Structure of C Programming

05/09/25

① Documentation Section/Comment Section (Optional)

Consists:- Author/ Date n Time/ Brief/ Description

Eg:- // durgaprasad
// 8th sept 2025
/* Learning C programming */

② Link Section (Needed)

```
# include <stdio.h>   → printf, scanf
# include <conio.h>
```

#→ preprocessor directives
① #include
↳ libraries
② #define
↳ constants

③ Defination section (Optional)

It defines symbolic constants.

Eg:- #define PI 3.14
↳ microdefination

④ Global declaration section (Optional)

Variables that are used in 1 or more func^n in a program are declared globally.

→ User defined functions can also be declared in Global Declaration Section.

⑤ Main ~~function~~ section

This section is compulsory having a program only 1 main function. It has 2 parts:- Declaration Variables & Executable part.

```
eg:- void main(){
        getch();
     }

     int main(){
        return 0;
     }
```

⑥ Subprogram section (Optional)

We can include all the user designed section.

SYNTAX

```
// Write a C program to print Hello World
// Author: Durga Prasad Maharana
// Date: 8th Sept 2025
/* Program to print Hello World */
# include <stdio.h>
int main(){
    printf("Hello World");
    return 0;
}
```

OR

```
# include <conio.h>
void main(){
    printf("Hello World");
    getch();
}
```

- Constants
  Types:- ① #define PI 3.14
  ② const int a=5

- Variables in C
  To store values in memory we can use variables. These are named memory locations.
  → There are 2 way to declare:-
  ① 2-step Method:-
  (i) Declaration :- Eg:- int a;
  (ii) Initialization:- Eg:- a=10;
  ② Declaration + Initialization: Eg:- int a=10;

- Data Types in C
  ① Primary :- int, float, double, bool, char, void
  ② Derived :- array, pointer, function, Dynamic Memory Allocation (DMA)
  ③ User Defined :- union, structure, Typedef
  ↳ syntax: type def int dpm;
  dpm=10;

① Primary Datatypes:-
① int:- printf("%d", a);
↳ format specifier

② Character:- char a='b';
printf("%c", a);
printf("%d", a); //98

Format specifier
int => %d
char => %c
float => %f
double => %lf
Unigned int => %u

Commands
① cd Desktop
② mkdir folder name
③ cd folder-name
④ touch file-name.c → //in order to create a file
⑤ nano file-name //open the file
↳ then ctrl+O → ctrl+X
⑥ gcc file-name.c -o name //compile a file
⑦ ./a.out //execute file universal
name

10/09/25

syntax of printf
① printf("statement");
② printf("Format Specifier", variable);

Syntax of scanf (In order to accept values from terminals we use scanf funcⁿ whose defination is present in the library has #include<stdio.h> library.

① scanf("format specifier", &(address of operator) variable
↳ User defined

Operators

→ Types of Operators :- (Based on Operands)

① Unary Operator
② Binary Operator
③ Ternary Operator

1) Unary Operator
↳ (i) Unary minus operator (-a)
(ii) Post increment & pre increment (a++, ++a)
(iii) Post decrement & pre decrement (a--, --a)
(iv) Logical not (!a)
(v) address of (&a) (vi) sizeof()

2) Binary Operator
Arithmetic Operators, Relational Operators, Logic Operators, (&&, ||)
Bitwise Operator (&; |; <<; >>; ^), Equality Operator (==, !=), Comma operator(,),
Assignment operator (=)

3) Ternary Operator : (syntax)
Condition? exp1 : exp2;

Q) WAP to create a calculator with given operations: +, -, *, %, /
Q) write a C program to calculate area & parameters of a circle, square & triangle

```
#include<stdio.h>
float areacircle(float r){
    return 3.14159*r*r;
}
float perimetercircle(float r){
    return 2*3.14159*r;
}
int main(){
    float r, s, b, h, a, c;
```

```
printf("Radius of circle: ");
scanf("%f", &r);
printf("Area: %.2f/n", areacircle(r));
    :
return 0;
}
```

12/09/25

- **Prefix Increament & Post Increament**

Eg:-
```
int y=10;
int x= ++y;
printf("%d", x);
```
↳ y [10] → 11, x [11] o/P:- 11

```
int y=10;
int x= y++;
printf("%d", x); //10
```
↳ y [10], x [10]
then y=10+1=11

pre
1st increment then assign
Post
1st assign the increment

- **Pre-decrement & Post-Decrement**

```
int y=10;
int x= --y;
printf("%d",x); //9
```

```
int y=10;
int x= y--;
printf("%d", x); //10
```

0: false
1: true

Binary Operators

→ `int a= printf("Durga");`
`printf("%d", a); //Durga5`

Q) `int a=4, b=6, result;`
`result= a>b && printf("Durga") || ("Puspal");`
`printf("%d", result);` // Puspal1

Soln:- a [4], b [6], result [ ] → 4>6 (No) :- 0 & _ = 0
0 || printf("puspal") = puspal (6)
as it is conditional statement puspal1

Q2) `result = a>b && printf("Durga") || ("puspal") && ("Lecture");`
`printf("%d", result);`
0, (6), (7) → puspallecture1

Q3) `int a= 1;`
`int b= 6;`
`int result1= a-- && ++b;` // 1 && 7 → 1 (a → 0)
or
↳ `int result2= --a && ++b;` // 0 && 7 → 0

→ **Comma Operator (,)**

```
int a= 5, 6;
int b= (5, 6);
printf("%d", a); //5
printf("%d", b); //6
```

2) int a=8, b;
b = (a++, ++a);
b1 = (a++, ++a);
printf("%d", b); //10
printf("%d", b1); //10

a
11 10 9 [8]

b = (8, 10) = 10
b1 = 10, 12 = 10

Bitwise

13/09/25

Q) int a = 10, b = 5
printf("%d", a & b); //0
printf("%d", a | b); //15 (1111_2)
printf("%d", a ^ b); //15 (1111_2)
printf("%d", a&b && b+1 || 0); //0
0 0 0

<<, >>

Q) int a = 10;
printf("%d", a<<2); //40 $10 \times 2^2 = 40$
printf("%d", a>>2); //2 $\lfloor \frac{10}{2^2} \rfloor = 2$
~~a = a<<b~~
printf("%d", ~a); //-11 $-(a+1) = -11$

→ 00001010 = 10
= 00101000 = 40
→ or $10 \times 2 = 20 \times 2 = 40$
$10 \times 2^2 = 40$

→ 1010 = 10
~0101 = 5

int a = 5, b = 6, c = 11;
printf("%d %d %d", a, b, c);
int d, e, f;
scanf("%d %d %d", &d, &e, &f);

Formating no's in programming output

→ float a = 123.123456789123 45
double b = 123.123456789123 45
printf("%f", a);
printf("%f", b);

→ int a = 12345;
printf("%d", a); 12345
printf("%6d", a); 12345 → Right justified
printf("%-6d", a); 12345
printf("%-06d", a); 123450 (increases value that's why this is wrong)

Syntax
%-d

float:

```
float a = 12345.12345678
printf("%8.2f", a);  // 12345.12
```

↳ 12345.123 → upto 2 decimal places
✓ 8 7 6 5 4 3 2 1

Syntax

%α.βf

where α = min^m no. of pos (digits), including (.) point

β = no. of digits after decimal point

Exception: If the value of α is < the no. of digits assigned before decimal point

[or]

↳ If α < the no. before decimal + β + 1 (decimal point) then digits before decimal point remains as it is, only changes are performed on or before decimal.

Practice

Pg:- 44, 45, 46

Merge 2 sorted arrays in mergetwosortedarrays.c

```
#include<stdio.h>
#define max 100
int main(){
    int p[max], q[max], r[max];
    int m,n;
    int i,j,k;
    printf("Enter length of first array: ");
    scanf("%d", &m);
    printf("Enter %d element in array in sorted order\n", m);
    for(i=0; i<m; i++)
        scanf("%d", &p[i]);
    printf("\n Enter length of 2nd array: ");
    scanf("%d", &n);
    printf("Enter %d element in array in sorted order\n", m);
    for(i=0; i<n; i++)
        scanf("%d", &q[i]);

    i=j=k=0;
    while((i<m) && (j<n)){
        if(p[i]<q[j]){
            r[k++]=p[i++];
        }
        else if(q[j]<p[i]){
            r[k++]=q[j++];
        }
        else {
            r[k++]=p[i++];
        }   r[k++]=p[j++];
    }
    while(i<m){
        r[k++]=p[i++];
    }
    while(j<n){
        r[k++]=q[j++];
    }
    printf("\nSorted Array:\n");
    for(int i=0; i<k; i++){
        printf("%d\n", r[i]);
    }
    return 0;
}
```

Control Statement

15/09/25

There are various types of control statements in C programming such as if, if-else, if-else ladder, nested if, switch, while, do-while, for.

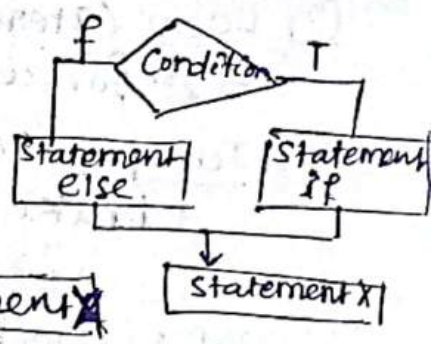
→ Start control - statements - if, if-else, if-else ladder

- if statement

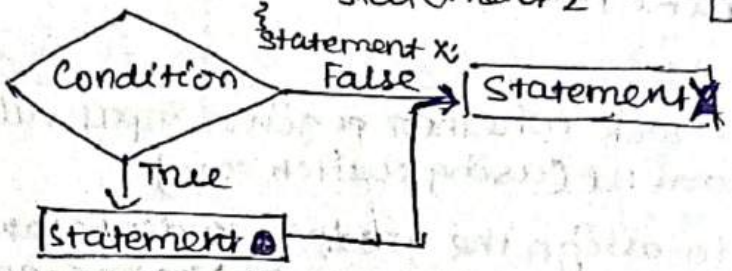
```
if (condition){
    statement
}
statement x;
```

- if-else

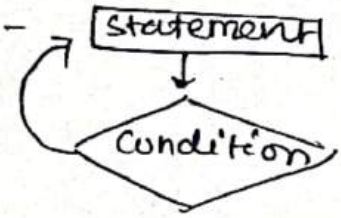
```
if (condition){
    statement 1
}else {
    statement 2
}
statement x;
```



- Start-control -



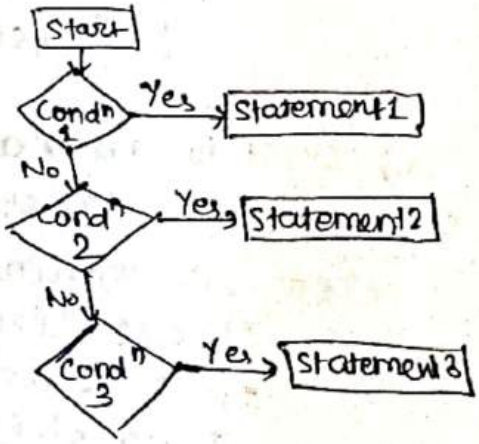
- End-control : Do-while
- Range specific control statement: while, do-while, for
- Condition specific control statement: while
- End-control -



Practice
Pg-58 (1 to 5)

* if-else ladder

```
if (condition 1) {
        // execute code if cond^n 1 is true
}
else if (condition 2) {
        //
}else if (condition 3) {
        //
}else {
        //
}
```



* switch :-

```
switch (controlling expression) {
    case 1:
        statement 1;
        break;
    :
    case n:
        statement n;
        break;
    default:
        statement d;
}
```

(label set)

* Control ~~segment~~ statement in C

The 3 categories of control statements are as follows:

① Decision Making (Selection)
- If, If-else, If-else ladder, nested if & switch case

② Loop (Iteration)
→ for, while, do-while

③ Jump (Branching)
→ brake, continue, go-to

17/09/25

Q6) WAP to check whether a given input value or char is vowel or consonent (using switch case).

Q7) WACP to assign the grade to students based on following marks criteria. Mark > 90: 'O', or >80 <=90: 'A'
>70 <= 80 : 'B' , >60 <=70: 'C', >50 <=60 : 'F'

---

* Nested Loop

```
if(cond1){
    if(cond2){
        //Statement1
    }else{
        //statement2
    }
else(cond3){
    if(cond3){
        //statement3
    }else{
        //statement4
} }
```

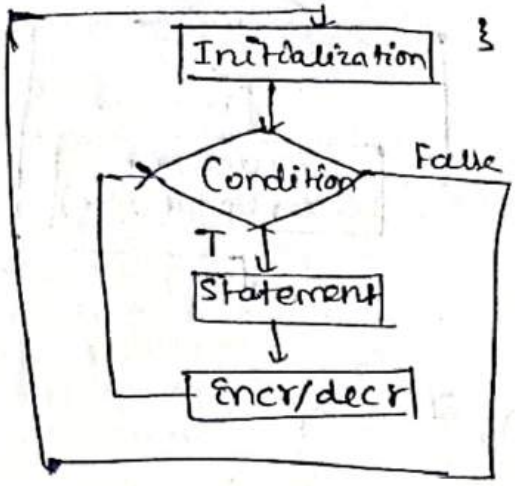
Q8) WACP to find maxm integer among 3 input values.

Q9) WACP that takes a C 3 integer value as length of a triangle side & determine the type of Δ.

If all sides are equal then equilateral
If 2 sides " " " isosceles
If no sides " " then it is scalene
If the sum of any 2 side is !> 3rd side not a valid Δ

② Loop (Iteration)

(i) for loop :- Syntax :- for(initialization; condition; update){
//code to be executed
}



Q) Write a C program to find factorial of a no.

```
int main(){
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    int fact=1;
    for(int i=1; i<=n; i++){
        fact= fact*i;
    }
    printf("Factorial of %d is
     %d", n, fact);
    return 0;
}
```

```
#include<stdio.h>
→ int main(){
    int i, j;
    for( i=0; j=0; i<=5; i++){
        printf("%d %d \n", i, j);
    }
    return 0;
}
```

//O/P

```
0 0
1 0
2 0
3 0
4 0
5 0
```

19/09/25

```
→ int main(){
    int i, j;
    for(i=1; j=0; i<=5; j<3; i++; j++){
        printf("%d %d \n", i, j);
    }
    return 0;
}
```

//O/P

```
1 0
2 1
3 2
```

```
→ int main(){
    int i, j;
    for( i=1; i<=5; i++){
        for(j=0; j<i; j++){
            printf("%d %d \n", i, j);
        }
    }
}
```

//O/P

```
1 0
2 0
2 1
3 0
3 1
3 2
4 0
4 1
4 2
4 3
5 0
5 1
5 2
5 3
5 4
```

```
→ int i, j;
  for(int i=1; j=0; ; i++){
      printf("%d %d \n", i, j);
  }
```

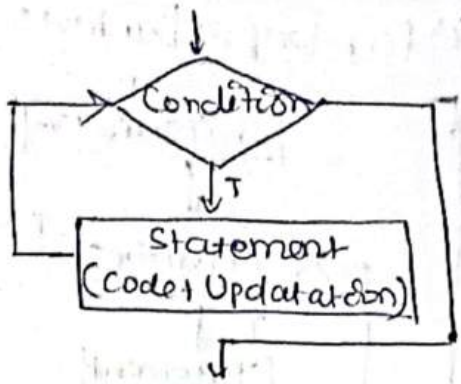
//O/P:

```
1 0
2 0
3 0
⋮
535 0
⋮
```

(ii) while loop :- syntax

```
while (condition) {
    // code to be executed
}   // updation
```

Flow Diagram



Code

```
#include<stdio.h>
int main(){
    int i=0;
    while(i<5){
        printf("%d\n", i);
    }   i++;
}
```

O/P :-
0
1
2
3
4

Q1) WAP that reverse the digit of a number provided by the user.

Q2) WAP using a while loop to count the no. of digits in a no. provided by the user.

Q1)

```
#include<stdio.h>
int main(){
    int n, rev=0, rem; printf("Enter n"); scanf("%d", &n);
    while(n!=0){
        rem = n%10;
        rev = rev*10+rem;
        n = n/10;
    }
    printf("Reverse digit no is %d", rev);
    return 0;
}
```

Q2)

```
#include<stdio.h>
int main(){
    int n, count=0, rem;
    while(n!=0){
        rem = n%10;
        n = n/10;
        count++;
    }
    printf("No. of counts in %d is %d", n, count);
    return 0;
}
```

→ int main(){
int i=0;
while(0); //condition but no statement
{ printf("%d", i);
}
printf("Hello");
return 0;
}

O/P :- 0Hello

→ int main(){
int i=0;
while(i++); {
printf("%d", i);
}
printf("Hello");
return 0;
}

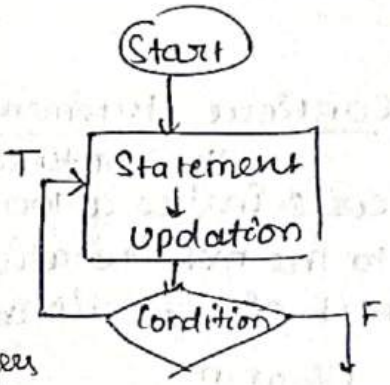
O/P :- 1Hello

i
~~0~~ 1 Hello
=> 1Hello

(iii) do-while :- syntax

```
do{
    //code to executed
    //updation
} while(condition);
```

Flow Diagram



H.A.

Q1) WAP using do-while loop to find the sum of all even no's btwn 2 given integers (start & end values)

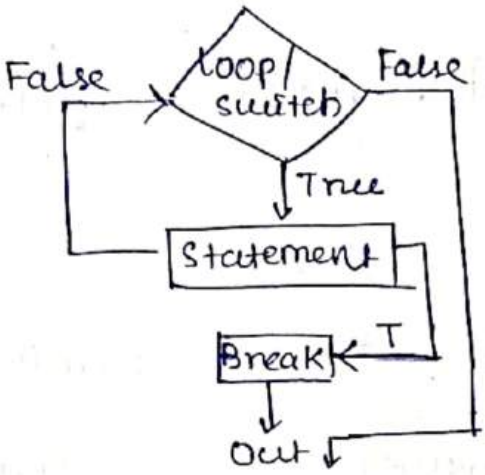
```
#include<stdio.h>
int main(){
    int i,n,m;
    printf("Enter 2 no.s ");
    scanf("%d %d",&n,&m);
    i=n;
    do{ if(i%2==0){
            printf("%d\n",i);
        }
        i++;
    }while(i<=m);
    return 0;
}
```

Q2) WAP using a do-while loop to check whether a given int no. is a strong no. or not.

```
#include<stdio.h>
int main(){
    int n,res,fact;
    do{
        rem=n%10;
        for(int i=0;i<n;i++){
            fact=fact*i;
        }res+=fact;
        n=n/10;
    }
    while(n!=0);
```

→ Break statement is used to ~~abort~~ exit ~~extract~~ a loop or switch statement prematurely, when a break is encountered control immediately exits the loop & execution continues with nxt statement after the loop or switch.

Diagram



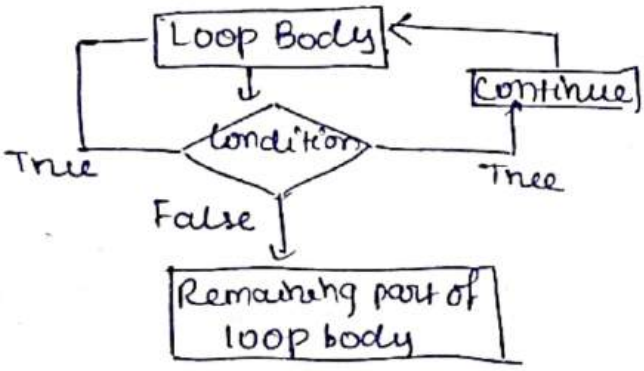

```
#include<stdio.h>
int main(){
int i;
for(i=1;i<=10;i++){
    if(i==5){
        break;
    }
    printf("%d", i);
}
return 0;
}
```

#O/P : 1234

→ Continue Statement :-

The continue statement is used to skip the remaining code inside a loop for the current iteration & move directly to the nxt iteration. It doesn't exit the loop but skips the rest of the code in the loop & continues with the nxt iteration.

Diagram

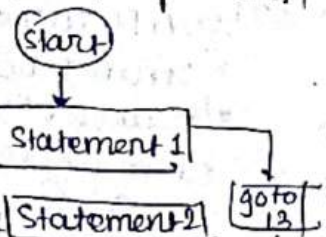



```
#include<stdio.h>
int main(){
    int i;
    for(i=1;i<=10;i++){
        if(i==5){
            continue;
        }
        printf("%d",i);
    }
    return 0;
}
```

#O/P : 12346789 10

→ goto statement :- This allows a program to jump to a labelled statement else where in the code, the label is an identifier followed by a semi-colon (;) & it marks a specific point in the code.

```
#include<stdio.h>
int main(){
    int i=0;
    loopstart:
        if(i<5){
            printf("%d\n",i);
            i++;
        }goto loopstart;
} return 0;
```



Q) Write a C Program that reads an integer from the users until a -ve no. is entered. Display the sum of all the no's entered.

```
#include<stdio.h>
int main(){
    int n,sum=0,i;
    printf("Enter n: ");
    scanf("%d",&n);
    ~~if(n>=0){ for(int i=0; ;i++){~~ if(n<0){
        ~~break; sum+=n;~~ break;
    ~~}~~ }
    ~~if(n<0){~~ else{
        ~~break;~~ for(i=0; ;i++){
    ~~}~~ sum+=n;
    ~~else{}~~ }
}
return 0;
}
```

a)