Date: / / Page no:

↳ Invetor (डेनिसरिची)

C- Dennis Ritche ↳ 1972 - Bell labratery in USA

C++ Bjarne stroustrup (1992)
(जार्न स्ट्रास्ट्रुप)

ASC → Amireance Standed code

Tokens – Single individucal Unit of a program

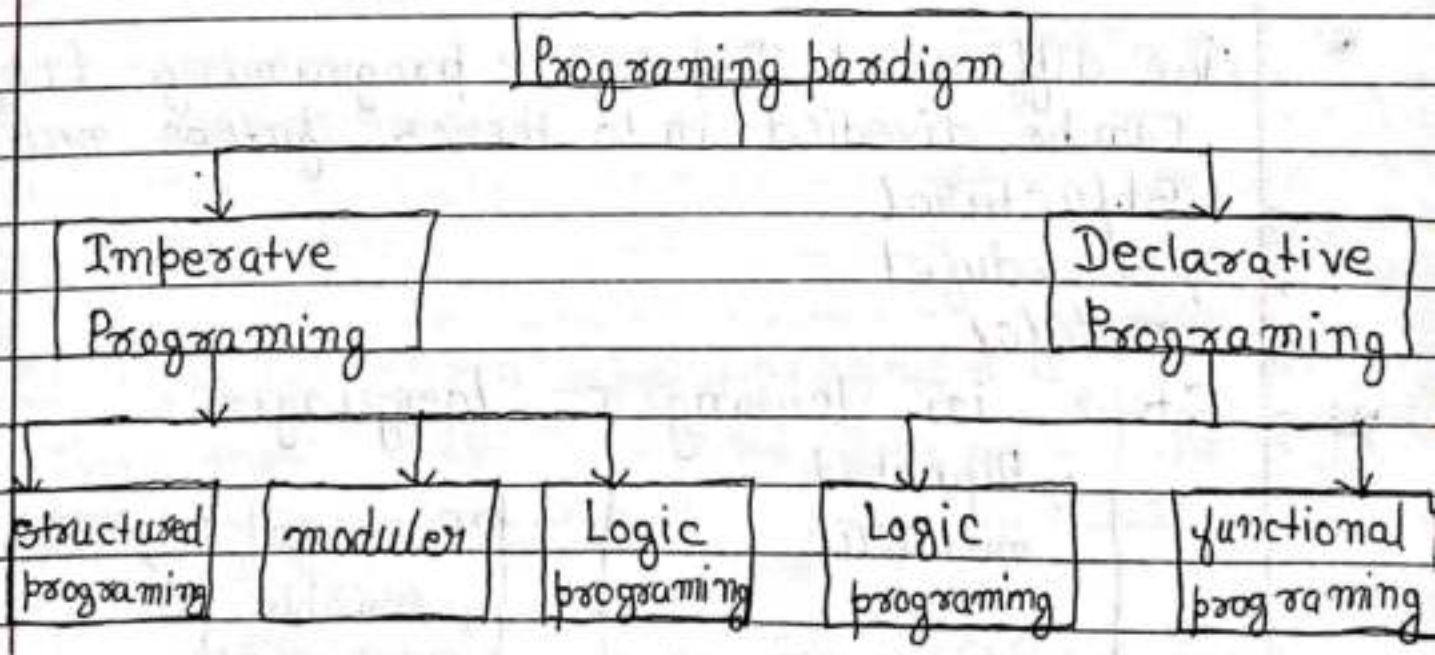
- ↳ Constann
- ↳ keyb (key woards)
- ↳ Variable
- ↳ Idnetifier

→ पहचान करना

↓

Instfuction - program

Programing paradigm



Date: / / Page no:

* The different Impesative programing languages in made up of well defined Sequence of inthuction to a computers. It in the aldent programing languages where the Soubce in a sesiev of commended which opecien with the computer thar to do and when in oldes to achieve the denised Senutd. Values ned in a vasiatle are chaged at program one time. To Conttot the commanded. Conth nttuctule much an loops or branches are integlated in to the code.

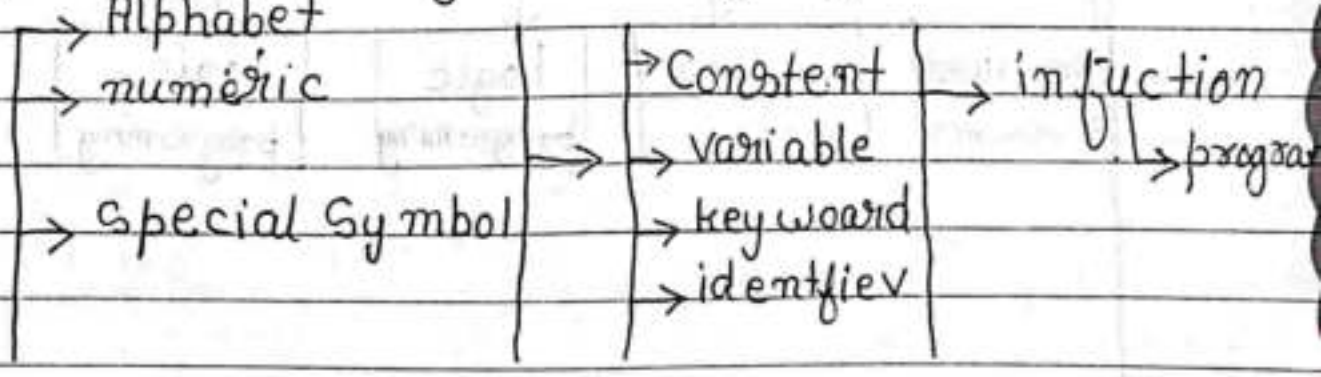
* The best note impesative programing Ianuges are

→ Pascal
→ C
→ Basic etc

* The different Impesative programing Languges Can be divedied in to theres futhes nub afdinate

→ Stfuctural
→ psocedufel
→ modufal

* → Steps in leasing c languages



Date: / / Page no:

Tokens

→ A Single Individual unit of a programing in called Tokens.

Tips of Tokens are / ÷

(i) Constants
(ii) Variable
(iii) Keywoard
(iv) Indentifiers
(vi) Characterss

→ C Constants can be divided in to two mojef constants

Primaly Contants	Secondaly Contants
→ Intgers	→ array
→ Characters	→ Structure
→ float	→ pointers
→	→ Unions
→	→ e.c.t
→	→ Enumeration

* primaly Constant :-

Rules for Counttucting Integers Constann

(i) An Integer constant must have atleamt one digit
(ii) It mouist have decimal point.
(iii) It can be either ponitive of nagative
(iv) If no Singn precearh an integer Constant It in anumed to be positive.
(v) No commam of blank are allowed with an Integ Constant.

Date: / / Page no:

→ Rules for conticting real constant Realy constant are often called as floating point. Contant. The can be written in two pelms

(i) flactional
(ii) exponential

① following suten must be alseved whiche Constucting

(i) A real constans most have in flactional from
(ii) If must have a decimal point
(iii) It could be eithes poitive of nagative
(iv) Default Sign in positive
(v) No commam of blanues are allowed withe real Constant.

② following suten must be alissved while Constu-cting real constants exprened in exponential from.

(i) The mantissa past and the exponetial past should be aepalated by a letter (e) of 'E'.
(ii) The mantissa past may have positive of negative sign
(iii) Default Sing of mantisa past in tve
(iv) The exponent must have atlent one digit, which most betve of-ve default Sign in tve.
(v) Range of real constant exprened in exponential from in — 3.4e38 to + 3.4e38
— 3.4×10^{38} to + 3.4×10^{38}

n) (i) Ac chalater point in a Single alphabet a Single digit of a Single Special Symbol enclosed within Sinale Inverted Commam.

(ii) Both the Inverted comman should point to the left.

Types of C - Variables

A particular type of variables can hold only the
Ex - An Integer variables can hold only the name type of contant, a real variable can hold only a real conntant, and a chatacter variable can hold only a chatacter constant.

The ruten for counticting differant type of contan are different howevers for consthucting variable names of all types the name net of ruten are applied there are given below.

A variable name in any combination of 1-31 alphabets digit of undercafe name compitern allown variable name whane length could be up to 247 chiatacters

Still it could be safes to stick to the rulen of 31 characters. Do not create unnecary long variable name as it add to you typing effeten.

The fist character in the variable name must be an alphalet.

No commam, of blow are allown between variable name.

No special Symbol other than under can b used in a variable name.

Keywords —

Keywords are this words whose meaning was already been explain to the compilires There are only 32 keywords in c.

→

auto	break	case
const	continue	default
double	else	enum
float	for	goto
int	large	register
short	signed	sizeof
struct	switch	typedef
unsigned	void	volatile

Basic data type :- c five fundamental data type

(I) Characters Char (Character)
(II) integer int (integer)
(III) flowaing point float (floating)
(IV) double flonting point double floating point
(V) void

Data type
- primary: void, int, Char, Double, float
- Derived: array, References, pointers
- user defined: Structure, union, Enumerati

Date: / / Page no:

These are decralued using char, int, floating double and void.

Type	memory size	miminal range
char	8	-2^8 to 2^8-1
Signed char	8	0 to $2^{16}-1$
int	16	-32768 to 32767
Signed int	16	-32768 to 32767
Unfined int	16	-32768 to 32767
Short int	16	0 to 65535
unsigned closint	16	0 to 65535
long int		
figned long int		
long long int		

local variable :- that are decled inside a funcation are called local variallen. There variallen are prefered to an amomatic variable. It can be used only by bettem that are inside the lock in which the witten are dealosed. There variablen exit only whis be there of quote in which they are declared in evective.

Operter :- operter in a Symbol to praffice mati of logic function c langwse & rich an operter and provide are provider The following 6 types of opertors.

(1) Airthemetic opexetoxs (2) Relatinal opexetoxs
(3) logical opexetoxs (4) Bitwise opexetoxs
(5) assignement opexetoxs (6) misc opexetoxs

Date: / / Page no:

Airthemetic operetoxs :- An airthemetic operatoxs methantical operetoxs, such as addition, Subtraction, multiplixcation, devision, modula Increement and decreement on numerical values (constants and variables)

m:- #include <Stdio.h>
#include <conio.h>
void main()
{
int a, b, ;
float c;
Clrscr();
printf("enter two numbers\n");
Scanf("%d%d", &a, &b);
c= a+b;
printf("\n Addition of %d and %d %f", a, b, c);
c = a-b;
printf("\n Subtraction of %d and %d is %f", a, b, c);
c = a*b;
printf("\n multiplixation of %d and %d is %f", a, b, c);
c = (float) a/b;
printf("\n Division of %d by %d" is %f", a, b, c);
a++;
printf("\n Increement in a is %d", a);
b--;
printf("\n Decreement in b is %d", b);
c= a%b;
printf("\n Remainder of %d by %d is %f", a, b, c);
getch();
}

Date: / / Page no:

Relational operators :- The relational operators (comparision operators) are used to check the check wheather two are equal or not, equal or ien than or greater than etc.
It returns if the relational relationship check pamen otheiwise it returns.

for Ex :- if we to numebrs 14 and 7, and if I say is 14 is greater than 7 this is thur hence, this check will returns, an the other hand if I say 14 in lem than 7 this is false hence it will returnes.

Operators	Discription	Ex:- a and b;
= =	check if two operators are equal	Where a=10, b=11 (a == b it returns 0
! =	check of two operators are not equal	(a, 1= b it returns
>	check of two operators no left is gra to than operand on right	(a>b) it returns 0
<	check of operand on left is matter than operaton on right	(a<b) it returns 1
> =	it ckek if operand on left is geater than or equal to operand on right	(a>=b) if returns
< =	it check of operandon left is reatler than or equal to operand right	(a<=b) it returns 11

```
#include <stdio.h>
#include <conio.h>
{
int a, b, result;
printf ("Enter any two numbers\n");
Scanf ("%d%d", &a, &b);
result = (a==b);
printf ("\n Is (a==b) is %d", result);
result = (a!=b);
printf ("\n (a!=b) is %", result);
result = (a<b);
printf ("\n (a<b) is %d", result);
result = (a>b);
printf ("\n (a>b) is %d", result);
result = (a<=b);
printf ("\n (a<=b) = %d", result);
result (a>=b);
printf ("\n (a>=b) = %d", result);
getch ();
}
```

Logical operator :- C languages support the following three logical operator.

Operators	Discription	Example
&&	Logical AND	a=0, b=1, (a&&b)=0 returns 0
||	Logical OR	(a||b)=1 returns 1
!	Logical OR	|a=1 returns 1, !b=0 returns 0

Date: / / Page no:

These operator are used to peryorm logical operations if used with conditional statments like of else.

With logical and Operator the result is true only if both the operents are true.

With OR operator the result will be trus of a single operator is true.

the logical not operator changes true value to false and false and false value to true.

```
#include < Stdio.h >
#include < Conio.h >
 Void main ()
 {
int a, b, c;
Clrscr;
printf (" enter any two binary digit \n");
Scanf (" %d %d", &a, &b);
c = (a && b);
printf (" logical AND on a and b given %d \n", c);
c = (a || b);
printf (" Logical OR on a and b is %d \n" c);
c = !a;
printf (" Logical Not on a is %d \n" c);
getch ();
 3
```

Bitwise operators in C ÷ Bitwise operators perform manupulation data at the bit level (binary digit) This operation also performs swifting of bits from right to left Bitwise operators are no applied in float, double, long double type of values.

Bitwise operators in C

Operators	meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise Complement
<<	Bitwise left
>>	Bitwise Right

* Bitwise AND ÷ The output of Bitwise AND is if the coresponding bits of two Depriants is, If either bits of operant is zero. the result of coresponding bits equlated is zero.

Ex ÷ 25 — 11001
12 — 1100

25 — 0001 1001
12 — 000 1100
8 — 0000 1000

Program :-

```
#include <stdio.h>
#include <conio.h>
 void main();
{
int a,b;
clrscr();
printf("enter two decimal numbers\n");
Scanf("%d%d", &a, &b);
printf("Bitwise AND will give result as:%d", a&b);
getch();
}
```

⊗ Betwise (OR) operators :- The output of Betwise OR is 1 if at least one corresponding bit of two operants.

```
    25 - 11001
    12 - 01100
  | 25 - 00011001
  29 - 00011101
     2^4 2^3 2^2 2^1 2^0
```

```
#include<stdio.h>
#include<conio.h>
 void main();
{
 int a,b;
clrscr();
printf("enter two decimal numbers\n");
Scanf("%d%d", &a, &b);
printf("Betwise OR will give result as:%d", a|b);
getch();
}
```

* Betwise∧(XOR) :- The result of Betwise XOR is 1 if the corresponding bits of two operands are opposite.

$$\begin{array}{rl} 25 = & 00011001 \\ \underline{\wedge 12 = } & \underline{00001100} \\ 21 = & 00010101 \end{array}$$

* Betwise complement (~) operator :- is an Urinary operator its changes 1 to 0 and 0 to 1.

$$35 - 00100011$$
$$\sim 35 - 11011100 \quad (280)$$
$$2^7 2^6 2^4 + 2^3 2^2$$

```
#linclude <Stdio.h>
#linclude< conio.h>
void main()
{
int a, b;
clrscr();
printf("enter any decimal number);
Scanf("%d%d", &a, &b);
printf("\n Betwise Complement of a is %d", ~a);
printf("\n Betwise Complement of b is %d", ~b);
getch();
}
```

Date: / / Page no:

* Betwise Right shift (>>) operators :- Betwise right shift all the bits to wards by certain number of speelfyed bits and it is denoted by (>>).

$$212 = 11010100$$
$$212 >> 2 = 00110101 \ (53)$$
$$q^5 q^3 q^2 q^0$$
$$212 >> 4 = 00\ 00110101 \ (13)$$
$$q^3 \ q^2 \ q^0$$
$$212 >> 7 = 00000000 \ (1)$$

* Betwise left shift (<<) operators :- Betwise left Shift operators. Shift all the bits towards left bit positions that have been vacated by the left shift operators with zero.

Symbol (<<)

$$212 = 11010100$$
$$212 << 2 = 01010000$$
$$212 << 4 = 01000000$$
$$212 << 7 = 00000000$$

Date: / / Page no:

Betwise left shift(<<) operators program :-

```
#include<stdio.h>
#include<conio.h>
void main()
int a = 2+2, i;
for (i=0; i<=2; i++)
{
printf("Right shift by %d = %d\n", i, num>>i);
}
for (i=0, i=2, i++)
{
printf("left shirt by %d = %d \n", i num<<i);
getch();
}
```

* Assignment operators :- The assignment operators are used to asign valued for variable for example, if we want to asign valued 10 to a variable as then we can do this by using the assignment operators, x=10 Here equal to operator (=) is used to asigned the value. However it has serieral other veriant i.e +=, -=, /=, x=, %.=

① Operators	Description	a & bare two variable
+ -	i+ add right operant to test operant assing the value in Left %. operant	a = 10, y = 5 a+b, i.e a = a+b a = 15, b = 5

Date: / / Page no:

(ii) -= Subtract right operants to left operants it assign to left operats — a-=b i.e a=a-b, a=5, b=5

(iii) x= multiply righ operants to left operant and assig to left operant — ax=b; ie a=ax b, a=50, b=5

(iv) /= Divid left operants to right operantand assign to left operants — a%=b; i.e a=a% b, a=0, b=5

```
#include<stdio.h>
#include<conio.h>
 void main()
{
int a=10, b=5;
a+=b;
printf("a=%d and b=%d \n", a, b);
a-=b;
printf("a=%d and b=%d \n", a, b);
ax=b;
printf("a=%d and b=%d \n", a, b);
a%=b;
printf("a=%d and b=%d \n", a, b);
getch();
}
```

* Miscell -- operators :- There are few important operators including Sizeof ?. Supoortet by the c languges.

	Operators	Description	Example
①	Sizeof()	returns the size of variable	Sizeof(a) where a is int returns 2 bytes. So it returns the actual variable
②	&	Relative the address of a variable	
③	⚠	pointer is a variable keeps the address	*a

* ?: conditinal operators :- if conditinal is true ? the value is executed else (:) another value is executed.

```
#include < stdio.h >
#include < conio.h >
void main()
{
int a, *p;
clrscr();
printf("enter value of a")
Scanf("%d", &a);
p = &a
printf("Address of a is %a", &a);
printf("Address that p holds is %a", p);
```

printf ("value that pointer p holds is %d", *p);
getch();
3

Operators	Description	Example
() function call	L to R	14 (higest)
[] Array Subscript	L to R	14
(int) pointer of string	L to R	14
->, array (number of string)	L to R	14
! Logical: NOT	R to L	13
~ one's compliment	R to L	13
- Unary minu	R to L	13
++ ncrement	R to L	13
-- decrement	R to L	13
& Address of	R to L	13
* value int	R to L	13
type cast	R to L	13
Size of	R to L	13
*	multiplication (L to R)	12
/ Division	L to R	12
% Modulo	L to R	12
+ addition	L to R	11
- Subtraction	L to R	11
<< left shift	L to R	10
>> Right shift	L to R	10
<	L to R	9
>	L to R	9
<=	L to R	9
>=	L to R	9

= =	equal to	L to R	8
! =	Not equal to	L to R	8
&	Bitwise AND	L to R	7
~	Bitwise Exor	L to R	6
!	Bitwise OR	L to R	3
&&	Logical AND	L to R	4
||	Logical OR	L to R	3
? :	conditinal	R to L	2
=, -=, *=, %=, /=	Assign	R to L	1
,	Comma, comma	L to R	0

* Comments in c :- Comments are used to provide information about line of code. It is wialy used for documenting code. There are two types of comment in c languges.

(1) Single line comment // ——— //
(2) multiple line comment /* ..... */

* C format Specifer :- The format Speeitfier is the string used in the formated input & output funcation. The format string determines the formet of the input & Output. It always starts with percentage char (%d)

The commonly used format Sepeeifier in printf funcation are (%d or %i)

(1) (%d or %i) :- it is used to print Sing interger value where Singned integer means that Variable can hold both positive and negative.

(ii) %u ÷ It is used to print unsigned integer value where the unsigned mens where a variable can hold only positive value.

(iii) %o ÷ It is used to print u the otal usigned integer value where octal integer value always starts with zero (0 to 7)

(iv) %x ÷ it is used to print the hexa decimal usigened integer where the hexa decimal value always starts with zero. In this format depeling alphabetical char. are printed in small vetter.

(v) %X ÷ it is used to print the hexa decimal unsigned integer but the alphabetica e. char. are printed in upper case.

(v) %f ÷ it is used to used to print the flating point value. By defult sixth value. after decimal.

(vii) %e or %E ÷ It is used for Scientific propagab It is also know is mantiso and exponent

(viii) %g ÷ it is used to print the decimal floating point values and it uses the fixed decision that is the value after the decimal be exactly the same. as

as the value in output.

(VI) %p ÷ if is used to print the address in a hexa decimal from.

(⊗) %c ÷ it is used to print the string values.

(XII) %s ÷ it is used to print the longe signed integer value.

(⊛) ASCII ÷ (American Standard Code for information intere)

It is char in coding c used for electronic Communication. Each char, or special char represented by some ASCII Code and each ASCII Code occupy → bits in a memory.

In c programing charrater variable does not contain a Charrater value itself whether The ASCII value of charrter. The ASCII value represent the Charater in number and each charater is assigned with some number from 0 to 27.

Date: / / Page no:

C formatt Specifier :-

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int k;
printf("printing ascii values formate to 127")
for (k=0, k=127; k++)
{
printf("\n the ascii value of %c is %d", k, k);
}
getch();
}
```

[Unit-3]

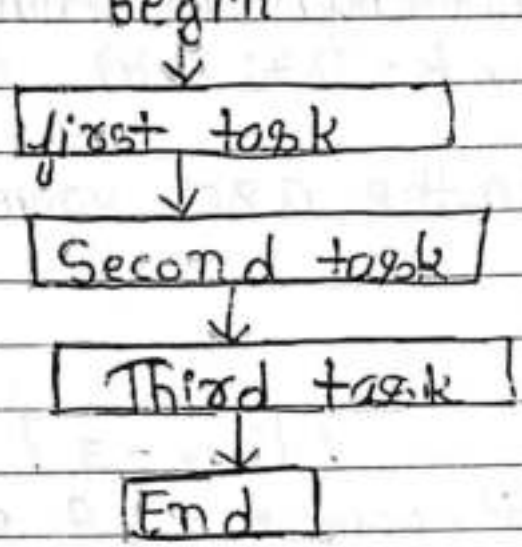
Contral Structure :- A statement that is used to controld the flow of execution in a program is called control Statemen I Combins intruction in to logical unit logical unit has one entry point and one exest pointe.

Types of Contral Structures

(i) Sequence
(ii) Selection
(iii) Repetation
(iv) Juncation Cell

Sequence ÷ Statement are executed in a Sepcified order no statement is skipped and no statements is executed more than ones

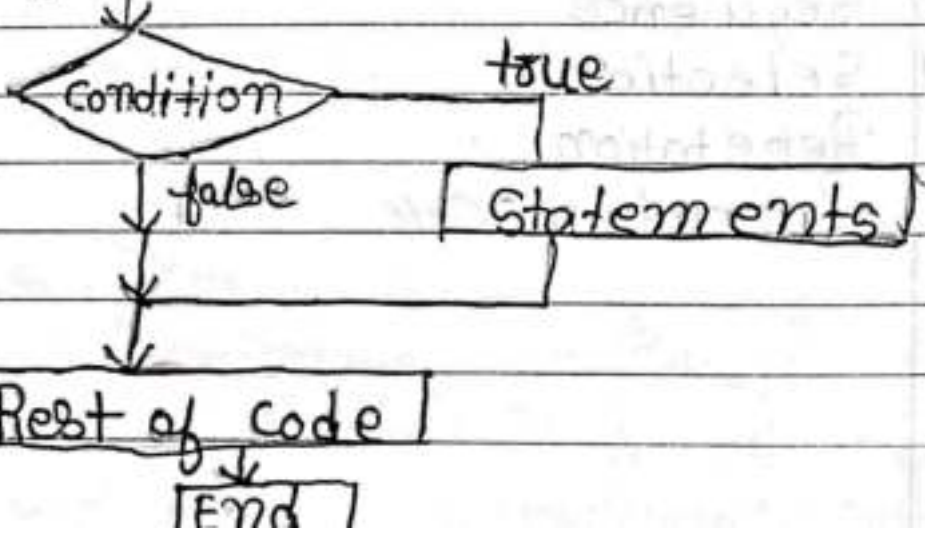
flowchart



Selection ÷ A selection a statement to executed on the basis of condition statements and excuted when the condition is true and iagore when it is false.

for example ÷ I.f, Ifelse, if Switch

flowchart



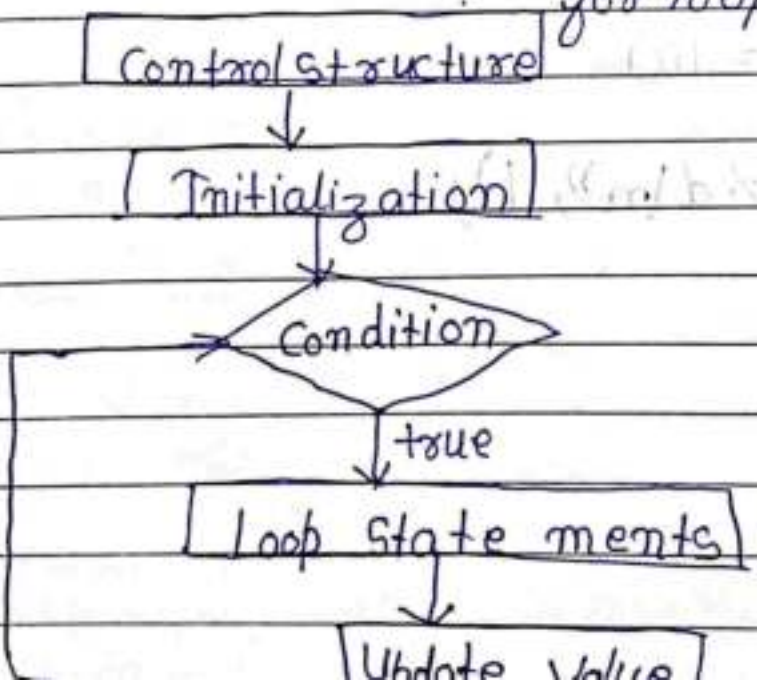
Date: / / Page no:

Programs

```
#include < stdio.h >
#include < conio.h >
Void main ()
{
int year;
clrscr ();
printf ("enter any year");
Scanf ("%d", & year");
if (year % 4 = 0)
printf (" It is a leap year")
else
printf (" It is not leap year")
getch ();
}
```

Reptation :- In this statements the statements are excuted, are excuted more than one time it is also know as Heration or loop.

for example :- While loop, Do while loop, for loop.



Date: / / Page no:

Write a program to print first in natural number.

[for]

```
Void main ()
{
int i;
clrscr ();
for (i=1; i<=10; i++)
{
printf ("%d \n", i);
}
getch ();
}
```

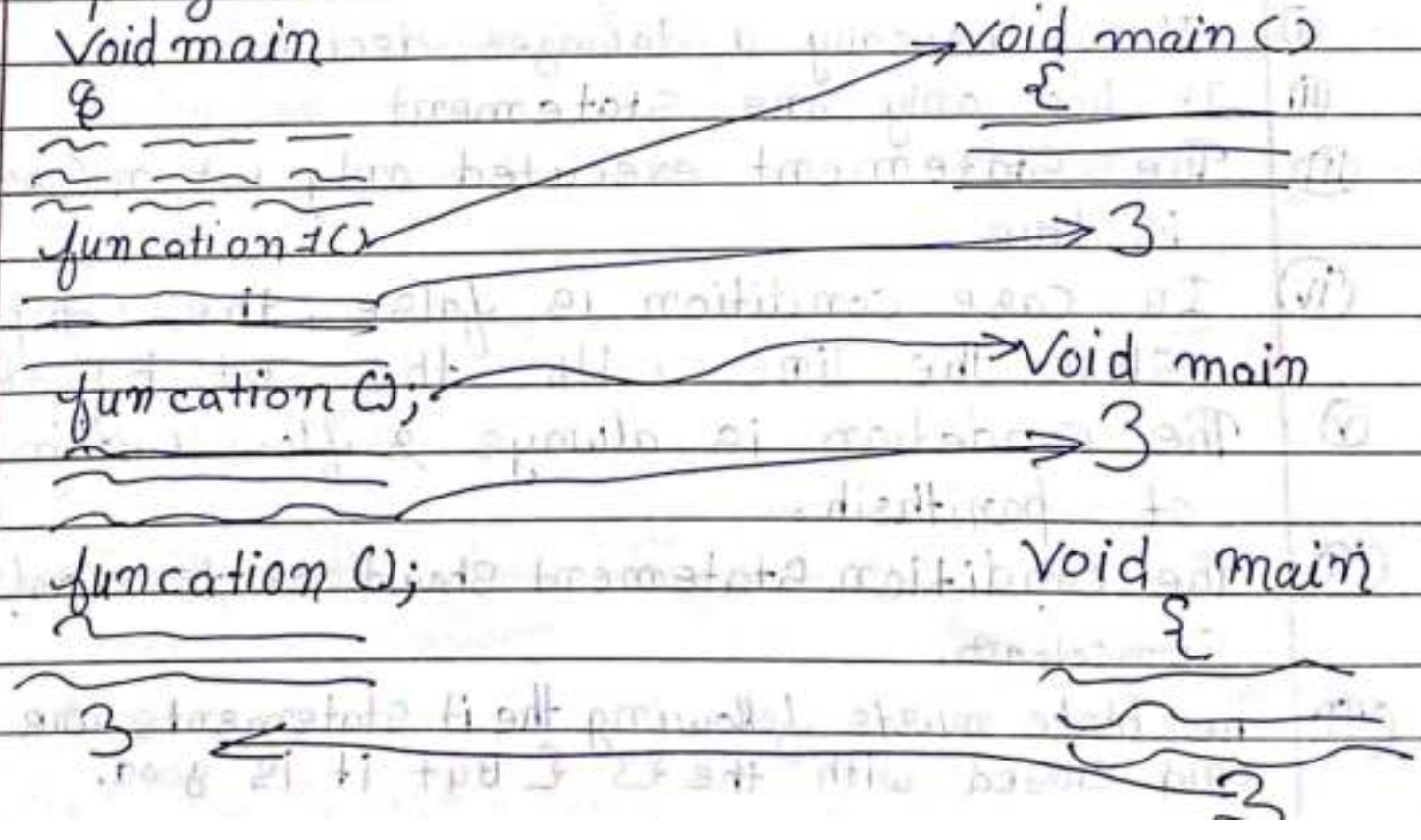
[While loop]

```
Void main ()
{
int i;
clrscr ();
i=1;
while (i<=10)
{
printf ("%d \n", i);
i++;
}
```

Do while :-

```
Void main()
{
int i=1;
clrscr();
do
{
printf("%d\n", i);
i++;
}
while(i<=10);
getch();
}
```

Funcation call :- It is used to call a piece code or statement. In this case the control jumps from the main program to that be solve the code the returns back to main program.



Date: / / Page no:

Descision making Statements

① Simple of Statement

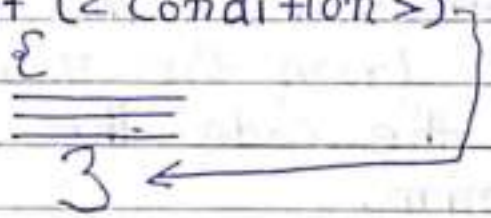
② it else Statement

③ Nested else Statement
else it ladder switch
Statement

Simple of Statement → It Statements in a power full descision making statement and is use to Control the How of.

Syntax :-

```
it (< condition >)
{
=========
}
```

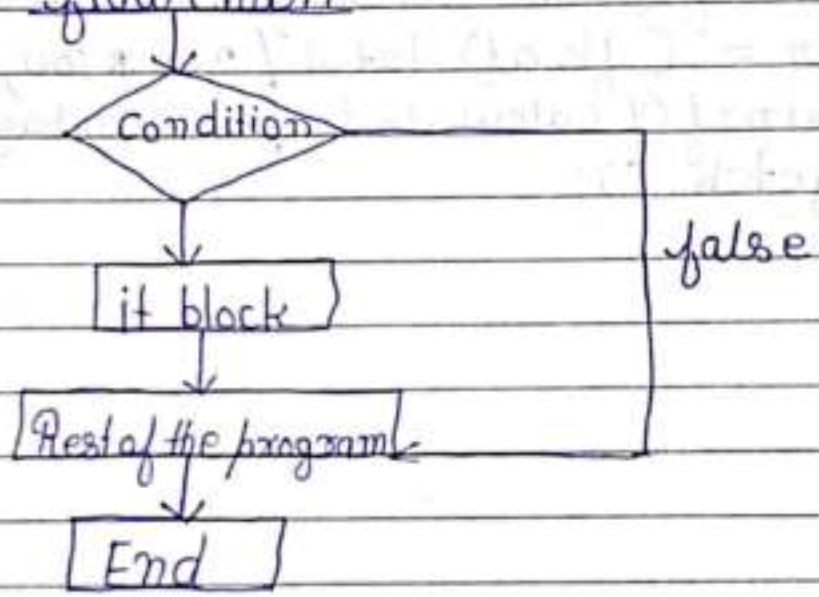


① It is basically a towages decision
② It has only one statement
③ The Statement executed only when (condition) is true
④ In case condition is false, the compilex skills the line with the ut block
⑤ The condetion is always s flow within a pair of parithsih.
⑥ The condition statement should not terminals with semicolons.
⑦ The state musts following the it statements are normals and closed with the {} But it is good.

(ix) If thelest Experience condition are skip true the statement block will be and thone best of the The program.

(*) If it test Exiperience, condition are false than statement block will be the programm best af excuted.

flow chart



(Q) Write af program to then a input is a marks as marks as three Subject PCM
Calculet the total averge af and presentage af three Subjects.

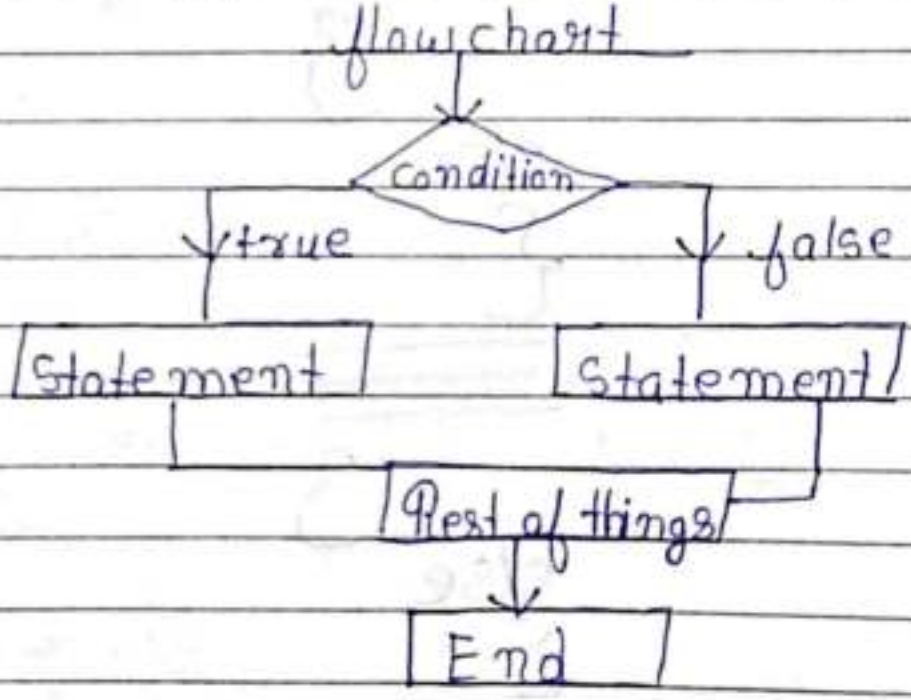
```
Void main()
{
int physics, chemistry, maths, total;
float avg , pres
clrscr();
```

```
printf ("enter marks of physics\n");
Scanf ("%d", &physics);
printf ("enter marks of chemistry\n");
Scanf ("%d", &chemistry);
printf ("enter marks of maths\n");
Scanf ("%d", &maths);
total = (Physics + chemistry + maths);
printf ("total marks are %d", total);
avg = (float) total /3;
printf ("the average marks are if", avg);
per = (float) total / 300 * 100;
printf ("Calculated percentage is if", per);
getch ();
3
```

if else statement :- of exigute only when the condition following is true if class nothing is F but in if-else either true block or false block will be exigutel and not be used without if.

Syntax :-

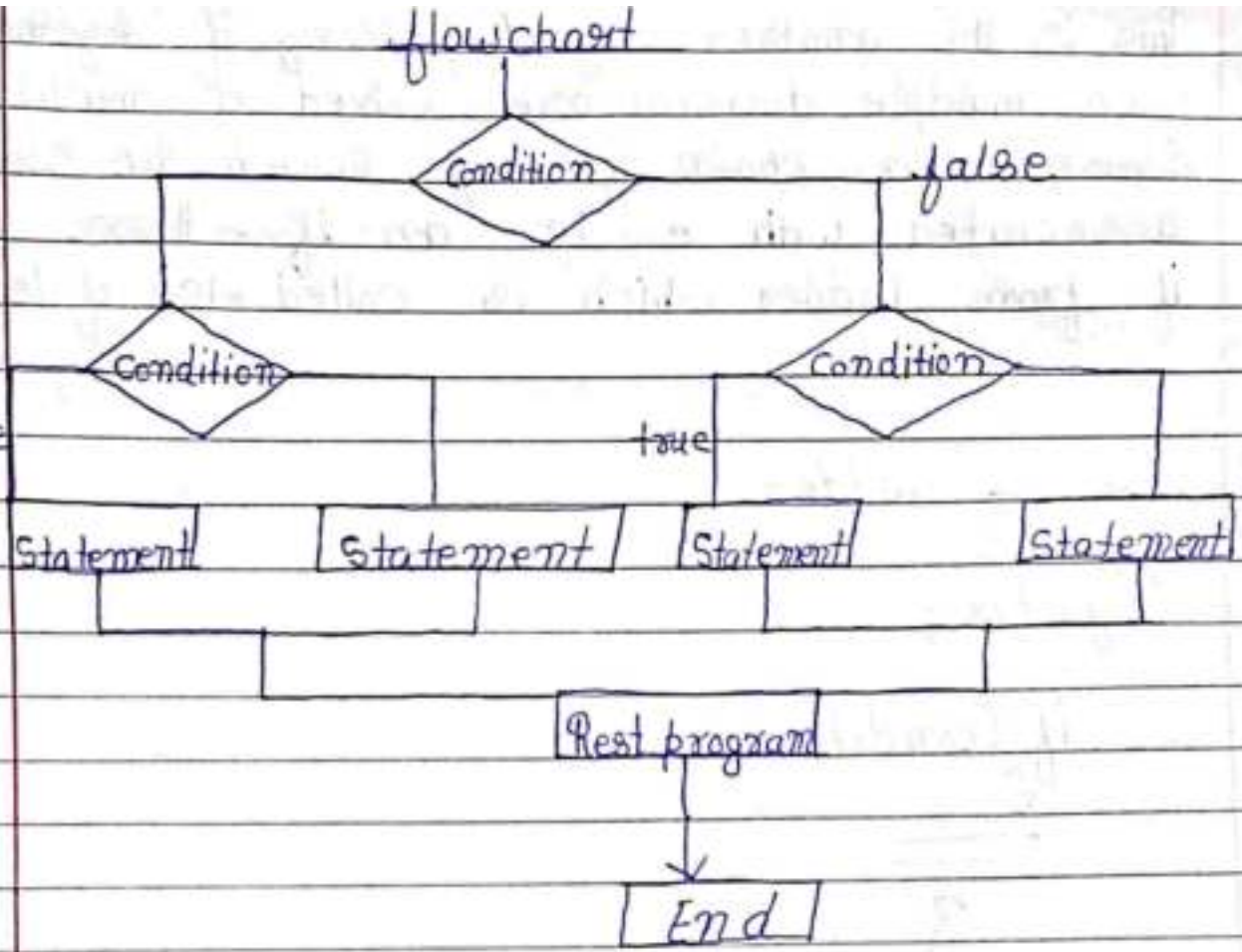
```
if (test condition)
{
Statement;
3
else
{
Statement
3
```



Nested if else statement :- Using of one if else statement in another if else statement is called Nested if else Control Statement when a series of decissions are envolve we my have to use more than one if on else statement in Nested from

Syntax :-

```
if (test condition)
{
    if (test condition)
    {
        ========
        ========
        ========
    }
    else
    {
        ========
        ========
        ========
    }
}
else
{
    if (condition)
    ========
    else
    {
        ========
        ========
    }
```



Nested if else

```
Void main()
{
int a, b, c;
clrscr();
printf("enter three numbers");
Scanf("%d%d%d", &a, &b, &c);
if (a>b)
{
if (a>c)
printf("a is greater");
else
printf("c is greater");
}
else
{
if (b>c)
```

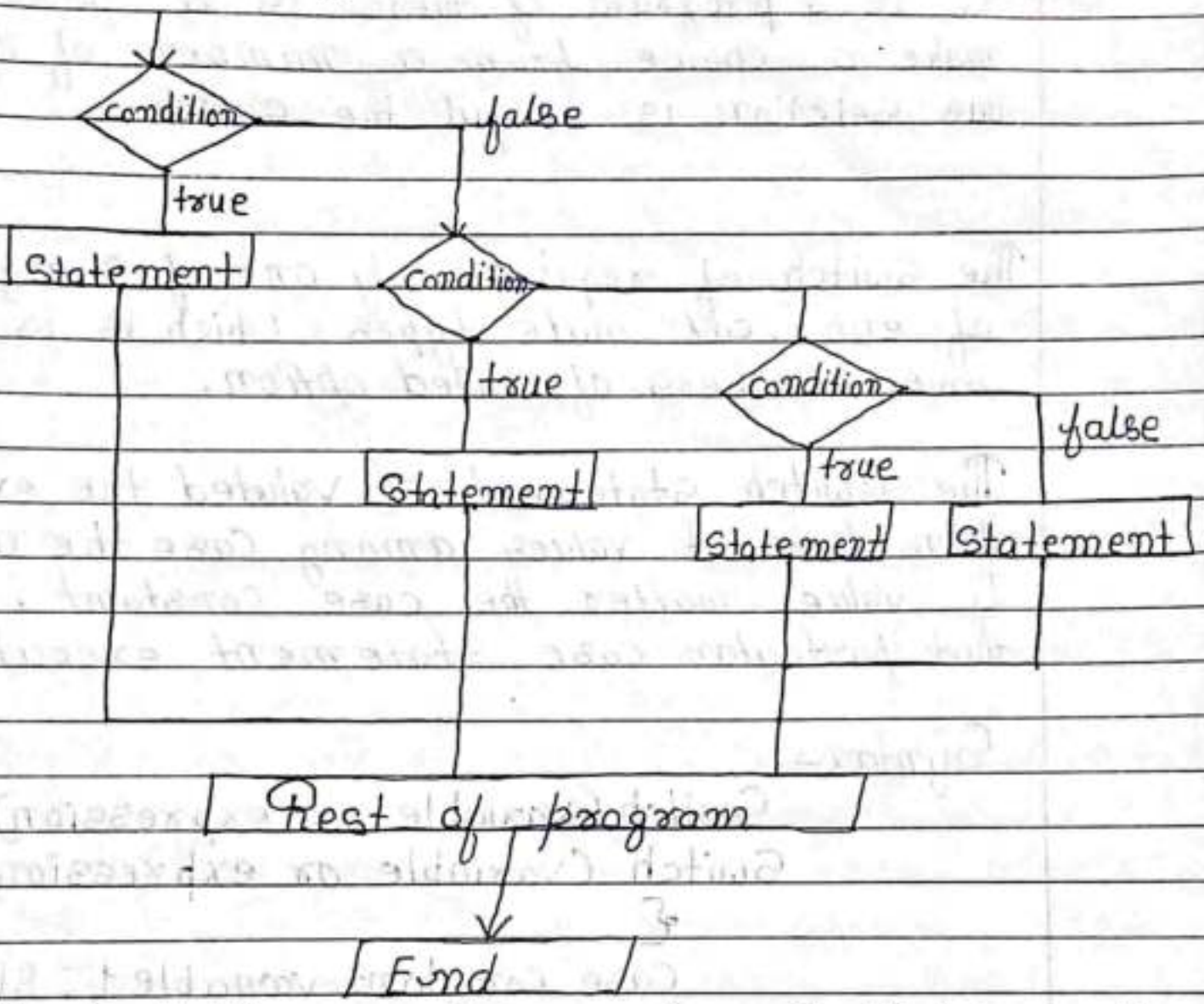
This is the another way of putting if together When mulitple decisiol are volved a multiple dession is a chain if's if switch the statement associated with e is an if . there for if from ladder which is called else if ledder.

else of ladder

Syntax :-

```
if (condition)
  £
  ===
  3
else if (condition)
  £
  ===
  3
else if (condition)
  £
  ===
  3
else
  £
  ===
  3
```

flowchart



The (Switch) case Statement :- Switch is another condional contrel Statement use. its seclet on option from several option based on given expression Value.

This is am alernetiec method to if else or s-else ladder The switch Statement causes the particulam group of Statment to be Choosen from several group the Selection is based on the current value of an expresion which is included-with Switch Statement.

Date: / / Page no:

Switch statement is a multiway branch statement the In a program if there is a possibility to make a choice from a number of options this selction is useful the switch

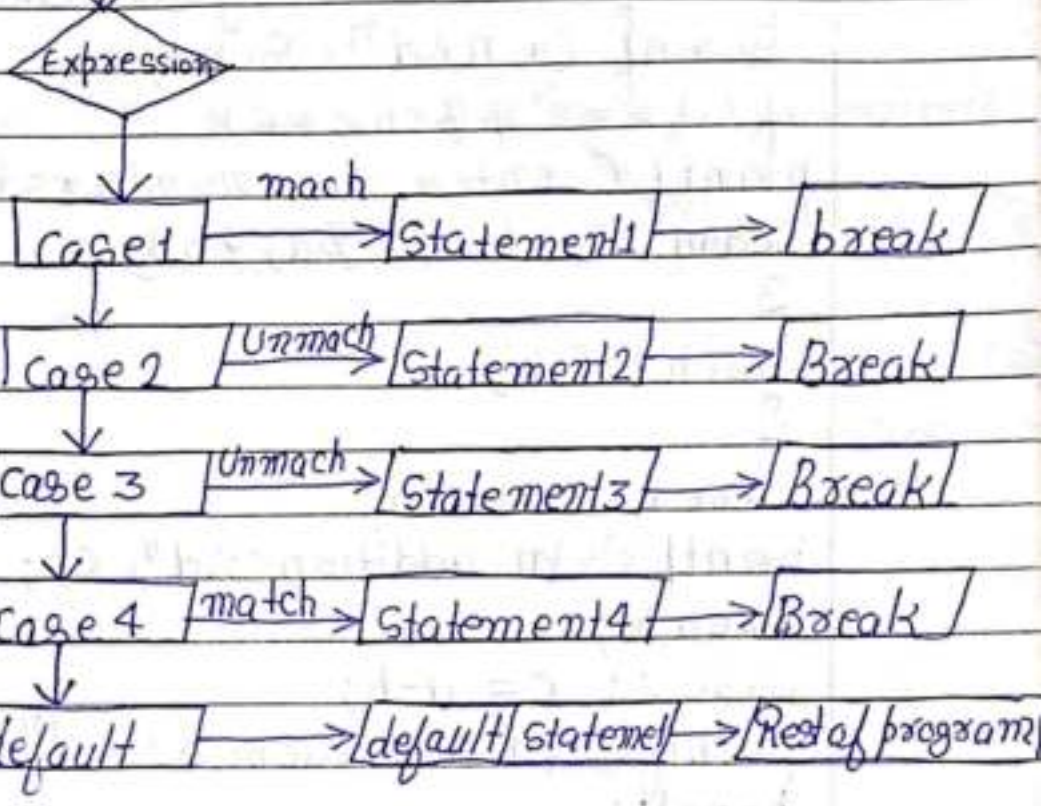
The switch of requted only one of $ argrment of ent , cat nate types . which is is cheak wave numbers of called option.

The switch statement a valueted the expersion then Looks it values among case the constant It value matter the case constant . Then the that partcular case statement executed.

Syntax :-

```
Switch (variable or expression)
Switch (variable or expression)
    {
    Case constant variable 1 : Block1;
    Case constant variable    Block 2;
                              break;
    Case constant value n:    Block;
                              Break;
    default : default block ;
```

flowchart



Write a program to provide ÷ multiple funcation such as 1 funcation 2 for substraction 3 for multiple 3 4 for division 5 for remender 6 for larger of two 7 exist using switch Statement.

```
Void main()
{
int a, b, c, ck;
clrscr();
printf("\n\t menu");
printf("\n\t [1] addition");
printf("\n\t [2] Subtraction");
printf("\n\t [3] multiplication");
printf("\n\t [4] Division");
printf("\n\t [5] Remindex");
printf("\n\t [6] Laigex of two");
```

```
printf ("\nt [7] exct");
printf ("\nt enter your choice");
Scanf ("n%d" &ch);
if (ch>= 1&&ch<=6)
printf (" enter two numbers\n");
Scanf ("%d%d", &a, &b);
3
Switch (ch)
{
Case 1: c = a+b;
printf ("\n addition: %d", c);
break;
Case 2: c= a-b;
printf ("\n subtraction: %d", c);
break;
case 3: c= a*b
print ("\n multiplication: %d", c);
break;
case 4: c = a/b;
printf ("\n Division= %d; c);
break;
case 6: if (a >b)
printf ("% disgeraler", a);
else
printf ("%d is gerater", b);
break;
case 7: printf ("ferminated by choice");
exit();
default: printf ("favalid choice");
3
getch();
3
```

Date: / / Page no:

Loop control Statements in c :-

Loop :- A loop is defined as a block of Statement which are repatitly executed for certain number of times. In other word in a code for group of many.

Why we are using loop :- The looping sing, simply the complex-problem in to the esely. It entry enables us to after the flow of program so that instant of writting the program same code agian and agin the can repeat the same code for finite no of times.

for example :- If we need in to print fix 10 natural number then of using the printf Statement to times, We Can printf is inside a loop which runs up in 10 equation.

Advantage of loops in c

- It provides for the usibility
- Using loop we do not need to write the same code agian and agian.
- Using data structurs arrays and linked

Date: / / Page no:

Types of loops

Threer are three types of loops.

(i) Do while (ii) While (iii) for

Do while :- The do while loop continues is execution unit a given condition satisfied. It is also called past tested loop.

It is used whet it is necessary to executed the loop at least once.

Syntax :- do

{

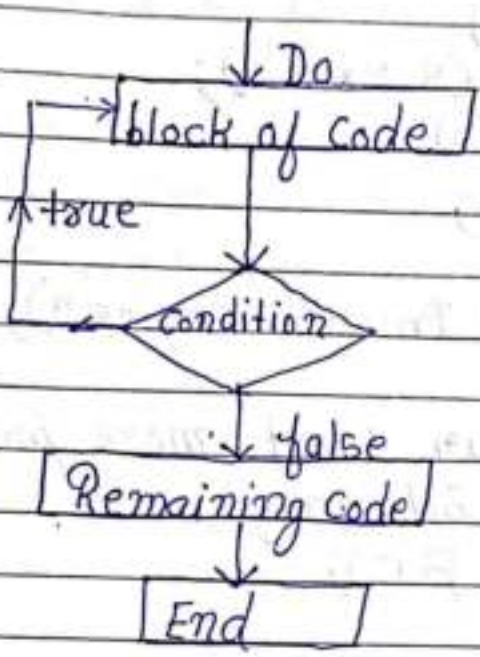
block of Code.

}

While (test condition);

Do while :- It is used whet it is necessary to executed the loop at least once.

flow chart



Program :-

```
#include<stdio.h>
#include<conio.h>
Void main()
{
Char C;
int choice, dummy;
do
{
printf("\n1. Hello");
printf("\n2. today is holiday");
printf("\n3. exit");
printf("\n enter choice");
Scanf("%d", &choice);
Switch (choice)
{
Case 1: printf("Hello");
        break;
```

```
Case 2: printf ("today is holiday");
        break;
Case 3: printf ("exit");
        exit (0);
        break;
default:
        printf ("Invalid choice");
}
printf ("Do your want more printf");
Scanf ("%d", &dummy);
Scanf ("%c", &c);
}
While (c == 'y');
getch ();
}
```

Q. Write of program to print multiple option and choice one of then, using switch and do while.

Write a program to print the table of any number (0).

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
printf ("enter an integer");
Scanf ("%d", &n);
for(int i= 1; i< 10; ++i)
```

```
Case 2: printf ("today is holiday");
        break;
Case 3: printf ("exit");
        exit (0);
        break;
default:
        printf ("Invalid choice");
}
printf ("Do your want more printf");
Scanf ("%d", &dummy);
Scanf ("%c", &c);
}
While (c == 'y');
getch ();
}
```

(Q) Write of program to print multiple option and choice one of then, using Switch and do while.

Write a program to print the table of any number (n).

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n;
printf ("enter an integer");
Scanf ("%d", &n);
for(int i = 1; i<= 10; ++i)
```

```
{
printf ("%d * %d \n", n, i, n * i);
}
getch ();
}
```

(1) Write a program to print the sum of fist 10 natural numbers.

```
#include <stdio.h>
#include <conio.h>
void main()
{
int sum=0;
for (int i= 1; i<=10; ++i)
  { sum += i
}
printf ("sum of the first 10 natural numbers %d \n", sum);
getch ();
}
void main()
{
int i sum=0;
printf ("The first 10 natural numbers is \n");
for (i=1; i<=10; i++)
{
sum=sum+i);
printf ("%d", i);
}
printf ("\n the num is %d \n", sum);
```

Date: / / Page no:

While loop :- The is while loop is C is to be used in the Senario where we dont know the no of illetraction in advance the block of Statement is executed in the while loop until conditions specified in which while loop is satisfied It is also called are tested loop.

Syntax :-

Initialization of loop variable

While (< condition)
{
=====
Inc\dec
}

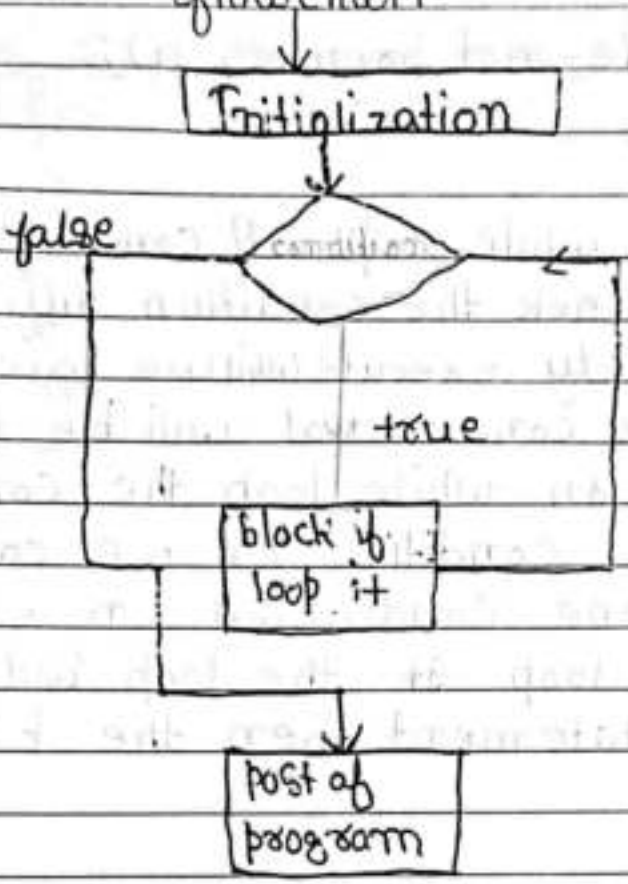
Program :-

```
Void main()
{
int n, i;
clrscr();
printf("enter number whose table is tobeprint")
Scanf("%d", &n);
i = 1;
do
{
printf("%d %d %d = %d \n", n, i, n*i)
i++
}
While (i<= 10);
getch();
}
```

Jay prakash Upadhyay

Date: / / Page no:

flowchart




```
# Void main()
{
int n, i;
clrscr();
printf("enter a number");
Scanf("%d", &n);
i=2;
While (i<n)
{
if (n%. i==0)
{
printf("%d is not prime", n);
break;
3
i++
3
```

```
if (i==n)
printf ("%d is not prime", n);
getch();
```

properties of while loop :- A conditional expressing is used to chek the condition diffine while loop resepteadely execute whitwe given condition is false. The conditional will be true if it returns 0. In while loop the conditional Sepresion is condition in we can have more than one conditional in expresion in the while loop If the loop body contains only are statement then the brets are optinal.

Q Write a program to calculate the factorial of any number using while loop.

Solve:-

```
# include<stdio.h>
# include<conio.h>
 void main()
{
int n, i, fact = 1;
clrscr();
printf("enter a number");
Scanf("%d", &n);
i=1
While(i<=n)
{
fact = fact x i;
 i++;
```

```
3
printf ("factorial of %d", n, fact);
getch ();
 3
```

(Q) Write of a program to calculate sum of all the digits in a where no is given by the user.

Ans⇒

```
#include<stdio.h>
#include<conio.h>
 void main ()
{
int n, r, sum = 0;
clrscr ();
printf ("enter a numbers");
scanf ("%d, &n);
while (n>0)
{
r = n % 10;
sum += r;
n=n/10;
3
printf ("Sum of all digit is %d", sum);
 getch ();
 3
```

Q. Iatriate a program to print wheater a number given by users Armstrong or not.

Ans⇒
```
#include <stdio.h>
#include <conio.h>
 void main()
{
int n, r, sum = 0, x;
clrscr();
printf("Enter a numbers");
Scanf("%d", &n);
 x = n;
while (n>0)
{
r = n % 10;
sum += (r*r*r);
n = n/10;
3
it (sum == x)
printf("I to is armstrong");
else
printf("No is not armstrong");
 getch();
 3
```

@ W.A.P to print fibonacci series (0, 1, 2, 3, 5...
W.A.P to print reverse of any number.

Ans ⇒
```
#include<stdio.h>
#include<conio.h>
 void main()
 {
int a, b, c, n, i= 1;
clrscr();
printf("take two intral elements of series");
Scanf("%d%d", &a, &b);
printf("How many elements do you want in series");
Scanf("%d", &n);
printf("%d\t%d\t", a, b);
while (i<=(n-2))
{
c= a+b;
printf("%d\t", c);
a=b;
b=c;
i++;
}
getch();
}
```

@ Latrate a program to print wheater a number given by users Armstrong or not.

Ans⇒

```
#include <stdio.h>
#include <conio.h>
 void main ()
 {
int n, r, sum = 0, x;
clrscr ();
printf ("Enter a numbers");
Scanf ("%d", &n);
 x = n;
while (n>0)
{
r = n % 10;
sum += (r*r*r);
n = n/10;
3
it (sum == x)
printf ("I to is armstrong");
else
printf ("No is not armstrong");
 getch ();
  3
```

Date: / / Page no:

@ W.A.P to print fibonacci series (0, 1, 2, 3, 5...
W.A.P to print reverse of any number.

Ans⇒
```
#include<stdio.h>
#include<conio.h>
 void main()
 {
int a, b, c, n, i = 1;
clrscr();
printf("take two intral elements of series");
scanf("%d%d", &a, &b);
printf("How many elements do you want in series");
scanf("%d", &n);
printf("%d\t%d\t", a, b);
while (i<=(n-2))
{
c = a+b;
printf("%d\t", c);
a = b;
b = c;
i++;
}
getch();
 }
```

for loop :- it executes the set of statements Until the time a paticular condition is accomplished. It is know as the open-ended loop.

In for loop we can have more than one initilization or incr/decr seperated by using comma operators and one condition as well.

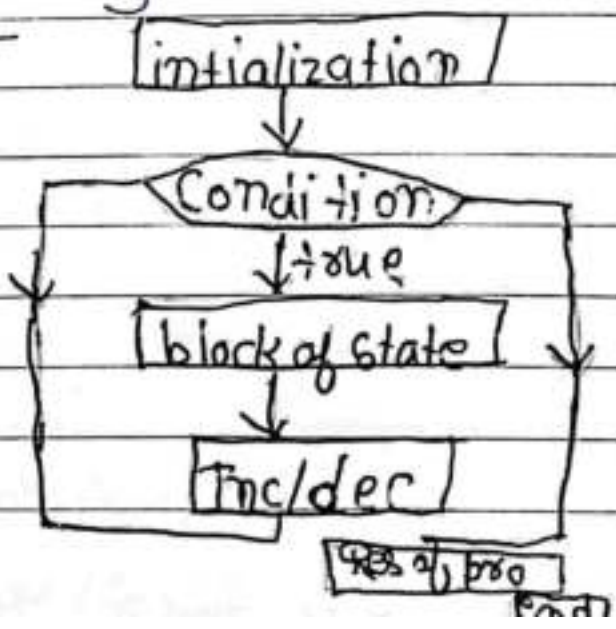
for loop is used to emalute the initilization part first then its check to the condition for true or false. If the condition is true then it excutes the set of statements of for loop. after that it evalute the inc/dec condition and again check the condition and repat the block of statements until the statements is false.

Syntax :- for (intialization, condition, inc/dec

{

}

flowchart :-



Program :-

```
Void main()
{
int i, j;
clrscr();
for(i=0, j=10; i<=j; i++, j--)
{
printf("%d\t%d/n", i, j);
}
getch();
}
```

Nested for loop :- There are nested for loop as well is which there are the outer for loop and inner for loop. In nested loop the inner loop is repeated for a given condition of outer loop.

Syntax :-

```
for(initialization; condition; inc/dec)
{
    for(initialization; condition; inc/dec)
    {
    ==========
    ==========
    ==========
    }
}
```

Program :-

```
Void main()
{
int i, j;
clrscr();
for (i=1; i<=5; i++)
{
for (j=i; j<=i; j++)
{
printf ("%d\t", j);
}
printf ("\n");
}
getch();
}
```

go to statement :- The gate statement is the jump statement which is same time also refund to as Unconditional jump statement It is used to jump from any where to anywhere with an function.

go to Statement

Syntax :-

```
(label)
go to level;
   ===
   ===
   ===
level
```

```
(label)
level
   ===
   ===
   ===
go to level
```

Date: / / Page no:

Disadvantages of using goto statement

(i) The use of goto statement is highly discouraged as it makes the program logic very complen.

(ii) Use of go to make the task of analysing and ienifing the correction of problem very difficult.

(iii) Use of goto can be simply avoided using either break or condinue statement

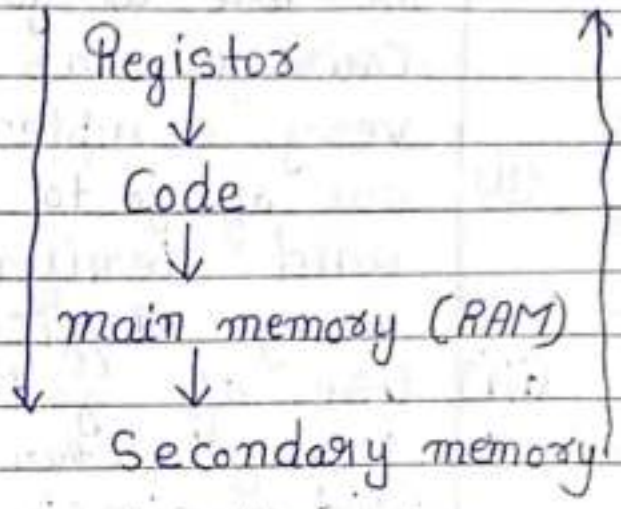
```
void main()
{
int num = 26;
if (num % 2 == 0)
  go to even
else
go to odd;
even: printf ("%d is even", num);
return;
odd: printf ("%d is odd", num);
getch();
}
```

```
void main()
{
Int n=1;
label=
printf("%d", n);
n++;
if (n==10)
goto (label);
getch();
}
```

Storage classes

- Storge classies

automatic	auto
static	int a
extern	i
rigester	



Storage classes in c are used to determine the life time visible, memory location end initial value of variable.

There are four type classes in C :-
(i) Automatic (ii) External (iii) Static
(iv) Resister

Automatic :- • Automatic variables are alocating the memory automatic automically at run-time.

The visibilty of automatic variables is limited to the block in which they are difined.

The Scopes of a automatic variable is limited to block in which they difined.

The automatic variable are intilize to garbage value by default.

The memory assigned to automatic variable gets free upon existing from the block.

- The keyboard used for defining automatic variable is auto.
- Every local variable is automatic in c by default.

Program :-

```
Void main ()
{
int a;
float b;
char c;
printf ("%d\t%f\t%c", a, b, c);
getch();
}
```

Static :-

- The variables defined as static specifier can hold their value between the muttede function call.
- Static local variables are visible only to the function or the block in which they are defined.
- A same static variable can be declare at many tim but can be assigned at any time.
- Default initial value of the static interger variable is 0 therirle null.
- The visible of static variable is limted to the file in which it declared.
- The key ward used to defiented is Static.

Program ÷ 1

```
Static char c;
Static int i;
Static float f;
Static char a[10];
  void main ()
  {
printf ("%c\t%d\t%f\t%s", c, i, f, a);
getch ();
  }
```

program ÷ 2

```
Void Sum()
{
Static a = 10;
Static b = 24;
printf ("%d\t%d", a, b);
a++;
b++;
}
void main()
{
int i;
for (i=1; i<=3; i++)
{
Sum();
}
}
```

Date: / / Page no:

13. Regestor :- (i) The variabes define as the regestor is alagated the memory in to the CPU resistor depending on the size of the memory remaining in the CPU.

(ii) We cannot deference the registor variable that is we cannot use ampercent operator (&) for the regesistor variable.

(iii) The excess time of risistor variable is faster the the autometic variable.

(iv) The initial default value of the resistor local variable is zero.

(5) The risiter keyword used for the variable which should be stored in the CPU register However, it is compailers choice wheather or not the variable can be Stored in the resistor.

(6) We can Store pointer in to the riegister that is a regisster can Store the address of a variable.

Program :-

```
Void main ()
{
  register int a;
  printf ("%d", a);
}
```

Date: / / Page no:

External :-

① The external storage class is used to tell the compiler that variable defined as extern is declared with n external linkage else where in the programe.

② The variables declared as extern are not alogating any memory. It is only declaration and intended to specify that the variable is declared some where in the program.

③ The defualt intial value of enternal integer type is zero and null other wise

④ We can only intialise the externi variable globaly that is we cannot intialize the external variable withs only block or method.

⑤ An external variable declared many times but can be initalized at only one.

⑥ If a variable is declared as external then the Compailer Searches for that variable to be initalised some where in the program. if it is not found then the compiler will show an error.

Date: / / Page no:

Program :-

```
int main()
{
extern int a;
printf ("%d, a);
return 0;
}
```

Program :-

```
int a;
int main()
{
extern int a;
printf ("%d", a);
return 0;
}
int main()
{
extern int a;
printf ("%d", a);
}
int a = 20;
```

Date: / / Page no:

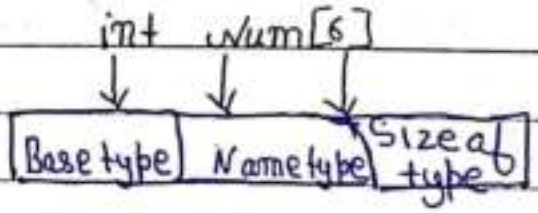
Array :- A array in any programing languges is a collection of similer items stored at contigious memory locatione it elements can be excessed randomly using Indices of an array. The can be used to store collection of parimitive data types such as int, float char, double etc. of any particular type. An array can store derived data types such as structures, pointers etc.

Why do we need arrays :-

We can use normal variables ($V_1, V_2 V_3 \dots V_R$) when can we have a small no of objects, but if we want to store a large no of instantes it becomes difficult to manage then with normal variables. The ideal of an array is to represent many instenes in one variable.

Array declaration :-

```
int a [3];
int a [3] = {1,2,33};
int a [3] = {1,1,13};
int a [3] = {3};
int a [3] = {0};
int a [3] = {1};
int a [3] = {0.13 = 33};
int a [] = {[0 --- 1]3};
```



Date: / / Page no:

Advantage of an array :-

(1) Random access of elements using array index access

(2) Use of few line of code as it weaters a single array of multiple elements.

(3) Easy access to all the elements.

(4) Travessal through the array becomes eay using a single loop.

(5) Sorting becomes easy as it can be accoplitsed by writing few line of code.

Disadvantage of an array :-

(1) it allows a fix no of elements to be enterd which is decided at the time of declaration Unlike a linked list, an array is not dynams.

(2) Insertion and delition of elements can be costly seince the elements are needed to managed in accordance with new memor alogation.

Accessing the array elements :-

```
Void main()
{
int a[5], i;
clrscr();
printf("enter elements in array\n");
for(i=0; i<5; i++)
scanf("%d", &a[i]);
}
printf("Array elements are\n");
for(i=0; i<5; i++)
printf("%d\t", a[i]);
getch();
}
```

Q) Write a program to reach an element in single dimensional array.

```
Void main
{
int a[10] se, i;
clrscr();
printf("enter elements in array");
for(i=0; i<10; i++)
{
scanf("%d", &a[i]);
}
printf("enter searching element");
Scanf("%d", &se);
for(i=0; i<10; i++)
{
if(se == a[i])
```

```
{
printf("element found at %d index", i);
                af position; i++);
break
if(i==10)
  3
printf("elements doesnot exist in array");
getch();
  3
```

Multdimesinal arrays in c :-

We can define multdimesinal array in Symbol word as and data in multi-dimesional arrays are storage in tabular from row for mager (Coloum mage) The genral from dimesinal in dimensional array in

data type array name [Size1] [Size2]..[Sizen];

```
int a[5];
int a[4][5] ⟶
      ↓   ↓
    (Row) Coloum
```

	0	1	2	3	4	
1						
2						
3						

int a[3][4][3];

Size of array :- The total number elements that can be Stored in a multdimensinal arrays can be calcuated by multiplying

Date: / / Page no:

- Two dimensinal array the simplest form of a multidimensinal array and case see 2-D array is an array of array.

- Elements in 2D array are company refered a[i][j] Whare i is the row numbers & j is colum numbers.

- Initializing 2-D arrays :-

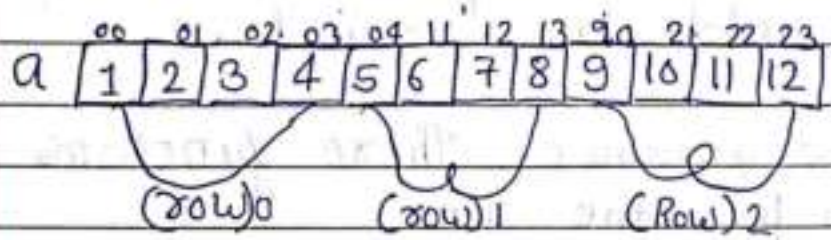
```
int a[3][2] = {1,2}, {3,4}, {5,6};
int a[3][2] = {1, 2, 3, 4, 5, 6};
```

WAP to acces 2D array in row major.

```
void main()
{
int a[3][4], i, j;
clrscr();
printf("enter elements in array\n");
for(i=0; i<3; i++)
for(j=0; j<4; j++)
{
scanf("%d", &a[i][j]);
}
}
printf("Entered elements in row major\n");
for(i=0; i<3; i++)
{
for(j=0; j<4; j++)
```

Date: / / Page no:

```
printf ("%d \t", a[i][j]);
}
printf ("\n");
}
getch();
}
```



i [0] 1, 2, 3

j [0] 1, 2, 3, 4, 0, 1, 2, 3, 4, 0 ... 4

1	2	3	4
5	6	7	8
9	10	4	12

function :- A function is a block of code that performs a specific task, suppose we need to create a program of a circle creation and colour is you need to create two function to solve.

(i) Create a circle function
(ii) Create a colour function

Types of function :- There are two types of function.

(i) Standard library functions
(ii) User-defind functions

Standard libary functions :- These are functions in c programing which are defined in libary functions.

ex :- printf in needs file stidio.h and getch in conio.h.

(@) User-difined function :- There functions are crated as per the need.

```
# Include < Stdio.h>
 int Sum (int, int);
 ==
 Void main()
 {
 ===
 C = Sum (a, b);
 ==
 3
int Sum (intx, inty)
{
int z;
z = x+y;
return z;
3
```

Program :-

```
#include <stdio.h>
#include <conio.h>
int Sum (int, int);
void main()
{
int a, b, c;
clrscr ();
printf ("enter two numbers \n");
Scanf ("%d%d", &a, &b);
c = Sum (a, b);
printf ("Sum of %d and %d is d", a, b, c);
getch();
}
int Sum (int x, int y)
{
int z;
z = x+y;
return z;
}
```

(Q) WAP to sweep value

```
void swap int, int);
void main ()
{
int a, b;
clrscr ();
printf ("enter two different values \n");
Scanf ("%d%d", &a, &b);
Swap (a, b);
getch ();
}
```

```
void Swap (int x; int y)
{
int z;
z=x;
x = y;
y = z;
printf ("the Swapped value are\n");
printf ("%d\t%d", x, y);
}
```

These are different approaches you can tak Save the same problem using functions

Ans:- There are four program that check whether the integer entered by the user is a prime number or not. The output of all the programs is the same is each Example but the approaches.

1st approach (take nothing & return nothing)

```
Void prime ();
void main();
{
clrscr ();
prime();
getch ();
}
Void prime ()
{
int n, i;
printf ("enter any no\n");
```

```
Scanf("%d", &n);
i= 2;
while (i<n)
{
if (n%.i==0)
{
printf ("%d is not prime");
break;
}
i++;
}
if (i == n)
printf ("%d is prime\n");
}
```

Take nothing and return Something (primeno

```
Void prime();
void main()
{
int f;
f = prime();
if (f==1)
printf ("Numbers is prime");
else
printf ("Numbers is not prime");
getch();
}
int prime()
{
int n, i;
printf ("enter any numbers");
```

```
Scanf ("%d", &n);
i = 2;
while (i<n)
{
if (n % i == 0)
{
return 0;
}
i++;
}
if (i == n)
return 1;
}
```

3rd approach take something and return.

```
Void prime (int x)
{
int i = 2;
while (i<x)
{
if (x % i == 0)
{
printf ("%d is not prime \n", x);
break;
}
i++;
}
if (i == x)
printf ("%d is a prime", x);
}
```

```
Void main()
{
int n;
clrscr();
printf("enter any number\n");
Scanf("%d", &n);
prime(n);
getch();
}
```

Take Something, return Something

```
int prime(int x)
{
int i;
i=2;
for(i;i<x;i++)
{
if(x%i==0)
return 0;
}
if(i==x)
return 1;
}
void main()
{
int n,f;
clrscr();
printf("enter any numbers\n");
Scanf("%d",&n);
f=prime(n);
if(f==1)
```

Date: / / Page no:

```
printf("%d is a prime numbers", n);
else
printf("%d is prime number", n);
getch();
}
```

Which approach is better ÷ pasing arguments & returning the value of function is better because function should perform a Speific

The parameter and the parameter recieved in a definition of funcation are called indevidual

There are two most poplar ways of pesing parameter are as follow.

Pass by value or call by value ÷ In this parameter pasing method value of actual paramater are copied to functions formaks paramater and the two types of paramater are stored in different memory location so any changes make insert the function are not reflected in the actual parameter of main program.

(11) Call by refrence or pass by refernce: In this parameter passing method, both actual and formal parameters referced to the same location, So any changes make inside the functions are actually reflected in actual parameters of the main program.

Program :-

```
void check (int *x)
{
 *x = 40;
}
void main ()
{
int n;
n = 20;
printf ("Refere Applying function n is %d \n", n);
  int x = 20
  int *x = &n;
```

W.A.P to Sweeping of two numbers

Program :-

```
void Swap (int *x, int *y)
{
int z;
z = *x;
*x = *y;
*y = y;
}
void main()
{
int a, b;
Clrscr();
printf ("enter two values");
Scanf ("%d%d", &a, &b);
printf ("values before calling functions are\n");
printf ("%d\t%d", a, b);
Swap (&a, &b);
printf ("Values after calling function are\n");
```

```
printf("%d\t\t%d", a, b);
getch();
}
```

following are some points about functions in C.

(I) Every C program has a function called main() that is called by operating System when a user runs program.

(II) Every function have a return type, if a function does not return a value, then is used as a write type.

(III) In C functions can return any type except arrays and functions, We can get around the limitation by returning pointer to array or pointer to function.

(IV) Im Empty parameter less means that the parameter. called with declear a function can be to declear a function that can by we called without any parameter We should use void() fun(void)

(V) In a C program, if a function is called before its declertion then the C compiler automatically assume the declearation of that function as void assume.

Date: / / Page no:

Recursion :- The process in which a function calls it self directly or indirectly is called recursion and the corresponding functions is called as recursive function. Using recursive algorithm certain problems can be solve quite easily. Certain for :

Example :- Fibonaci Series, Quicksort, merge sort etc.

• Recursion :-

```
int factorial (int x)
{
if (x = 0 || x == 1)
return 1;
else
return x * factorial (x-1);
}
Void main ()
{
int n, fact;
clrscr ();
printf ("enter numbers");
Scanf ("%d", &n);
fact = factorial (n);
printf ("the factorial of %d is %d", n fact);
getch ();
}
```

Date: / / Page no:

Difference b/w direct and indirect recursion

A function is called direct recursive if it call it self and a function is called indirectes recursive if it call at another fun and that another then calls the present fun

Example :- Direct recursion

```
Void fun ()
{
fun ();
}
```

indirect recursion

```
Void fun 1 ()
{
fun 2 ();
}
Void fun 2 ()
{
fun 1 ();
}
```

Difference b/w

A recursive fun is taild recursive when recursive call is the last thin exiquet by fun a recursive fun. taild non recursive when recursive call is exiquet in the insiddle by the function.

Program :-

```
void printfun (int test)
{
if (test<1)
return;
else
{
printf ("%d\t", test);
printfun (test-1);
print ("%d\t" test);
}
}
void main ()
{
int test = 3;
printf fun(test);
getch();
}
```

Structure :-

Structure :- A Structure is a user a defined data type in c. Structure cerates a data type that can be use to group iteam of possild possibilly different type Struct keyward is used to woard structure.

```
Struct Student
{
char Sname [25];
int S rollno;
Char S branch [5];
float S marks;
};
Struct-student - S1, S2
};
```

How to declare Structure variable?

Ans⇒ A Structure variable can either be declered by structure - declaration like basic type.

```
Struct Student
{
char Sname [25];
int S rollno;
Char S branch [5];
float S marks;
};
Struct Student
};
```

Date: / / Page no:

Structure member can be initalize by declaction

```
Struct point
{
int x=0;
int y=0;
};
```

// Compilation Cerness

The structure is single whice data type is declaction no memory is allocated when variable are poinded structure member can be intialize by 3.

Program :-

```
Struct point
{
int x;
int y;
};
void main()
{
Struct point (P);
P= {1, 2};
printf ("%d \t %d", P.x, P.y);
getch();
}
```

Date: / / Page no:

Array of structure :-

Program

```
Structure Student
{
int &roll;
float &marks;
};
Void main()
{
Struct Student s[10];
int i;
clrscr();
printf("enter value in structure\n");
for(i=0; i++)
{
Scanf("%d %f", &s[i].&roll &s[i].&marks);
};
printf(" entered in to is\n");
for(i=0; i<10; i++)
{
printf("%d \t %f \n", &[i] s.roll, &[i] &marks);
};
getch();
};
```

Date: / / Page no:

Pointer structure :- like primitive data tips, we can have pointer to a structure. If we have a pointer to structure members are accessesed Using arrow operaters.

Program :-

```
Struct point
{
int x, y;
};
Void main()
{
Struct point P1 = {1,2};
Struct point *P2 = &p;
printf("%d\t%d\n", P2 -> y);
getch();
}
```

limition of structure :- In structure, structure provider.

A structure is helpfull tall to handle a group logically related data items but it have some limition.

(I) Structure don't allow the struct deta type to be treated by build data type.

(II) like can not used operaters like plas mixmus, dixelty on the structure Variable but can be Used adifferent ways.

Date: / / Page no:

```
Struct marks
{
int Sub1;
int Sub 2;
int Sub 3;
int total;
3;
main ()
{
int i;
Struct marks Student[3] = {{4
                           {
                           {

Struct marks total
for (i=0; i<=2; i++)
Student[i] total = Student[i] Sub1
                   Student[i] Sub 2
                   Student[i] Sub 3;
total Sub1= total Sub1 + Student[i] Sub 1;
total Sub 2 = total Sub2 + Student [i] Sub2;
total Sub 3 = total Sub 3 + Student [i] Sub 3;
total, total = total+ Student [i]  total;
3
point (" Student+ total\");
for (i = 0; i<=2; i++)
printf (" Student [%.d]%.d/n ", i+1 Student[i];
printf (" \n Student-%.d ", i+1 );  total);
getch ();
3
```

Date: / / Page no:

Array program :-

```
#include<iostream>
Using namespace Std;
Void main()
{
int array[5];
array[0] = 10;
array[1] = 20;
array[2] = 30;
array[3] = 40;
array[4] = 20;
Cout<<"Array Contents;"<<endl
for(int n=0; n<5; n++)
{
Cout<<"array["<<n<<"] = "<<array[n]<<endl;
}
}
```

Array program :-

```
#include<iostream.h>
#include<conio.h>
void main()
{
int i, a[10];
Cout<<"Enter the elements in a array\n";
for(i=0; i<10; i++)
Cin>>a[i];
Clrscr();
for(i=0; i<10; i++)
Cout<<"a["<<i<<"]"<<a[i]<<endl;
getch();
}
```

Array program :- 2

```
#include<iostream.h>
#include<conio.h>
 void main()
{
int i, a[10], b[10], c[10];
clrscr();
cout<<"enter the elements in a array A\n";
for(i=0; i<10; i++)
 cin>>a[i];
cout<<"enter the elements in a array B\n";
for(i=0; i<10; i++)
 cin>>b[i];
for(i=0; i<10; i++) [i]=a[i]+b[i];
clrscr();
cout<<" Sum two arrays\n";
for(i=0; i<10; i++)
cout<<" c["<<i<<"]="<<c[i]<<endl;
getch();
}
```

Example :-

```
Struct node
{
 int data1;
 char data 2;
Struct node *link;
};
```

In above example link in a pointer to structure of type node. Hence the structure node is a self-referencial structure with link as the referencing pointer.

An impotant point to consider is that the pointer should be initilazed properly before accessing by default it contags garbage value.

Types of self-referential structure

① Self-referential structure with single link
② Self-referential structure with multiple link

Self-referential structure with single link

- these structure can have only one self-pointers as their members.
- The following example with shows -Show the following example to connect the object of a self-referential.

Self-referential structre a with multiple link can have move than one self-pointers. many completed data structre can be easily construsted using thise structre. Such structres can easily connect to move than one nodes.

- Program :-

```
#include<stdio.h>
#include<conio.h>
struct node
{
int data;
struct node * prev. link;
struct node * next. link;
```

```
};
Void main()
{
Struct node n1, n2, n3;
n1. prev_link = Null;
n1. Next_link = Null;
n1. data 100;
n2. prev_link = Null;
n2. next_link = Null;
n2. data = 200;
n3. prev_link = Null;
n3. next_link = Null;
n3. data = 300;
n1. next_link = &n2;
n2. next_link = &n3;
n2. prev_link = &n1;
n3. prev_link = &n2;
printf("%d\t", n1, data);
printf("%d\t", n1, next_link -> data);
printf("%d\t", n1, next_link -> next_link -> data);
printf("%d\t", n2, prev_link -> data);
printf("%d\t", n2. data);
printf("%d\t", n2. next. link_data);
printf("%d\t", n3 prev_link -> prev_link -> data);
printf("%d\t", n3 prev_link -> data);
printf("%d\t", n3. data);
getch();
}
```

Date: / / Page no:

Union :- Like structures, Union is a where all the members shares same memory location in the following program both x & y share the same location. And if we change x, we can see the changes reflected in y also.

Program :-

```
#include <stdio.h>
 Union test
 {
int x;
float y;
};
void main()
{
Union test t;
t.x = 2;
printf("After making x = 2\n values are
                x = %d and y = %f\n", t.x, t.y);
t.y = 10.2;
printf("After making y = 10.2\n Values
 are x = %d and y = %f\n", t.x, t.y);
 getch();
}
```

Union Application :-

Union can be useful in many Suitation where we want to use the same memory for two or more members, for example Suppose we want to implement a binary tree data structure, where each leaf node a double data values by each in terma node has pointer two children but no data we decleare it as.

```
Struct node                    Union node
{                              {
Struct node * left;            Union node * left
Struct node * right;           Union node * right
double data;                   double data
};                             };
```

Typedef :- the c programing languege provids a keyword called typedef. Which you can use to give a type of a new name. following is an example to define a term for same data type.

```
typedef Unsigned char B;
```

After the definition, the indentifer B can be used as an for three type Usigned char, for example :-

```
B b1, B2;
```

Where b_1 is b_2 are variable of type Unsiged char.

By convention Upper case letter use for B definitions to remind the User that the type main the Symbolls

but you can Use lower case as follows typedef Unsinged Charb;

you can Use typedef to give a name to User define data type as well. for example you can used typedef with Structure to define a new data type and then Use that datatype to define Structure data type divectly as follows

```
#include < Stdio.h >
#include < conio.h >
typedef Struct Books
{
Char tittle [50];
char author [50];
Char subject [100];
int book - id;
} Book;
Void main()
{
Book Book;
Strcpy (book tittle, " C programming");
Strcpy (book author Y. kanittkar);
```

Date: / / Page no:

```
Strcpy (book.Subject"; C programming");
book.book id = 6452;
printf ("Book tittle: %s\n", book.tittle);
printf ("Book author: %s\n", book.author);
printf ("Book Subject: %s\n", book.Subject);
printf ("Book id: %d\n", book.book.id);
}
```

difine is a c directive with is used to difine alias.

```
#include <stidio.h>
# define HYD "Hyderabad"
 void main()
{
printf ("%s" HYD);
}
```

Difference between typedef & # define

(I) Typedef is limited to geving Symbolic name two types only wheres #defin can be Use to define an alias for Values as well for example:
you can define 3.14 as pi.

(II) Typedef interpretation is performed by the compiler where are # difine Statement are performed by preprocessor directive.

Date: / / Page no:

(iii) define should not be terminated with semicolon where typedef has to be terminated

(iv) # define will just copy the defination value at the point of use while typedef is the actual definition of new type.

(v) typedef follows the scope is which if a new types is defin in a scpae then.

(vi) The new type name will only be visible till the scope is there but in case of # define when prepossor # define it replaces all the occurences and there is no scope rule is followed.

End