100;
        da=bs*90/100;
}
else
{
        hra =500;
        da=bs*98/100;
}
gs= bs+da+hra;
printf("Total Salary = %f",gs);
}
```

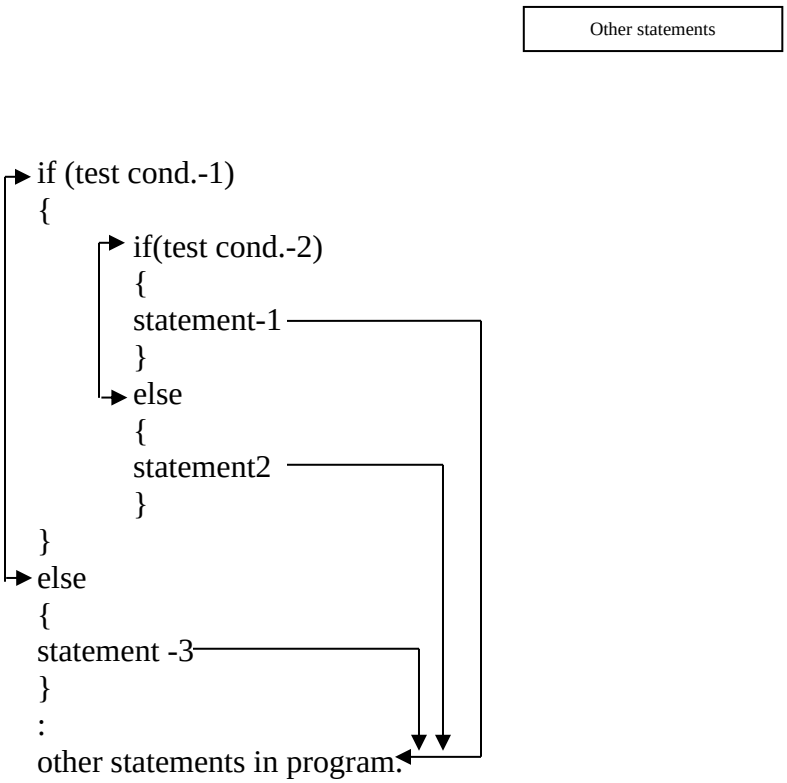


(iii) *Nested if (else) statement:*

When an if… else is included within an if … else statement, it is called as-

Nested if-else statement.





Example: Program to Check Leap Year (using nested if-else)

```
main()
{
   int year;
   printf("Enter a year: ");
   scanf("%d",&year);
   if(year%4 == 0)
   {
      if( year%100 == 0)
      {          // year is divisible by 400, hence the year is a leap year
         if ( year%400 == 0)
            printf("%d is a leap year.", year);
         else    printf("%d is not a leap year.", year);
      }         // year is divisible by 4, hence the year is a leap year
      else     printf("%d is a leap year.", year );
   }
   else    printf("%d is not a leap year.", year);
}
```

Program: **One company decides to give bonus as 2% to all employees & 5% to all its managers having (salary) > 10,000 & if balance > 5000. Find increased salary after adding bonus.**

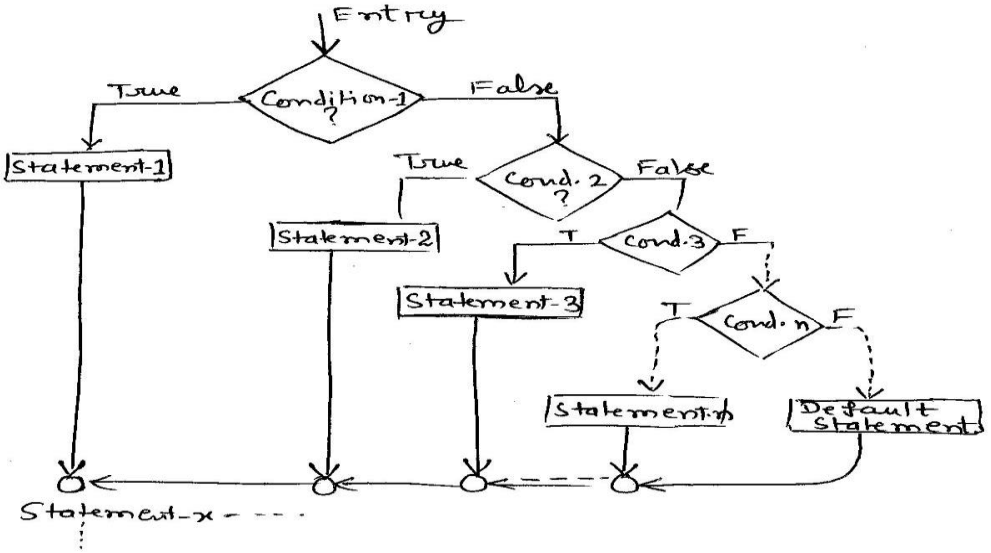
```
main( )
{
float balance, bonus, salary;
printf("Enter Balance, salary");
```

```
scanf("%f", & balance);
scanf("%f", & salary);
if (salary > 10,000)
{
        if (balance>5000)
                bonus=.05 * salary;
        else
                bonus=.02 * salary;}
else
{
        bonus=.02* salary;
}
salary=(salary+bonus);printf("salary now=%f ",salary);
```

The Else IF Ladder:

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

- when multipath decisions are involved, then we use a combination of else-ifs together.
- A multipath decision is a chain of ifs in which the stmt. associated with each else is an if



Example: Program to Check Leap Year (using if-else-if ladder)

```
main()
{
  int year;
   printf("Enter a year to check if it is a leap year\n");
  scanf("%d", &year);
   if ( year%400 == 0)
    printf("%d is a leap year.\n", year);
  else if ( year%100 == 0)
        printf("%d is not a leap year.\n", year);
         else if ( year%4 == 0 )
                printf("%d is a leap year.\n", year);
                 else
                        printf("%d is not a leap year.\n", year);
  }
```

Program-Statement: Declare the result of student
as if he gets marks > 79 and <=100- honors
input marks from keyboard. > 59 and <=79– Ist div.
> 49 and <=59– IInd div.
> 39 and <=49- IIIrd div.
else (<= 39) – Fail.

```
main()
{
int marks;
printf("Enter the marks");
scanf("%d", &marks);
printf("\nresult is:");
if(marks > 79 && marks<=100)
{
        printf("1st div. with Honors");
}
else if (marks > 59 && marks<=79)
        {
                printf("1st div.");
        }
        else if (marks > 49 && marks<=59)
        {
                printf("2nd Div.");
        }
        else if(marks > 39 && marks<=49)
        {
                printf("3rd Div.");
        }
        else
        {
                printf("Fail");
        }
}
```

Example: A company insures its employees in the following cases:

- If the employee is married
- If the employee is unmarried, male & above 30 yrs.
- If the employee is unmarried, female & above 25 yrs.

Otherwise the employee is not insured.
Marital status, gender and age of employee are inputs. Then WAP to determine if the driver is to be insured or not.
Now we will solve above program by 2 ways: using logical operators & without using logical operators.

(A) ***using logical operators.***

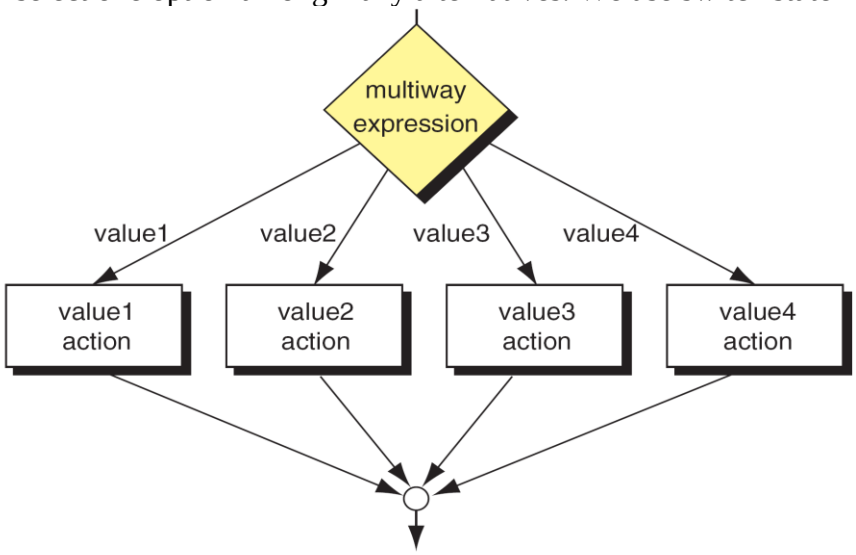
```
main()
{
char gender, ms;
int age;
printf("Enter age, gender, marital status");
scanf("%d%c%c", &age,& gender,&ms);
if((ms=='M') || (ms= ='U'&& gender = ='M'&&age>30) || (ms = = 'U'&& gender = = 'F'
&& age>25))
printf("Emloyee is insured");
else
{printf("Employee is not insured");}}
```

(B) ***Without using logical operators:***

```
main()
{
char gender, ms;
int age;
printf("Enter age, gender, marital status);
scanf("%d%c%c", &age, & gender, &ms);
if(ms=='M')
printf("Employee is insured");
else
{
if (gender = = 'M')
{
if (age > 30)
printf("Employee is insured");
else
printf("Employee is not insured");
}
else
{
if (age > 25)
printf("Employee is insured");
else
printf("Employee is not insured");
}
}
```

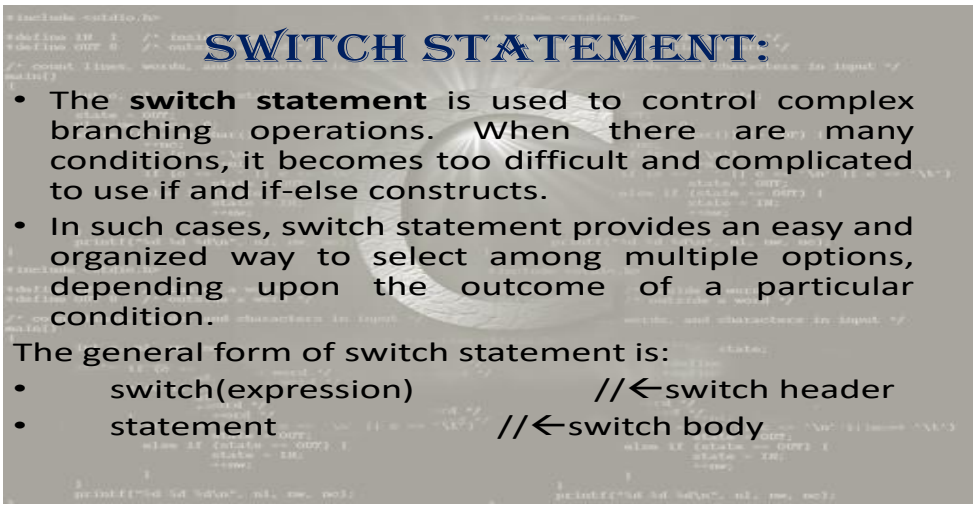
Switch Statements (Multiway Selection)

Here we can select one option among many alternatives. We use switch statement here.



Switch Decision logic:

Switch statement is used when we have many alternatives. Since *if-else if* becomes complex. The switch statement tests the value of a given variable (or expression) against a list of case values (options/alternatives) & when a match is found, a block of statements associated with that case is executed.



The General form of switch:

```
switch (integer expression)
{
case <constant> 1:
        {statements;
        statements;}
        break;
case <constant> 2:
        statement-block
        break;
:               :
```

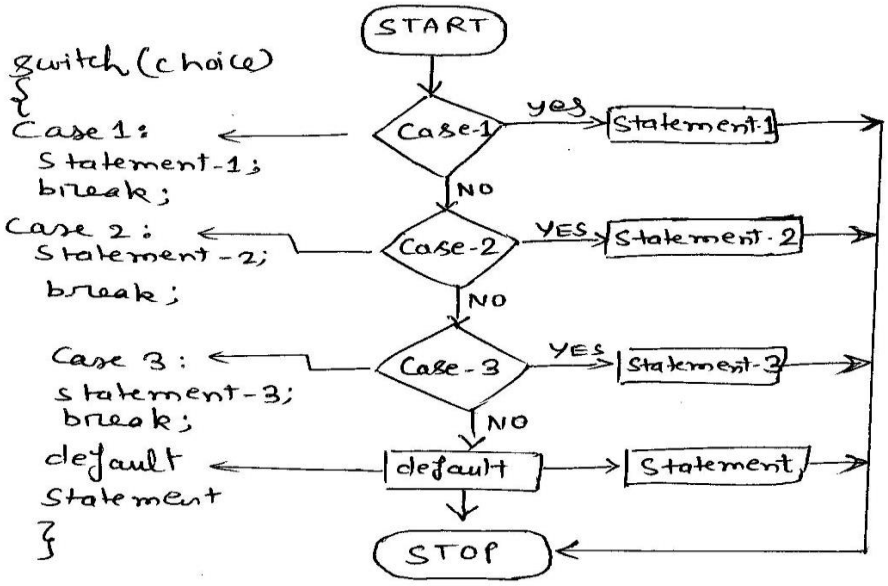
```
case n:
        statement-block
        break;
default:
        stmt-block      }
```

Example :

```
 void main()
{
int i=2;
switch(i)
{
case 1:
        printf("I am in case 1\n");break
case 2:
        printf("I am in case 2\n");
case 3:
        printf("I am in case 3\n");
default:
        printf("I am in default \n");
        }
}
```

o/p → I am in case 2
I am in case 3
I am in default

But if we will insert **break** in each case then only one o/p → I am in case 2.



Statement-x

KEY POINTS: SWITCH STATEMENT

- **Switch header** :consists of keyword switch followed by **switch selection expression** enclosed within parentheses.
- Switch selection expression must be of integral type (i.e. integer type or character type).
- **Switch body** consists of a statement. The statement constituting switch body can be a null statement, expression statement, labeled statement, flow control statement, compound statement etc.
- Generally, switch body consists of a compound statement, whose constituent statements are case labeled statements, expression statements, flow control statements and an optional default labeled statement.

- Case labels of case labeled statements constituting body of switch statement should be unique i.e. no two case labels should have or evaluate to the same value.
- There can be at most one default labeled statement within the switch body.
- switch statement is executed as follows:
  - switch selection expression is evaluated.
  - The result of evaluation of switch selection expression is compared with the case labels of the case labeled statements until there is a match or until all the case labeled statements have been examined.
    - ✓If the result matches with the case label of a case labeled statement, the execution starts from the matched case labeled statement and all the statements after the matched case labeled statement within the switch body gets executed.
    - ✓If no case label of case labeled statements within the switch body matches the result , the execution starts with the default labeled statement, if it is present, and all the statements after the default labeled statement within the switch body gets executed.
    - ✓If none of the case label matches the result and there is no default labeled statement present within the switch body, no statement within the switch body will be executed and the execution continues from the statement following the switch statement.

○

- Order of cases is not important
  We can write as:

```
Case 10:
Statement …..
Case 599:
Statement …..
Case 305:
Statement …
```

- Char values are also permitted in case & switch. because char values are replaced by the ASCII values of these characters.
  As :-

```
        Char c;
switch (c)
{
        case 'v':
        ----
        case 'a':
        -----
        }
}
```

- break takes the control outside the switch body.
- switch stmts. are very useful to write menu driven programs.

Prog. *To make a program of calculator.*

```
        Addition operation:option1
        Subtraction operation:option 2
        Multiplication operation:option 3
        Division operation:option 4


main()
{
        int a,c;
        int b;
        int opt;
printf("Select an option (1….4):");
scanf("%d", &opt);
switch (opt)
{       case 1:scanf("%d%d",&a,&b); c = a +b; printf("%d",c);break;
        case 2: scanf("%d%d",&a,&b);c = a -b; printf("%d",c);break;
        case 3: scanf("%d%d",&a,&b);c = a *b; printf("%d",c);break;
        case 4: scanf("%d%d",&a,&b);c = a /b; printf("%d",c);break;
default:printf("Select a value between 1 & 4,  try again");}return 0;}
```