:- Algorithm :-

Algorithm is a step-by-step procedure or a set of rules for solving a specific problem or performing a task. It is a sequence of instructions that takes an input, processes it, and products

and output. Algorithm can be expressed in various forms, including natural languages, pseudocode or programming language.

There are some basic steps to make an algorithms:-

(i) Start :- start the algorithm.

(ii) Input :- Take the input for values in which the algorithm will execute.

(iii) Conditions :- perform some conditions on the inputs to get the desired output.

(iv) Output :- printing the Output.

(v) End :- End/stop/exit the execution.

*characteristics of a good algorithm:

(i) precision :- The steps are precisely stated (defin

(ii) Uniqueness :- Results of each steps are uniquely defined and only depends on the input and the result of the preceding steps.

(iii) finiteness :- The algorithm stops after a finite number of instructions are executed.

(iv) Input :- It receives input.

(v) output :- It produces output.

(vi) generality :- The algorithm applies to a set of inputs.

· Example of an algorithm:

• Swap two numbers with a third number/variable.

→ Algorithm :-

1) start

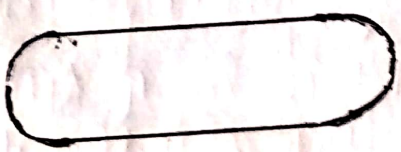
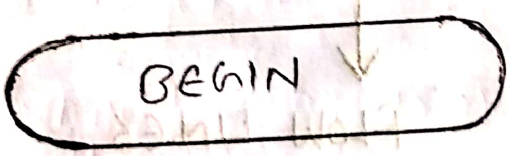
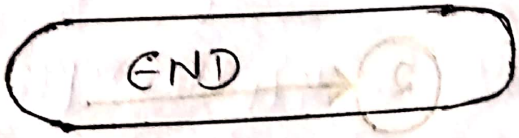
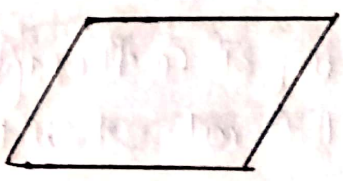
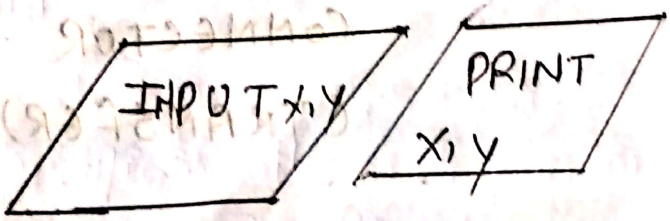
2) Take 2 numbers as input.
3) Declare another variable as "temp".
4) Store the first variable to "temp".
5) Store the second variable to the first variable.
6) Store the "temp" veriable to the 2nd veriable.
7) Print the first and second variable.
8) End.

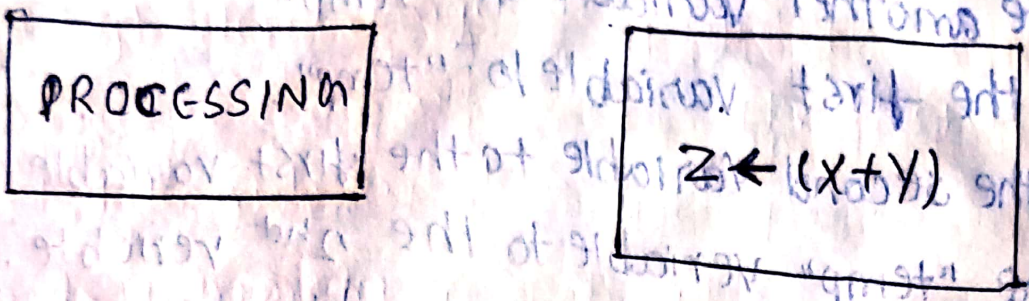
:- FLOW CHART :-

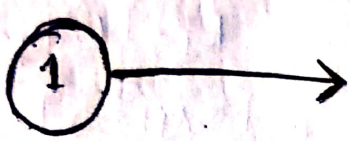
A flowchart is a graphical representation of the sequence of operations for the solution of the a problem. Program flowcharts show the sequence of instructions written in a single program or subroutime.

Flow chart (in a single) uses boxes of different shapes to denote different types of instructions. The actual instructions are written within these boxes using clear and concise statements.

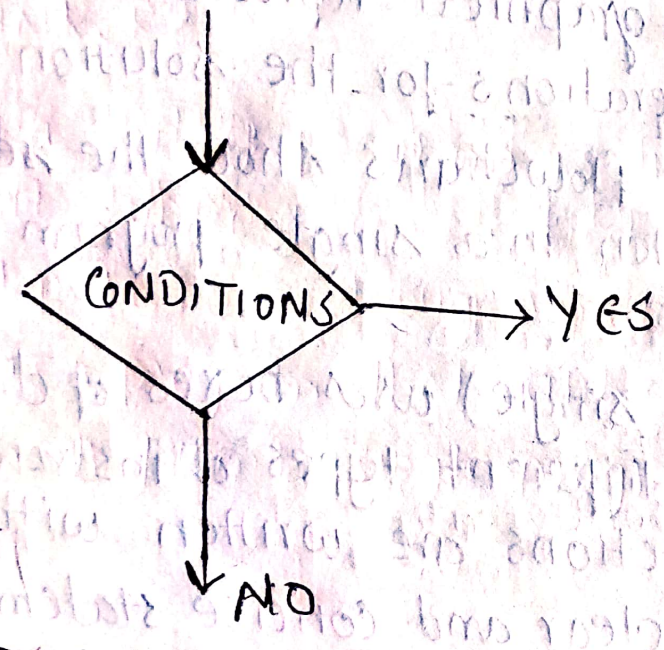
:- FLOWCHART SYMBOLS :-

SYMBOL	EXAMPLE
	
TERMINAL	
	
INPUT/OUTPUT	

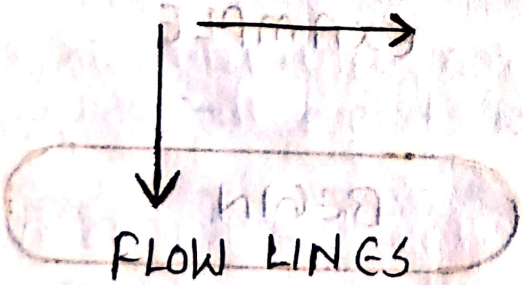




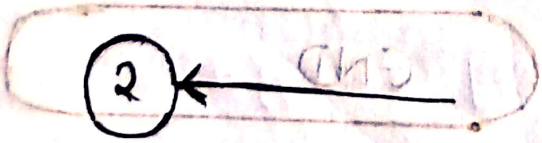
ENTRY CONNECTOR



DECISION DIAMOND



FLOW LINES



CONNECTOR
(TRANSFER)

-: Advantages of using flowcharts :-

A flowchart is a pictorial representation of the solution of a problem, it is easier for a programmer to understand the logic of a program.

- Effective joining of parts of a system
- Efficient coding.
- systematic Debugging.
- Systematic Testing.

-: Limitations of Flowcharts :-

- Take more time to draw
- Difficult to make changes.
- Non-standardization.

* COMPILER *

A program which translates a high-level language program into a machine language program is called a complier.

It checks all kinds of limits, ranges, errors, etc. But its program execution time is more, and occupies a larger part of the memory, It has slow speed and low-efficiency in memory utilization.

* Interpreter *

An interpreter is a program which translates statements of a high-level language program into [illegible] It translates one statement of the

program at at a time. It reads one statement of a high-level language program, translates it into machine code and executes it. Then it reads the next statements and translates.

An interpreter is a small program as compared to complier. It occupies less memory space, so it can be used in a smaller system which has limited memory space.

LINKER

Usually a longer program is divided into a number of small subprograms called modules. It is easier to develop, test and debug smaller programs. A linker is a program that links (combines) smaller programs to form a single program.

2 Introduction 'C'

The 'C' Language is developed by Dennis Ritchie for creating system applications that directly interact with the hardware devices such as drivers, kernels etc.

C programming is considered as the base for other programming languages, that is why it is known as mother language.

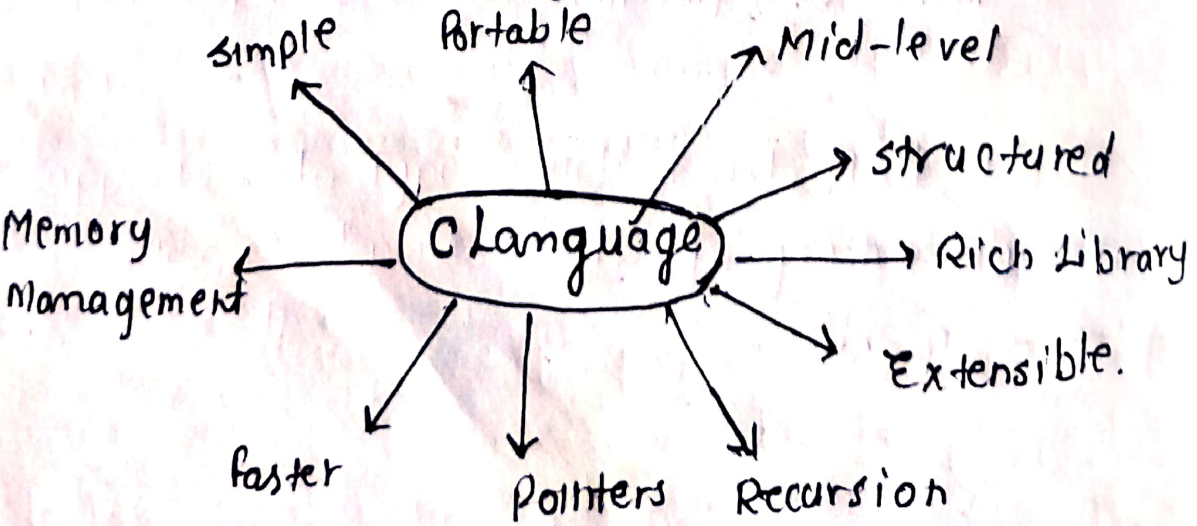
:- History of C language :-

C programming language was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A.

It was developed to overcome the problems of previous languages such as B, BCPL, etc.

Initially, C language was developed to be used in UNIX operating system. It inherits many features of previous languages such as B and BCPL.

:- Features of C language :-



etc.

—: Variables :—

A variable is the name of memory location, It is used to store information. It's value can be altered and reused several times..

Variable are key building elements of the C Programming language used to store and

modify data in computer program.

syntax:

datatype Variable Type;

- float a

* Variable Definition:

The process of reserving memory space for the variable to keep its contents during program execution is known as a variable definition. It is based on data type and connects the variable name with a particular address of sufficient size.

Example of variable definition:

```
#include<stdio.h>
int main(){
//Variable definition
 int age=25;
 float salary=2500.5;
 char initial='J';
 return 0;
}
```

-: Key words :-

Keywords in C are reserved words that have predefined meanings and part of the C language syntax. These keywords can not be used as variable names, function names, or any other

identifiers within the program except for their intended purpose.

The compiler recognise a defined set of keywords in the C programming language. These keywords have specialized purposes and play critical roles in establishing the logic and behaviour of a program.

- Characteristics of C keywords :-

- Reserved: The C language reserves keywords are those keywords that cannot be used as identifiers in program. using a keyword as a identifiers in program, variable name will cause a compilation error.

- Predefined meaning :- Each keyword has a specific meaning that is assigned by the C language. These meanings are built into the C language's grammer and syntax and the compiler interprets them accordingly.

- Specific use :- Keywords are designed for specific purposes and contexts, within the C language. They define control structures, data type and flow types, and other language constructs. Attempting to use a keyword outside of its intended purpose will result in a compilation error.

Ex :-
auto, break, int, float, char, continue, do while, else, double, long, for etc.

→ C Identifiers :-

C identifiers ; represent the name in the c program, for example, variables, functions, arrays, structures, unions, labels etc. A identifier can be composed of letters such as upper case, lower case letters, underscore, digits ; but the starting letter should be the underscore or a alphabet.

We can say identifiers is a collection of alphanumeric characters that begins either with an alphabetical character or an underscore, which are used to represent various programming elements.

- Rules for constructing C identifiers.
- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digits, or underscore.
- It should not begin with any numerical digit.
- Identifiers are case sensitive.
- Keywords can not be represented as an identifier.
- The length of the identifiers should not be more than the 31 characters.
- Identifiers should be written in such a way that it is meaningful, short and easy to read.

C operators :-

An operator is a simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc.

- following are the different types of operators :-

1) Arithmetic operations :-

1) Addition operator (+) : The addition operator adds two operends stogeather.

- substraction operator (-) : The scecond operand is substracted from the first operand via the substraction operator.
- multiplication operator (*) : The operator is used to multiply the two operands.
- Division Operator (/) : The first operand and the second operand are divided using the division operator.
- modulus operator (%) : The modulus operator determines the remainder of the division between two operands.

2) Relational operators :- Relational operators assess the relationship between values by comparing them. They return either true (1) or false (0).

- Equality operators (==) : If two operands are equal the equality operators rarify this.

- Inequality operator (!=): The inequality operator determines whether two operands are equal or not.
- Greater than operator (>):
- Less than operator (<):
- Greater than or equal to operator (>=) It determines if the first operand is more than or equal to the second operand.
- Less than equal to operator (<=):

3) Shift operators:
A binary numbers bits can be moved to the left or right using shift operators.
- Left shift operator :- It moves the bits of the first operand to the left by the number of places indicated by the second argument.
- Right shift operator (>>)

4) Logical operators :- Logical operators perform logical operations on boolean value and return either true (1) or false (0).
- Logical & operator (&&): The logical AND operator returns true if both operends are true.
- Logical OR operator (||): The logical OR operator return true if atleast one of the operands is true.
- Logical NOT operator (!); The logical NOT operator negates the value of the operand)

Data type in C

Each variable in C has an associated data type. It specifies the type of data that the variable can store like integer, character, floating double, etc. Each data type requires different amounts of memory and has some specific operations which can be performed over it.

The data types in C can be classified as follows :-

1. Primitive Data types :- primitive data types are the most basic data types that are used for requirenting simple value such as integers, float, characters etc.
Data types int, char, float, double, void.

2) Derived data types :- The data types that are derived from the primitive or built in datatypes are reffered to as Derived Data types. Ex :- array, pointers, function.

3) User defined data types :- The user-defined data types are defined by the user himself.

Ex :- structure, union, enum.

(1) To print table of any number by do loops.

(2) write a program to generate first 100 prime numbers

(3) Difference between while and do-while.

while	do-while
Condition is checked first then statement is executed.	Statement is executed at least once, thereafter condition is checked.
It might occur statement is executed zero times. If condition is false.	At least once the statement is executed.
No semicolon at the end of while while (condition)	semicolon at the end of while. while (condition);
Variable in condition is initialized before the execution of loop.	variable may be initialized before or within the loop.
while loop is entry controlled loop	do-while loop is exit Controlled loop

⇒ control statements in C:-

control statements in C are programming constructs that are used to control flow of execution in a program. They allow a programmer to specify conditions that determine which statements are executed and which are skipped; to loop through statements until a condition is met, or jump to a different part of a program.

- control statements serve to control the execution flow of a program.
- The if statement, for loop, while loop, switch statement, break statement and continue statement are C's most widely used control statements.
- for loop:- It's a control statement, which sets a variable 'i' to 0 and the performs a block of code as long as 'i' is less than number.
- while loop:- It sets a variable 'i' to 0 and then perform a code block if 'i' is less than number.

⇒ Advantages:-

- Control statements allow programmers to control the flow of execution in their program.
- The allow programmers to handle different scenarios and conditions in their programs.

⇒ disadvantages:-

- OVA use of control statements can make code difficult to read and maintain.
- Complex control statements can be diffi cult to debug and can introduce bugs in the code.

(Array in C)

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data types in c programming language, which can store primitive type data such as int, char, double, float etc. It also has the capability to store the collection of derived data types, such as pointers, structure etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number. C array is beneficial if you have to store similar elements.

Properties of arrays:

- Each element of an array is of same data type and carries the same size i.e. int = 4 bytes.
- Element of the array are stored at contigious memory locations where the first element is stored at the smallest memory location.
- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

→ Declaration of C Array.

We can declare an array in c language in the following way.

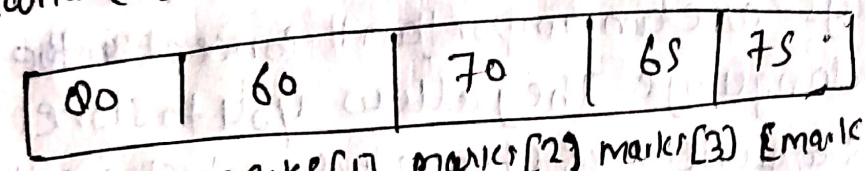
```
data-type array_name [array_size];

int mark [5];
```

→ Initialization of C Array.

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

```
marks[0] = 90; // initialization of array.
marks[1] = 60;
marks[2] = 70;
marks[3] = 65;
marks[4] = 75;
```



→ initialization of Arrays :-

→ Two Dimensional Array in C.

The two dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

Declaration of two dimensional Array in C.

The syntax is given below.

data-type array-name[rows][columns];

ex int twodimen[4][3];

Variable in C language

A variable in C language is the name associated with some memory location to store data of different types. There are many types of variables in C depending on the scope, storage class, lifetime, type of data etc. they store etc.

"A variable in C is memory location with some name that helps store some form of data and retrieves it when required. we can store different type of data in the variable.

Syntax

→ Single variable

Data_type Variable_type = value;

→ Multiple variable.

Data_type variable_name, variablename2;

- Rules
  - → Not start from digit/number
  - → white space is not allowed.
  - → underscore is allowed
  - → shouldn't be keyword.

Types

- Local variables.
- → Global variables.
- → Static variables.
- → ~~Automatic variables.~~
- → Automatic variables.
- → Extern variables.
- → Register variables.

① Local variables in C:
A Local variable in C is a variable that is decleared inside the function or a block of code. It's scope is limited to the block or function in which it's declared.

Ex; #include<stdio.h>
void function()
{
int x = 10; //local variable.
printf("%d", x);
}.

② Global variable - A Global variable in C is a variable that is decleared outside the function or a block of code. It's scope is the whole program.. i.e. we can access the global variable anywhere in the c program after it is decla-red.

Ex;
#include<stdio.h>
int x = 10; //Global function

void Function1(){
Print("Function1; %d\n", x); }

int main()
{
Function1();
return 0;

③ Static variables :-

A static variable in C is a variable that is defined using the static keyword. It's can be defined only once in a C program and its scope depends upon the region where it is declared (can be local or global).

syntax :

static data_type variable_name = initial value;

④ Automatic variables :-

All the local variables are the automatic variables by default. They are also known as auto variables. Their scope is local and lifetime is till the end of the block, we can use auto keyword to define the auto variables.

syntax :

auto data_type variable name;

⑤ External variable :

External variables in C can be shared between multiple C files. We can declare an external variable using extern keyword.

The scope is global and they exist between the multiple C files.

syntax :

extern data_type variable name.

Functions in C

A function is a set of statements that when called perform some specific tasks. It is the basic building block of a program that provides modulatory and code reusability.

Syntax of Functions in C

The function's ~~can~~ syntax can be clivide into three aspects -

1) Function Declearation.
2) Function Definition -
3) Function calls.

i) Function declearation

In a function declearation, we must provide the function name, its return type, and the numer and type of its parameter. A function ~~tell~~ declearation tells the ~~parameter~~ compiler that there is a function ~~which is given~~ with the given name defined somewhere else in the program.

syntax -

return_type name_of_the_function (P1, P2);

```
int sum (int a, int b);
```

- int → return type
- sum → Function name
- int → Parameter type
- a, b → parameter name

② function Definition

- The function definition consist of actual statements which are executed when the function is called.

syntax:

return_type Function_name (Para1_type Para1_name, Para2_type, Para2_name)] header.

{

// body of the function

}

③ Function call

A function call is a statement that instructs the compiler to execute the function, we use function name and parameter in the function call, #in-

~~int~~ void ~~hell~~ print hello();

int main() {

printHello();

return 0; }

void printHello() {

Printf("Hello\n"); }

function
- declaration
- Calling
- Definition

working of a fun
- declearation
- defining
- Calling.
- execution.
- returning.

Types of functions:-

1. library functions.
2. user-defined function.

(i) Library functions

A library function is also reffred to as a "built-in function". A compiler package already exist that contain these functions, each of which has a specific meaning, and is included in the package. Built-in functions have the advantages of being directly usable without being defined, whereas user defined functions must be decleared and defined before being used

Ex: Pow(), sqrt(), strcmp() etc

(ii) ~~Library~~ User defined functions

functions that the programmer creates are known as User-Defined functions or "tailor-made functions", User defined functions can be improved and modified according to the need of the programmer. Whenever we write a function that is case-specific and is not defined in any header file, we need to define and declare the function according to the syntax.

Ad. Changable function can be modified and improved as per need.

- The code of these functions is reusable in other programs.

Call by value vs call by reference

While calling a function, we pass a value of variables to it, such functions are called "call by value"	While calling the function instead of passing value of variable we pass the address or location of variables known as "call by reference"
In this method value of the each variable in the calling function is copied into corresponding dummy variable of the called function.	In this method address of the each and every variable in the calling functi. is copied in the dummy variable of the called function.
With this method, the changes made to the dummy variables in the calling func. have no effect on the values of actual variables in calling functions.	With this method, using addresses we would have access the actual variables and hence we would be able to manipulate then
In call-by value we can't alter the value of actual variables through the function calls	In this method we can alter the variable value through function calls
value of variable passed by simple technique.	Pointers variable are necessary to define to store the address value of variables.

→ Data types :-

Each variable in C has a associated data types. It specify the type of data that the variable can store like, integer, floating, char, etc. Each data type requires different ammount of memory and has some specific operations which can be performed over it.

→ classification of data types :-

---

① Primitive data types :- The data types that are most basic datatypes that are used for requesting simple values such as integer, float char etc.

② Int - (Integer data type).

The integer data type in C is used to store the integer numbers (positive, negative) : octal, values hexdecimal values and decimal values can be store in d- int data type in C. ((4 bytes) %d)

③ Character Data tyes (char)

Character data types allows it's variable to store only a single character. The size of char data type is 1 byte it is the most basic data type in C.

④ Float data types :- In C programming float data types is used to store floating - point value. Float is used to store decimal and exponential values.

(iv) void Data type

The void data type in C is used to specify that no value is present. It does no provide a result value to its caller. It has no values and no operations, It is used to represent nothing.

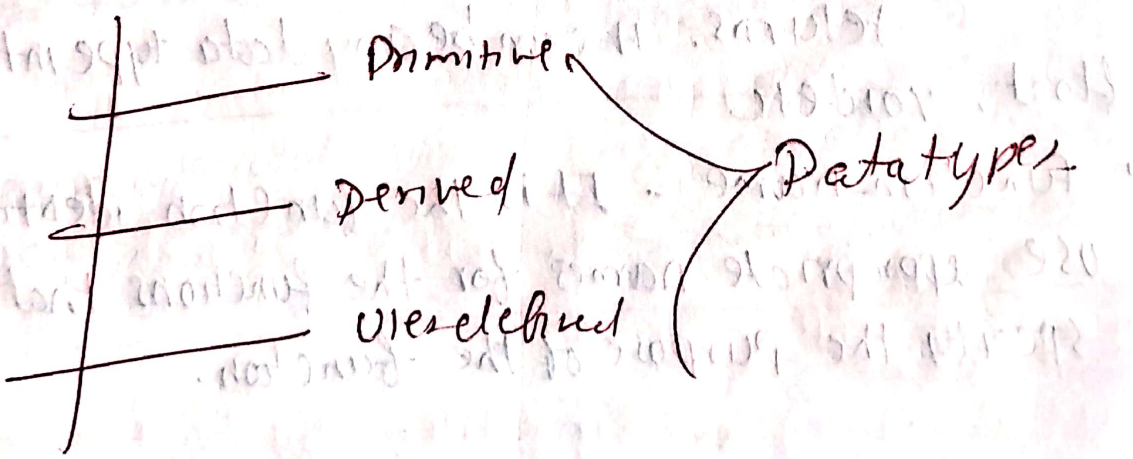
(v) Double Data type.

A Double data type in C is used to store decimal number (number with floating values) with double precision. It is used to define numeric values which hold numbers with numeric values in C.

(2) Derived datatypes:- The data types that are derived from the primitive or built-in datatypes are reffered to Derived data types. Ex; array, pointers, functions

(3) User defined datatypes:- The user-defined data types are defined by the user himself.

Ex, Structure, Union, enum.



Function prototypes in C

The C function prototype is a statement that tells the compiler about the function name, its return type, numbers & data types of its parameters. By using this information, the compiler cross checks function parameters and their data type with function definition and function call.

Function prototype works like a function declaration where it is necessary where the function reference or call is prerest before the function definition but optional if the function definition is present before the function call in the program.

Syntax

```
return_type
Function_name ( Parameter_list);
```

where

- return_type. It is the data types that function returns. It can be any data type int, float, void etc
- Function_name - It is the function identifier use appropriate names for the functions that specify the purpose of the function.

- Prototype_list :- It is the list of parameters Par- that a function expects in parentheses a parameter consist of data types and name.

Argument v/s Parameters

when the function is called, the values that are passed during the call are called as arguments.	The values which are defined at the time of the function prototype or definition of the function are called as parameters
These are used in function call statement to send values from calling function to the receiving function.	These are used in function headers of the called function. to receive the value from the argument.
During the time of call each argument is always assigned to the parameter in the function definition.	Parameters are local variables which are assigned values of the argument when the function is called
They are also called actual parameters.	They are also called formatted parameters

Strings in C :-

A string in C programming is a sequence of characters terminated with null character '\0'. The C string is ~~that the~~ stored as an array of characters. The difference between array and a C string is that the string in C is terminated with a unique character '\0'.

C string Declaration syntax :-

Declaring a string in C is as simple as declaring a one-dimensional array. The basic syntax of declearing a string is given below

```
Char string_name[size];
```

In the above syntax string_name is any name given to the string variable and size is used to define the length of the string, i.e the number of characters strings will store.

C string initialization :-

A string in C can be initialized in different ~~ways. We will~~

① Assigning a string literal without size

string in c can be assigned without size. Here, the string is given a name str acts as a pointer because it is an array.

```
char str[] = "Hello world";
```

② Assigning a string literal with a predefined size.

string literals can be assigned with a predefined size. But we should always account for one extra space which will be assigned to null character.

```
char str[50] = "Hello world";
```

③ Assigning character by character with size.

```
char str[14] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

④ Assigning character by character without size.

:- Unions :-

The Union is a user-defined data types in C language that can contain elements of the different data types just like structure. But unlike structures, all the members in C union are stored in the same memory location. Due to this, only one member can store data at the given instances.

:- Syntax of union in C :-

The syntax of the union in C can be divided into three steps which are as follow)

- C union declaration.
- Defining a union.
- Access union member.
- Initialization of Union in C

① C union declearation :- In this part, we only declare the template of the union, i.e, we only declare the member's names and data types set along with the name of the union. No memory is allocated to the union declaration.

```
union unionname{
  datatype member1;
  datatype member2;
  };
```

② We need to define a variable of the Union type to start using Union members. There are two methods using which can define a union variable.

① With Union declaration.

② ~~Without~~ After Union declaration.

① Defining Union variable with declaration.

```
Union Unionname{
    datatype member1;
    ,,  2;
,,
} var1, var2, ...;
```

③ Initialization of Union in C.

The initialization of a union is the initialization of its members by simply assigning the value to it.

```
var1.member1 = some value.
```

⊖ # Only one member can contain some value at a given instance of time.

Union V/S Structure	
The size of the union is the size of its largest member.	The size of the structure is equal to or greater than the total size of the all of it's member.
The only one member can contain data at the same time.	The structure can contain data in multiple members at the same time
It is declare using the structe keyword.	It's declared using the union keyword.

:- Pointers :-

A pointer is a variable that stores the memory address of another variable. Instead of holding a direct value, it holds the address where the value is stored in the memory. There are 2 important operators that we are used in pointers concepts i.e.

① Dereferencing operator (*) used to declare pointer variable and access the

value stored in the address.

② Address operator (&) used to return the address of the variable or access the address of a variable to a pointer.

:- Types of Pointer :-

① Integer pointers :- These are the pointer that points the integer values. The pointer are pronounced as Pointer to integer.

② Array pointer :-
Pointer and array are closly related to each other. Even the array name is pointer to it's first element. They are also known as pointer to Arrays.

③ structure pointer :-
The pointer pointing the structure type is called structure pointer. or pointer to structure.

④ function variable pointers :-
function pointers point the functions. They are different form the rest of the pointers in the sense that instead of pointing to the data, they point to the code.

⑤ Double pointer :- In c language, we can declare define a pointer that stores the memory address of another pointer such pointers are called double pointer.

Compiler V/S Interpreter

① The compiler saves the machine language in form of machine code on disks.	The interpreter does not save the machine language
② Compiled codes run faster than the interpreter	Interpreter code run slower than the compiler.
③ Linking - loading model is the basic working model of compiler	The interpretation model is the basic working model of the Interpreter.
④ The compiler generate the output in the form of (.exe).	The interpreter does not generate any output.
⑤ The Any change in the source program after the compilation requires recompiling the entire code.	Any change in the source program during the translation does not require translation of entire program
⑥ Errors are displayed in compiler after compiling together at the current time.	Errors are displayed in every single line.