#preprocessor

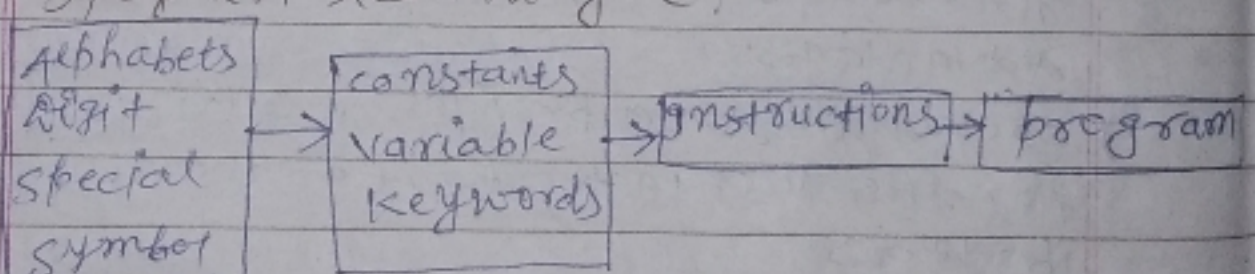
'C' Language

Date: / /
Page:

C is a procedural programming language. It was initially developed by dennis ritchie at bell laboratories in 1972. It was mainly developed as a system programming to write an operating system. The main features of the C language include low-level memory access, a simple set of keywords and a clean style, these features make C language suitable for system programmings like an operating system or compiler development.

→ C is a middle level language. It reduces the gap b/w high level language and low level language that is why it is known as middle level language.

Step in learning C:



Alphabets: A, B, ---- V, Z
a, b, ---- y, z

Digit: 0, 1, 2, ---- 9

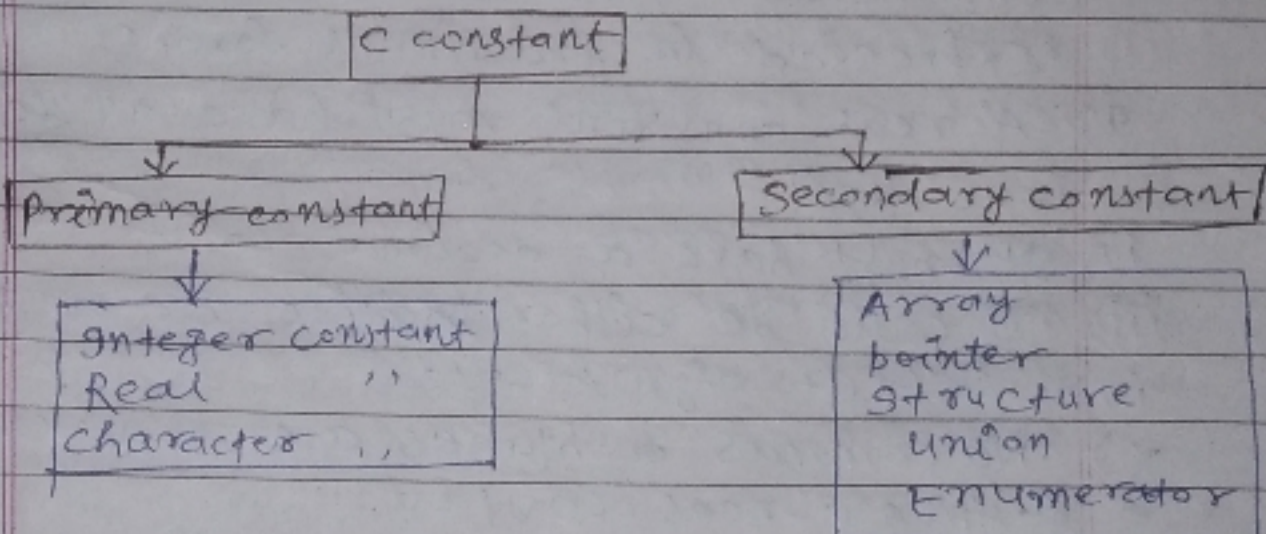
Special symbol: ~, ! @, #, %, ^, &, *, (), -, +, =/\
{} [] : ; " <> , ?/

Constant:

A constant is a number, character or character string that can be used as a value in a program. We constants to represent floating-point, integer, enumeration, or character values that cannot be modified.

→ constant are divided into two major categories:

i) Primary constants
ii) Secondary constants



Integer constant

→ These are sequence of numbers from 0 to 9 without decimal point or fractional part.
→ It require minimum two byte & max 4 byte
→ Integer constants could either be positive or negative or may be zero.
→ No commas or blanks are allowed within an integer constant.

→ Range of integer constant depends upon the compiler. For 16 bit compiler range will be -32768 to 32767.

ex:- 10, 20, 426, +782, -8000 etc

Real constant:

→ Real constants are often known as floating point constant.

→ The real constants could be written in two forms fractional form and exponential form.

Expressed in fractional form:

i) A real constant must have at least one digit.

ii) It must have a decimal point

iii) It could be either positive or negative

iv) Default sign is +ve

v) No commas or blanks are allowed within a real constant.

ex:- 2.5, 5.215, -48.579, 2.0

Expressed in exponential form

→ It is usually used if the value of constant is either too small or too large.

→ It is represented in two parts. The part appearing before 'e' is called

mantissa, whereas the part following e is called exponent.

→ Following rules must be observed while constructing real constants. e.
Expressed in exponential form:-

i) The mantissa part and the exponential part should be seperated by a letter e or E.
ii) The mantissa part may have a +ve or -ve sign.
iii) Default sign of mantissa part is +ve.
iv) Range -3.4e38 to 3.4e38

0.00342 → $3.42e^{-4}$
2456.123 → $2.4561e^{+3}$

Character constant

→ A character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas.
→ Both the inverted commas show point to the left.
→ Maximum length of a character constant can be 1 character.

Ex: 'A'
'I'
'5'
'='

but not 'A', '-3', '3.4'
'Rishabh'

Variable:

In programming language a program store data for perform some operation on that data, the memory location on which the data is stored is known as variable.

rule for constructing variable:

i) variable name is any combination of 1 to 31 alphabets, digit or underscores. some compiler support variable name length up to 247 character.

ii) first character in the variable name must be an alphabets or underscore.

iii) no commas or blanks are allowed within a variable name.

iv) no special symbol other than an underscore can be used in a variable name. v) no two successive underscores are allowed

Keywords:

→ Keyword are word whose meaning has already been explained to the C compiler.

→ The keyword cannot used as variable name

→ Keywords are also called Reserved word

→ There are only 32 keyword available in C.

Ex:

auto	default	float	register	typedef
break	do	for	return	union
case	double	goto	short	unsigned
char	else	if	signed	void
const	enum	int	size of	volatile
continue	extern	long	static	while
			struct	packed
			switch	

Structure of c program:

→ every c program contain a no. of several function

→ Each fuc^n of it performs task independently

Documentation section: set of comment

Include header file section
Global declaration section

main( )
{
Declaration part
Executable part
}
user defined fuc^n
{
}

Some important point related to c:

→ c program comprise of series of statement

→ All statement are entered in small case letter

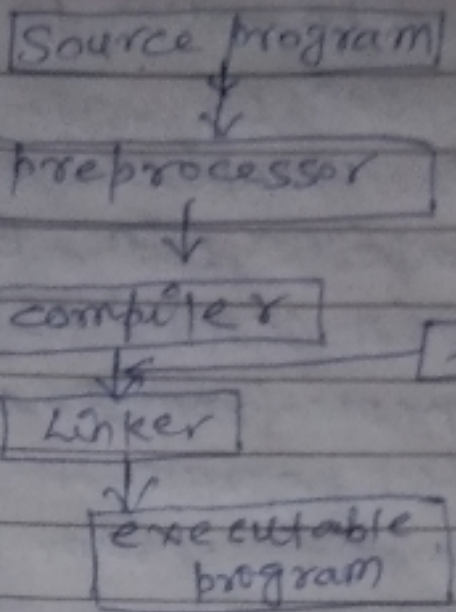
→ every c statement end with (;)

→ blank space inserted b/w two word to improve the readability of the statement.
→ It is not necessary to fix the position of statement in program.
→ Users can also write one or more statements in one line seperating them with a semicolumn (;), Hence it is often called from language.
→ opening & closing braces should be blanced
→ comment about program should be enclosed within /* */
→ Any no. of comments can be written at any place in the program
→ comment can be split over more than one line
→ any Variable used in the program must be declared before using it.

Scanf()

Scanf() is a predefined fucn
→ Scanf() take the value from keyboard
→ return 0; means → program has successfully executed & it return integer.
→ void main(); means → it returns null value.

compilation & Execution:



→ To type c program need editor.
→ once the program has been typed it need to b converted to machine language before the machine can execute it. this conversesion done by compiler.
→ C compiler are available with and without editor.
→ The enviroment where you find the compil editor, debugging tool, linking. tracing & testing tool is called integrated develop enviroment (IDE)
There are several such IDE available i marked tergeted towards different operat system as follow:

	operating system	compiler
1.	MDOS →	Turbo C, Turbo c++, Microsoft c
2.	window →	visual c++, Berland c++
3.	Linux →	gcc

Steps need to follow to compile and execute 1st c program:

i) → Start the compiler at C > prompt
The compeler (TC: EXE is usually present in C:\TC\BIN directory)

ii) Select new from the file menu

iii) Type the program

iv) Same the program using F_2 under a proper name (ex: program1.C)

v) use ctrl+ F9 to compile & execute the program

vi) use alt + F5 to view the operation.

Problem solving by the computer:

Problem solving by the computer involves following step:

i) Problem defination
ii) Analysis
iii) Algorithm
iv) Flowchart
v) pseudo code
vi) coding
vii) Running the program
viii) Debugging
ix) Testing
x) Documentation

Problem defination: Problem solver should understand the problem throughly in terms of requirements.

Problem analysis: problem analysis is the process of defining a problem and decomposing

overall system into smaller parts to identify possible inputs, processes and output associated with the problem.

The following task is performed:

i) specifying the objective:- First, we need to known what problem is actually being solved. Making a clear statement of the problem depends upon the size & complexity of the problem.

ii) specifying the output: Before identifying inputs required for the system, we need to identify what comes out of the system. The best way to specify output is to prepare some output forms and required format for displaying result.

iii) specifying the input:- After having specified the outputs, the input data required for the system need to be specified as well. One needs to identify the list of inputs required and the source of data.

iv) specifying processing requirements: when output and input are specified, we need to specify process that converts specified inputs into desired output. If the proposed program is to replace or supplement an existing one, a careful evaluation of the present processing procedures needs to be made, nothing

any improvements that could made.

Algorithm: An algorithm is an effect step-by-step procedure for solving a problem in a finite number of steps. A algorithm itself is diveision of a problem into small steps which are ordered in sequence and easily understandable.

an algorithm must posses following characteristics:

i) Finiteness: an algorithm should have fini number of steps and it should end aft a finite time

ii) Input: an algorithm maye have many inputs or no inputs at all.

iii) output: It should result at least one output

iv) Definiteness: Each step must be clear, well-defined and precise. There should be no any ambiguity.

v) Effectiveness: Each step must be simple & should take finite amount of time.

Algorithm notation:

1) name of algorithm
2) step number

3) Explanatory comment
4) Termination

Q) 1) Algorithm to calculate simple interest
Step 1: Start
Step 2: Read principle(P), time(T) & rate(R)
Step 3: Calculate S.I = $P*T*R/100$
Step 4: print S.I as simple interest
Step 5: Stop

2) Algorithm to compute the area of circle.
Step 1: Start
Step 2: Read radius(r)
Step 3: Calculate area = 3.14*r*r
Step 4: print Area of a circle
Step 5: Stop

Advantages of an algorithm:

1. Effective communication: since algorithm is written in English like language, it is simple to understand step-by step solution of the problems.
2. Easy Debugging: well-designed algorithm makes debugging easy so that we can identify logical error in the program.
3. Easy and efficient coding: An algorithm acts as a blueprint of a program and helps during program development.

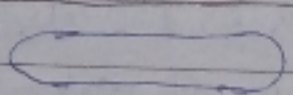
4) independent of programming language: ~~Programming~~ An algorithm is indep of programming languages and can be easily coded using any high level language.

Disadvantages of algorithm:

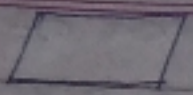
1) Developing algorithm for complex proble would be time consuming and difficu to understand.
2) understanding complex logic through algorithms can be very difficult.

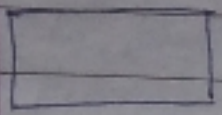
Flowchart: Flowchart is basically a pictorial or diagrammatic representat of an algorithm using standard symbol They are classified into two parts:
i) program flowchart & ii) system flowchart

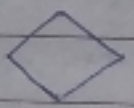
Program flowchart:

Flowchart symbol	Symbol name	Description
	Terminal (start or stop	Terminals are used to represent start & stop of the flow chart
	Flowline or arrow	Flow lines are used to connect symbols used in flowchart & indicate direction of flow

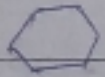
Date: __/__/__
Page: _____

 input/output parallelogram are used to read input data and output or display information

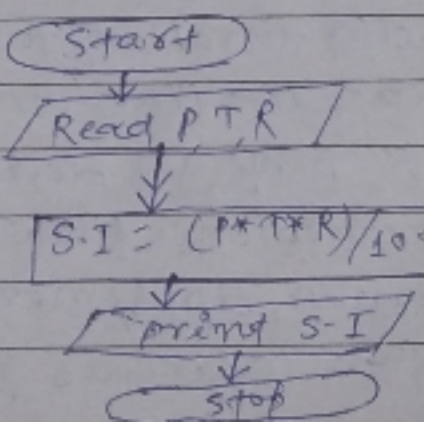
 process rectangels are generally used to represent process. for ex:- Arithmetic operations, ~~data~~ movement e.t.c.

 Decision Diamond shapes are generally used to check any condition or take decision of for which there are two answers, they are yes (true) or no (false)

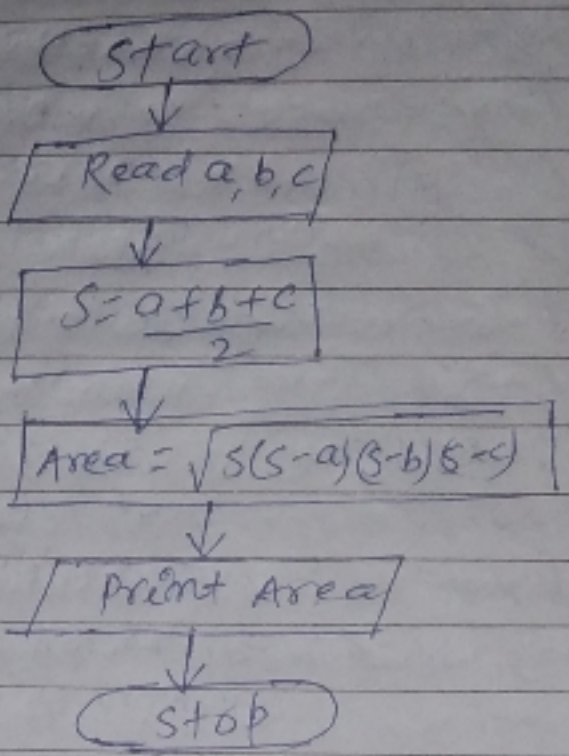
 connector It is used to connect or join flow lines.

 ⇒ repetition / loping

8) Flowchart example for calculation simple interest.



Ex. Draw flowchart to find area of traingle when its 3 sides are given.



Advantages of flowchart

1) effective communication: Flow charts are better way of communicating the logic of the system.
2) Effective Analysis: Using flowchart problem can be analyzed more efficiently.
3) Easy debugging & efficient testing: The flow chart helps in debugging & testing process.
4) Efficient coding: The flowcharts are very useful during program development phase.
5) Proper documentation: Flowcharts serves as a good program documentation, which is needed for various purpose.

6) Efficient program maintenance: maintenance of operating programs becomes easy with the help of flowchart.

Disadvantages of flowchart:-

1) complex logic: For complicated logic, flowchart becomes complex & clumsy.
2) Difficulty in modifications: If change is required in the logic then flowcharts needs to be redrawn and requires a lot of time.

Pseudocodes:- Pseudocode is structured english for describing algorithms concisely. It is made up of two words, namely. pseudo meaning imitation and code meaning instructions. It is not a real programming code.

ex:- write pseudocodes for calculation of simple interest.

Step 1:- START
Step 2:- READ P, T, R
Step 3: S.I = P*T*R/100
Step 4: PRINT S.I
Step 5: STOP

note:- Generally used keywords are capitalized while preparing pseudo code.

Different class statement in C

Instruction:

There are basically three types of instruction in C.

1) type declaration instruction
2) Arithmetic instruction
3) control instruction

Type declaration Instruction

→ This instruction is used to decleare the types of variables being used in the program.

→ Any variable used in the program must be decleared before using it in any statement.

→ The type declaration statement is written at the begining of mains function.

Ex; int bas;
float radius;
char name;

→ while declaring the type of variable we can also initialize it as shown below

Ex: int i=10, j=25;
float a=1.5 b=a+20;

→ The order in which we define the variables is sometime important & sometime not

Ex: int J=10, i=25;
is same as int i=25, J=10;

However
float a=1.5, b=a+3.1; is right
but float b=a+3.1, a=1.5; is not right.

→ The following statement would work which int a,b,c,d;
a=b=c=d=10;

but int a=b=c=d=10; not work

Arithmetic instruction:-

→ To perform arithmetic operations b/w constants and variables.

→ A 'C' arithmetic instruction consits of a variable name on the left hand of = variable names & constants are on the right side of =.

→ The variables and constants appearing on the right hand side of = are connected by arithmetic operator leke +, −, * and /.

Ex: int a, b, c, d=10;
c= a+b*d/2

→ A c arithmetic statement could be of three types. These are as follows:

1) Integermode arithmetic statements:
This is an arithmetic statement in which all operands are either integer variable or integer constants.

Ex: int a, i, b, c;
i = i+1;
c = a+b;

2) Real mode arithmetic statements: This is an arithmetic statement in which all operands are either real constants or real variable.

float a, b, c
c = a+b*2.5;

3) Mixed mode arithmetic statements: This is an arithmetic statement in which some of the operands are integer and some of the operand are real.

float a, b;
int c
b = a*c;

note: modulus (%) operator cannot applied on a float.

+ using %. sign of remainder is always same as the sign of numerator.

-5%2 = -1
5%-2 = 1

* arithmetic operations can be performed on ints, floats & char.
* There is no operator for performing exponentiation operation. Thus following statements are invalid.

Ex: a = 3**2; b = 3^2;

* exponentiation is done by pow c library function.

ex: 3^5 in arithmetic be = pow(3,5)

3^2 " " " = pow(3,2)

→ arithmetic operation b/w an integer and integer always yield an integer result

i-e (int = int ⊕ int) { ⊕ → arithmetic operator (+, −, /, *)
real = real ⊕ real
real = real ⊕ int

control instruction :- control instruction enable as to specify the order in which the various instruction in a program are to be executed by the compiler's.

→ control instruction determine the 'flow of control' in a program.

→ There are four types of control instruction in C

1) Sequence control instruction
2) Selection or decision " "
3) Repetition or loop " "
4) case control Instruction

Data type

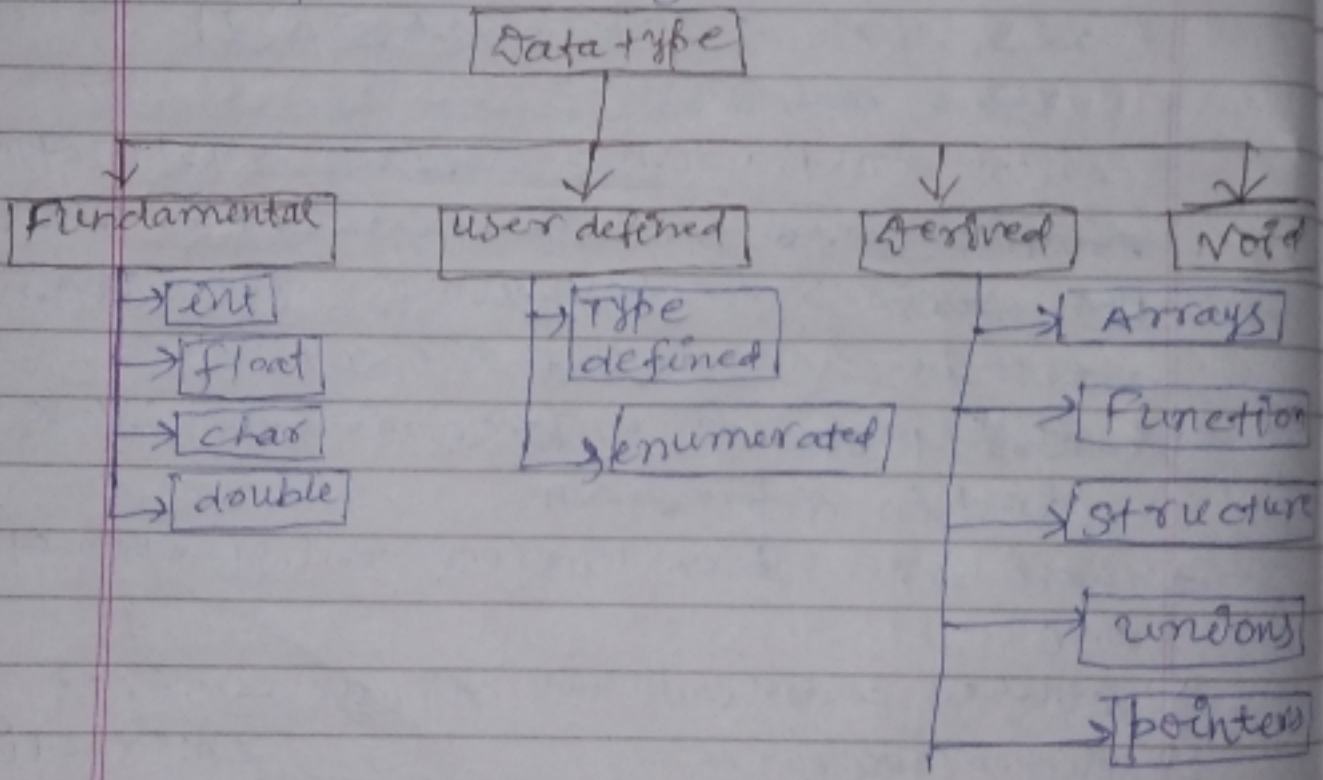
Data type in c

A data type in c programming is a set of values and is determined to act on those values. c provides various types of data-types which allow the programmer to select the appropriate type for the variable to set its value.

→ In 'c' data types can categorised in

1) Built-in data type/fundamental
2) Derived ,,
3) user defined ,,

Data types in c are shown with the help of following fig.



Data type	size (byte)	Range	Format Specifier
char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
short or int	2	-32,768 to 32,767	%i or %d
unsigned int	2	0 to 65535	%u
long int	4	-2147483648 to 21...47	%ld
unsign... long int	4	0 to 4294967295	%lu
float	4	3.4e-38 to 3.4e+38	%f or %g
double	8	1.7e-308 to 1.7e+308	%lf
long double	10	8.4e-4932 to 1.1e+4932	%Lf

Int:

Integer data type is used to store a value of numeric type. memory size of a variable of integer data type is dependent on operating system.

ex: 8 bit computer → -128 to 127

16 bit " → -32,768 to 32,767

valid	Invalid
-248	3,333
14020	34.0
27246	+3432
0	999999
32760	

1) Integermode arithmetic statements:
This is an arithmetic statement in which all operands are either integer variable or integer constants.

Ex: int a, i, b, c;
i = i+1;
c = a+b;

2) Real mode arithmetic statements: This is an arithmetic statement in which all operands are either real constants or real variable.

float a, b, c
c = a+b*2.5;

3) Mixed mode arithmetic statements: This is an arithmetic statement in which some of the operands are integer and some of the operand are real.

float a, b;
int c
b = a*c;

note: modulus (%) operator cannot applied on a float.

+ using % sign of remainder is always same as the sign of numerator.

-5%2 = -1
5%-2 = 1

* arithmetic operations can be performed on ints, floats & char.
* There is no operator for performing exponentiation operation. Thus following statements are invalid.

Ex: a = 3**2; b = 3^2;

* exponentiation is done by pow c library function.

ex: 3^5 in arithmetic be = pow(3,5)
3^2 " " " = pow(3,2)

→ arithmetic operation b/w an integer and integer always yield an integer result

i-e (int = int ⊕ int)
real = real ⊕ real
real = real ⊕ int

⊕ → arithmetic operator (+, −, /, *)

control instruction : control instruction enable as to specify the order in which the various instruction in a program are to be executed by the computer's.

→ control instruction determine the "flow of control" in a program.

→ There are four types of control instruction in C

1) sequence control instruction
2) Selection or decision " "
3) Repitition or loop " "
4) case control instruction

→ Default declaration assumes a signed no.

Unsigned

→ This can be used for both int & char
→ It can also be used in combination with long or short
→ It is used to create an unsigned integer.

Long

→ This is applied to int data type.
→ When applied to int, it almost doubles the length in bits of the data type that is modulus.

Ex: 16 bit int data → 32 bit int data by long int.

→ When long is applied double roughly doubles the precision.

Short:

→ This makes the size of an int half.
→ ASCII acceptable size for int is 16 bits smallest & most compilers have 16 bit int.
→ Most of systems, Provide size of a short int is that of int.

Arrays

In c language arrays are reffered to as structured data types. An array is defined as finite ordered collection of homogenous data, stored in contiguous memory locations.

example where arrays are used:-

→ to store list of employee or student names
→ to store marks of students
→ or to store list of no. or character etc.

operators & expressions

c operators

operators in c language are the special kind of symbols that performs certain operations on the data.

→ There are 15 operator in c.
→ c operator are broadly classified into three main categories

i) unary operators ii) Binary operators
iii) ternory operators

unary operators

A unary operator in c is an operator that takes a single operand in an expression or a statement.

→ The c unary operator are

1) unary minus
2) Logical not operator
3) Bitwise complements

Date: / /
Page:

Unary minus:

→ Any +ve operand associated with unary minus operator gets its value changed to negative.

ex:- a = 3, b = 4 c = a + (-b).
= 3 + (-4) = -1

int x = -5;

Binary operator:

A binary operator in C is an operator that takes two operands in an expression or a statement.

→ Binary operator are classified into 4 categories

i) Arithmetic operator iii) Relational operator
ii) Logical operator iv) Bitwise operator

i) Arithmetic operator:

→ Arithmatic operator are used to perform the basic arithmetic operation such as addition, substraction, multiplica

→ C do not provide exponential operat it is performed by POW( ) library fun.

→ modulus operator (%) cannot be used with floating point number.

3+4-7 = 0 (solve left to right) 3+4*5 = 23
3-4+7 = 6 (")

SSIS

operations	operator	Precedence	Association
Addition	+	2	Left to right
Substraction	-	2	"
multiplication	*	1	"
Division	/	1	"
modulus	%	1	"

Arithmetic expression:

→ Expression involving arithmetic operator is called an arithmetic expression
→ It takes one or more operands and connects them by an arithmetic operator. These operands are either integer or floating point number.
→ Arithmetic expression are evaluated from left to right
→ Expression involving high priority operators are evaluted first.
→ If an unary minus is present in the expressions. Then the term associated with unary minus must be evaluated before the other expression.

Logical operators:

Logical operators are part of binary operators these operators are specifically used when we are going to combine two or more requirements together. These can be used in many conditional and relational expressions.

→ There are three such operator: AND, OR and NOT.

→ AND and OR operators are binary and NOT is unary operator.

operator	meaning	precedence	Associat
&&	Logical AND	2	L to R
||	" OR	3	"
!	" NOT	1	"

Evaluation of logical expression:

→ Expression involving logical operators called logical expression.

→ Result of a logical expression is either TRUE or FALSE

→ If expression involving AND, OR and NOT operation, the expression involving NOT must be evaluated first, then the expression with AND and finally the expression having OR should be evaluated.

eg: a&&b||c&&(!b)
2&&4||3&&(!4)
2&&4||3&&(0)
1||0 = 1

i) Logical AND operator: The '&&' operator returns true when both the conditions under consideration are satisfied. otherwise it

returns false. For ex:-
a&&b returns true when both a and b are true (i.e non zero)

ii) Logical OR operator:- The '||' operator returns true even if one (or both) of the conditions under consideration is satisfied. otherwise it returns false.

ex: a||b returns true if one of a or b or both are true.

iii) Logical NOT operator: The '!' operator returns true the condition in consideration is not statisfied. otherwise it returns false.

ex: !a returns true if a is false, i.e when a=0

1. write a program to display 1 if input no. is b/w 1-100 otherwise 0. use logical AND(&&) operator.

```
main()
{
  int x,z;
  clrscr();
  printf("enter numbers");
  scanf("%d",&x);
  z=(x>=1 && x<=100 ? 1:0);
  printf("z=%d",z)
```

o/p enter no: 5 z=1
101 z=0

2. write a program to display 1 if inputed number is either 1 or 100 otherwise 0 use the logical OR (||) operator.

```
main()
{
  int x,z;
  printf("enter numbers");
  scanf("%d", &x);
  z = (x==1 || x==100 ? 1:0);
  printf("z = %d", z);
```

operator/ enter nos: 1

z = 1

enter nos = 100

z = 1

enter nos = 101

z = 0

Bitwise operator: perform with only (0,1)

These operators are used to perform bit operations. Decimal values are converted in binary values which are the sequence of bits and bit wise operators work on these bits

→ To perform the bitwise operations c provides six operator.

→ These operators work with int and char type data. They cannot be used with floating point numbers.

operators	symbol name	Meaning
&	Ampersand	Bitwise AND
|	pipeline	Bitwise OR
^	Caret	XOR
~	Tilde	1's complement
<<	Double less than	left Shifting of bits
>>	Double greater than	Right shifting of bits

AND
a = 00000100
b = 00000011
a&b = 00000000
*

OR
a = 00000100
b = 00000011
a|b 00000111
+

XOR
a = 00000100
b = 00000011
a^b = 00000111
similar → 0
desimilar → 1

Ex: a = 10 and its binary equivalents is 1010
b = ~a
b = ~(1010)
= 0101

Ex: a<<1 (left shift with one)

1	0	0	0	0	1	0	0	Before shift
0	0	0	0	1	0	0	0	After shift

Ex: a>>1 (Right shift with one)

1	0	0	0	0	1	0	0	Before
0	1	0	0	0	0	1	0	After

→ In Right & left Shift we use ~~Zero~~ to berforan these operators.

P-1. write a program to shift input data two bits right.

```
main()
{ int x,y;
  clrscr();
  printf("Read the input x");
  scanf("%d", &x);
    x>>=2
    y=x
  printf("Right shift data is %d", y);
}
```

output

o/p Read the i/p x: 8
the right shift data is 2.

note: shifting two bit right means

$y = \frac{n}{2^s}$ Eg: $y = \frac{8}{2^2} = \frac{8}{4} = 2$

P-2 write a program to shift inputed data by 3 bits to the left.

```
main()
{ int x,y;
  clrscr();
  printf("Read the intput x");
  scanf("%d", &x)
    x<<=3
    y=x;
printf("the left shifted data is %d", y);
```

Relational operators:

→ Relational operators in C are commonly used to check the relationship between the two variables. If the relation is true, then it will return value 1. otherwise, it returns value 0.

→ There are six relational operator in C

operator	precedence	Associativity	Example	Relational value
>	1	Left to Right	5>4	1
<	1	"	10<9	0
>=	1	"	11>=5	1
<=	1	"	10<=10	1
==	2	"	2==3	0
!=	2	"	3!=3	0

Increment/Decrement operators:

increment operator (++):

This operator is used to increment the value of an integer quantity by one. This is represent by '++' symbol & this symbol can be placed before or after the integer variable.

Ex: a = ++b
this is equivalent to
b = b+1
a = b;

similarly
a = b++
this is equivalent to
a = b;
b = b+1;

Note: (++a or --a) is uses before then value is Increment/decrement.
(a++ or a--) increament then use.

P-1. write a program to show the effect of increarment operator as suffix.

```
main()
{ int a, z, x=10, y=20;
  clrscr();
  z = x * y++;
  a = x * y;
  printf("\n %d, %d", z, a);
}
```

o/p = 200, 210

P-2. write a program to show the effect of increment operator as a prefix.

```
main()
{ int a, z, x=10, y=20
  clrscr();
  z = x * ++y
  a = x * y
  printf("\n %d, %d", z, a);
}
```

o/p: 210 210

Decrement operator

This operator is used to reduce the value of an integer quantity by one. This is represented by '--' symbol & this symbol can be placed either before or after an integer variable.

Ex: a=--b equivalent to

b=b-1

a=b;

semillarly,

a=b-- equivalent to

a=b;

b=b-1

conditional operator:

The conditional statements are the dec making statements that depends upon t output of the expression. As a conditional operator works on three operands, so it is known as the ternary operator. The operands may be an expression, constants Variables. It starts with a condition, hen it is called a conditional operator.

→ conditional operators return one value if the condition is true & the returns another value if the condition is false

special operators of C:

1) comma operator(,)
2) Size of ()
3) Address operator (&)
4) Dereferencing operator(*)
5) Dot operator(.)
6) Arrow operator(→)

comma operator (,)

The comma operator is basically associated with for statement. It is also used to link two or more related expression together. In such a case the expressions are evaluated in left to right fashion.

→ We can obtain the value of the combined expression from the value of the rightmost expression.

ex: sum(a=12, b=22, a+b)
Value of rightmost expression = 12+22 = 34

Size of ( ) operator (14)

→ The size of ( ) operator returns the size of operand. (i.e no. of byte)

→ It should be written in lowercase letters.

→ The operand may be constant, variable or a data type.

→ It is normally used to determine the size of arrays and structures.

→ It also used for dynamic memory allocation.

Ex: x = size of(int);
size of(sum);

→ The sizeof operator is normally used to determi the lengths of arrays and structures when their sizes are not known to programmer. It is also used to allocate memory space dynamical to variables during execution of a progra

Assignment operator

An assignment operator is the oper
used to assign a new value to a vari
→ we have seen the usual assignme
operator '='
→ This is basically associated with a
arithmetic operators & bitwise operato

operator	Assignments	shorthan A
+	a = a+b	a+ = b;
-	a = a-b	a- = b;
*	a = a*b	a* = b;
/	a = a/b	a/ = b;
%.	a = a%. b	a%. = b;
&	a = a&b	a& = b;
|	a = a|b	a| = b;
^	a = a^b	a^ = b;
<<	a = a<<b	a<<=b;
>>	a = a>>b	a>>=b;

→ In addition, c has a set of shorthan
assignment operators of the form

Var	operator	=	exp ;

Decision making & branching

Decision making & branching is one of the most important concepts of computer programming. Program should be able to make logical decisions based on the condition provided.

→ In most of the programs, it is necessary to

i) select a set of statement from several alternatives.

ii) skip certain statements depending on some conditions in the program & continue the execution from some other point.

iii) Repeat a set of statement for unknown no. of times or until a specified condition is fulfilled under such situation we make use of control statements-they are generally called control ~~structure~~ structure. C provides a widely range of control statements to control sequence of statements being a C program.

→ C provides a variety of condition control statement such as

i) if statement
ii) if-else "
iii) nested-if statement
iv) switch statement

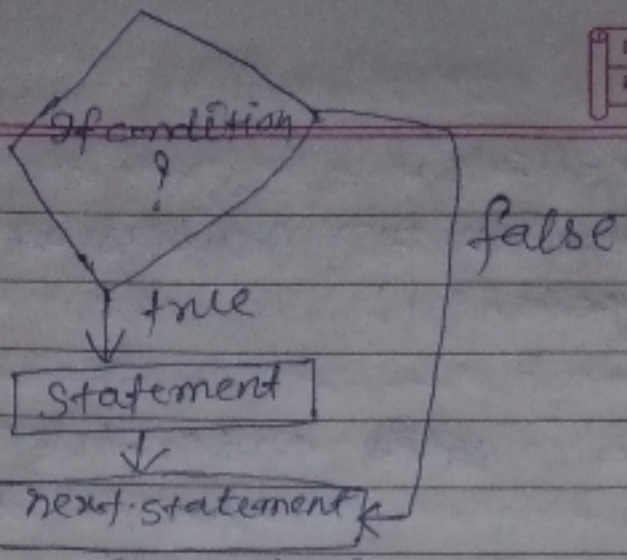
IF

→ If statement is a powerful decision-making statement. It is called a one way branch.
→ The logical condition is tested which results in either a true or false.
→ If the result of logical test is true then the statement that immediately follows 'if' is executed.
→ If the logical condition is false, then control transfers to the next executable statement.
→ The condition may be an expression containing constants, variable or logical comparisons.
→ The condition must be written within the parentheses.

Syntax

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

where condition; It is logical expression that result in true/false
statement: a simple/compound statement



FLOW chart

P-1. write a program that accept a nos & prints if it is odd no.

```
#include (stdio.h)
main()
{
    int number;
printf("enter the no.\n");
scanf("%d", &number);
  if( (number % 2) != 0 );
printf("%d", is an odd number");
    }
```

o/p Enter the no. 13
13 is odd no.

P-2 write a program to check equivalend of two no. use if statement.

```
#include<stdio.h)
#include<conio.h)
   void main()
    { int m, n;
      clrscr();
printf("\n Enter the two numbers");
scanf("%d, %d", &m, &n);
```

```
if(m-n=0);
printf("\n two numbers are equal")
    getch();
        }
```

o/p :- enter two no: 5,5
two nos are equal.

IF-ELSE Statement

→ The if-else statement takes care of true as well as false condition. It has two blocks one block is for if and is executed when the condition is true the other block is of else & this execute when the condition is false.

→ The else statement cannot be used without if.

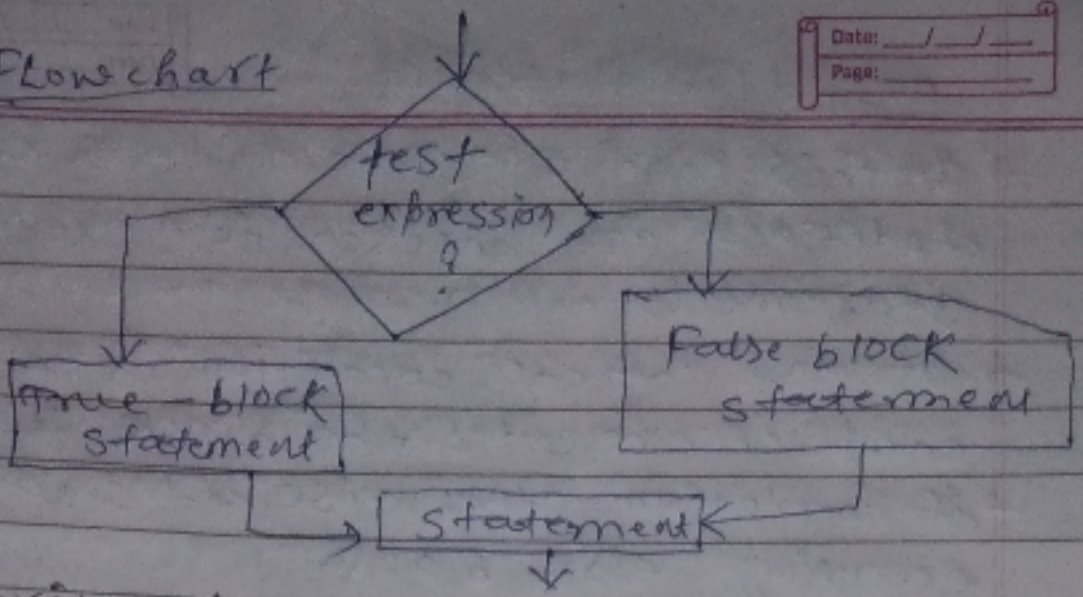
→ No multiple else statements are allowed with one if.

Syntax

```
if(condition)
    {
    True block statements
    }
else
    {
    false-block statements
    }
```

note: If we write only one statements then we are not use the ('{') this bracket.

Flowchart



P-1. write a program accepts two nos & check whether they are equal.

```
#include<stdio.h>
#include<conio.h>
void main
  {
    int a,b;
    clrscr();
  printf("Enter the two number");
  if(a-b=0) scanf("%d,%d",&a,&b);
  printf("They are equal");
  else
    printf("They are not equal");
        } getch();
          }
```

note: '\n' is escape sequence in c language.

Nested if statement

→ In this kind of statement no. of logical conditions are checked for executing the statements.

→ If any logical condition is true the compiler executes the block followed by if condition, otherwise it skip and executing else block.

→ In if-else statement else block is executed by default after failure of condition.

Syntax

```
if (condition1)
{ if (condition2)
    { statement1
    }
  else
    { statement2
    }
}
else if (condition3)
{ statement3;
}
else
{
  statement4;
}
```

P-1 WAP to calculate largest of three no. using if-else.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a,b,c;
    clrscr();
printf("Enter the threenumber");
scanf("%d,%d,%d",&a,&b,&c);
if(a>b && a>c)
printf("Largenst no. is a=%d");
else if(b>a && b>c)
 printf("Largest no is b=%d");
else if(c>a && c>b)
printf("largest number is c=%d");
    getch();
}
```

Loop control structure

Loop control statements in c are used to perform looping operations until the given condition is true. control comes out of the loop statements once condition becomes false.

→ Loop control program consist of two segments

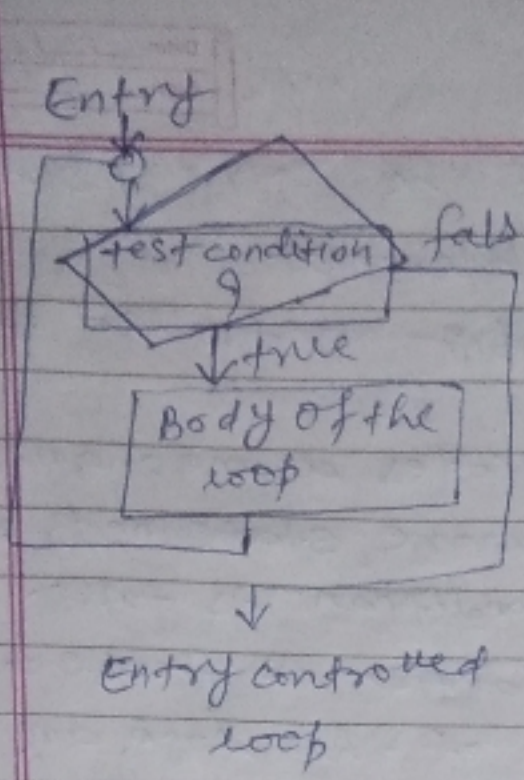
i) body of loop

ii) control statement (testing condition)

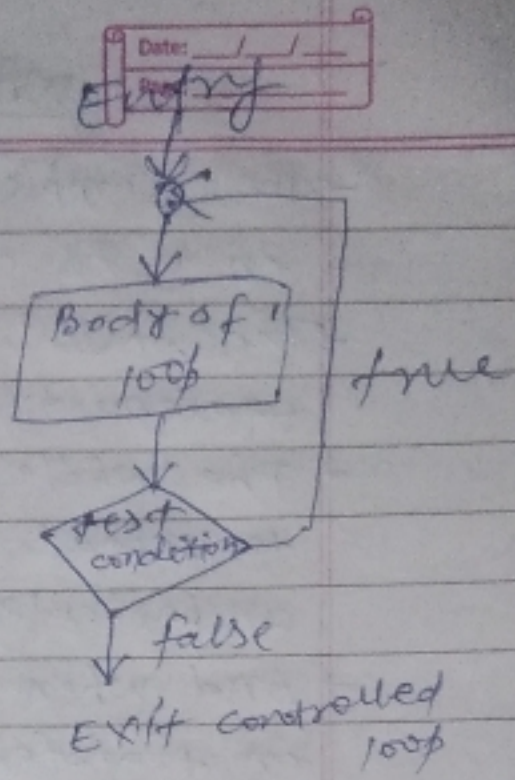
→ Depending on position of the control statement in loop control structure may be classified as entry & exit controlled loop.

→ Entry control loop: In entry control loop the control of conditions are tested before the start of loop execution. If the condition are not satisfied the body of loop will not be executed. EX: for, while

→ Exit-control loop: The test is performed at the end of the body of loop. Body is executed unconditionally for the first time



Entry controlled loop



Exit controlled loop

→ Thus in looping, in general, would include the following 4 steps:

i) Setting & initialization of a conditional variable
ii) Execution of the statements in the loop
iii) Test for a specified values of the condition variable for executing the loop
iv) increamenting or updating the condition variable.

→ 'C' language provides for three constructs for loop operations they are

1) The 'while' statement
2) The 'do' statement
3) The 'for' statement

While

→ The simplest of all the looping structure in c is 'while' statement.
→ The 'while' is an entry-controlled loop statement.
→ The 'while' loop provides a mechanism to repeat one or more statements while a particular test condition is true.
→ And when test-condition is false control in transferred out of the loop, and exits the program continues with the statement immediately after the body of the loop.
→ Body of loop may have one or more statements.
→ The braces are needed only if the body contains two or more statements.
→ If the condition is never update and condition never becomes false, then the computer will run into an infinite loop.
→ If the control condition evaluates to false then the statements enclosed in the loop are never executed.

Syntax

```
while (test condition)
{
    body of the loop
}
```

→ The condition being tested may use relational or logical operators.

Ex:- while (i<=10)

while (i)10 && (b<15||c<20))

→ Instead of incrementing a loop counter we can even decrement it.

Ex:-

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

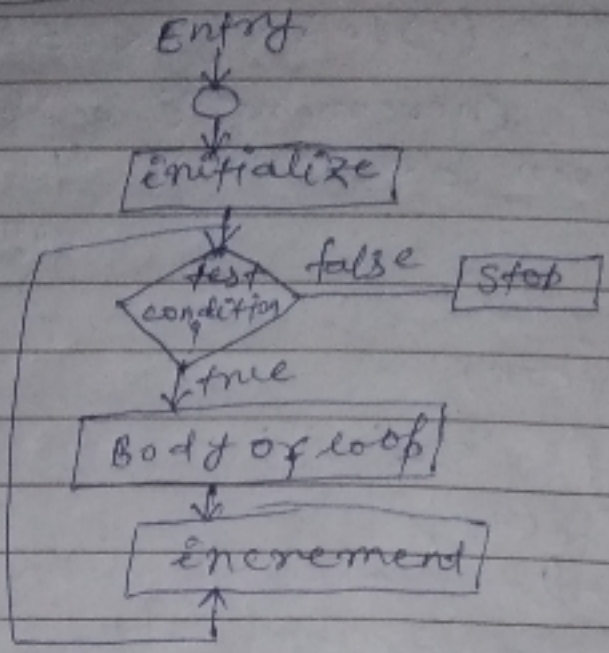
```
main
{ int i=5
while (i>=1)
{
printf("welcome to computer world");
i= i-1
}
}
```

→ It is not necessary that a loop counter must only be an int. It can even be a float.

```
main()
{
float a=10.0
    while(a<=10.5)
    {
printf("welcome to computer world");
printf("to computer programming);
        a=a+0.1
    }
}
```

→ Even floating point loop counters can be decremented

Flow chart



P-1. WAP to print your name 9 times using while loop.

```
main
{   int x=1;
        while(x<10)
        { printf("\n Hello");
          x++;}
}
```

P-2 WAP to add 10 consecutive numbers starting from 1. use the while loop.

```
main()
{
int a=1, sum=0;
clrscr();
while(a<=10)
```

{

printf("%d", a)

sum = sum + a;

a++;

}

printf("\n sum of 10 no is %d", sum);

}

P-3 WAP to calculate the sum of no. from m to n.

```
#include<stdio.h>
#include<conio.h>
main()
{
  int m, n sum=0;
  clrscr();
  printf("\n Enter the value of m & n");
  scanf("%d, %d", &m, &n);
  while(m<=n)
  {
    sum = sum+m;
    m = m+1
  }
  printf("\n sum is %d", sum);
  getch();
}
```

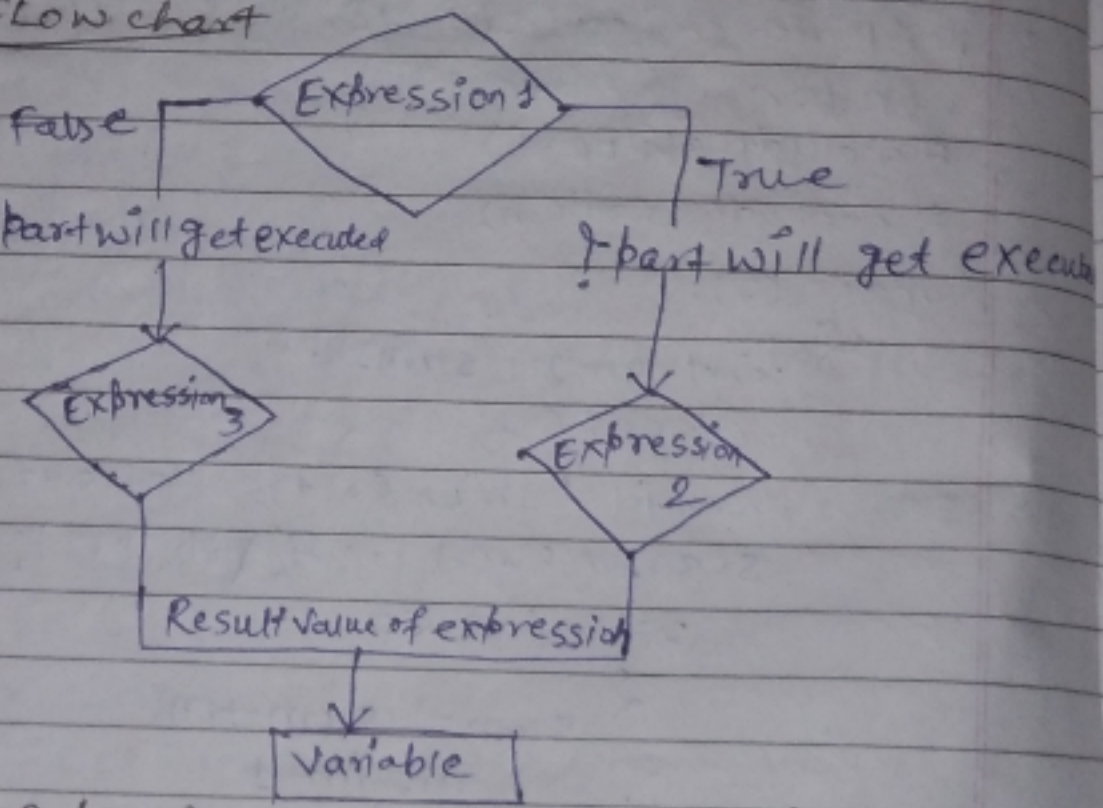
Conditional or ternary operator (?:)

The conditional operator is kind of similar to the if-else statement as it does follow the same algorithm as of if else statement.

Syntax

Expression 1 ? Expression 2 : Expression 3

FLOW chart



Q) WAP to check a no is positive or not.

```
#include<...>
void main()
    {
    int x;
    clrscr();
printf("Enter the number");
scanf("%d", &x);
```

x>0? printf("Positive number"): printf("Negative number");

getch();

}

8) Program to store the greatest of two numbers.

```
#include<stdio.h>
void main()
  {
     int z,a,b;
     printf("Enter the no");
     scanf("%d%d", &a,&b);
    z= a>b? a:b;
  printf("Greater no is %d", z);
       getch();
          }
```

break

→ The break can be used in loop body or in switch body.

→ The purpose of break is to terminate loop's execution immediately as it encounters.

continue statement: This statement skips the rest of the loop statement and starts the next iteration of the loop to take place. ~~Below is the program~~

'do'while

Syntax

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

Q) Wap to print A no. 1 to 10 using do loop

```
#include<stdio.h>
int main()
{
    int a=10
    do
    {
        printf("value of a is %d\n", a);
        a++;
    } while( a<20);
    getch();
}
```

for while

A "For" loop is used to repeat a specific block of statements to a known number of times. The for-loop statement is a very specialized while loop, which increases the readability of a program.

Syntax

For(initialize counter; test counter; increment counter)

```
{
  execute the statements;
}
```

note: The for loop must have two semi-colon b/w the opening & closing parenthesis.

5) WAP to print your name 6 times by for loop.

```
void main()
{
  int i;
  for (i=1; i<=6; i++)
  {
    printf("\n Rishabh");
  } getch();
}
```

Difference b/w break & continue statement:

	Break	continue
1)	The break statement is used to exit from the loop constructs.	The continue is used to continue the next iteration in the loop.
2)	It is used in loop & switch case	It is used only in loop
3)	The program control is transfer outside the loop.	control remains in the same loop.
4)	Syntax 'break'	Syntax 'continue'

8) WAP to use break & continue statement

→
```
void main()
{ int i;
  clrscr();
for(i=1; i<=10; i++)
{
    if(i=5);
    break;
    printf("%d", i);
    }
    getch();
}
```

→
```
void main{
  int i;
  clrscr();
for(i=1; i<=10; i++)
  {
    if(i==5);
    continue;
  printf("%d", i);}
    getch(); }
```

Arrays

→ Array's are a kind of data structure that can store a fixed-size sequential collection of same data type.

→ All array consist of continuous memory allocation.

→ Array is a group of variable

Declaration of array

int a[ ]; error

int a[5];

data type → array name [size];

Ex: int a[5]

a[0]	a[1]	a[2]	a[3]	a[4]
9	8	7	6	5

note: i) If you declare the size of array and assign the element of array in box is less than its size then the blank box contain atomaticly zero value in its.

ii) If you assign the element of array in box's greater than its size then its show error.

→ int S[5]

S[0]	S[1]	S[2]	S[3]	S[4]
11	9	8	6	22

Address ~ 200 202 204 206 208

↳ integer take two bite

* Length of arry = (upper bond - Lower bound + 1)

B) WAP to read & display 'n' no using an array

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n,i,a[n];
printf("Enter the number of array|n");
    scanf("%d",&n);
printf("Enter for(i=0; i<n; i++)
elemund of array") scanf("%d",&a[i]);
    getch();
}
```

Array operation

- Traversal
- copying
- Reversing
- sorting
- Insertion
- Deletion
- Searching
- Merging

① Traversal : Traversal means a access each array element for a specific

purpose, either to perform an operation on them, & counting the total number of elements. Or,
Print all the array elements one by one.

```
#include<stdio.h>
#include<conio.h>
void main()
 {
 int i, a[5]={2,3,4,8,0};
  for(i=0; i<5; i++)
   {
    printf("a[%d]=\t", i);
    printf("%d\n", a[i];
   }
  getch();
 }
```

8) WAP to find biggest no. in array.

```
#include<stdio.h>
#include<conio.h>
 void main() {
int a[50], n, i, big;
   clrscr();
printf("Enter the number of array\n");
scanf("%d", &n);
```

element

```
printf("Enter the [redacted] in array\n");
for(i=0; i<n; i++)
{ scanf("%d", &a[i]);
    big = x[0];
   for(i=0; i<n; i++)
     {
     if (a[i] > big)
     big = x[i]
      }
     }
printf("\n %d is the bigger number in
      array", big);
     getch();
      }
```

8) WAP to implement linear search.

```
#include<stdio.h>
#include<conio.h>
  void main()
{ int a[10], i, n, num;
    clrscr();
printf("Enter the no. of array\n");
scanf("%d", &n);
printf("Enter the element of array\n");
for(i=0; i<n; i++)
scanf("%d", &a[i]);
```

```
printf("\n Enter the numer to search");
scanf("%d" &num)
 for(i=0; i<n; i++)
   {
    if( a[i]==num)
   {
   printf("%d is persent at %d \n", num, i+1);
    break;
    }
    }
     if (i==n)
  printf("%d is not persent in array");
   getch();
       }
```

Binary search

→ Binary search is a search algorithm that is used to find the position of an element in a sorted array.

8) WAP C program to search an element in array using binary search.

```
#include<stdio.h>
# include<conio.h>
   void main()
   {
   int a[10], n, i, num, l, u, mid;
      clrscr();
```

```
printf("Enter the number of array\n");
  scanf("%d", &n);
printf("Enter the element in array");
  for(i=0; i<n; i++)
  scanf("%d", &a[i]);
printf("Enter the number to search");
 scanf("%d", &num);
  l=0;
  u=n-1;
  while(l<=u)
    {
      mid=(l+u)/2;
  if(a[mid]==num)
    {
    printf("Number is at %d\n", mid);
  return;
    }
  elseif(a[mid]>num)
    u=mid-1;
  else
      l=mid+1;
    }
printf("Number is not found at array");
      getch();
        }
```

	a[0]	a[1]	a[2]	a[3]	a[4]
a[5] =	5	7	9	11	13

Search = 7

mid = $\frac{0+4}{2} = 2$

a[2] = 9

9 > 7

∴ u = mid - 1 = 1

then

mid = $\frac{0+1}{2} = 0$

a[0] = 5 ⇒ 7 > 5

then l = mid + 1

l = 0 + 1 = 1

u = 1

∴ mid = $\frac{1+1}{2} = 1$

a[1] = 7 search completed at (2).

Search = 11

mid = $\frac{0+4}{2} = 2$

a[2] = 9

9 < 11

∴ l = mid + 1 = 2 + 1 = 3

& mid = $\frac{3+4}{2} = \frac{7}{2} = 3$

a[3] = 11 Search completed at 3 (4)

Bubble sort

Bubble sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

8) WAP to sort the array using bubble sort.

```
#include<stdio.h>
#include<conio.h>
void main()
{ int a[15], n, i, j, temp;
  clrscr();
printf("Enter the no. of array\n");
 scanf("%d", &n);
printf("Enter the element of array\n");
 for(i=0; i<n; i++)
 scanf("%d", &a[i]);
  for(i=0; i<n; i++)
   {
    for(j=0; j<n-1; j++);
   {
     if(a[j]> a[j+1])
      { temp= a[j+1];
        a[j+1]= a[j];
        a[j]= temp;
      }
     }}
printf("After sorting\n");
  for(i=0; i<n; i++)
   printf("%d\n", a[i]);
      getch();
}
```

2D-array

→ collection of 1D array
→ Row and column rebresentation
Declaration :- array name [size][size];
Ex: int a[10][10]
store (10*10) = 100 elements
* Elements in 2D-arrys are commonly referred to by a[i][j] where i is row & j is the column.
* Two dimensional array can be seen as a table with i rows and j columns where the row number ranges from 0 to (i-1) & column number ranges from 0 to (j-1).
→ a[3][3] represented as:

	column 0	col 1	col 2
Row 0	a[0][0]	a[0][1]	a[0][2]
Row 1	a[1][0]	a[1][1]	a[1][2]
Row 2	a[2][0]	a[2][1]	a[2][2]

8) WAP to display 2D-array.

```
#include<stdio.h>
#include<conio.h>
    void main( )
    { int a[3][3], i, j;
    clrscr();
    printf("Enter the elements of array\n");
    for(i=0; i<3; i++)
        {
```

```
for(J=0; J<3; J++)
scanf("%d", &a[i][J]);
}
printf("2D-array element is\n");
 for(i=0; i<3; i++)
 for(J=0; J<3; J++)
 printf("%d\t", a[i][J]);
   getch();
  }
```

8) WAP to add two matrices in c.

```
#include<stdio.h>
# include<conio.h>
  void main()
   { int a[3][3], b[3][3], c[3][3], i, J;
   clrscr();
printf("Enter elements in 1st matrices\n");
 for(i=0; i<3; i++)
  {
   for(J=0; J<3; J++)
   scan("%d", &a[i][J]); }
printf("Enter elements in 2nd matrices\n");
  for(i=0; i<3; i++)
    {
     for(J=0; J<3; J++)
     scan("%d", &b[i][J]);
          }
```

```
printf("Addition of two matrices is \n");
 for(i=0; i<3; i++)
  {
   for(j=0; j<3; j++)
    {
     c[i][j] = a[i][j] + b[i][j]
   printf("%6d\t ", c[i][j]);
    }
  }
   getch();
    }
```

8) WAP to multiply two matrices in c.

```
#include<stdio.h>
#include<conio.h>
 void main()
    { int a[3][3], b[3][3], mult[3][3], i, j,
            k;
  clrscr();
printf("Enter the 1st matrices \n");
  for( i=0 ; i<3; i++)
  { for(j=0; j<3; j++)
    scanf("%d", &a[i][j]); }
printf("Enter the 2nd matrices \n");
  for(i=0; i<3; i++)
  { for(j=0; j<3; j++)
```

```
scanf("%d", &b[i][j]); }
printf("Multiplication of two matrices is\n");
for(i=0; i<3; i++)
{ for(j=0; j<3; j++)
{ mult[i][j]=0
for(k=0; k<3; k++)
mult[i][j]=mult[i][j]+(a[i][k] * b[k][j]);
printf("%6d\t", mult[i][j]);
}
}
getch(); }
```