Module 2 : (part 1)

Formatted & unformatted Input & output statements

- printf()
- scanf()
- getchar()
- gets()
- getch()
- putch()
- puts()
- putchar()

unformatted Input & output functions

* unformatted I/O functions are the functions which do not specify the type of data & the way it should be read or written.

- getchar()
- putchar()
- getch()
- putch()
- gets()
- puts()

#include<stdio.h> - If getchar(), putchar() are used in program.

#include<conio.h> - If getch(), getchar() or putch() are used in program.

Scanned by CamScanner

~~getch() & putch() :-~~

* getchar() & putchar() :-

getchar() :- This function reads a single character from the standard input device.

* There is no parameter within the parentheses.

Syntax :- ch-var = getchar();

where ch-var → character type variable to which an accepted character is assigned

putchar() :- This function prints a single character on the screen.

Syntax :- putchar(ch-var);

ch-var → character variable which is enclosed within the parantheses.

Example :- Write a program (WAP) to read a character & print a character using unformatted I/O statement?

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char a;
    clrscr();
    a=getchar();
```

Scanned by CamScanner

```
    putchar(c);
    getch();
}
```

Output :- c ↵ Note: Indicates Enter is pressed.
c

* gets() & puts() :-

gets() :-
* This function is used to read a set of characters from the Keyboard.

Syntax is,

```
gets(string);
```

where, string is a sequences of characters & it is of type char.

puts() :-
* This function displays the sets of characters on the screen.

Syntax is,

```
puts(string);
```

where, string is a sequence of characters

Example :- WAP to read a name & print name using unformatted I/O statements.

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

Scanned by CamScanner

```
    char a[10];
    gets(a);
    puts(a);
}
```

Output :- program ↵ Note : Indicates
program Enter Key is press

* getch() & putch() :-

* getch() :- This function reads a char from the Keyboard & copies to the m area.
* No arguement or parameter is requi
* This function does not wait for the user to press "Enter" Key.

Syntax :- ch_var = getch();

* putch() :- This function outputs or dis a character stored in the memory.

* The variable should be passed as a parameter to the function.

Eg :-

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char a;
    clrscr();
    a = getch();
    putch(a);
```

Scanned by CamScanner

```
    getch();
}
```

output :- c

Note :-

* The typed character will not be echoed (displayed) on the screen if we use getch() function.

* The typed character will be echoed (displayed) on the screen if we use getche() function.

Eg:-
```
#include<stdio.h>
#include<conio.h>
void main()
{
    char a;
    clrscr();
    a=getche();
    putch(a);
    getch();
}
```

output :- c c
↓ ↳ output
Input

P.T.O

Scanned by CamScanner

Formatted Input & output Statements

- scanf()
- printf()

* Formatted I/O functions enable the user to specify the type of the data & the way in which it should be read in & written out.

* Formatted Input function :-

- scanf()

* scanf() :- It is a function used to read the data of any type such as integer, character, float, double & string from the keyboard.

Syntax :- scanf("format string", &var1, ----, &varn)

where,
- format string → consists of 1 or more format specifiers beginning with %.
- &var1, ---- &varn → List of address of variables.

* Rules :-

1. The format string should be enclosed within a pair of double quotes.

Scanned by CamScanner

2. The address of variable list should be separated by commas.

3. The values for the address-list variable should match in number, type & the order of the variable.

4. The address of var list should not be enclosed within double quotes.

Datatype	Format specifier	Meaning
int	%d	Reads decimal integer value
	(long int) %ld	Reads long integer value
	%u	Reads unsigned integer val
	%hd	Reads short integer value
	%o	Reads octal value
	%x	Reads hexadecimal value
char	%c	Reads character value
	%s	Reads a string
float	%f	Reads floating point numb
	%e	Reads floating point numbe exponential.
double	%lf	Reads long floating point numbers or double.

* Integer input :-

* %d is used as a format specifier

* Eg:- scanf("%d", &number);

Scanned by CamScanner

* If there are more than 1 ~~variable~~ format specifier within format string then specify them by zero blank space.

Eg:- scanf("%d%d%d", &a, &b, &c);

* We can accept data through the field specification.

* Field specification means number of columns or fields reserved for the input.

Eg:- scanf("%3d", &num);

If the input data to be assigned to num is 2345, then only 234 is stored in memory location.

* Character input :-

%c is used as a format specifier.

Eg:- scanf("%c", &char);

* If character input is a name, then a sequence of characters that constitutes the name.

* The scanf() to read a name with %c is

scanf("%10c", name);

Scanned by CamScanner

* If the input name is chaithracoorg then only first 10 characters (chaithraco) is stored in memory.

Note :- Before accepting the string, the variable name should be declared as a 1-dimensional array of characters.
i.e., char name[10];

Another way is,

```
scanf("%s",name);
```

* Floating-point number Input :-

* %f is used as a format specifier.

eg:-
```
float a;
scanf("%f",&a);
```

* Mixed mode Input

* scanf() can accept more than one type of data (mixed mode).

* The number of input variables, data type & their order should be proper.

eg:-
```
int a;
float b;
char c;
scanf("%d%f%c",&a,&b,&c);
```

Scanned by CamScanner

* Formatted Output function :-

- printf() :-

* This function display the data on the screen.

Syntax :-

```
printf("format string", var1, ---- varn);
```

where,

- format string → consists of 1 or more format specifier.
- var1, var2, ..., varn represent the values of variable.

Data type	Format	Meaning
int	%d	To print decimal integer
	%o	To print octal value
	%u	To print unsigned integer
	%x	To print hexadecimal value
	%hd	To print short integer value
float	%f	To print floating pt number
	%e	To print floating pt number
char	%c	To print character value
	%s	To print string
double	%lf	To print double
long int	%ld	To print long integer value.

Scanned by CamScanner

* Integer output :- Syntax :- %wd
where w → Indicates number of colum
%d is used as format specifier.

Eg:- printf("%d", num);

Eg:- num = 3456;

~~then~~ printf("%4d", num);

Then output is pictorially shown as:-

3	4	5	6

if → num = 345
printf("%4d", num);

output :

	3	4	5

num = 798
printf("%2d", num);
output : [7|9|8]
* If the space specified is less than requirement then compiler provides sp to print.

* For +ve integers the field will be allocated using right justification.
* For -ve integers → Left justification.

Eg:- num = 345;
printf("%-5d", num);

output :

3	4	5		

* If you want to fill all the empty space by 0, then write

printf("%05d", num);

output :-

0	0	3	4	5

Scanned by CamScanner

* Floating point output :-

Syntax:- %w.xf

where w is the number of columns.

x is number of columns after decimal point

① Eg:- a = 98.578;

printf("%7.2f", a)

output :- | | | 9 | 8 | . | 5 | 7 |

② a = 98.578;

printf("%7.2f", a);

output :- | 9 | 8 | . | 5 | 7 | | |

③ a = 98.578; → round off to 6

printf("1.1f", a);

output :- | 9 | 8 | . | 6 |

* Character output :-

%c & %s are used as a format specifier for single character & string respectively.

char book[20]; printf("%s", book);

① Eg:- string is "programming in C"

then o/p will be,

| | | | p | r | o | g | r | a | m | m | i | n | g | | i | n | | C |

② Eg:- char book[20];
printf("%20.10s", book);

										p	r	o	g	r	a	m	m	i	n

Example 1:- WAP to show the usage of formatted output function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int N= 4567;
    clrscr();
    printf("%d\n", N);
    printf("%.5d\n",N);
    printf("%-5d\n",N);
    printf("%06d",N);
    getch();
}
```

output :

4	5	6	7

	4	5	6	7

4	5	6	7	

0	0	4	5	6	7

Example 2:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
    float n = 2.345;
    clrscr();
```

Scanned by CamScanner

```
printf("%f \n", n);
printf("%8.2f \n", n);
printf("%4.2f", n);
getch();
}
```

output :- 2.345000
2.35
2.35

<u>possible questions</u>

1. Explain the unformatted I/O statements?
2. Explain the formatted I/O statements?

Scanned by CamScanner

Chaithra H.E

Module 2 :- (Continuation) (part 2)

Branching & Looping :-

2 way selection :- if
if-else
nested if-else
switch statement
goto statement
} Decision making statements

Loops : for
while
do-while
break and continue
} Loops

1) ~~[illegible]~~

Simple Statement :-

- only one statement enclosed within pair of braces such as '{' and '}'.

Compound statement :

Set of statements enclosed within pair of braces { and }.

Control Statements :-

The statements that are used to control the flow of execution of program is called control statement.

By, Chaithra H.E

Scanned by CamScanner

Types :-

- Control Statements
  - Sequential Statements
  - Branching Statements
    - Conditional branch statement
      - simple-if
      - if-else
      - Nested-if
      - else-if ladder
      - switch
    - unconditional branch statement
      - goto
      - break
      - continue
      - return
  - Loop Statements
    - for
    - while
    - do-while

* Sequential Statements

The programming statements that are executed sequentially, one after the other are called as sequential statements.

```
Statement -1;
statement -2;
:
:
statement n;
```

Statement 1 is executed first, then Statement 2 ~~and~~ and continues till Statement n.

Example :- Write a C program to convert Fehrenheit to celsius.

Formula : $C = (5/9)(F-32)$

Scanned by CamScanner

```
#include<stdio.h>
void main()
{
  float f, c;
  clrscr();
  printf("Enter degree in Fehrenheit\n");
  scanf("%f", &f);
  c=(float) 5/9 * (f-32);
  printf("Result =%f\n", c);
  getch();
}
```

<u>Output</u> :- Enter degree in Fehrenheit
40
C=4.444
Result =4.444

* <u>Branching statements</u> :-

the statement which alters the sequence of execution of program are called branching statements.

- <u>Conditional branch statement</u>

The statements that alter the sequence of execution of program based on the condition.

It is also called as decision statement

→ if
→ if-else
→ Nested if-else
→ else-if ladder
→ switch

Scanned by CamScanner

1) **if :-** if statement is a decision statement which is used to execute statement or a set of statements conditionally.

- It is a one-way decision statement.
- Here condition is tested which results in either a true or a false value.

- If the condition is true then the statement which is inside the body of if will be executed.

- If the condition is false then the next statement which is executable i.e., outside the body of if will be executed.

Syntax :-

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

Condition → Logical expression that results in true or false.

statement → simple or compound statement

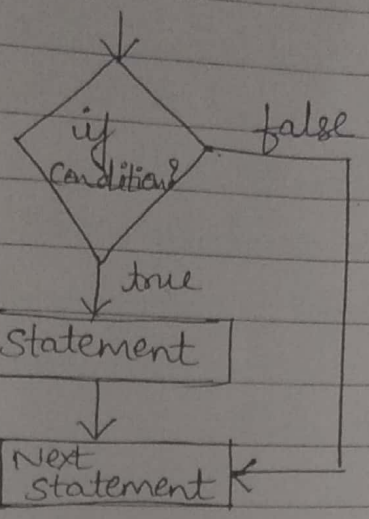
simple statement : single statement

compound statement : set of statements.

Note :- Braces are not required for simple statement.

Scanned by CamScanner

Flowchart :-



Example1:- WAP to accept a number & check whether it is even number & print the same.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n;
 clrscr();
 printf("Enter a number\n");
 scanf("%d",&n);
 if (n%2==0)
 printf("number is even");
 getch();
}
```

output :- Enter a number
10
number is even

P.T.O

Scanned by CamScanner

Example 2:- WAP to find the largest of 2 numbers using if statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a, b, big;
  clrscr();
printf("Enter 2 numbers\n");
scanf("%d%d",&a,&b);
big=a;      // assigning a to big
if(b>big).
big=b;
printf("Largest number is %d", big);
getch();
}
```

Output :-

```
Enter 2 numbers
10 20
Largest number is 20.
```

2) if-else :-

It is a 2-way decision statement i.e., one set of statement will be executed when the condition is true & another set of statement will be executed when the condition is false.

Scanned by CamScanner

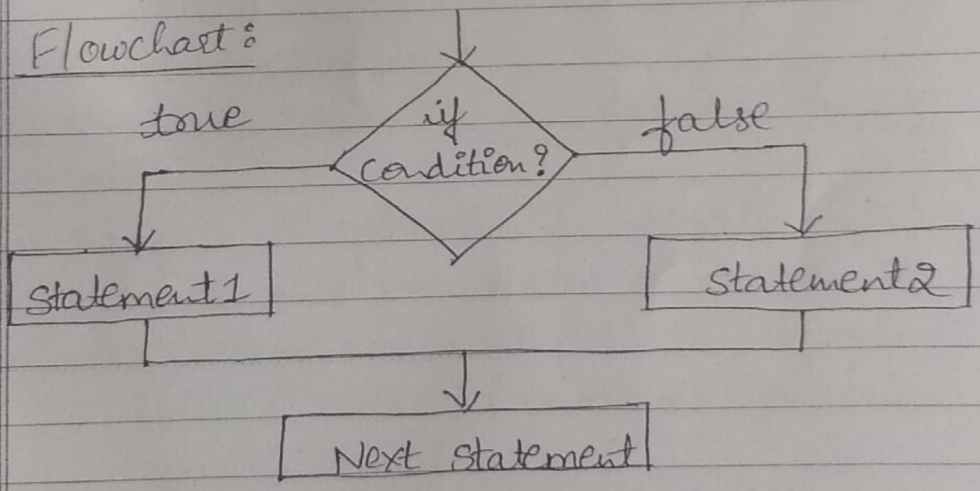
Syntax :-

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

The condition is tested. If the condition is true, then statement 1 is executed. otherwise, statement 2 is executed.

The statement 1 and statement 2 may be simple or compound statements.

Flowchart:



Example 1:- WAP to check whether number is even or odd.

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

Scanned by CamScanner

```
    printf("Enter a number\n");
    scanf("%d", &n);
    if(n%2==0)
    {
        Printf("even number");
    }
    else
    {
        printf("odd number");
    }
    getch();
}
```

Output

Enter a number

5

odd number.

Lab program

1) a) Design & develop a C program to read a year as an input and find whether it is leap or not. (Also consider end of centuries).

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int year;
    clrscr();
    printf("enter the year\n");
```

Scanned by CamScanner

```
scanf("%d", &year);
if((year%4==0)&&(year%100!=0)||(year%400==0))
{
printf("%d is a leap year", year);
}

else
{
printf("%d is not a leap year", year);
}

getch();
}
```

Example 2: WAP to find the largest of 2 numbers using if-else.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a, b;
    clrscr();
    printf("enter a and b values\n");
    scanf("%d%d", &a, &b);
    if(a>b)
    {
      printf("a is greater");
    }
    else
    {  printf("b is greater");
    }
```

Scanned by CamScanner

```
getch();
}
```

3) <u>nested-if</u> :-

An if or if-else statement enclosed within another if or if-else statement is called nested-if statement.

An action is performed based on many decision, this is also called as <u>multi-way</u> decision statement.

<u>Syntax</u> :-

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

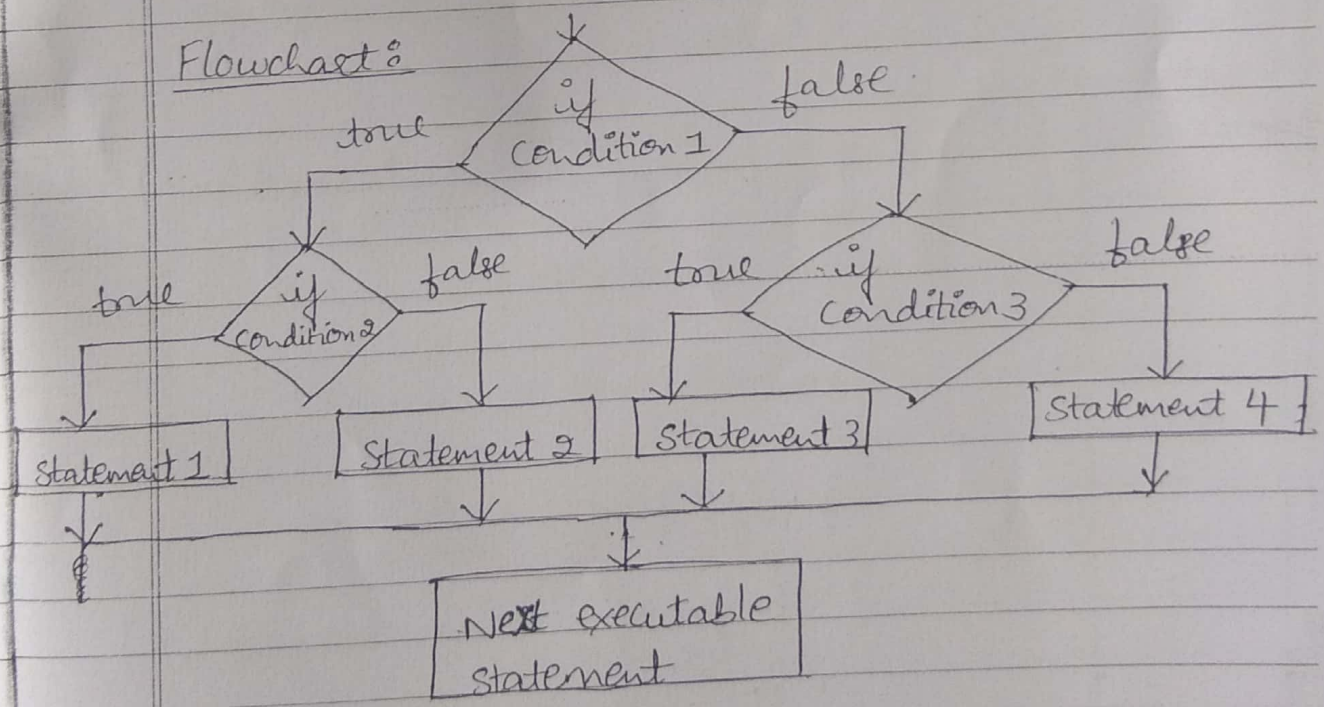
Scanned by CamScanner

```
else
{
    Statement 4;
}
}
```

In the above syntax,

* Statement 1 is executed if condition 1 & condition 2 are true.
* If the condition 1 is true & condition 2 is false then statement 2 is executed.
* If the condition 1 is false then control transfers to the else part.
* In the else part if the condition 3 is true then statement 3 is executed, else statement 4 is executed.

Flowchart:



Scanned by CamScanner

Example:- WAP to find the largest of 3 numbers using nested-if

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

  int a, b, c;
  clrscr();
  printf("enter a, b and c values\n");
  scanf("%d%d%d", &a, &b, &c);
  if (a>b)
  {
    if (a>c)
    {
      printf("a is largest\n");
    }
    else
    {
      printf("c is largest\n");
    }
  }else
  {
    if (b>c)
    {
      printf("b is largest\n");
    }
    else
    {
```

Scanned by CamScanner

```
    printf(" c is largest \n");
  }
}
  getch();
}
```

4) <u>else-if ladder</u> :-

when an action is to be performed based on many decisions, then this statement is used, so it is called multi-way decision statement.

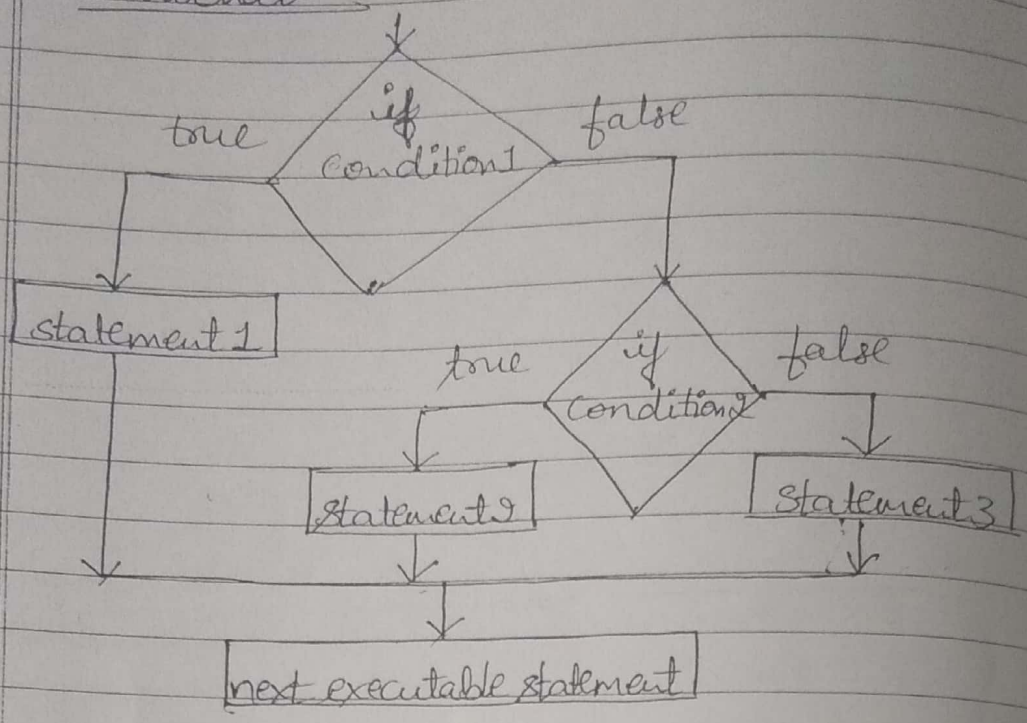
It is a form of nested-if statement where <u>nesting</u> is <u>allowed</u> only in the <u>else part</u>.

<u>syntax</u>:-

```
if (condition1)
{
  statement1;
}
else if (condition2)
{
  statement2;
}
else
{
  statement3;
}
```

Scanned by CamScanner

Flowchart



Example :- WAP to check whether a number positive, negative or zero.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n;
    clrscr();
    printf("enter a number\n");
    scanf("%d",&n);
    if(n>0)
    {
      printf(" number is positive");
    }
    else if(n<0)
    {
      printf(" number is negative");
    }
```

Scanned by CamScanner

```
else
{
    printf("number is zero");
}

getch();
}
```

5) <u>Switch Statement</u> :-

Switch statement provides a multiple way of branching.

It allows user to select any one of the several alternatives, depending on the value of expression (choice).

Depending upon the value of expression, the control transfers to a particular <u>case label</u> (branch) & all the statements followed by that case label are executed.

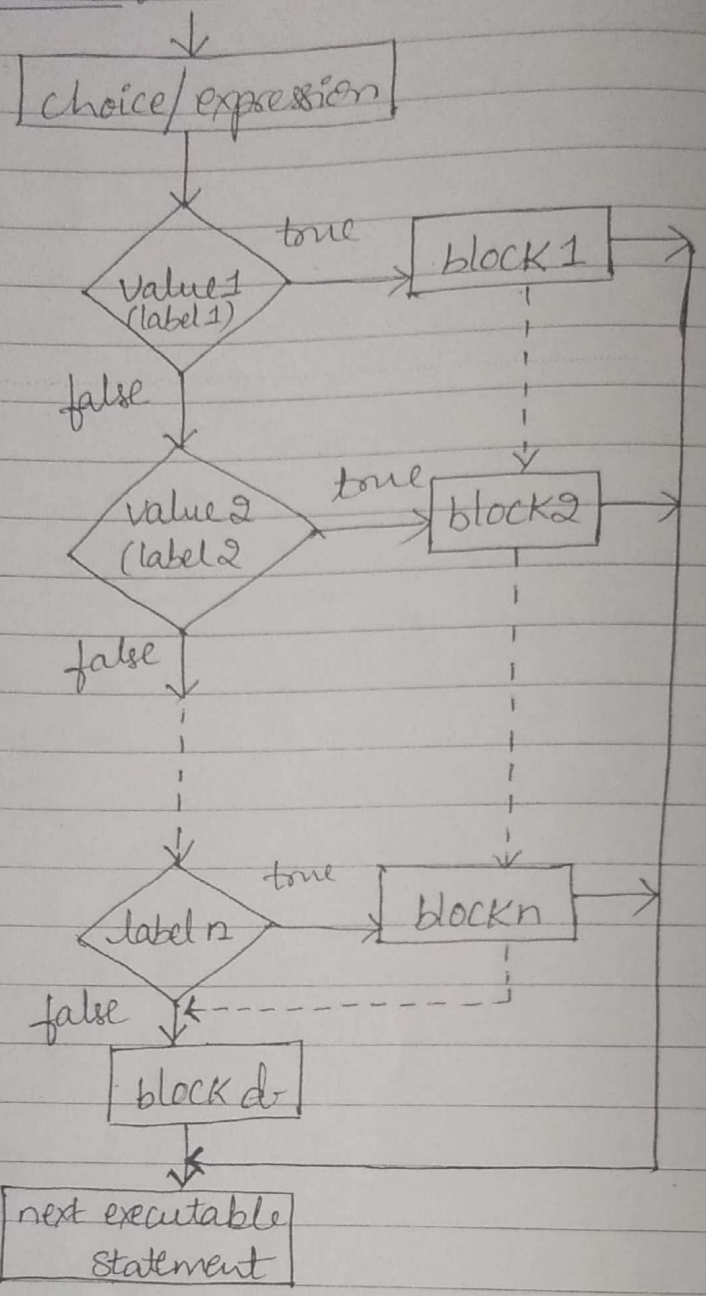
<u>Syntax</u> :-

```
switch(choice/expression)
{
    case label1 : block1;
                  break;
    case label2 : block2;
                  break;
    :
    case labeln : blockn;
                  break;

    default : block d;
}
```

Scanned by CamScanner

Flowchart :-



Example :- WAP to simulate the functions of a simple calculator.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    float a, b, res;
    char op;
    clrscr();
```

Scanned by CamScanner

```
printf("Enter the expression\n");
scanf("%f%c%f", &a, &op, &b);
switch(op)
{

  case '+': res=a+b;
            break;

  case '-': res=a-b;
            break;

  case '*': res=a*b;
            break;

  case '/': if(b!=0)
            {res=a/b;
            }
            else
            {
             printf("Div. by zero\n");
             exit(0);
            }
            break;
  default : printf("Illegal operator\n");
            exit(0);

  }
  printf("%f%c%f=%f\n", a, op, b, res);

}
```

Break statement prevents the code running into the next case.

Scanned by CamScanner

6) goto statement :-

By using goto statement, the control of a program can be transferred from one statement to the specified statement without any condition.

Syntax :- goto label;

⋮

label: statement;

where, goto is a keyword
label is any identifier.

when goto statement is encountered the control is transferred to the label after which statement is executed.

a) goto END;

⋮

END:

⋮

Here, The statements immediately followed by the goto will be skipped, while control jumps to label END. This type is known as a forward jump.

b) START:

⋮

goto START;

Scanned by CamScanner

Here, the statement inside label start will be executed. The label START: is placed before the goto statement. This type is known as a backward jump.

Example :- WAP to find sum of 10 natural numbers using goto statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i, sum;
    clrscr();
    i=0;
    sum=0;
    top: sum=sum+i;
         i++;
         if (i<=10)
    goto top;
    printf("sum=%d\n", sum);
}
```

* WAP to find whether the number is armstrong or not.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
    int n, d1, d2, d3, sum;
```

Scanned by CamScanner

```
clrscr();
printf("Enter the value of n\n");
scanf("%d", &n);
d3 = n%10;
d2 = (n/10)%10;
d1 = n/100;
sum = pow(d1,3) + pow(d2,3) + pow(d3,3);
if(sum==n)
printf("Armstrong number");
else
printf("Not an armstrong number");
}
```

Lab program

2) Design and develop a C program that accepts 3 coefficients (a, b and c) of a quadratic equations ($ax^2+bx+c=0$) as input and compute all possible roots & print the possible roots for a given set of coefficients. Also print the message in case of zero valued coefficients.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
  float a,b,c,d,r1,r2,rp,ip;
  clrscr();
  printf("Enter 3 coefficients\n");
  scanf("%f%f%f", &a, &b, &c);
  if(a==0||b==0||c==0)
  {
    printf("Invalid input");
```

```
}
else
{
    d = (b*b) - (4*a*c);
    if (d==0)
    {
        r1 = -b/(2*a);
        r2 = r1;
        printf("r1=%f\t r2=%f\n", r1, r2);
    }
    else if (d>0)
    {
        r1 = (-b + sqrt(d)) / (2*a);
        r2 = (-b - sqrt(d)) / (2*a);
        printf("r1=%f\t r2=%f\n", r1, r2);
    }
    else
    {
        rp = -b / (2*a);
        ip = sqrt(fabs(d)) / (2*a);
        printf("r1= %f + %f i\t
            r2 = %f - %f i\n", rp, ip);
    }
}
    getch();
}
```

Loops :-

while
do - while
for

1) **while :-**

* A set of statements may have to be repeatedly executed till a certain condition is reached. So we use while loop.

* while is a loop where set of statements will be executed repeatedly as long as the condition is true. When the condition becomes false then control of program will terminate the while loop.

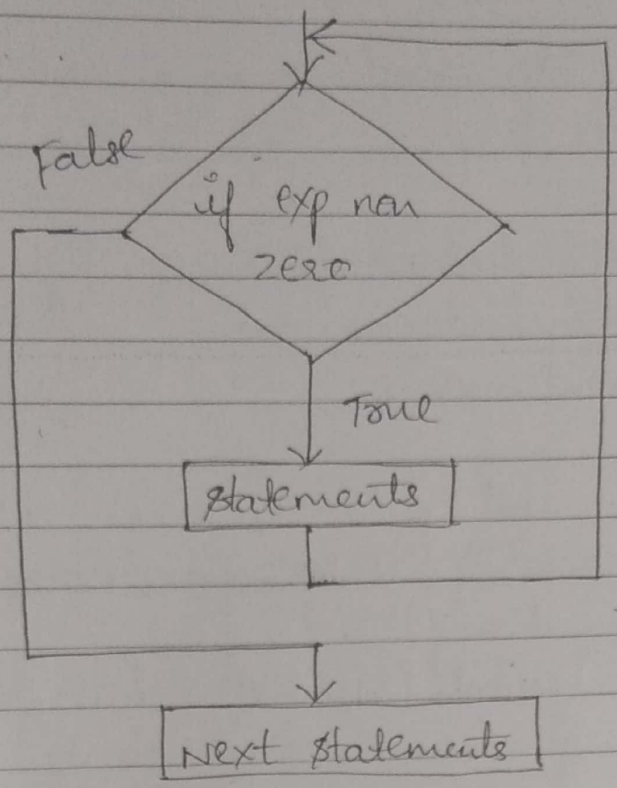
Syntax :-

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

where, while is a Keyword.
exp is the expression which is evaluated to TRUE or False.

Scanned by CamScanner

Flowchart:-



1) WAP to accept name & print it for a specified number of times.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char name[20];
    int ntimes, i=0;
    clrscr();
    printf("enter name\n");
    scanf("%s", name);
    printf("How many times to print name\n");
    scanf("%d", &ntimes);
    while( i < ntimes)
    {
        printf("%s", name);
        i++;
    }
```

Scanned by CamScanner

```
    getch();
}
```

2) WAP to find sum of given series using while loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n, i=0, sum=0;
    printf("Enter number of elements\n");
    scanf("%d", &n);
    while (i<=n)
    {
        sum= sum+i;
        i++;
    }
    printf("sum of series = %d", sum);
    getch();
}
```

3) WAP to compute sum of series $1+x+x^2+---+x^n$.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n, i=0, sum=0, x;
    clrscr();
    printf("Enter value of x and n\n");
```

Scanned by CamScanner

```
scanf("%d%d", &x, &n);
while(i<=n)
{
    sum = sum + pow(x, i);
    i++;
}
printf(" sum of series = %d", sum);
getch();
}
```

4) WAP to compute GCD & LCM of 2 numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int m, n, p, q, gcd=0, lcm=0;
    clrscr();
    printf(" Enter 2 numbers\n");
    scanf("%d%d", &m, &n);
    p=m;
    q=n;
    while(n!=0)
    {
        r=m%n;
        m=n;
        n=r;
    }
    gcd=m;
    lcm=(p*q)/gcd;
    printf(" gcd = %d", gcd);
    printf(" lcm = %d", lcm);
    getch();
}
```

Scanned by CamScanner

Note :- Refer text books for more example programs.

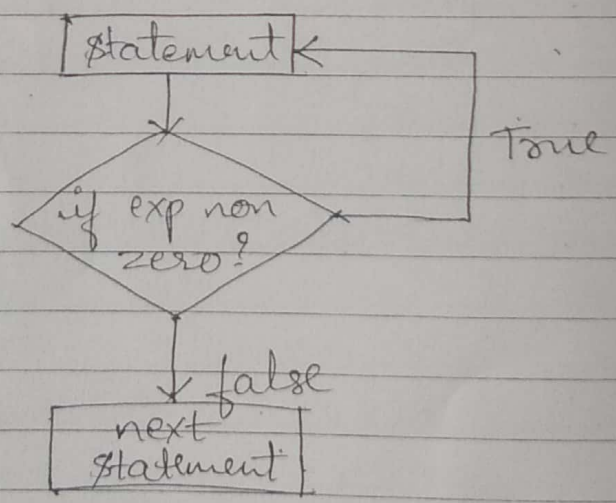
2) do-while :-

* It is a loop where set of statements will be executed repeatedly as long as the condition is true.
* It is a post-testing loop where code will be executed atleast once, then condition will be checked.

Syntax :-

```
do
{
    statements;
}
while(exp);
```

Flowchart :-



1) WAP to convert find sum of first n integers.

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

Scanned by CamScanner

```
{
    int n, i=0, sum=0;
    clrscr();
    printf("enter n\n");
    scanf("%d", &n);
    do
    {
        sum = sum+i;
        i++;
    } while (i<=n);

    printf("sum= %d", sum);
    getch();
}
```

3) <u>for loop :-</u>

* Loop in which code will be executed for a specified number of times.

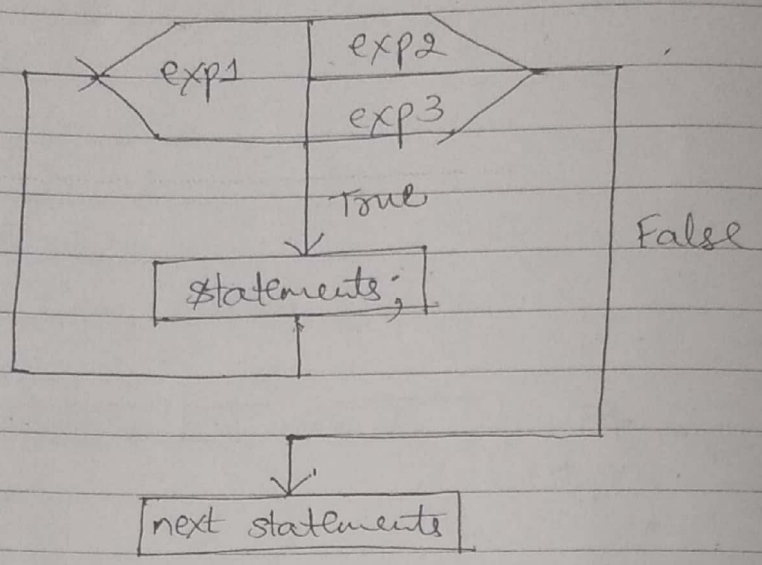
* If we know well in advance as how many times a set of statements have to be repeatedly executed, then for loop is the best choice.

<u>Syntax :-</u>

```
for(exp1; exp2; exp3)
{
    statements;
}
```

Scanned by CamScanner

where exp1 is initialization of variable
exp2 is condition
exp3 is increment/decrement.



1) WAP to print N Numbers.

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

Scanned by CamScanner

2) WAP to find factorial of given number using for loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i, fact=1, n;
  clrscr();
  printf("enter n\n");
  scanf("%d", &n);
  for(i=1; i<=n; i++)
  {
    fact = fact * i;
  }
  printf(" factorial=%d", fact);
  getch();
}
```

3) WAP to generate n fibonacci series.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i, n, fib1=0, fib2=1, fib3;
  clrscr();
  printf("enter n\n");
  scanf("%d", &n);
  if(n<2)
  {
    printf(" fib series doesn't exist");
  }
```

Scanned by CamScanner

```
else
{
  printf("%d%d", fib1, fib2);
  for(i=3; i<=n; i++)
  {
    fib3 = fib1+fib2;
    printf("%d", fib3);
    fib1=fib2;
    fib2=fib3;
  }
}
  getch();
}
```

4) WAP to find whether a number is prime or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n, i, count=0;
  printf("Enter n\n");
  scanf("%d", &n);
  for(i=1; i<=n; i++)
  {
    if(n%i==0)
    {
      count++;
    }
  }
  if(count==2)
  {
```

Scanned by CamScanner

```
    printf("n is prime number\n");
  }
  else
  {
    printf("n is not prime number");
  }
  getch();
}
```

5) WAP to add even numbers & odd numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n, i, esum, osum;
    clrscr();
    printf("Enter n\n");
    scanf("%d", &n);
    esum=0;
    osum=0;
    for(i=1; i<=n; i++)
    {
        if(i%2==0)
          esum=esum+i;
        else
          osum=osum+i;
    }
    printf("sum of even numbers=%d\n", esum)
    printf("sum of odd numbers=%d\n", osum
    getch();
}
```

Scanned by CamScanner

6) WAP to add numbers which are divisible by 5 in a given range.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n1, n2, i, sum;
  clrscr();
  printf("Enter range n1 & n2\n");
  scanf("%d%d", &n1, &n2);
  sum=0;
  for(i=n1; i<=n2; i++)
  {
    if(i%5==0)
    {
      printf("%d\n", i);
      sum=sum+i;
    }
  }
  printf("sum=%d", sum);
  getch();
}
```

Scanned by CamScanner

Lab program 3:-

Design & develop a C program to find the reverse of an integer number Num & check whether is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n, rev=0, digit=0, temp; clrscr();
    printf("Enter an integer number\n");
    scanf("%d", &n);
    temp=n;
    while(n!=0)
    {
      digit = n%10;
      rev=rev*10+digit;
      n=n/10;
    }
    printf(" Given number is %d\n", temp);
    printf(" Reverse=%d\n", rev);
    if(rev==temp)
    printf(" Number is palindrome\n");
    else
    printf(" Number is not palindrome\n");
    getch();
}
```

Scanned by CamScanner

Nested loop :-

A loop enclosed within another loop is called nesting of loops.
It is used when a set of statements are to be executed when 2 different conditions are satisfied.

Eg:-
```
for(i=1; i<=m; i++)
{
    for(j=1; j<=n; j++)
    {
        statements;
    }
}
```

Here outer loop executes m times. Each time the control enters inside, the inner loop is executed n times.

1) WAP to print the following pattern

(2) (1)

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

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

```
int i, j;
clrscr();
printf("enter n\n");
scanf("%d", &n);
for(i=1; i<=n; i++)
{
    for(j=1; j<=i; j++)
    {
        printf("*");
    }
    printf("\n");
}
getch();
}
```

→ No. of rows

→ How many times * should be printed in a row

2) WAP to print following pattern

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

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i, j;
    clrscr();
    printf("enter n\n");
    scanf("%d", &n);
```

Scanned by CamScanner

```
for (i=1; i<=n; i++)
{
  for (j=1; j<=i; j++)
  {  printf("%d", i);
  }
  printf("\n");
}
getch();
}
```

WAP to find whether a given number is Armstrong or not.

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
{
  int n, d1, d2, d3, sum;
  clrscr();
  d1 = (n%10);
  d2 = (n/10) % 10;
  d3 = (n/100);
  sum = pow(d1,3) + pow(d2,3) + pow(d3,3);
  if (sum==n)
   printf(" Armstrong num");
  else
   printf(" Not armstrong");
  getch();
}
```

Scanned by CamScanner