UNIT-2

Decision Making and Arrays.

* Branching And Loops :-

* Branching Statements :-

We will implement these kind of statements in we want to execute some set of statements if the condition is satisfied.

There are 2 types:

1) Variants of if
2) Switch

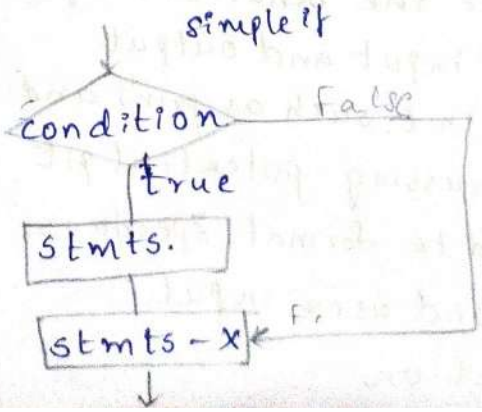
* Varients of if :
⇒ simple if
⇒ else if
⇒ else if ladder
⇒ nested if

* Simple if :-

Syntax: If condition statements

```
if (condition)
{
    statements;
}
```

Flowchart :-



* Write a C p
#include <s
int main (
{
int age
scanf("
if (age
{
Print
}
Print f
return
}

* else if :
Syntax: if
{

flow chart

st

s.

nts in
ts if

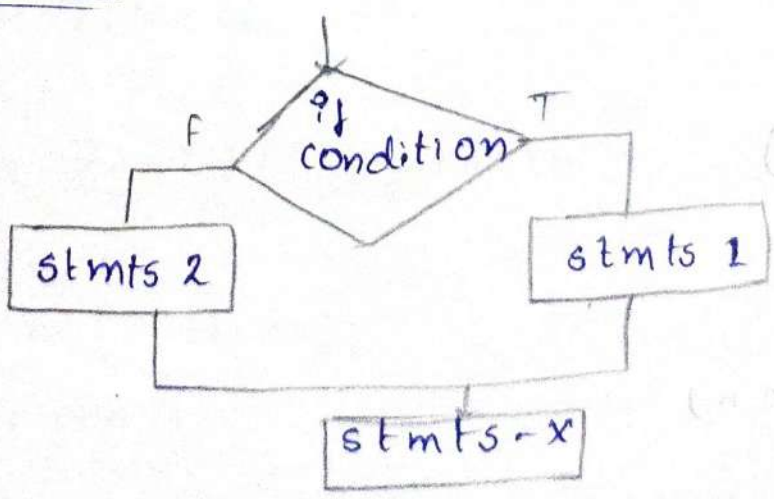
* Write a C program whether a person is retired

```
#include<stdio.h>
int main()
{
   int age;
   scanf("%d", &age);
   if (age >=60)
   {
      Printf("Retired\n");
   }
  Printf("Hello");
  return 0;
}
```

* else if :-

Syntax :- if (condition)
{
stmts 1;
}
else
{
stmts 2;
}
stmts-x;

Flow chart :-



* Write a C program whether a person is eligible to vote or not.

```
#include <stdio.h>
int main()
{
    int age
    scanf("%d", &age);
    if(age>=18)
    {
      Printf("Eligible to Vote\n");
    }
    else
    {
      Printf("Not Eligible to vote\n");
    }
    Printf("Hello");
    return 0;
}
```

* 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 sta
}
sm ts-x;
flow cha
```

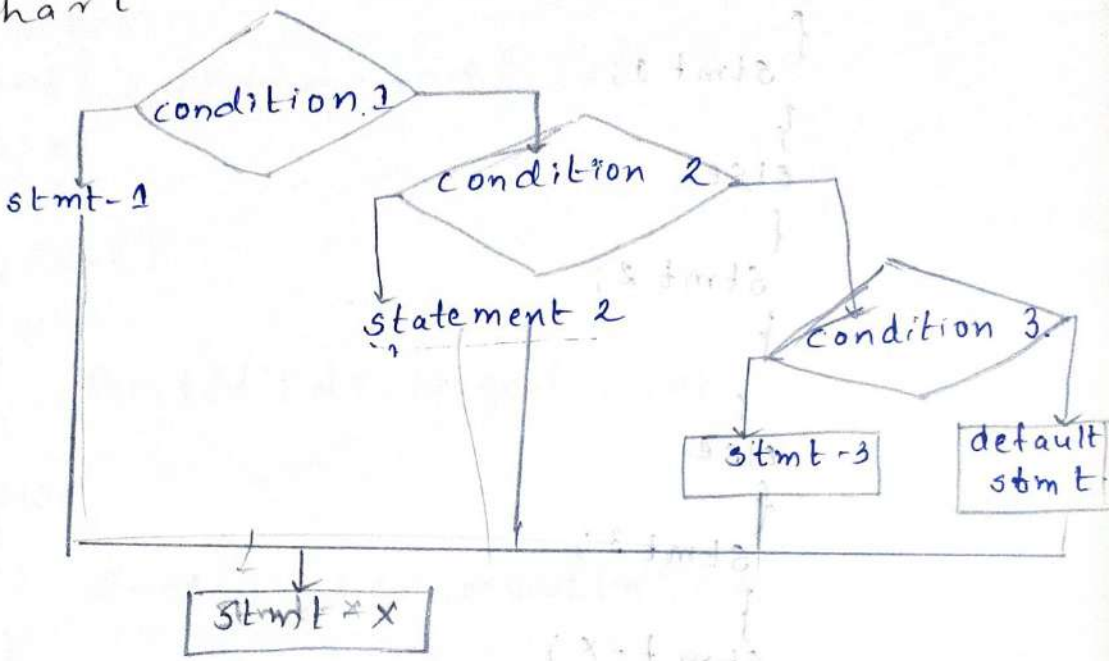
```
str
```

* Write a

```
#include
int main
{
  int n
  scan
  if (n=
  {
    Pri
  }
  else if
  {
    }
    Prin
    ret
}
```

```
else
{
default statement;
}
smts-x;
```

flow chart



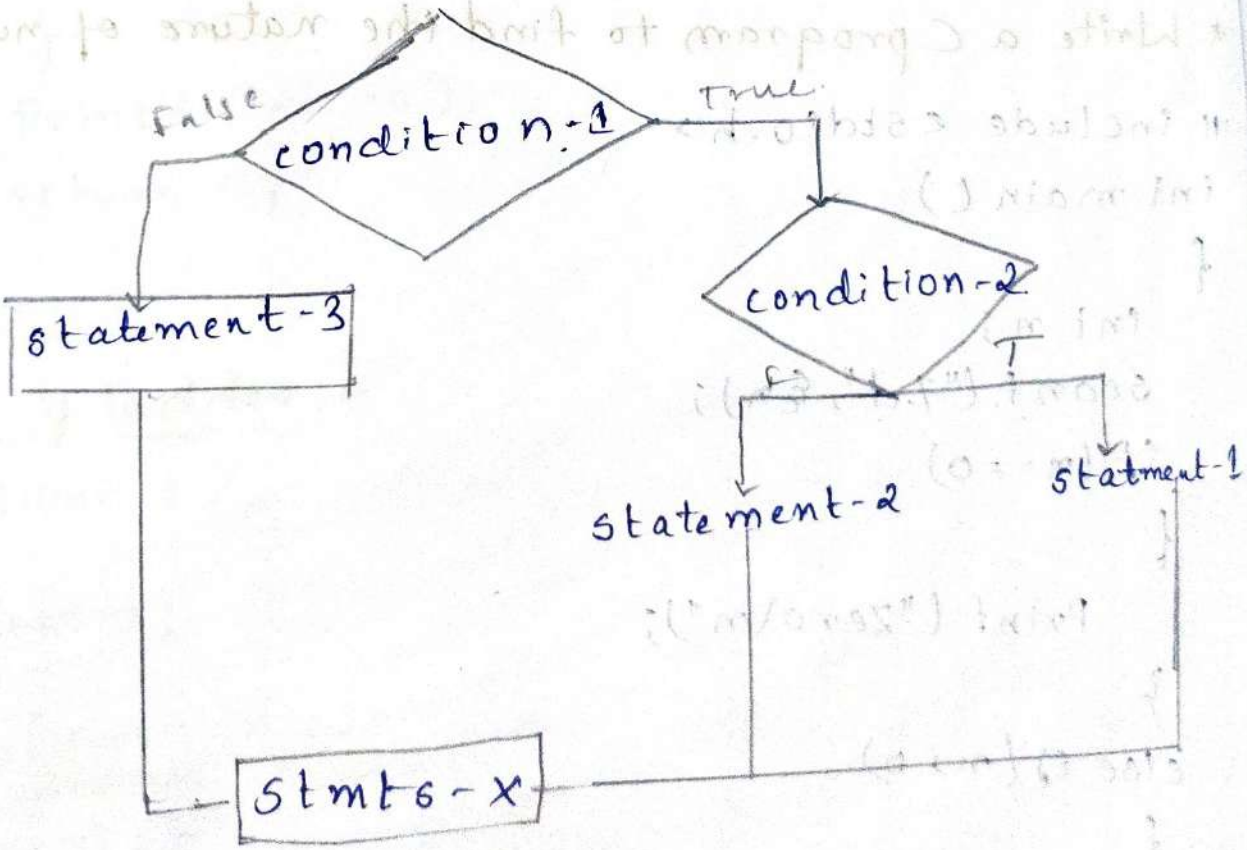
* Write a C program to find the nature of number.

```
#include <stdio.h>
int main()
{
   int n;
   scanf("%d", &n);
   if (n==0)
   {
      Printf ("zero\n");
   }
   else if (n>0)
   {
      Printf ("Negative\n");
   }
   Printf("Hello);
   return 0;
}
```

* Nested if:-

Syntax:-

```
if(condition 1)
{
if (condition 2)
{
 stmt 1;
}
else
{
 stmt 2;
}
 }
else
{
 stmt 3;
}
stmt-x;
```



* Write a C program to find the largest of 3 numbers

```
#include<stdio.h>
int main()
{
    int a,b,c;
    scanf("%d %d %d",&a,&b,&c);
    if(a>b)
    {
      if(a>c)
      {
        Printf("%d is largest\n",a);
      }
      else
      {
          Printf("%d is largest\n",c);
      }
    }
    else
    {
        if(b>c)
        {
          Printf("%d is largest\n",b);
        }
        else
        {
            Printf("%d is largest\n",c);
        }
    }
    return 0;
}
```

* Switch:-

⇒ Switch is a multiway decision making statement.

⇒ It is a branching statement in C.

⇒ The switch statement provides another way to decide which statement to execute next

⇒ The switch statement evaluates an expression, then attempts to match the result to one of several possible cases.

⇒ Each case contains a value and a list of statements

⇒ The switch case statement is used when we have multiple options and we need to perform a different task for each option.

⇒ The flow of control transfers to statement associated with the first case value that matches.

⇒ often a break statement is used as the last statement in each case's statement list.

* Keywords used in Switch statement:-

⇒ The keywords used are

- * Switch
- * case
- * default
- * Break

Syntax:-

switch (

{

case <e

case

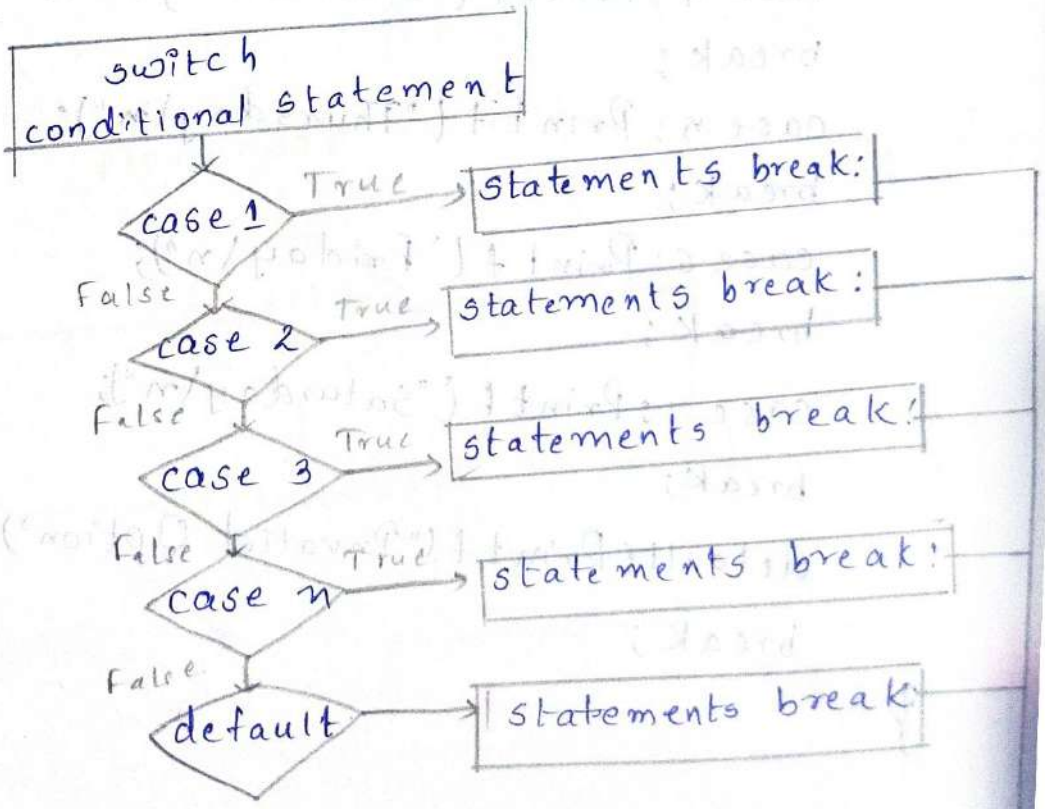
case

cas

Syntax:

```
switch (<exp>)
{
case <exp-val-1>: statements block-1
                  break;
case <exp-val-2>: statements block-2
                  break;
case <exp-val-3>: statements block-3
                  break;
case <exp-val-N>: statements block-N
                  break;
    default: default statements block
}
Next-statement;
```

Flowchart:



* Write a C program to print the day of a week using switch case

```
#include <stdio.h>
int main()
{
    int n;
    scanf("%d", &n);
    switch(n)
    {
        case 1 : Printf("Sunday\n");
        break;
        case 2 : Printf("Monday\n");
        break;
        case 3 : Printf("Tuesday\n");
        break;
        case 4 : Printf("Wednesday\n");
        break;
        case 5 : Printf("Thursday\n");
        break;
        case 6 : Printf("Friday\n");
        break;
        case 7 : Printf("Saturday\n");
        break;
        default : Printf("Invalid Option");
        break;
}
```

Loops :-

If I wan
and again
There ar
1) Entr
2) Exi
* For e
Do

Synta
varia
whi
{
s
}

Loops:-

If I want to repeat same set of statements again and again then we use loops.

There are 2 types of loops

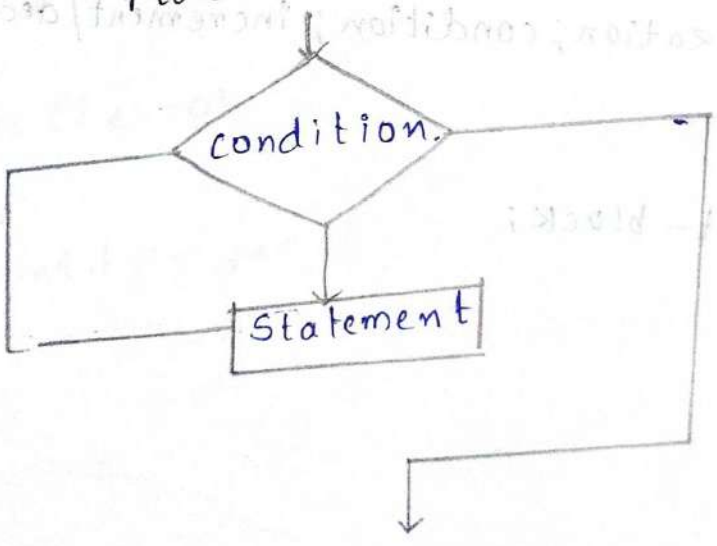
1) Entry control
2) Exit control.

* for & while are entry control loops. whereas Do while is exit control loop.

Syntax: While

```
variable initialization;
while (condition)
{
 statements;
 variable increment or decrement ;
}
```

flowchart

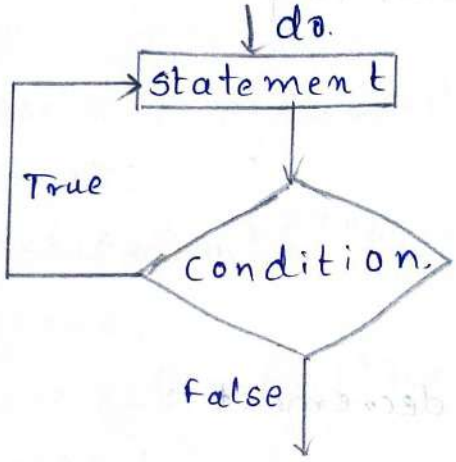


nxt stmt.

Syntax:

```
do.
{
  ....
  .....
}
while (condition)
```

flowchart:

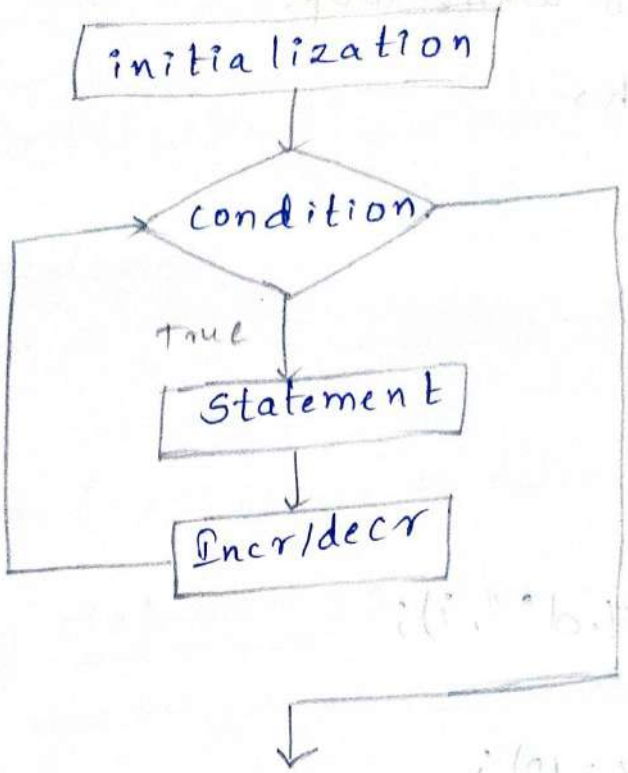


* for loop:-

Syntax:

```
for (initialization; condition; increment/decrement)
{
  statement-block;
}
```

flowchart



* Write a C program to print the numbers from 1 to 10 using while loop.

```
#include<stdio.h>
int main()
{
    int i;
    i=1;
    while(i<=10)
    {
      Printf("%d", i);
       i++
    }
    return 0;
}
```

* Write a C program to print the numbers from 1 to 10 by using do while loop.

```
#include<stdio.h>
int main()
{
    int i;
    i=1;
    do
    {
        Printf("%d ", i);
        i++
    }while(i<=10);
    return 0;
}
```

* Write a C program to print the numbers from 1 to 10 by using for loop.

```
#include<stdio.h>
int main()
{
    int i;
    for(i=1; i<=10; i++)
    {
        Printf("%d ", i);
    }
    return 0;
}
```

* Nested
for (init
{ for (i
{
}
}
* NOTE:
* Jumpi
⇒ These
⇒ There
1) Brea
2) cont
3) En
4) go
* Br
The
contr
Synt
bre
flow

* Nested for loop:

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

* NOTE:- for (;;) leads to infinite loop

* Jumping statements:-

⇒ These are also known as unconditional statements

⇒ There are 4 types:

1) Break
2) continue
3) Exit
4) go to.

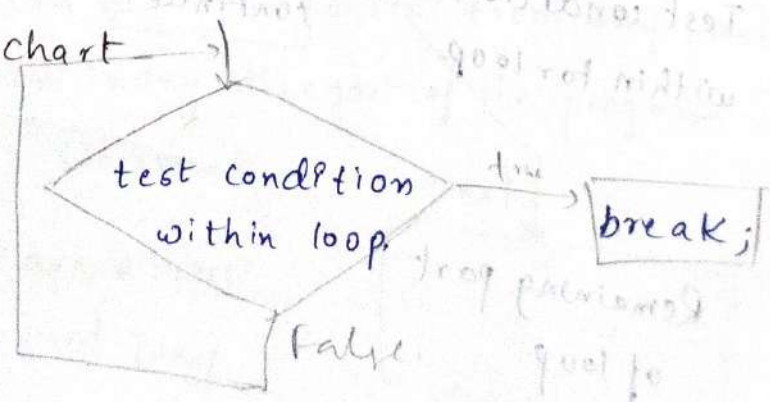
* Break statement:-

The break statement is used within the looping control statements, switch statement and nested loops.

Syntax for the break statement is:

```
break;
```

Flowchart



* Write a C program to implement break

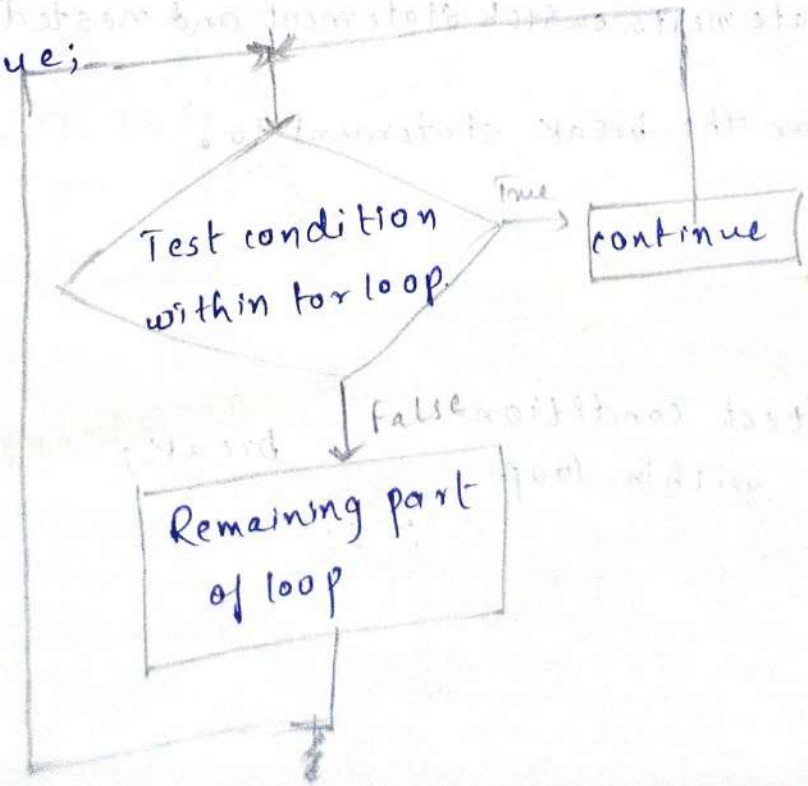
```
#include<stdio.h>
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==5)
        break;
        else
        Printf("%d", i);
    }
    return 0;
}
```

* continue:-

It is sometimes desirable to skip some statements inside the loop. In such cases, continue statement is used

Syntax:

```
continue;
```



* Write a C program to implement continue.

```
#include<stdio.h>
int main()
{
    int i;
    for(i=1; i<=10; i++)
    {
        if(i==5)
        continue;
        else
        printf("%d ",i);
    }
    return 0;
}
```

Output:-

1 2 3 4 6 7 8 9 10.

* Exit:

This ~~prog~~ function is used to come out of the program

* Go to:

* go to statement is used for unconditional jumping.

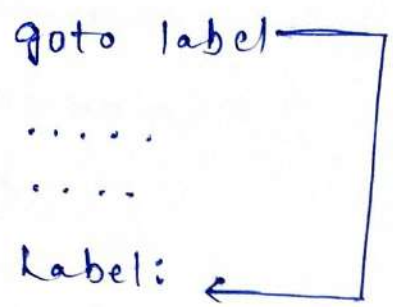
* We can move the control from any part of the Program to any other part of the program with the help of goto statement

* There are 2 types

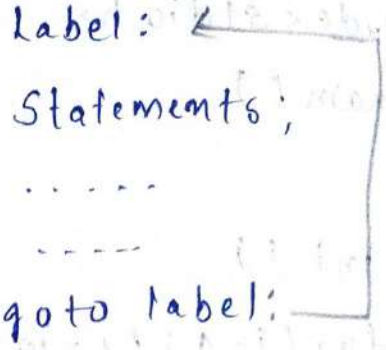
1) Forward jump

2) Backward jump

* Forward jump



Backward jump



* Write a C program to implement goto ~~statement~~

```
#include<stdio.h>
int main()
{
    int n = 1;
label:
    Printf("%d ", n);
    n++;
    if(n<=10)
    goto label;
    return 0;
}
```

output:

1 2 3 4 5 6 7 8 9 10

53) Write a C program to print fibonacci numbers upton

```
#include <stdio.h>
int main ()
{
    int n, i, a, b, c;
    scanf ("%d", &n);
    a = 0;
    b = 1;
    Printf ("%d %d ", a, b);
    while (i <= n-2)
    {
        c = a+b.
        Printf ("%d ", c);
        a = b;
        b = c;
        i++;
    }
```

* Write a C program to check whether a given number is prime or not.

```
#include <stdio.h>
int main()
{
    int n, c = 0, i;
    scanf("%d", &n);
    for (i = 2; i < n; i++)
    {
        if (n%i == 0)
        c++
    }
    if (c == 0)
    Printf("%d is prime", n);
    else
    Printf("%d is not Prime", n);
    return 0;
```

* Write a C program to generate the following Pattern

1)
```
1
2 2
3 3 3
4 4 4 4
```

2)
```
1
1 2
1 2 3
1 2 3 4
```

1)
```
#include <stdio.h>
int main()
{
    int i, j, n;
    scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
      for(j=1; j<=i; j++)
      printf("%d", i);
      printf("\n");
    }
}
```

2)
```
#include <stdio.h>
int main()
{
    int i, j, n;
    scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
      for(j=1; j<=i; j++)
      printf("%d ", j);
      printf("\n");
    }
```

3)
```
#include <
int main
{
  int
  scan
  for(
  {
  }
}
```

4)
```
#in
int
{
```

3) * * * *
   * * * *
   * * * *
   * * * *

```
#include<stdio.h>
int main()
{
  int i,j,n;
  scanf("%d",&n);
  for(i=1;i<=n;i++)
  {
    for(j=1;j<=n;j++)
    printf("* ");
    printf("\n");
  }
}
```

4) *
   * *
   * * *
   * * * *

```
#include<stdio.h>
int main()
{
  int i,j,n;
  scanf("%d",&n);
  for(i=1;i<=n;i++)
  {
    for(j=1;j<=i;j++)
    printf("* ");
    printf("\n");
  }
}
```

Arrays:-

Defn: An array is a fixed size sequenced collecti
of elements of the same data type

Examples:
⇒ List of employes in an organization
⇒ List of products and their cost
⇒ Test scores of a class of students
⇒ List of customers and their telephone numbers etc

* Types of arrays:-
⇒ One-dimensional arrays
⇒ 2-D arrays
⇒ Multidimensional arrays.

* 1-D arrays:
A list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable or a 1-D array.

* Declaration of 1-D array:
⇒ Datatype arrayname [size];
where size represents the max no. of elements that an array can hold.

* Intialisation of 1-D array:-
⇒ The initialisation of 1-D array can be done in 2 ways
1) Compile time
2) Runtime

* Compile time:
The different
follows:
1) Initialise
ex: int a[

2) Partial
Ex: int a[

3) Initia
Ex: int
a[0]

4) Initia
Ex: int

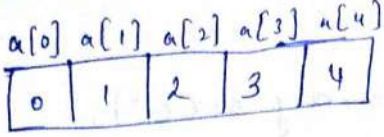
* Run-
for(
scan

* Compile time:-

The different types of compile time intilisation are as follows:

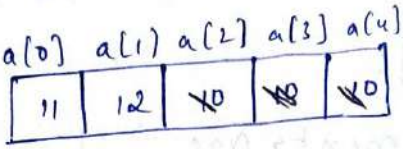
1) Initialise all the values:

ex: int a[5] = {11, 12, 13, 14, 15}.



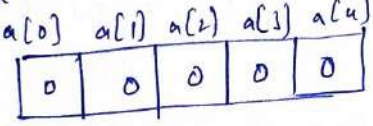
2) Partial initialization:

Ex: int a[5] = {11, 12}



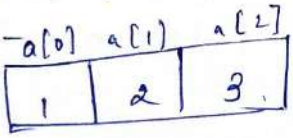
3) Initialize with all zeroes:

Ex: int a[5] = {0}



4) Initialise without size.

Ex: int a[ ] = {1, 2, 3}



* Run-time:-

```
for(i=0; i<n; i++)
scanf("%d", &a[i]);
```

* Write a C program to read and display the elements of 1-D array.

```
#include <stdio.h>
int main()
{
    int a[20], n, i;
    Printf("Enter the array size");
    scanf("%d", &n);
    Printf("\n Enter elements:");
    for(i=0; i<n; i++)
    scanf("%d", &a[i]);
    Printf("\n The elements are :");
    for (i=0; i<n; i++)
    Printf("%d", a[i]);
    return 0;
}
```

* Write a C program to find the sum and average of the list of elements.

```
#include <stdio.h>
int main()
{
    int a[20], n, i, s=0;
    Printf("Enter the array size");
    scanf("%d", &n);
    Printf("\n Enter elements:");
    for(i=0; i<n; i++)
    scanf("%d", &a[i]);
    Printf("\n The elements are :");
    for(i=0; i<n; i++)
    Printf("%d", a[i]);
    for(i=0; i<n; i++)
    s=s+a[i];
    Printf("Sum = %d \n Average = %f", s, (float)s/n);
}
```

* Write a C program to find minimum and maximum elements in the list.

```
#include<stdio.h>
int main()
{
    int a[20], n, i, min, max;
    Printf("Enter the array size");
    scanf("%d", &n);
    Printf("\nEnter elements: ");
    for(i=0; i<n; i++)
    scanf("%d", &a[i]);
    Printf("\n The elements are : ");
    for(i=0; i<n; i++)
    Printf("%d", a[i]);
    min = max = a[0];
    for(i=0; 1<n; i++)
    {
        if(a[i] < min)
        min = a[i];
        if (a[i] > max)
        max = a[i];
    }
    Printf("Minimum element = %d \n Maximum.
                element = %d", min, max);

    return 0;
}
```

* 2-D Array :-

1D array variables store a list of values only but there could be a situations where a table of values have to be stored which need 2D arrays.

Ex: 1) Periodic table
2) Sales information of a company
3) Data in spread sheets etc.

-) The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns. A Particular value in a matrix can be accessed by using two subscripts such as Vij here v denotes the entire matrix and Vij refers to the value in the ith row and jth column. An array of arrays is Known as 2-D array.

* Declaration of 2-D array :-

Syntax:

datatype arrayname [size 1] [size 2]

Ex: int a[10][10]

* Initialisation of 2-D arrays :-

There are 2 types 1) Compile time 2) Runtime

* Compile time :-

The different ways of initialising in compile time are as follows.

→ int a[2][3] = {1, 2, 3, 4, 5, 6}; →

1	2	3
4	5	6

→ int a[2][3] = {{1, 2, 3}, {4, 5, 6}}; →

1	2	3
4	5	6

→ int a[2][3] = {1, 2, 3, 4}; →

1	2	3
4	0	0

→ int a[2][3] = {

→ int a[ ][3] = {

* Runtime - (1n

for (i=0; i<

{

for (j=0;

scanf ("

* Write a C pro

2-D arrays.

#include <st

int main (

{

int a[5]

Print f ("

scan f (

Print f (

for (i=

{

for

sca

}

Print

for

{

to

Pr

P

}

ret

}

→ int a[2][3] = {{1,2,3},{4,5}}; →

1	2	10
4	5	10

→ int a[ ][3] = {1, 2, 3, 4, 5}; →

1	2	3
4	5	10

* Runtime - Initialisation:

```
for(i=0; i<r; i++)
{
   for(j=0; j<c; j++)
   scanf("%d", &a[i][i]);
```

* Write a C program to read and display content of 2-D arrays.

```
#include<stdio.h>
int main()
{
   int a[5][5], r, c, i, j;
   Printf("Enter the order: ");
   scanf("%d %d", &r, &c);
   Printf("Enter the elements: \n");
   for(i=0; i<r; j++)
   {
      for(j=0; j<c; j++)
      scanf("%d", &a[i][j]);
   }
   Printf("The elements are: \n");
      for(i=0; i<r; i++)
   {
      for(j=0; j<c; j++)
      Printf("%d", a[i][j]);
      Printf("\n");
   }
   return 0;
}
```

* Write a C program to implement matrix transpose.

```
#include <stdio.h>
int main()
{
    int a[5][5], r, c, i, j;
    Printf("Enter the order: ");
    scanf("%d %d", &r, &c);
    Printf("Enter the elements:\n");
    for(i=0; i<r; i++)
    {
        for(j=0; j<c; j++)
        scanf("%d", &a[i][j]);
    }
    Printf("The elements are:\n");
    for(i=0; i<r; i++)
    {
        for(j=0; j<c; j++)
        Printf("%d", a[i][j]);
        Printf("\n");
    }
    Printf("The transpose is:\n");
    for(i=0; i<c; i++)
    {
        for(j=0; j<r; j++)
        Printf("%d", a[j][i]);
        Printf("\n");
    }
    return 0;
}
```

* Multi-D
C allows
limit is
Syntax:
type-ar
where
eg: int
int

A 3D
array
and a
initi

* Multi-Dimensional Array:-

C allows arrays of three or more dimensions. The exact limit is determined by the compiler.

Syntax:-

type-array-name [s1] [s2] [s3] ... [sm];

where si is the size of the ith demension.

eg: int a[3][3][3];

int b[5][4][5][3];

A 3D array is essentially an array of arrays of arrays: it's on array or collection of 2D arrays and a 2D array is an array of 1D array.

Initializing 3rd array: