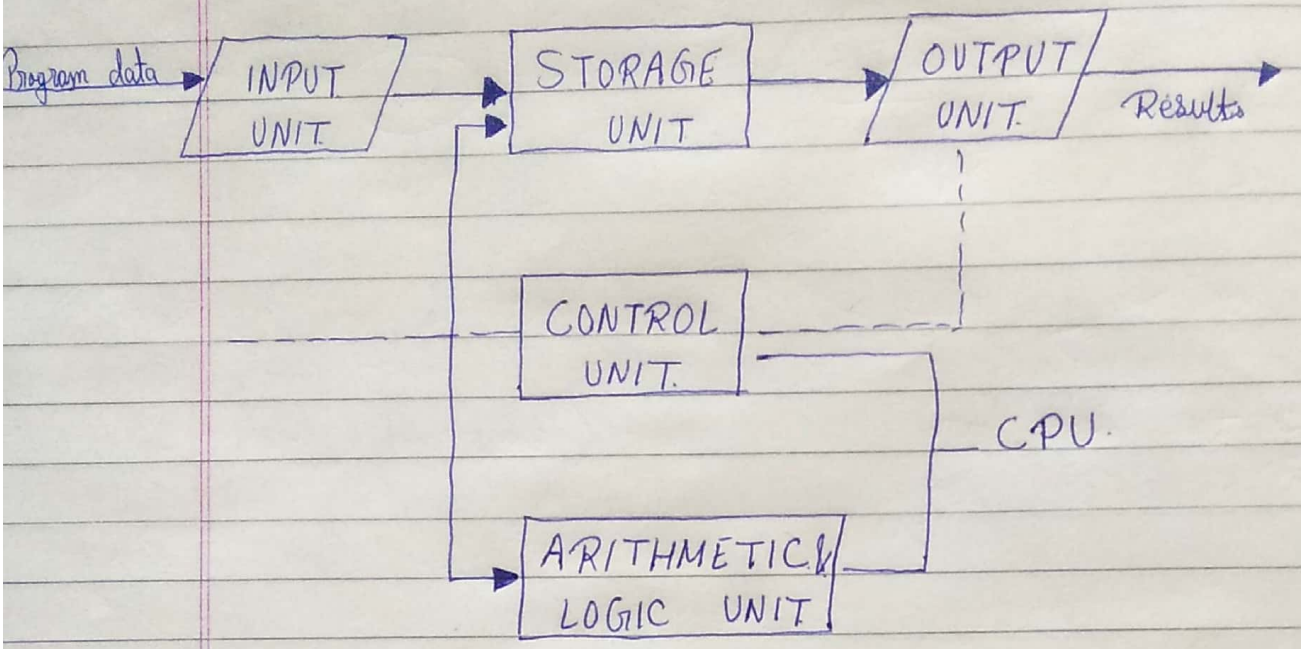
Intro To 'C'

* Block diagram of Comp & its function



1) Input unit :- Input unit is a unit that accepts any input device. The input device is used to input data into computer system.

function of IU :-

1) It converts inputted data into binary codes
2) It sends data to main memory of computer

2) CPU - CPU is called brain of computer. An electric circuitry that carries out the instruction given by a computer system.

CPU is sub-classified into 3 parts.

1) Control unit :- It manages the various components of the computer. It read instructions from the memory & interpret the changes in the series of signals to activate other parts of computer.

It controls & co-ordinate input output memory & all other units

(ii) ALU :- ALU performs simple arithmetic operations such as +, -, *, / & logical operations such as >, <, =<, >= etc.

(iii) Memory unit :- Memory is used to store data & instructions before & after processing. It is used to store data temporary or permanently

* Function of CPU :- 1) It control all the parts, software and data flow of computer
2) It performs all operations
3) It accepts data from input device
4) It sends information to output device
5) Executing programs stored in memory
6) It stores data either temporary or permanent basis
7) It performs Arithematical & logical operations

3) Output unit :- It is a unit that constituents a no. of output device. An output device is used to show the result of processing.

* fun" of output device :- 1) It accepts data or information sends from main memory of computer
2) It converts binary coded information into HLL or inputted language

Classification of fun^{n} :

1) No arguments & No return values
2) with arguments & no return values
3) with arg & with return values
4) No (arg) & with (return) values

arg → input
return → output

* No arg & No return

return → (Void) fun((void)) → arg

```
{
   ____
   ____
}
```

calling :- fun(-);

* with arg & no return

```
void fun(int x)
{
   ____
   ____
}
```

fun(10);

* With arg. with return

```
int fun(char x)
{
   ____
   return (13);
}
```

int x = fun('g');

* No arg & with return

```
float fun(void)
{
   ____
   return 34.16;
}
```

float x = fun();

Library

stdio.h
- printf();
- scanf();

Conio.h
- clrscr();
- getch();

String.h
- strlen();
- strrev();

Graphics.h
- setcolor();
- setbkcolor();

Math.h
- rand();
- Pow();

dos.h
- getdata();
- gettime();

data-types

- primitive
  - int
  - char
  - float
  - void
- derived
  - Array
  - String
  - Pointer
- user-defined
  - Structure
  - union
  - typedef
  - enum

* ASCII :- Americans std code for information interchange

A → 65
⋮
Z = 90

a = 97
⋮
z = 122

0 - 48
1 - 49
⋮
9 - 57

→ 35
. → 32

Modify operators

- Increment
  - Pre
  - Post
- decrement
  - Pre
  - Post

eg: int a = ?, b = 3

```
b = a++ + b--;
a = a-- + ++b;
b = ++a + --b;
```

a = 8
b = 12

Method
Pre
Subs
Eva
Assign
Post

int a = 5

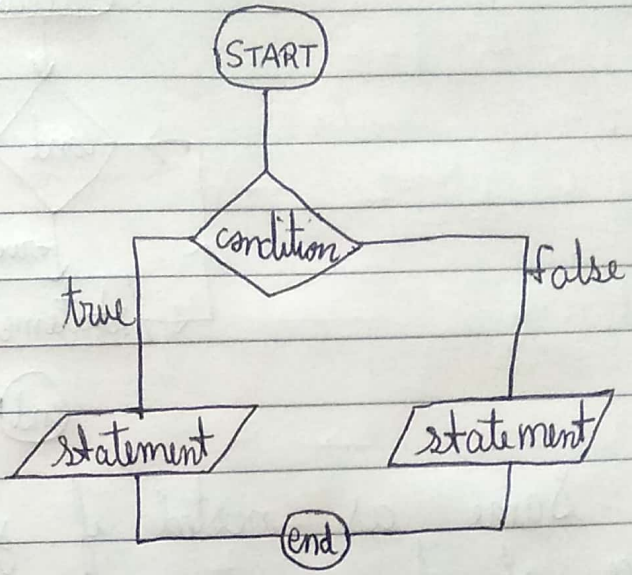
a = a+++a

a = a+ ++a; ✗

a = a++ +a ✓

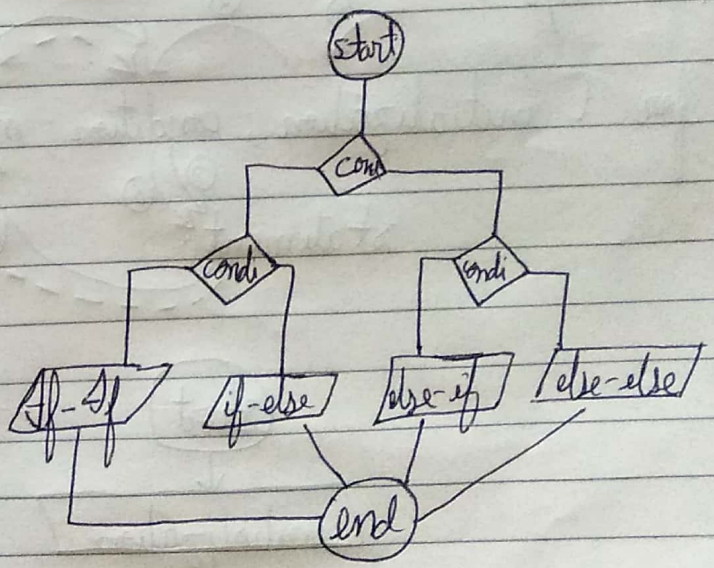
priority goes to modify operator.

* if else



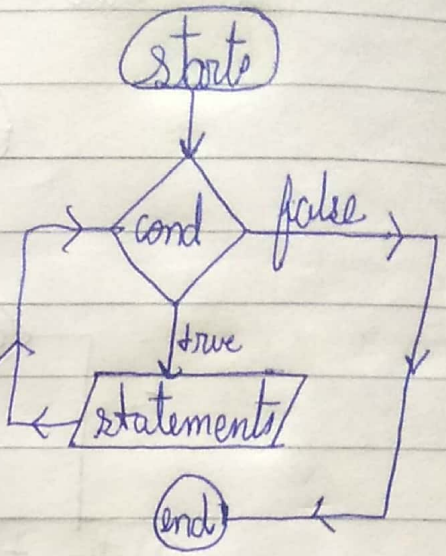
* Nested if

```
if (cond.)
{
    if (condi)
    else
}
else
{
    if (condi)
    else
}
```



while

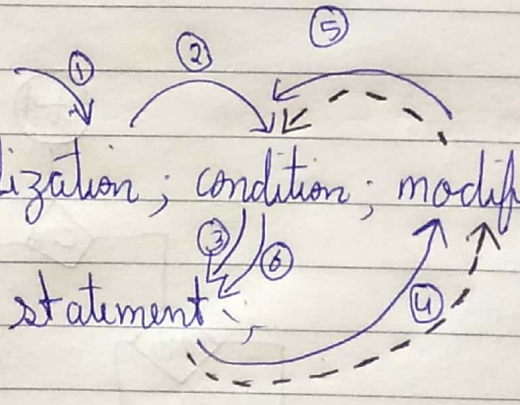
```
while (condition)
{
   statements;
}
```

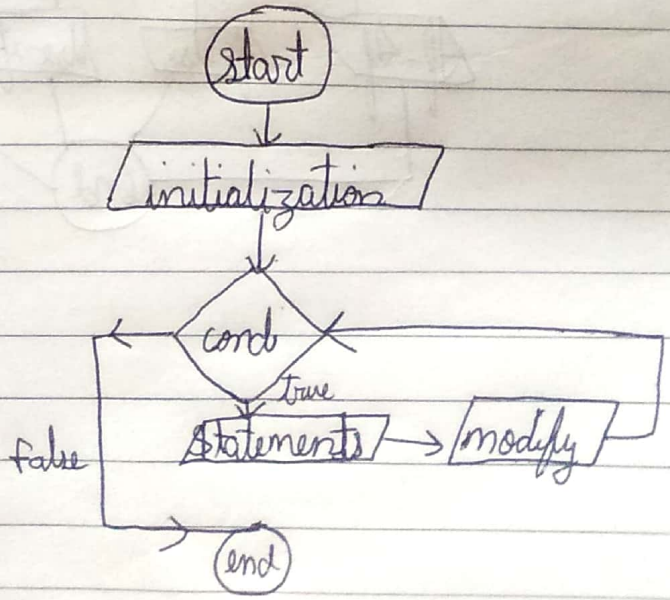


* Nested while :- Same as nested if, just replace while in place of if.

* for loop :-

```
for ( initialization ; condition ; modify)
    statement ;
```

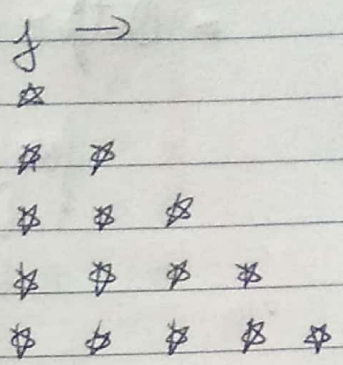




Pattern Programs

→ by using both increment operater

```
for (i=1 ; i<=5 ; i++)
{
    for (j=1 ; j<=i ; j++)
    {
        Printf ("*");
    }
    printf ("\n");
}
```



→ by using both decrement operator

```
for (i=5 ; i>=1 ; i--)
{
    for (j=5 ; j>=i ; j--)
    {
        printf ("*");
    }
    printf ("\n");
}
```

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

```
printf ("%d", i);
printf ("%d", j);
```

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

- Outer operator → Increment & inner is → decrement

```
for (i=1; i<=5; i++)
{
    for (j=i; j>=1; j--)
        printf("*");
}
printf("\n");
}
```

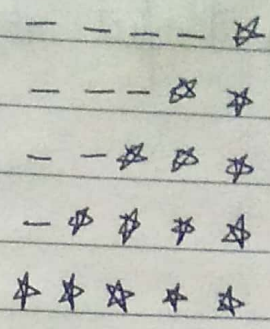
```
*
* *
* * *
* * * *
* * * * *
```

```
printf("%d", i);
printf("%d", j);
```

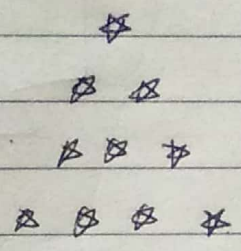
```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

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

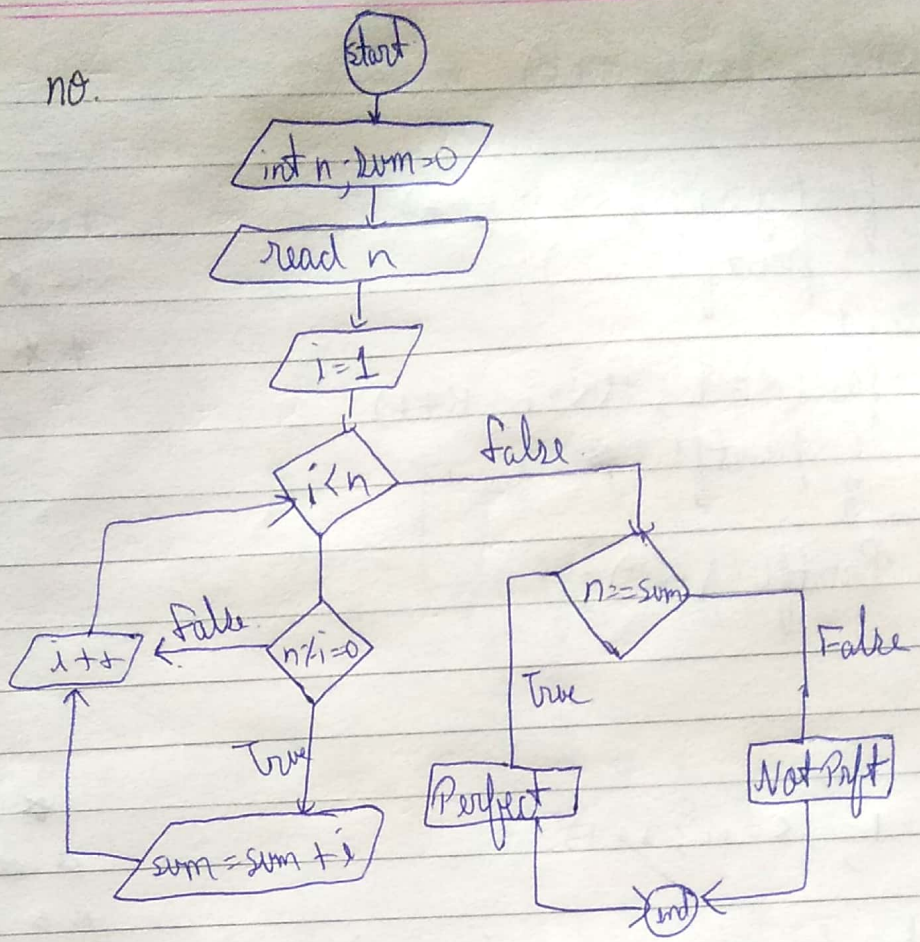
```
for (i=1; i<=5; i++)
{
    for (j=i; j<5; j++)
    { printf(" ");
    }
    for (k=1; k<=i; k++)
    { printf("*");
    }
    Printf("\n");
}
```



```
for (i=1; i<=n; i++)
{
    for (j=1; j<=c/2-i; j++)
    {
        Printf(" ");
    }
    for (k=i; k<=2i-1; k++)
    {
        Printf("* ");
    }
    Printf("\n");
}
```



* Perfect no.



* Prime no.

```
for (i=1; i<=n; i++)
  {
     if (n%i==0)
     { count++;
     }
  }
if (count==2)
  Pf("Prime no.");
else
  Pf("Not prime");
```

* Strong No: 145

$1! + 4! + 5! = 1 + 24 + 120 = 145$

* Perfect No :- 28 : 1 + 2 + 4 + 7 + 14 = 28

* Armstrong No :- 153

$1^3 + 5^3 + 3^3 = 153$

* Recursion :- function calling itself is called Recursion

Q:- factorial of a no. using recursion

```
main()
{ int n, res;
  Pf("Enter n value:");
  Sf("%d", &n);
  res = fact(n);
  printf("result: %d", res);
}
int fact(int n)
{
   int res;
   if (n==0)
      res = 1;
   else
      res = n * fact(n-1);
   return res;
}
```

Array :- In a primative data type we can only store one value, if we try to store other too, then it replace the previous one.

by using a single array, we can store n no. of elements bt of same data type (char / int / float)

eg: Direct initialization . int arr[5] = {10, 20, 30, 40, 50};
this is local Declaration

Global Declaration

```
int arr[5];
for (i=0; i<5; i++)
{ {
    scan ("%d", & arr[i]);
  }
  { print ("%d", arr[i]);
  }
}
```

Q:- largest no in array.

```
large = arr[0];
for (i= 1; i < n; i++)
  {
    if (arr[i] > large)
      large = arr[i].
  }
printf ("large : %d \n", large);
```

* Insert a element in an array :-

eg :- n = 6
key = 100
loc = 2

```
for (i=n-1; i>=loc; i--)
{
    arr[i+1] = arr[i]
}
arr[loc] = key;
```

* String :- It is a sequence of character that are representated as a single data item

* length of string using user define function

```
int strlen(char[]);
void main
{
    char str[30];
    int len;
    printf("Enter String :")
    gets(str);
    len = strlen(str)
    printf("Length : %d", len);
}
int strlen(char X[])
{ int i=0, count=0;
  while(X[i]!='\0')
  {
    count++;
    i++;
  }
  return count;
```

* Swapping in string using user define fun".

```
i = 0.
j = strlen(X);
while (i<j).
  {
     temp = x[i];
     x[i] = x[j];
     x[j] = temp;
     i++;
     j--;
  }
```

* Upper to lower.
upr to lwr

→
```
int i=0
while (x[i] != '\0')
{
   if ( x[i] >= 65 && x[i] <= 90)
   {
      x[i] = x[i]+32.
   }
      i++;
}
```

* C → Dennis Ritchie. → [redacted] Bell labs
1972 - 1973
New Jersey, US.

Cmp. - Compare:

* Concat ✓ Merging of two strings

→ String :- No need to use address operator, we pass base address only (%s)

→ Array :- We passes address manually by for loop (%c)

→ No problem to use printf but if we use scanf with %s than in output if we use space than compiler will terminate at space & print only element before space. So use gets instead of scanf.

* Pointer :- Pointer is a variable that points the address of another variable. Pointer is used to allocate memory dynamically ie at run time.

* Union :- Union is a special data type available in C that allows to store different data types in the same memory location. You can define a union with many number, but only one member contain a value at a given time. Union provide an efficient way of using the same memory location for multi-purpose.

Neel Kamal
Date
Page

* Structure :- Structure is a user defined data type in C. Structure creates a data type that can be used to group items of possibly different types into a single type. 'struct' keyword is used to create a structure

* File handling :- A file represents a sequence of bytes on the disk where a group of related data is stored. file is created for permanent storage of data. It is a readymade structure.