Operator

Date 15-Mar-22

Expression:

Expression is the combination of two things

1- Operator
2- Operand.

1- Operator:

An operater is a symbol that tells the compiler to perform specific mathematical or logical manipulation

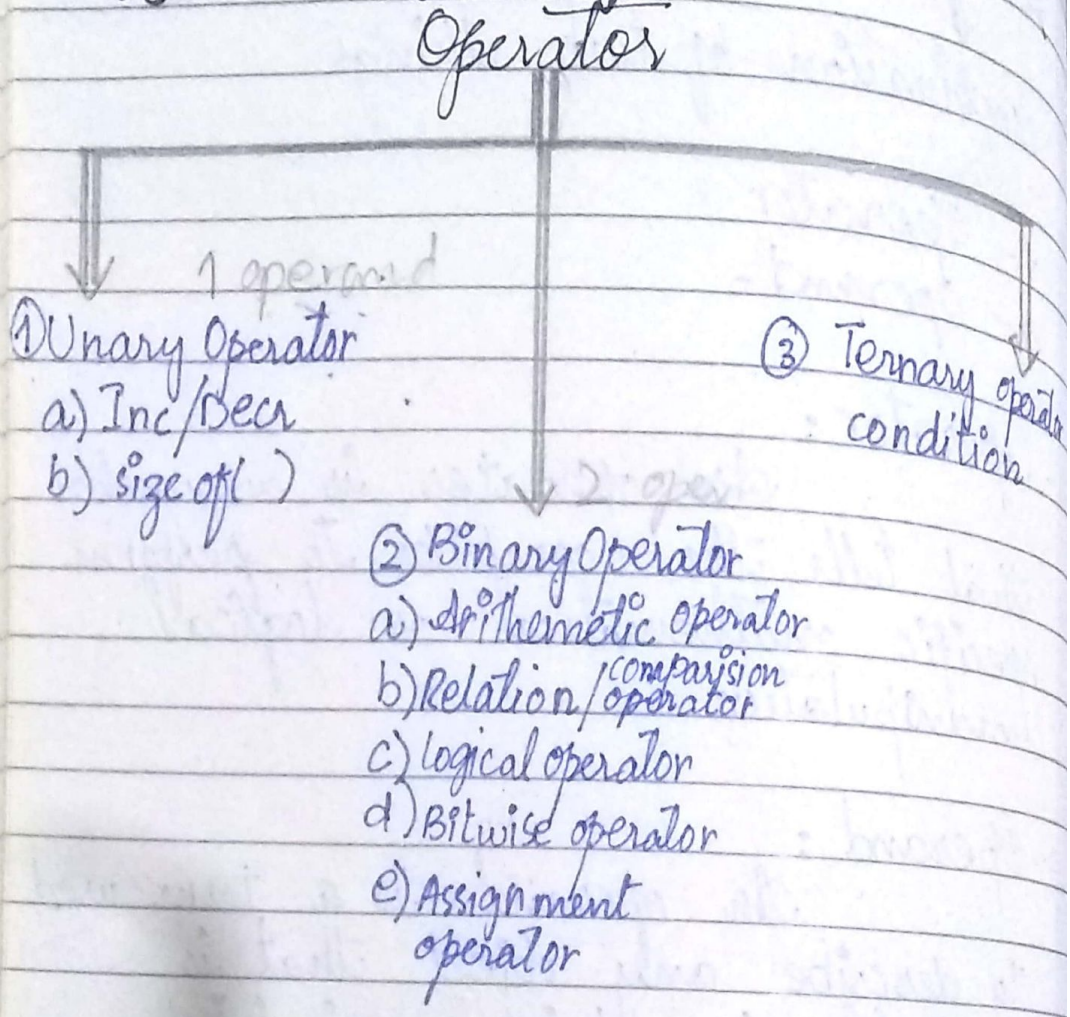
2- Operand:

An operand is a term used to describe any object that is capable of being manipulated

Example: in "1+2" the "1" & "2" are operand and the plus symbol is the operator:

Date

Types Of Operator:



1-Unary Operator:

A type of operator that work with one operand known as unary operator `--, ++, --` The above operator are used with single operator as follows `a++, --x, --a`.

2- **Binary Operator:**

A type of operator that work two operand is known as Binary operator

-, +, *, /, %

The above operator are used with Two operand as follows.

a+b, x/y, x*y, a%b etc

3. **Ternary Operator:**

Condition operator is a ternary operator because it work on three operands. The conditional operator behaves like an if else statement.

Syntax:

Condition? value-if-true : value-if-false

Unary Operator Types:

a:
1- Increament Operator:

The increament operator is used to increase the value of a variable in an expression

⇒ Two Types:

1- Pre-increament operator: Pre increment value is first incremented & then used inside the expression.

e.g ++m //m is a variable

is mai pehle karwale kiyaa

2- Post-increment operator: The post-increment value is first used inside the expression and then incremented.

is mai baad mai karwale hai

e.g m++

Example:

```
int main( ){
  int x;
  x = 200;
  ++x; OR x++;
  cout << x;
  return 0;
}
```

output in 201

Date 15-Mar-22

2- Decrement Operator:

The decrement operator is used to decrement the value of a variable in an expression.

Types:

1- Pre decrement operator: In the pre-decrement, value is first decremented & then used inside the expression.

e.g --m

2- Post-decrement operator: In the post decrement, value is first used inside the expression and decremented.

e.g m--

Example:

```
int main ( ){
int a;
a = 15;
a--; OR --a;
cout << a;
return 0;
}
```

output
14

Date 21-Mar-22

b) Size Of( ) Operator:

It is used to find out the size of datatypes or variable.

```
int main( ){
  cout << size of (char);
}
```

Cout << "The size of Datatype" << size of (char) << "Bytes";

Pg no 14

2- Binary Operator:

a) Arithematic Operator:

An arithematic operator is a mathematical function that takes two operands & perform a calculations

E.g:

+ - / % *

Addition: Suppose you have two variable A hold 20 & B hold 20 then

Date

Add two operands

e.g A+B will give 40

```
#Include <iostream>
int main ( ){
int A = " 20"
int B = " 20"
int C = " A + B "
cout << C ;
}
return 0 ;
}
```

```
# include <stdio.h>
int main ( ){
int a = 20;
int b = 20;
int c = a+b;
Printf ("%d", c);
return 0;
}
```

subtraction (-) ; Subtraction second operands from the first.

e.g if A - B will give 0 if A=20 B=20

```
int main ( ){
int A = " 20"
int B = " 20"
int C = A - B
cout << C ;
return 0 ;
}
```

Date

3- Division : Divide Two operand
e.g A/B will give 1 A = 20 , B = 20

```
int main ( ) {
int A = "20"
int B = "20"
cout << A/B ;
return 0;
}
```

4- Multiplication: Multiplies both operands
e.g A * B will give 400.

```
int main ( ) {
int A = "20"
    B = "20"
Cout << A * B ;
}
return 0
```

5- Modulus: Modulus operator reminder of after dn an inleger division

```
int A = '20'
int B = '20"
cout <<
return 0;
}
```

Date

e: Assignment Operator:

An Assignment operator used to assign a newvalue to a variable.

Example: +=, -=, /=, %=.

```
int main() {
int x = 10;
cout << x;
return 0;
}
```

```
#include <stdio.h>
int main() {
int x = 10;
x + 40;
Printf("%d", x)
```

```
int main() {
int x = "10";
x += "30";
cout << x;
```

40

```
int main() {
int x = "10";
x -= "30";
cout << x;
return 0;
}
```

-20

Date

b- Relation Operator:

A relational or comparision operator is used to check the relationship between two operands

Example:

a > b Here > is a Relation operation it check.
if a is greater than b or not.
if the relation is true, it return 1 where as if the relationship is false it return 0.

equal to not equal
= = , ! = , > , < , > = , < =.

II) AB

```
int main ( ) {
    a = 10 ;
    b = 20 ;
    Cout<< a % b ;
    return 0 ;
}
```

relation

```
"
int x = 5
int y = 3
cout<< x == y
return 0 ;
}
```

Date 22-Mar-22

Logical Operator:

logical operator are used to check whether an expression is true or false, if the expression is true, it return 1, whereas if the expression is true, return 1, where as if the expression is false it return 0.

e.g:
logical AND &&, logical OR ||, logical NOT!

logical && : logical AND, true only if all the operands are true.

e.g:
exp1 && exp2

Example:

```
int main(){
int x = 5;
int b = 3;
cout<<(x>3 && x<10);
return 0;
}
```

Date

Logical OR ||: logical OR, True if atleast one of the operands is true e.g exp1 || exp2

Example:

```
int main(){
int a = 5;
int b = 3;
cout<<(a>b || a<b);
return 0;
}
```

Logical NOT !: logical NOT, true only if the operand is false

e.g !exp.

Example:

```
int main(){
int a = 5;
int b = 3;
cout<<(!(a>b && a<b));
return 0;
}
```

Date

d- Bitwise Operator:

Bitwise operator are used to perform individual bits of number it can be used only integer type value not float, double etc

Types:

1- Bitwise AND &
2- Bitwise OR |
3- Bitwise XOR ^
4- One's complement ~
5- left shift <<
6- Right shift >>

(Binary: 1, 2, 3 — Unary: 4, 5, 6)

our hardware work on bit so we need bitwise

1- Bitwise AND &:

Example:

```
int a = 12;
int b = 14;
int c;
c = a & b
cout << "The result is" << c;
return 0;
}
```

2|12
2|6 — 0
2|3 — 0
1 — 1
1100₂

2|14
2|7 — 0
2|3 — 1
1 — 1
1110₂

AND
1100₂
1110₂
1100

8+4=12

2	3
1	1

2	2
1	0

2	28
2	14 — 0
2	7 — 0
2	3 — 1
	1 — 1

2- Bitwise OR | :

Example

```
int a = 12 ; b = 10 ;
int c ;
c = a | b
cout << "The result is " << c ;
return 0 ;
}
```

11100

14

2	10
2	5 — 0
2	2 — 1
	1 — 0

2	12
2	6 — 0
2	3 — 0
	1 — 1

1010

1100

1110

$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$8 + 4 + 2$

(14)

Date

3- Bitwise XOR (exclusive OR) :

Example

```
int a=12 ; b=10;
int c;
c = a^b
cout << "The result is:" << c;
return 0;
}
```

1010
1100
0110
$0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$
$4 + 2$
6

XOR

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

In exclusive OR like value give ans 0 & unlike give 1.

Date

4. One's Complement ~ :

Example:

```
int main( ){
int a = 10;
int z;
z = ~a;
Printf ("%d", z);
return 0;
}
```

2	10
2	5 — 0
2	2 — 1
	1 — 0

1010

int has 2 byte 2bytes - 16bit

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	0	1	0

15	14
0	0

Not gate

1	1	1	1	1	1	1	1	1	1	0	1	0	1
2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

1	1
2^{15}	2^{14}

65525

Date

5 Left Shift <<

Example

```
int main( ){
int a = 10;
z = a<<1
Printf ("%d", a);
return 0;
}
```

0 0 0 0 0 0 0 0 0 1 0 1

0 0 0 0 0 0 0 0 1 0 1 0

Date 18-Apr

Flow Of Control / Control Structure:

- Sequence (Step by Step)
- Selection Statement / condition, if, else switch
- Iteration Statement
- Jumping Statement

ab tk pee the ithe Phr padhe hai

→ Selection Statement:

① Condition:

We use the if ... else statement to run one block of code under certain condition and another block of code under different conditions

② if - Statement:

Syntax

```
if (condition){
  // body of if statement
}
```

Date

Example :

```
int main ( ){
if (50 > 10) {
Print f ("50 greater than 10");
}
Print f ("Sorry");
return 0;
}
```

agar ye wrong hua to

ye Print hoga

Output = 50 greater than 10

③ If else statement :

if ... else

Syntax

```
if (condition) {
// block of code if condition is true
}
else {
// block of code if condition is false
}
code after if ... else.
```

Agar ye true na ho to → ye wala Print hoga | agar ye bhi true na hua to ? | yeh print hoga aur agar yeh bhi false hua to Blank

Date 19-Apr

Example:

```
int main( ){
int x = 10;
int (x < 20){
Printf ("Welcome");
}
else{
  Printf ("Try again");
  }
return 0;
}
```

```
Printf ("Sorry");
}
```

④ else if Statement:

Syntax if statement

Syntax

```
if (Condition){
// block of code
}
else if (condition 2){
// block of code
}
else{
{// block of code}
```

Date

```
int main(){
  int x = 22;
  if (x < 10){
  Printf("NO");
  }
  else if (x < 20){
  Printf ("Try again");
  }
  else {
  Printf ("Yes");
  }
  return 0;
  }
```

5- Short hand if else (Ternary operator)

Syntax.

Variable = (Condition) ? (expression True : expression false.

Example if.. else

```
int main(){
int n = 20;
if (n < 18){
Printf("Welcome");
}
else{
Printf("Good day");
}
return 0;
}
```

like
if else ki condition ko
short kardia. Date

Ternary Operator Example:

```
int main ( ){
    int x = 20;
(x<20)? Printf("welcome"):Printf("
"Good day");
return 0;
}
```

Good day.

⑥ Nested if... else statement:

Sometimes, we need to use an if statement inside another if statement This is known as nested if statement

Syntax:

```
// outer if statement
if (condition 1){
 // statements
 // inner if statement
  if(condition 2){
  // statement
  }
```

Example:

```
int main (){
int mark = 100;
if (mark >=50){
Printf("You passed");
if (mark == 100){
Printf("Perfect");
}
}
else{
   Printf("you failed");
}
return 0;
}
```

→ Condition k andar condition paraha hai

Output: You passed

④ Switch:

The switch statement allow us to execute a block of code among many alternative

Syntax

```
switch (expression){
case X:
// code block
break;
case y:
// code block
break;
default
// code block
}
```

Date 19-Apr

Example:

```
int main ()
int day = 3;
Switch (day) {
case 1:
Printf ("Monday");
break;
Case 2:
Printf (" Tuesday");
break;
Case 3:
Printf ("Wednesday");
break;
Case 4:
Printf ("Thursday");
break;
default:
Printf ("Try again");
}
return 0;
}
```

This one denote the case

if There is no break then it will give all the case after 3 +..

Output- wednesday

Date 25-Apr-22

2- Iteration Statement:

Loop

Loop statement are used to execute the block of code repeatedly for a specified number of time or until it meets a specified condition

1- For loop:

For loop is used when we want to execute block of code known times

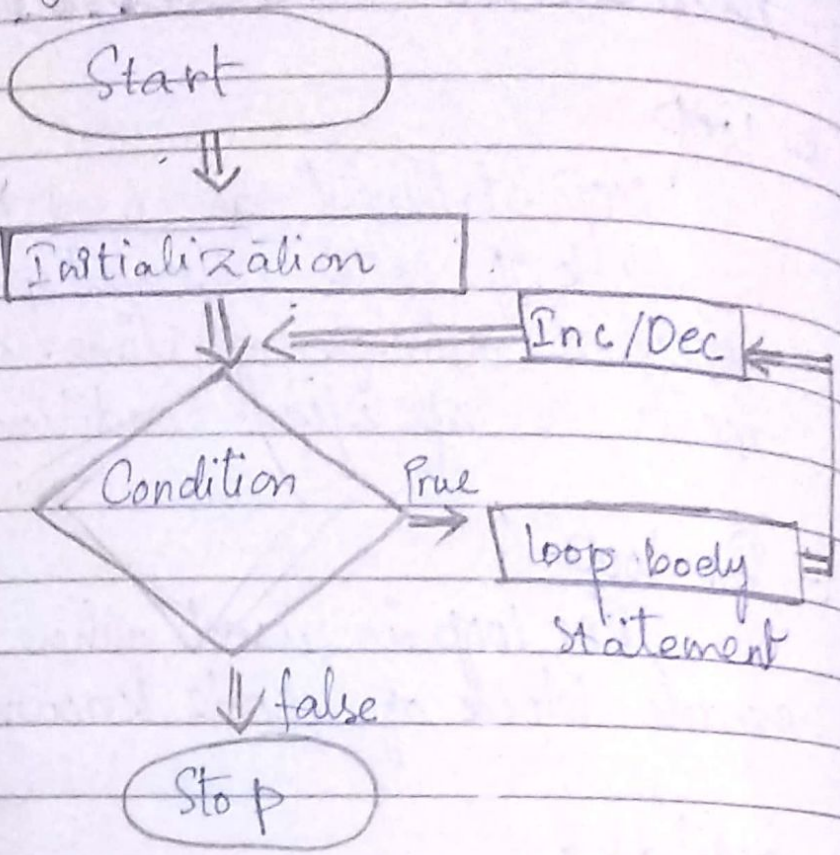
Syntax:

```
for(intialization, condition, updation){
  // body of loop
}
```

Part → 1) Initialization
2) Condition
3) Update → Increment / Decrement
4) Body of loop

Date

Flow Chart




```
int main ( ){
int i ;          Initial  cond   updat
for(int i=5 ; i>0 ; i--){
Printf("%d\n", i);
}
return 0;
}
```

5
4
3
2
1

```
int i;
for(int i =1 ; i<=5 ; i++)
Printf("%d/n", i);
}
return 0;
}
```

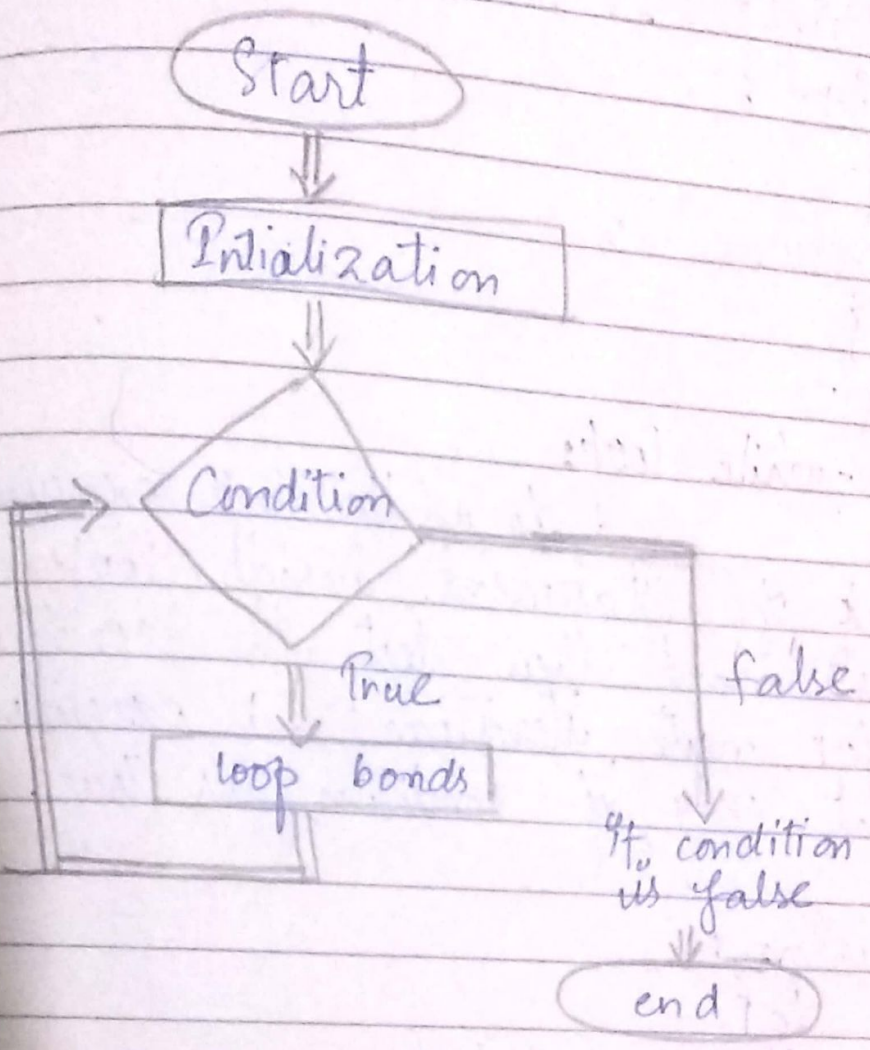
1
2
3
4
5

Date

②- while loop:

while loop will execute a block of statement as long as a test expression is true.

Flow Chart:



Syntax → only condition.

```
while(Condition){
  // body of the loop
   inc/dec
}
```

i=i+1

Date

Example:

```
int main() {
  int i = 1
while(i<=5){
 Printf("%d \n", i);
 i++;
}
return 0;
}
```

1
2
3
4
5

9/may.

③ do-while loop:

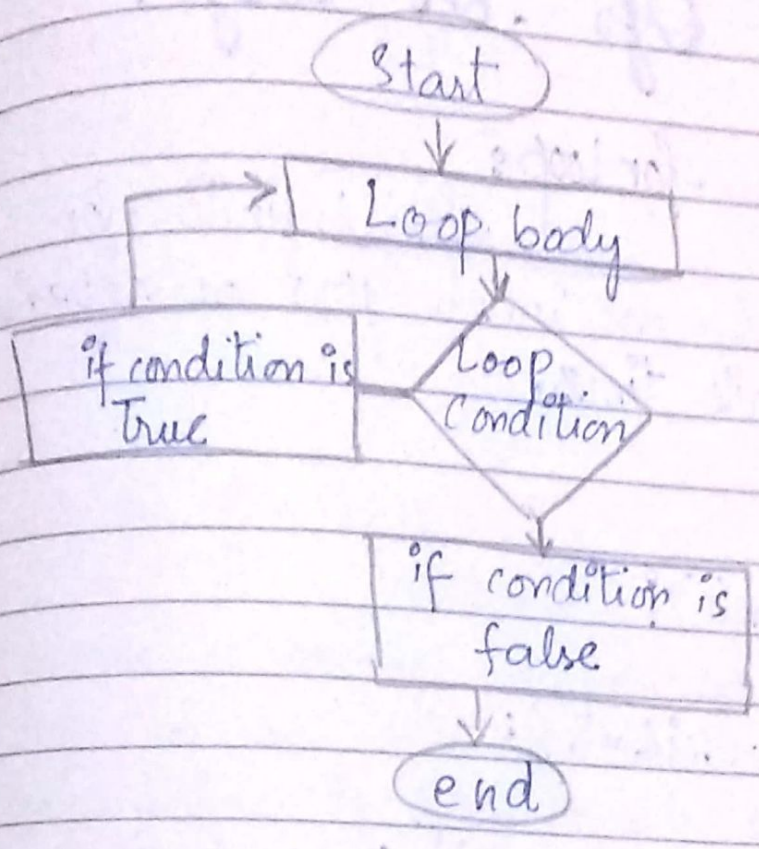
do while loop execute block of statement inside loop body first and then test the condition for next iteration and executes next only if condition is True.

Syntax:

```
do{
  // body of loop
}
while (Condition);
```

FLOW CHART

Date 9-May-22



Example:

```
int main( ){
int i = 1;
do {
Printf ("%d\n", i);
++i ;
}
while (i<=5);
return 0;
}
```

Types Of For loop:

i- Infinite for loop:

An Infinite for loop is the one which goes on repeatedly for infinite times.

```
int main (.){
int i;
for(i=1 ,i<=5);
{
Printf ("Hellow world");
}
return 0;
}
```

ii) Empty for loop:

Empty for loop is the one which has got no body in simple words, it contains no statement or expressions in it's body it can be used for time consuming purpose.

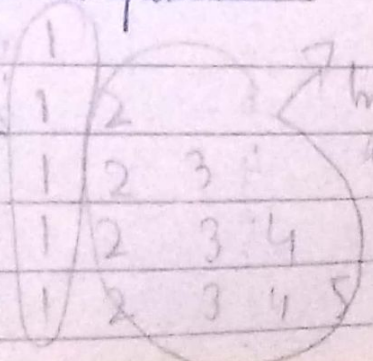
```
int main ( ) {
for (i = 1 ; i <= 5 ; i++ ) {
}
return 0;
}
```

iii) Nested for loop:

Nested for loop refers to the process of having one loop inside another loop. we can have multiple loops inside one another. The body of one for loop contain the other and so on.

Syntax:

```
for (initialization ; test ; update)
{
for (initialization ; test ; update)
{
// body of the inner loop.
}
{// body of outer loop // optional
}
```



Date

Example:

```
int main ( ) {
int i , j ;
for (i = 1 ; i <= 5 ; i++ )
{
for (j = 1 ; j <= 5 ; j++ )
Printf ("%d" , j );
Printf ("/n") ;
}
return 0;
}
```

outer loop → inner loop

Output

```
1
2
3
4
5
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

("/n") kiwaja se alag alag line peraya mean (verticle) aur i = 2 tha tu do dafa output aya.

```
1
2
3
4
5
```

5- Jump Statement:

Jump statement are used to alter or transfer the control to other section or statements in your program from the current section

a- Break jump statement
b- continue jump statement
c- goto jump statement

a- Break Jump statement:

Break statement is used to end the loop immediately after the encounter.

It is used to terminate the loop and program control resumes at the next statement following the loop.

Example:

```
int main() {
int a = 1;
while (a<=10)
{
if (a == 5)
break;
Printf ("%d, a);
a++
}
return 0;
}
```

output

1 2 3 4

Date

(5) Continue jump Statement:

Continue jump statement like any jump statement interrupts or changes the flow of control during the used in loop

Example:

```
int main ( ) {
int i;
for(i=1; i<=10 ; i++)
{
if (i==3)
}
continue
{
Printf ("%d\n";i);
}
return 0;
}
```

→ matlab jese hi i=3 ai to ase continue karde aur 3 print nahi hoga aur neche jump hojae ga

output

1
2
4
5
6
7
8
9
10

Date

⑤ goto jump statement:

goto jump statement is used to transfer the flow of control to any part of the program desired.

o- OR -o

It is used to transfer the program control from one statement to another statement (where label is define)
There are two way to ~~control~~ call goto statement.

Syntax 1
```
goto label;
// block of statement
label:
```

Syntax 2:
```
label:
// block of statement
goto label;
```

Example:
```
int main () {
int a = 1
Start:
Printf ("%d\n", a);
a++;
if (a <= 10)
goto start;
return 0;
}
```

Date

→ Derived Datatypes

1- Arrays
2- Function
3- Pointer
4- Reference

1- Arrays:

An Array can be defined as an infinite collection of homogenous (similar type) element. Array are always stored in consecutive (specific) memory location.

means same datatype k values 1 sath store hoage.

```
int a = 5;
    b = 10;
    c = 20;
```

```
      0   1   2   3
int a| 5 | 10| 20| 30|
```

No.

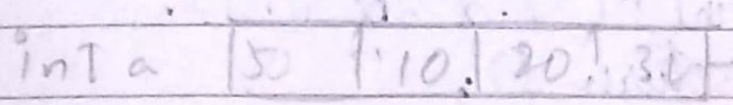
Date

Types Of Arrays:

1- One - dimensional arrays
2- Two - dimensional arrays
3- Multi-dimensional arrays

1- One - Dimensional arrays:

A type of array in which all elements are arranged in the form of a list is known as 1-D array or single dimensional array a linear list.



↔ Declaring 1-D array:

Syntax:

datatype identifier [length]
int marks [5];

↔ Initialization:

The process of assigning values to array elements at the time of array declaration is called

Date

initialization.

Syntax.

datatype identifier [length] = [list of values]

int x [ ] = {20, 30, 40, 50, 35}
0 1 2 3 4

Example:

```
int main (){
int x [ ]= {25, 50, 40, 30, 20};
Printf (" %d ", x);
Printf ("%.d", x [1]);
return 0;
}
```

Searching In Array

Searching is a process of finding the required data in the array searching become more important when the length of the array is very large.

There are two techniques to searching

1- Sequential Search
2- Binary Search

No.

Date

1- Sequential Search:

Sequential search is also know as linear or serial search.

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
a =	2	8	7	9	15	19	3

1- Visit the first element of the array and compare its value with value with the required value
2- If value of array matches with desired value, the search is complete.
3- If the value of array does not match move to next element and repeat some process.

if we have Keyword 9

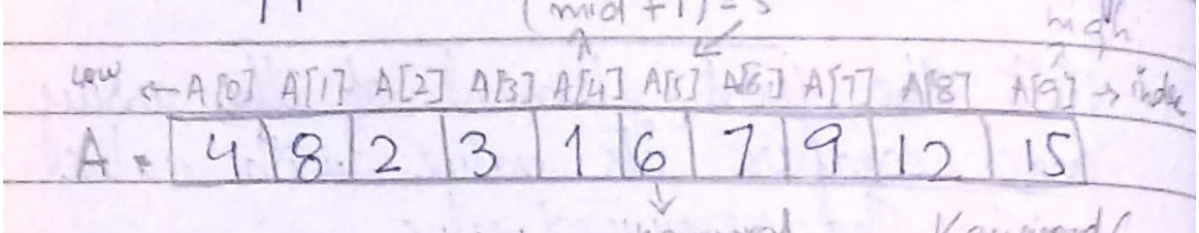
9 a = | 2 | 8 | 7 | 9 | 15 | 19 | 3 |

It will check word by word when Keyword match the 9 will stop.

Date

2-Binary Search:

Binary Search is a quick method of searching for value in the array binary search is very quick but it can only search an sorted array it cannot be applied on an unsorted array.



Formula: we gain our keyword at mid+1

$$Mid = low + \left(\frac{high - low}{2}\right)$$

$$Mid = 0 + \left(\frac{9 - 0}{2}\right)$$

$$M = \frac{9}{2} = 4.5 = 4 \rightarrow \text{index mai}$$

agar 2.5, 1.5, 3.5 and 4.5 ke jo bhi aye to hum 2 1 3 4 etc lenge respectively agar 2 series ho to 1 mai .5 decrease kr k or ek mai increase kr lenge ↔ Agar mid point pr hame hamara keyword nahi mila to hum agar keyword age ho to (mid+1) aur peeche ho to (mid-1) karenge. aur phir wo us k mid banjata hai (4-1)=3 Ye mid hoga aur agar keyword is se bhi peeche hoga to phir -1 kar dete hai

Date

Sorting Array:

Sorting is a process of arranging the value of array in a particular order. An array can be sorted in two order.

1- Asending Order:

5	10	15	20	25	30

2- Desending Order:

30	25	20	15	10	5

There are 2 Techniques
1- Selection Sort
2- Bubble Sort

Selection sort:

Selection sort is a techniques that sort an array it selects an element the array and move to it's proper position.

Date

1- Find the minimum value in the list
2.- Swap it with value in the first position.
3- Sort the reminder of the list excluding the first value.

5	4	2	3	6
0	1	2	3	4

min ← swap → location

Man size = 5
min = 5
location = 2
min se chota

2	4	5	3	6

min ← swap → loca

2	3	5	4	6

min ← → location

2	3	4	5	6

→ In ascending order

ye 2 values leker greater then aur less then se compare karta hai.

Date 17 – May.

2- Bubble Sort:

Bubble sort is also known as exchange Sort, it repeatedly visits the array and compares two item at a time.

	0	1	2	3	4	5	6
A =	4	2	1	5	3	7	6

No of elements = 7

4 > 2 → Swap

2 4 1 5 3 7 6

4 > 1 - Swap

2 1 4 5 3 7 6

4 > 5 - no swap because 4 < 5

2 1 4 5 3 7 6

5 > 3 - swap

2 1 4 3 5 7 6

5 > 7 - no swap because 5 < 7

2 1 4 3 5 7 6

7 > 6 - Swap

2 1 4 3 5 6 7

This is one step then after do it for "2" this process is continue till series arrange in ascending order

work as matrix Date 17-May

2. Two - Dimensional Array:

Two D-array can be considered as table that consists of rows & columns. Each element in 2-D array referred with the help of two index: One index indicates row and second indicates the column.

→ Declaration:

Datatype Arrayname [row] [Column];

→ Initialization:

Datatype Arrayname [row] [column] = {val

int x [2] [3] = {2, 3, 4, 5, 6

2 × 3 = 6

| 1 | 2 | 3 | 4 | 5 | 6 |

$$\begin{bmatrix} 2 & 3 & 4 \\ 5 & 8 & 7 \end{bmatrix}$$ → matrix

Example:

```
int main ( ){
int i=0, j=0;
int arr[4][3] = {(1,2,3), (4,5,6), (7,8,9), (10,11,
for(i=0; i<4; i++){   // Row
for (j=0; j<3; j++){  // Column
Printf("%d][%d]= %d\n", i, j arr[i][
}
}
return 0;
```

arr,

Date

Output:

arr [0] [0] = 1
arr [0] [1] = 2
arr [0] [2] = 3
arr [1] [0] = 4
arr [1] [1] = 5
arr [1] [2] = 6
arr [2] [0] = 7
arr [2] [1] = 8
arr [2] [2] = 9
arr [3] [0] = 10
arr [3] [1] = 11
arr [3] [2] = 12

In Matrix

	0	1	2	
0	1	2	3	—
1	4	5	6	—
2	7	8	9	—
3	10	11	12	—

row

Column

Date 17-May

3- Multi Dimensional Array:

Multidimensional arrays can be defined as an array of array.

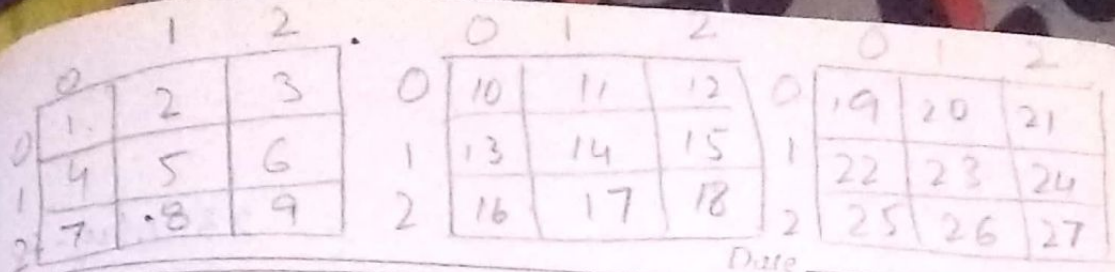
→ Declaration:

Datatype arrayname [size1] [size 2]..... [Sizen]

Note: Size of ~~array~~ Multidimensional array can be calculated by multiplying the size of all the dimension.

Size of a [5][2] = 10

Size of a [5][2][2] = 20



Date

```
int main(){
int i, j, x;
int a[2][2][2] = {{(1,2),3,4}},(5,6),(7,8)
int a[3][3][3] = {{(1,2,3)(4,5,6),(7,8,9)},
                  {(10,11,12),(13,14,15),(16,17,18)},
                  {(19,20,21),(22,23,24),(25,26,27)}};
Printf("Any Msg");
for(i=0; i<3; i++){
for (j=0; j<3; j++){
for (x=0; x<3; x++){
Printf("%d\t", a[i][j][x]);
}
Printf("\n");
}
Printf("\n");
}
return 0;
}
```

arr[0][0][0] = 1
arr[0][0][1] = 2
arr[0][0][2] = 3
arr[0][1][0] = 4
arr[0][1][1] = 5
arr[0][1][2] = 6
arr[0][2][0] = 7
arr[0][2][1] = 8
arr[0][2][2] = 9
arr[1][0][0] = 10
arr[1][0][1] = 11
arr[1][0][2] = 12

Date 23-May-22

Pointer:

Pointer is a variable that hold address of another variable of same datatype.
The size of pointer variable is 2 bytes while working with pointer we need two unary operator

1) & → Address of operator.
2) * → value at address operator.

Syntax of pointer.

Datatype * variablename.

Pr address

```
int main ( ) {
int i = 10;
int * Pr;
Pr = &i;
Printf ("%d", Pr);
Printf ("%d", *Pr);
return 0;
}
```

Date

&: address of operator help us get the value that has been stored in a memory address:

For Example:

if we have a variable given the name X stored in a address 0x538 and storing the value 10.

→ The & (address of operator) will return 0x538

→ The ~~dereference~~ operator (*) will return 10

dereference → de-reference

Types Of Pointer:

i - void Pointer:

A pointer which is declared by the help of void keyword is called void pointer.
It can hold any type of address of float, int, double void pointer also know as generic pointer.

Example

```
void main(){
int a = 10;
P = &a
Printf("%d", *(int*)(P));
return 0;
}
```

xehamien (int*) — iska value batata hai

agar yahan & operator hota To ye hamme iska address batata

Agar humien Char

output 10

Date

2- Null pointer:

A pointer variable that is initialize with the NULL value at the time of pointer declaration is called NULL pointer.

Syntax: Data type *variable = 10;

```
int main (){
int * P = NULL
Printf ("%d" * P );
  return 0;
}
```

Null pointer doesn't point any memory location.

Output khali ayega.

Date 30-May-22

3- Wild Pointer:

A pointer variable that not initialized with any address is called wild pointer. It is also known as bad pointer because it holds the address of random memory location.

```
int main(){
int *P;
Printf("%d", *P);
return 0;
}
```

Syntax

datatype *var-name.

Date

4- Dangling Pointer:

A pointer variable that hold the address of inactive area location than the pointer is called Dangling pointer

Syntax

* Var name

```
int main ( ){
int * P;
{
 int a = 5;
 P = &a;
}
P;
```

Output

Blank

5- Pointer to pointer:

A pointer variable which hold a address of another pointer variable is called pointer to pointer.

Syntax.

```
** variable name;
```

Example:

```
int main( ) {
int a = 5, *P; **q;
P = &a
q = &P
Printf ("%.d", a );
Printf ("%.d", &a);
Printf ("%.d", P);
Printf ("%.d", &P);
Printf ("%.d", *P);
Printf ("%.d", q );
Printf ("%.d", &q);
Printf ("%.d", **q);
return 0;
}
```

Variable name	Value	Address
a	5	100.5
*P	100.5	1002
**q	1002	1004

5
100.5
100.5
1002
5
1002
1004
1002

Jiske page() ho to wo function hota hai we use function to call function agn & again.

Date 06-06-2022

Function:

Function is the block of statement which can call by another function is called function.

-OR-

Function is a set of statement That takes input do some specific computation and produced output.

Syntax

```
ReturnType functionname
{
  body of statement
}
```

Example

```
# include <Stdio.h>
void fname( )
{
  Printf ("I am learning C language");
}
int main( )
{
fname ( ) {
return 0;
}
```

hum is se multiple times solve kar sakte hai-

Prototype
Body of function
calling
Arguments etc

There are 2 Types 1- Built in function
2- User defined function.

1- Built in function:

Built in function are also called library function. we do not need write them o urselves. we directly use ithen function in our code. These function are placed in the hearfiles of C for example <C math>, <String>, <Stdio.h>.

2- User defined function:

C language also allow it user to define their own functions. These are the user defined function. we define the function any where in the program and then call these function from any part of the code.

T.

Date

9 - May - 22

1- write a C language program to print table of 2 using for loop

2- write a C language program to print table of 5 using do while.

3- Write a code to print the following star pattern.

16 - May - 22

4 - write a code to Access Array element

5 - write a code to change an array element

6 - write a code 1D array.

7 - Make a calculator using function.