Operators & Expressions (U.U.T.P)

An operator is a symbol that helps to perform certain mathmatical calculation and logical manipulation. Eg; +, −, *, ==, =, >, <. The data item that operator acts upon is called operands. On the basis of number of operands they take, operators can be classified into:-

(a) Unary operator
(b) Binary operator
(c) Ternary operator.

(a) Unary operator :- The operator which use only one operand for operation is called unary operand. Eg :- increment and decrement i.e a++, --a, ++a etc.

(b) Binary operator :- The operator which use two operand for operation is called binary operator. Eg ; +, -, =, <, >, i.e 2+2, 2-3

(c) Ternary operator :- It is also know as conditional operator. It requires three operands. It evaluate the condition and then assigned the required expression.

Syntax :-

Condition ? expression 1 : Expression 2;

If condition is true then expression 1 is evaluted and this becomes the value of the conditional expression and if condition is false, then expression 2 is evaluated.

Example :-

Find a greatest among three numbers

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a, b, c, large;
clrscr();
Printf("\n Enter three numbers : ");
scanf("%d %d %d", &a, &b, &c);
large = a>b ? (a>c ? a:c) : (b>c ? b:c);
Printf("\n Largest number is %d", large);
getch();
}
```

Date :

Type of operators (on the basis of operation);

1. Arithematic operator
2. Relational operator
3. Logical operator or Boolean operator
4. Assignment operator
5. Increment & Decrement operator
6. Conditional operator (tenary operator)
7. Comma operator
8. Size of operator

(1) Arithematic operator :-

It is used for arithmetic operation like addition, substraction, multiplication & division.

Operator	Meaning	Example
+	Addition operator	a+b
-	Substraction operator	a-b
*	Multiplication operator	a*b
/	Division operator	a/b
%	Modulus operator	a%b

(2) Relational operator :-

Operators that are used for comparing relation between operands. In C programming we can compare the variables using the relational operator It is also called comparision operator as it is used to compare any two expression.

Operator	Meaning	Example
<	is less than	a<b
<=	less than equal to	a<=b
>	greater than	a>b
>=	greater than equal to	a>=b
==	equal to	a==b
!=	Not equal to	a!=b

(3) Logical operator or Boolean operator:-
Logical operators are used for combining logical expressions which are finally evaluated as true or false according to operator use. Logical operators are used for logical decision making in C-programming.

Operator	Meaning	Example
&&	logical AND	(a>b)&&(a<c)
|| or ||	logical OR	(a>b)||(a>c)
!	logical NOT	!(a==b)

(4) Assignment operator:-
It is used for assigning a value or a result of an expression to an identifier. It is the combination of arithmetic operator & assignment opertor & performs the task. They are also known as short hand operator.

Operators	Statement with Short hand operator	Statement with Simple assignment
+=	a += 1	a = a + 1
-=	a -= 1	a = a - 1
*=	a *= 1	a = a * 1
/=	a /= 2	a = a / 2
%=	a %= b	a = a % b

Here, Arithmetic operator performs it's operation first and their assignment operators assigns the value.

) Increment / Decrement operators :-

These operators are also known as unary operator which performs operation upon one operand. There are two types of ~~Normal~~ numeric operator.

(a) Prefix :- ++ variables, -- variables,
(b) Postfix :- variables ++ ; variable --;

(i) Eg; m = 5
y = ++m;
The value of y & m would be 6.

(ii) Eg; m = 5;
m = 5;
y = m++;
The value of y is 5 & m is 6

Page No.:
Date :

(a) Prefix operator :-

It first adds 1 to the operand & then result is assigned to the variable on the left.

(b) Post-fix operator :-

It assigns the value to the variable on the left first and then increases or decreases the operand.

(6) Conditional operator (Tenary) :- (? :)

Conditional operator requires three operands. It evaluates the condition & then assigns the required expressions.

Condition ? expression 1 : expression 2.

If condition 1 is true then expression 1 is evaluated and this becomes the value of the conditional expressions & if condition 1 is false, then expression 2 is evaluated.

Eg :- a = 5;
b = 9;
x = (a > b) ? a : b

(7) Comma operator :- It is used to link related expression together. It evaluates the expression from left to right and the value of right post expression is the value of combined expression.

res = (num1, num2);

seperator :- In the functional call declaring variable etc.

Page No.:
Date :

Eg; int num1 = 10, num2 = 20;

(8) Sizeof operator ( );-

The size of operator gives the size of the data type of variable in terms of bytes occupied in memory

Eg int x; char n;
n = size of (x); size of (n);

Expression :-

An expression is the combination of variables, constant & operators. In C every expression results same value that can be assigned to the variable.

⇒ Precedence of arithmetic operators:-

High priority :- *, /, %
Low priority :- +, -.

Rules for evaluation of expression :-

(1) Paranthized sub expression from left to right are evaluated.
(2) If parantheses are nested the evaluation begins with the inner nested sub expression.
(3) The precedence rule is applied in determining the order of application of operators in evaluating sub expression.
(4) The associativity rule is applied when two or more operators of the same precedence level appeared in a sub expression.
(5) Arithmetic expression are evaluated from left to right using the rule of precedence.

Precedence & Associativity

Operator precedence determine the grouping of term in an expression. This affects how an expression is evaluate. Certain operators have higher precedence. For example; multiplication operator has higher precedence then addition operator. A list of operator with there precedence is given below:-

Category	Operator	Associativity
postfix	() [] ->	left to Right
Unary	++ --	Right to left
multiplicative	* / %	left to right
additive	+ -	left to right
relational	< <= > >=	left to right
equality	== !=	left to right
logical AND	&&	left to right
logical OR	||	left to right
conditional	?:	Right to left
assignment	=+, =-, =*, =/ =	Right to left
comma	,	left to right

Example

1. $x = 7+3*2$
   $= 7+6$
   $= 13$

2. $y = 5-1.2*4$
   $= 1*4$
   $= 4$

3. $y = 5*4\%2$
   $= 0$

Control Statement

In C programming the program is executed from top to bottom. Each ~~in str~~ instruction are executed only once. Sometimes, we might need to break the sequence of the programe and jump from one instruction to another as per the requirement. Control statements are those statement which control the flow of execution of programe. Mainly control statements are of followings categories:-

(1) **Branching**

(a) Conditional statements

- (i) If statement
- (ii) If ... else statement
- (iii) If ... else If statement
- (iv) Nested if ... else statement

(b) Switch case statements

(2) **Looping**

- (a) while loop
- (b) Do ... while loop
- (c) for loop

(3) Jumping statement

(a) break;
(b) continue;
(c) Goto;
(d) exit();

(1) Branching

It is also known as selective control structure in which selection is made on the basis of condition. When the condition is true the programe will go towards certain statement & otherway when the condition is false. With the help of branching the programmer can also give user the choices to executed the selective instruction. Branching can be classified into two types:-

(a) Conditional statement
(b) Switch case

(a) Conditional statement

(i) ~~if statement~~:-

~~It is the simplest format~~ conditional statement. ~~The statements are executed if the test expression (condition) is true. When condition is false the control must get of the if statement structure & the statement outside the structure run.~~

Page No.:
Date :

(2) Conditional or control structure :-

It specifies the order in which various ~~instr~~ instructions or statements are to be executed in a Program. There are four types of control structure.

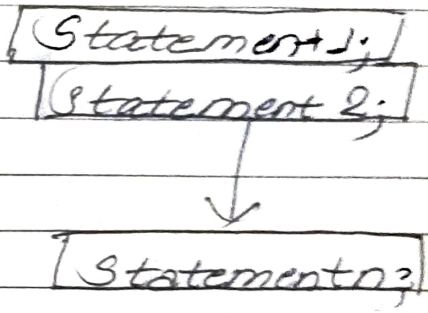
(i) Sequence :-

It is a control structure in which statement are executed sequentially

Syntax :-

```
Statement 1;
Statement 2;
:
:
Statement n;
```

Flowchart



(ii) Selective / decision :-

It is used to control the flow of program based upon condition. Conditional Statement are mainly used for decision making various decisions control statement

(a) if Statement :-

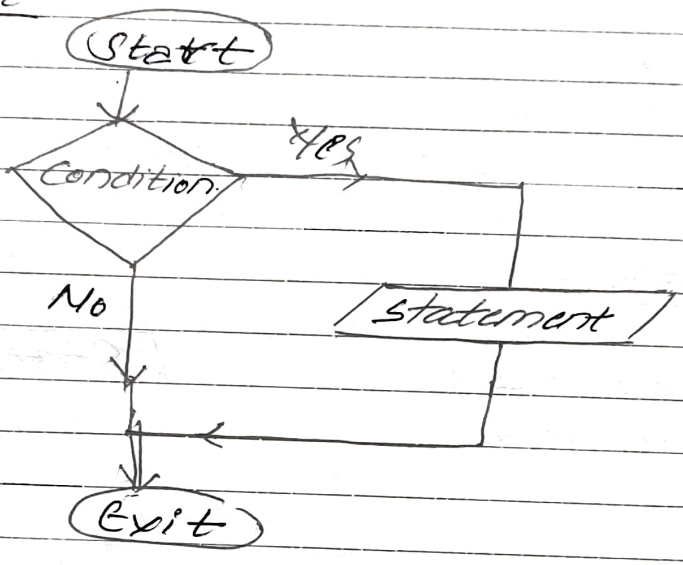
It is the simplest format conditional Statement. If the given condition is true then it will execute the given Statement otherwise the statement outside this structure is execute

Page No.:
Date :

Syntax

```
if (condition)
{
  statement / expression;
}
```

Flowchart



WAP to enter user's age & display you can cast vote. If the user is >=18.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
clrscr();
printf("\n Enter your age :");
scanf("%d",&a);
if (a>=18)
{
  printf("\n You can vote");
}
getch();
}
```

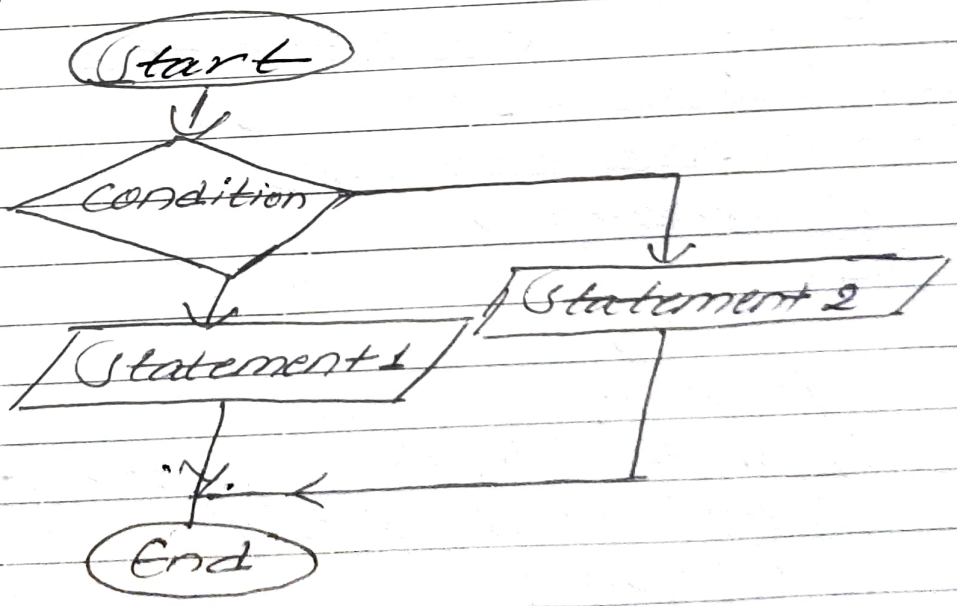
(b) if else statement :-

In another form of selective control structure which can handle both true and false state of a given condition.

Syntax

```
if (condition)
{
 statement 1;
}
else
{
 statement 2;
}
```

Flowchart



WAP to check whether the number enter is odd or even

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

```
int a, x;
clrscr();
printf("\n Enter any integer : ");
scanf("%d", &a);
x = a % 2;
if (x == 0)
{
  printf("\n %d is even number", a);
}
else
  {
    printf("\n %d is odd no", a);
  }
getch();
}
```

WAP to check whether the num is +ve or -ve

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a;
  clrscr().
  printf("\n Enter any number : ");
  scanf("%d", &a);
  if (a>0);
  {
    printf("\n %d is +ve no", a);
  }
  else (a<0)
  {
    printf("\n %d is -ve no", a);
  }
```

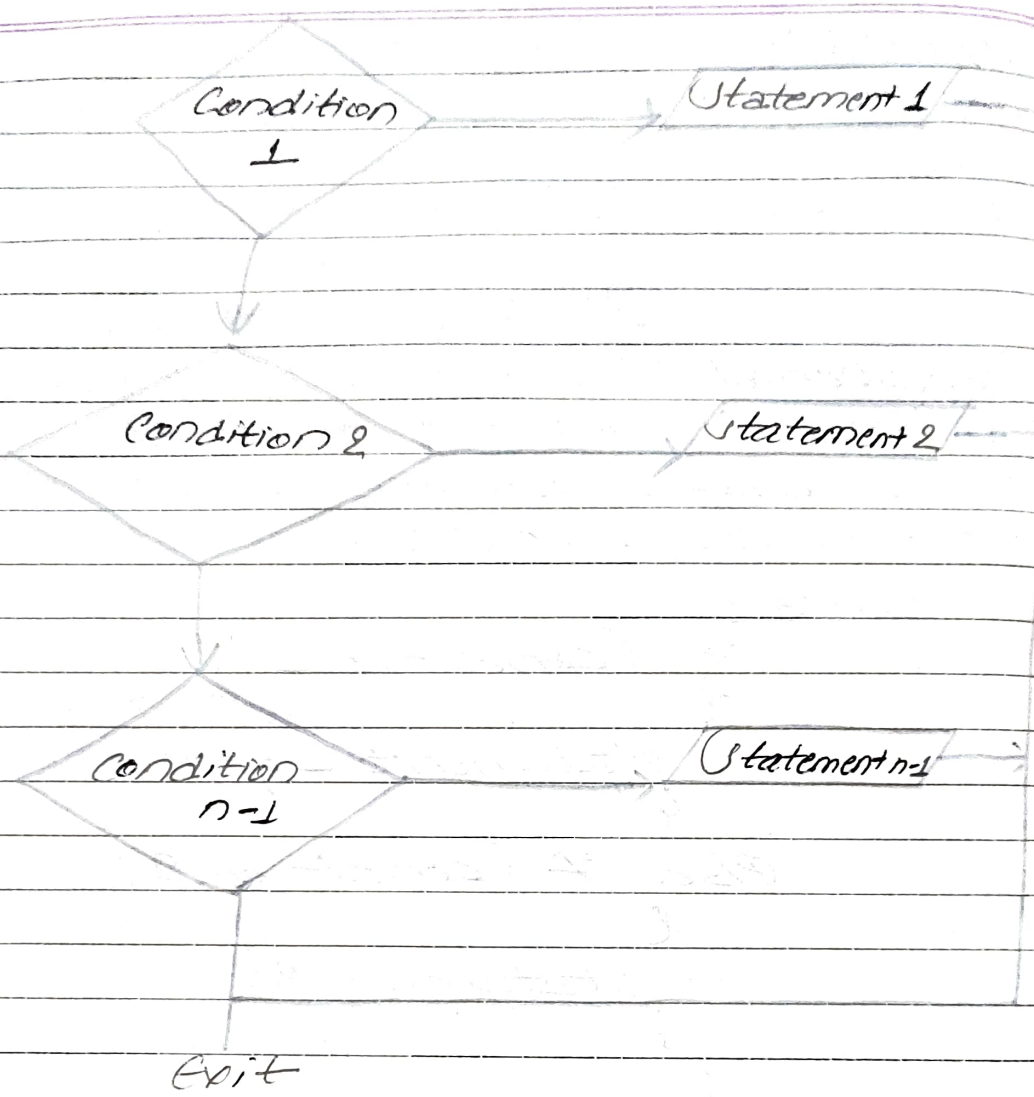
getch();
}

if else ... if statement

It is also known as multiple conditional statement or multipath conditional statement or if else ladder. It is used when we have to check two or more conditions.

Syntax

```
if (condition 1)
{
  statement 1;
}
else if (condition 2)
{
  statement 2;
}
:
else if (condition n-1)
{
  statement n-1;
}
else
{
  statement n;
}
```




```
#include <stdio.h>
#include <conio.h>
void main()
{
int a;
clrscr();
printf("\n Enter any integer : ");
scanf("%d", &a);
if (a>0)
{
printf("\n %d is +ve no", a);
```

```
}
else if (a<0)
{
printf("\n %d is -ve no", a);
}
else
{
printf("\n %d is zero", a);
}
getch();
}
```

#WAP to find the greatest among three numbers.

```
#include<stdio.h>
#include<conio.h>
  void main()
{
  int a, b, c;
  clrscr();
  printf("\n Enter any three number:");
  scanf("%d %d %d", &a, &b, &c);
  if(a>b && a>c)
  {
  printf("\n %d is the greatest", a);
  }
else if (b>a && b>c)
{
printf("\n %d is the greatest", b);
}
else
```

```
{
printf("\n %d is the greatest", c);
}
getch();
}
#      /* Smallest no */

#include <stdio.h>
#include <conio.h>
void main()
{
int a, b, c;
clrscr();
printf("\n Enter three no:");
scanf("%d %d %d", &a, &b, &c);
if(a<b && a<c)
{
printf("\n %d is the smallest", a);
}
else if(b<a && b<c)
{
printf("\n %d is the smallest", b);
}
else if(c<a && c<b)
{
printf("%d is the smallest", c);
}
getche(); else
{
printf("\n Numbers are equal");
}
getch();
}
```

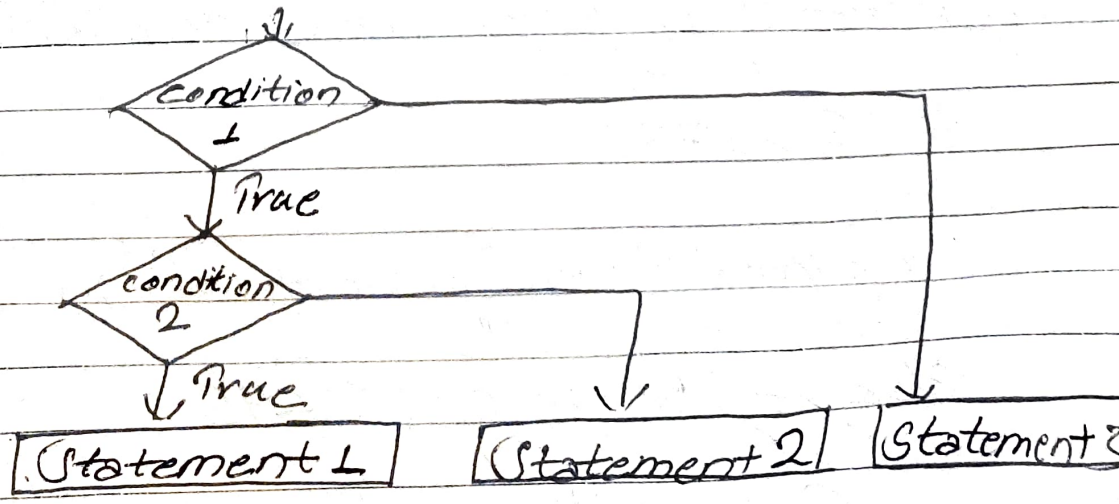
(d) Nested if else Statement.

An entire if else statement can be written within the body of if part or else part of another if else statement is called nested if else statement.

Syntax

```
if (condition )
{
    if (condition 2)
    {
      statement 1;
    }
    else
    {
      statement 2;
    }
else
{
  statement 3;
}
```

Flowchart



Page No.:
Date :

WAP to enter marks of five different subjects & calculate total marks & Percentage & display division on the basis of following criteria.

Percentage	Division
P >= 75	Distinction
P >= 60 & P < 75	First division
P >= 45 & P < 60	Second division
P >= 35 & P < 45	Third division
P > 35	.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int eco, comp, eng, math, acc, t;
float P;
Printf("\n Enter marks of economics, English, Maths, Accounts & Computer:");
scanf("%d %d %d %d %d %d", &eco, &eng, &math, &acc, &comP);
t = eco + com + eng + math + acc;
P = t / 5.0;
Printf("\n Total marks %d \n Preentage %f", t, P);
if (P >= 75)
{
Printf("\n Distinction");
}
Else if (P >= 60 && P < 75)
{
Printf("\n 1st div");
}
```

```
else if (p>=45 && p<60)
 {
  printf("In 2nd div");
 }
else if(p>=35 && p<45)
 {
  printf("In 3rd div");
 }
else
 {
  printf("In You have failed");
 }
getch();
}
```

(2) Switch case Statement:-

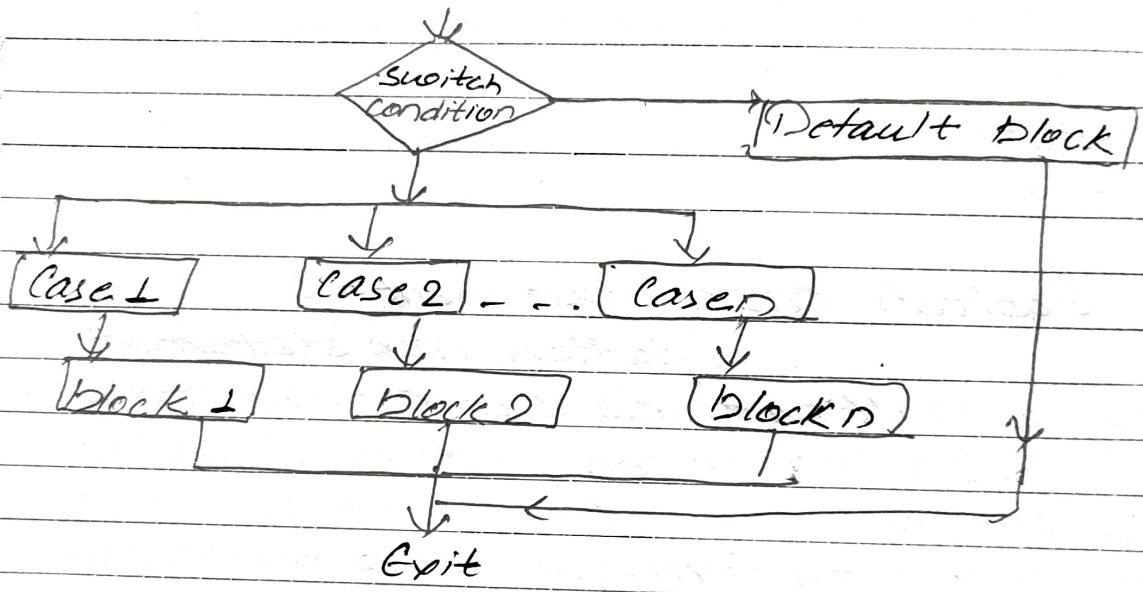
Switch case Statement is a multipath decision making Statement. If..else ladder can perform same task as switch case but as the number of authority increases selection process becomes complex. In case of if...else and the main difference between if...else & switch case is that if..else makes selection in a serial way whereas in switch case it is done in parallel way.

Syntax

```
switch (expression)
 {
  case constant 1;
  Statement 1;
   break;
```

:
:
case constant n-1
statement n-1;
break;
default;
statement n;
}

Flowchart




```
#include<stdio.h>
#include<conio.h>
  void main()
{
 int ch;
  clrscr();
Printf("\n Enter choice:");
 Scanf("%d", &ch);
 switch(ch)
  {
```

```
case 1 : printf ("\n First");
         break;
case 2 : printf ("\n Second");
         break;
case 3 : printf ("\n Third");
         break;
default : printf ("\n wrong choice");
}
getch();
}
```

WAP to enter two number and ask the user to display sum, diff & product according to their choice using switch case statement

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a, b, s, ch;
clrscr();
printf("\n Enter two number:");
scanf("%d %d", &a, &b);
printf("\n Press 1 for add");
printf("\n Press 2 for subtract");
printf("\n Press 3 for multiplication");
printf("\n Enter choice:");
scanf("%d", &ch);
switch(ch);
{
case 1 : s = a+b;
```

```
        printf("\n Sum is %d", s);
        break;
    case 2: s=a-b;
        printf("\n Difference is %d", s);
        break;
    case 3: s=a*b;
        printf("\n Product is %d", s);
        break;
    default: printf("\n exit");
    }
    getch();
}
```

Difference between if--else & switch

if -- else	Switch
1. In if else selection appropriate option is done in serial fashion	1. In switch case selection appropriate option is done in parallel way
2. An expression is evaluated & the code block is selected based on the result of expression	2. An expression is evaluated & code block is selected based on the value of expression
3. It takes decision on the basis of true or false	3. It takes decision on the basis of equality
4. It doesn't require break statements since only one of the block of code is executed at a time.	4. It needs break statement to avoid execution of the block of code just below the currently executing block.

(2) Repetitive / Looping :-

A loop is a repetitive control structure which execute a block of program statements repeatedly for specified number of time or till the given condition is satisfied. There are mainly three types of loop :-

(a) while loop
(b) do while
(c) and for loop

(a) while loop :-

It executes a block statement until the given condition is true. It checks the condition at first, if it is true than it executes the statement otherwise it gets out from the loop structure.

Syntax

```
initialization;
while(condition)
{
    statement;
    increment / decrement;
}
```