Being learning 'C'

Kapil Gupta
D-1 Batch
31/08/2020

History of C- language

C- seems to be strange name for programing Language but now it is the most popular language. bcoz it is a structured middle level - machine independent language.

→ The root of all Modern language is ALGOL that is introduced in 1960s. ALGOL gives the ~~which~~ Concept of structured prog. language for CS Community.

→ In 1967 Martin Richard developed a lang. called BCPL primarily for writting software. then In 1970 A new lang. was introduced 'B' using many features ~~by~~ of BCPL by Ken Thompson.

→ Then 'C' was evolved from ALGOL, BCPL & B by DENNIS RICHIE @ Bell laboratory in 1972.

C ~~was~~ has many concept of these languages And added the concept of datatypes and other powerful features.

Since 'C' was developed along with Unix- Operating system it strongly associated with unix operating system.

Advantages of C- language

→ It is case sensitive, middle level and structured programing language.

→ It combines the features of High-level & low-level language.

→ It is highly portable and can be used for scripting system applications.

→ 'C' is structured programming language that allows a complex program to broken into smaller one which called function.

Machine Language

→ Collection of Binary digits or Bits that the computer reads and interprets.

→ It doesn't need any kind of language translator

→ Machine language is the only language that a computer is capable of understanding.

Assembly Language

→ It is a low level programing language. prog. written in it is compiled by an assembler.

→ It requires assembler to convert assembly lang. code to machine language. Assembler is an example of system software.

→ Every assembler has it's own assembly language which is designed for one specific computer architecture.

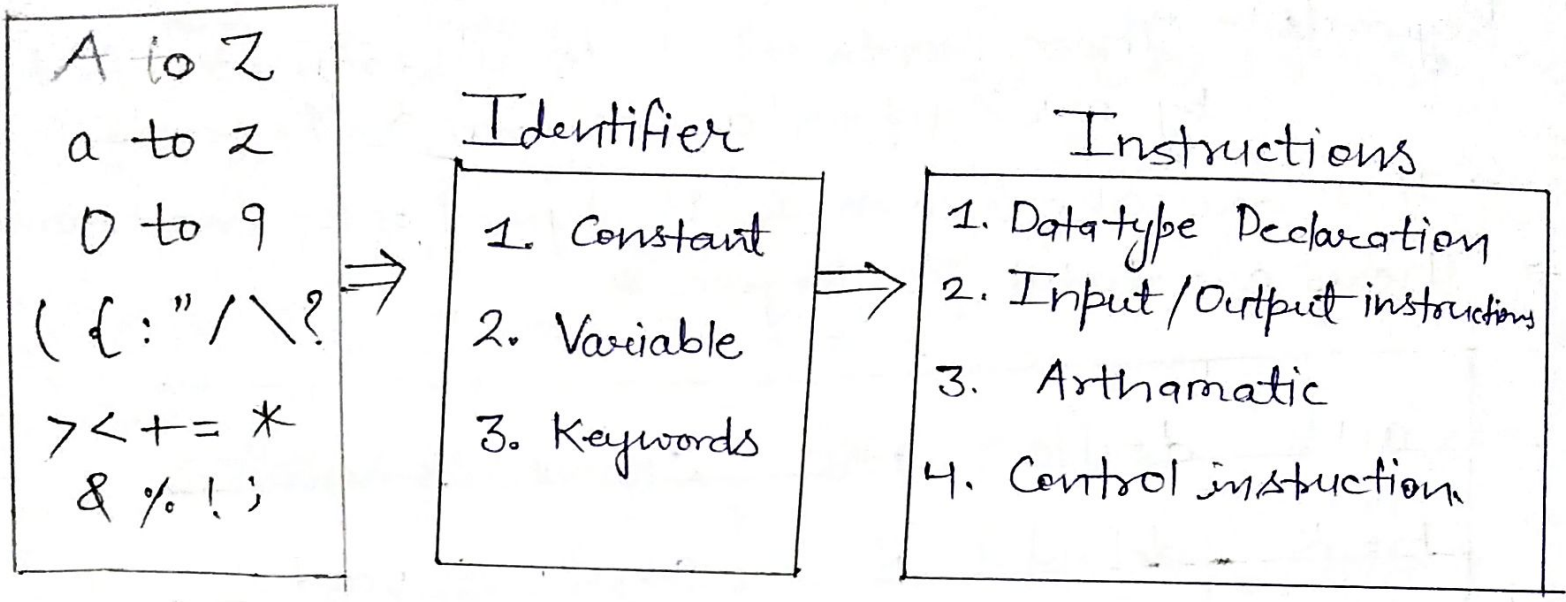
High-level Language

→ It's code is almost written in natural language. It is more like human lang. less like machine language.

→ It requires compiler to convert high level language code into simpler one.

→ It is a computer programming language that is limited by computer designed for a specific job and is easier to understood.

Being learning 'C'



Constant :- Any information which doesn't change it's value is constant.

Data = information = Constant (Same Thing)

⇒ Two catagories are there

Primary
- Integer (Ex:- 25, -26)
- Real (3.14, 0.25, 2.0)
- Character ('a', 'b' '4')

Secondary
- Array
- String
- Pointer
- Union
- Structure
- Enumerator

Variables:- It refers to unfixed data which can change it's value further. Variables are the name of memory locations where we store data.

→ It must contain only alphabate, digits or underscore.

→ Variable's name's first letter can't start with digit.

Identifier :→ Smallest identifying unit in program which make a complete scene.

Keywords :- Those words which is already identified by our compiler are known as keywords.

→ These are also known as Predefined or Reserved words.

→ There are total 32 Keywords.

auto	double	goto	signed	unsigned
break	default	if	sizeof	void
Case	enum	int	static	volatile
char	else	long	struct	while
Continue	extern	register	switch	do
Const	for	return	typedef	float
Short	Union			

Note:- ① Write these keywords only in small letter Otherwise compiler wouldn't identify them.

② We can't used to these 32 Keywords as variable.

Instructions :- Program statement are called instruction. Instruction are command for program.

- → Data type Declaration instruction.
- → Input - Output instruction.
- → Arithmetic instruction.
- → Control instruction.

There is NO BODMAS rule in 'c' language.

Operators :-

- Unary Operator
- Arithmetic Operators
- Bitwise operator
- Relational Operator
- Logical Operator
- Conditional Operator
- Assignment Operator.

① Unary Operators :- The operator which needs only one operand to work are known as unary operator.

Similarly
i) Binary Operator :- Which need two operand.
ii) Ternary :- which need three operand.

Ex :- '+' '-' '++' '--' sizeof().
↑ ↑ ↑ ↑
Positive Negative increament Decreament

← These all are Unary Operator.

Pre & Post increament :- They both work same but Decrement they have difference in priority.
Post increament have least priority lesser than Assignment operator also.

n++ (Post increment) ⇒ when operator is used later or after the variable / Const.

++n (Pre increment) ⇒ when operator is used ahead of that.

Ex ① #include<stdio.h>
#include<conio.h>
void main()
{ int n = 2;
clrscr();
n++; // n = n+1
Printf("%d", n);
getch(); }

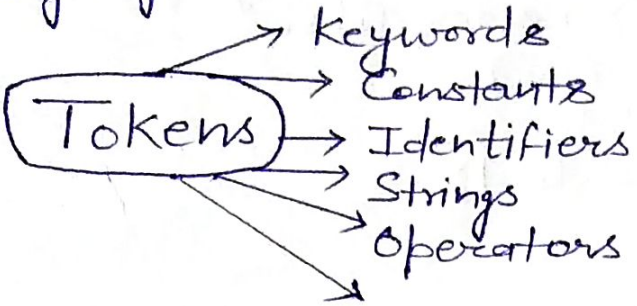
Output ⇒ 3

Ex ② Void main()
{ int n = 2;
clrscr();
n++; // n = n+1 ← Post
Printf("%d", n);
++n // n = n+1 ← Pre.
Printf("%d", n);
getch();

Output 45

Tokens in C

The individual words or smallest units in C-language are known as C-Tokens.



Format Specifier ⇒

It is a ways to tell the compiler what type of data is in a variable during taking input Scanf() or printing using printf().

Some examples are %.c - for characters etc.
%.d - for integers
%.f - for float

Q Explain Limitations of getchar and Scanf.

Ans = ① Scanf()

Scanf with %.S Or %WS can only read string without whitespace.
i.e. they can't be used for reading a text containing more than one word.

```
Scanf("%.S", name);
```

② getchar()

It used to read a single character from terminal
We can use this function to repeatedly read succeive single character from input and place them into a character array.
Thus entire line of text can be read and stored in an array. The reading is terminated when newline character ('\n') is entered and null character is then inserted @ end of string.

```
char ch
ch = getchar();
```

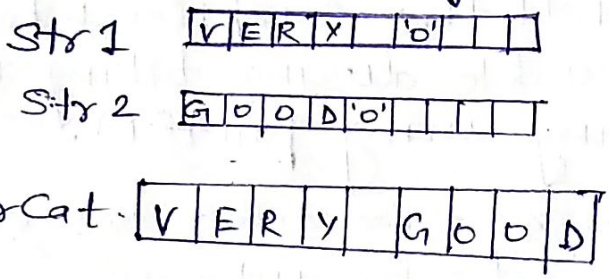
③ gets()

It is used for reading a string of text containing whitespaces is to use the library function gets available in <Stdio.h> header file.

```
char line [80]
gets (line)
Printf ("%s", line);
```

④ StrCat :→

This function joins two strings together when function strcat is executed string ② is appended to string ①.



⑤ Strcmp

This function compresses two string identified by the arguments and has a value 0 if they are equal.

```
Strcmp(name 1/name 2);
```

⑥ Strcpy :- This function is used to copy the content of one string to another string

```
Strcpy (Str 1, Str 2)
```

It assign the contents of Str 2 to Str 1.

⑦ Strlen :- The function counts and returns the no. of character in a string.

It takes the form `n= Strlen (Str);`

Headerfiles

Headerfile is a file with extension .h which contains defination of any predefined function which will be used in our program.

Ex :- <Stdio.h> → Standard input/output

Printf(); Scanf();

<Conio.h> → Console input/Output

Clrscr(); getch();

Algorithm

Basically it is step by step sol^{n} of any given problem.

Algo. has a definite begning with definite end with a definite no. of steps.

It can be written in any natural no. like Hindi Or English & it is used to solve complex program according to human readability.

Ex: add two numbers :-

Step 1 input A
Step 2 input B
3 Read A & B
4 C = A+B
5 Write C

DATA TYPE :- These are used to define a variable before to use in a Program. These are defined as data storage format that a variable can store a data to Perform a specific notation.

Primitive Data type :- Those data-types which are keywords are are known as Primitive data types.

- int
- char
- float
- double
- void.

⇒ There may be some more userdefined data types also.

Printf() :- Printf() is a predefined function, It is not any keyword.

→ It should by written only in small letter.

⇒ Two type of Messages

- Printing text as it is.
- Printing value of Expression or value of variable.

→ It is predefined output function.

Scanf() :- It is a predefined Input function

Scanf ("Format Specifier", variable adress);

Arithmetic Instructions :- An instruction which is used to manipulate data using operators, is known as Arithmetic instruction.

Ex:- 2+3

operands / Data

Operator

Sizeof() Operator⇒ Sizeof() operator tell us about the size or memory taken by data types, Variables, Constants.

Use of () is not necessary we simply can use space also.

Ex:-
```
main()
{ int x,a,b,c,d;
  float k;
  Double d;
  char c;
  Clrscr();
a= Sizeof(x); // 2 = int
b= Sizeof(k); // 4 = float
c = Sizeof(d); // 8 = Float
d = Sizeof('c'); /// 1 = char
Printf("%d%d%d%d", a,b,c,d);
   getch(); }
```

→ Output of the following Program will be :-

2 4 8 1

(2) **Arithmetic Operators:-** There are '5' Operator under it.

* / % ← Here Multiply & divison are samely prior.

\+ − ← lesser prior than above three.

⇒ Since there is no BODMAS Rule so associativity rule is followed from left to Right.

(3) **Bitwise Operator** :- There are '6' Operator.

Bitwise AND	&
" OR	|
" XOR	^
" NOT	~
Right Shift	>>
Left Shift	<<

& Operator				
0	&	0	=	0
0	&	1	=	0
1	&	0	=	0
1	&	1	=	1

Ex:- 7 & 11.

Change them into Binary

7 = 0111

11 = 1011

7&11 ⇒ 0011 ⇒ 3

| Operator (Or Operator)

0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1

Ex:- 7 | 11

7 = 0000000111
11 = 0000000001011

7 | 11 = 0000000001111 = 15

XOR operator (^)

0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0

Ex 7 ^ 11

7 = 0111
11 = 1011

7 ^ 11 = 1100 ⇒ 12

Right Shift (>>)

Ex:- 7 >> 2 ⇒ It means we have to shift two binary digit from right & add-up two zeros from left.

7 ⇒ 0111

7 >> 2 ⇒ 0001 (11) ← Here two binary digits are vanished.

7 >> 2 ⇒ 0001

similiarly with left shift. with change in direction.

(4) Relational Operator :- Total 6

< > <= >= ← 4 Are More prior

== . != ← less prior.

Note:- Relational Operator always yield result either 0 or 1.

→ Here 1 means true or 0 means false.

Logical Operator :-

① NOT - !
② AND - &&
③ OR - ||

Note:→ NOT operator (!) is also a unary operator. (!) Operator inverts truth value.

Control Instructions :-

① → Desicion Control instruction.
② → Iterative Control instruction.
③ → Switch case control instruction.
④ → goto control instruction.

① Desicion Control Instruction :- Also known as selection Control operator.

→ if
→ if-else
→ Conditional operator (?:)

Example :- // Write a program to check no. is positive or not.

If

```
main()
{ int n;
  clrscr();
  Printf("Enter any number");
  scanf("%d", &n);
  if (n>0)
  { Printf("No. is (+)ve"); }
  if (n<=0)
  {Printf("No. is non-positive");
   getch(); }
```

If-else

```
main()
 { int n;
   clrscr();
 Printf ("Enter any number");
  scanf (" %d", &n)
  if (n>0)
  { Printf (" Positive Number"); }
  else
  { Printf ("Non-(+)ve"); }
   getch();
```

→ If there's only one statement in 'if-Block' then we don't need curly braces {} there same with else.
→ If may be alone, if-else may together but not else alone.
→ Never use semi-colon(;) after with if or else.

Flowcharts

It is graphical representation of a given prog.
Here we use following kinds of symbol.

① Start and Stop 

② Input & Output 

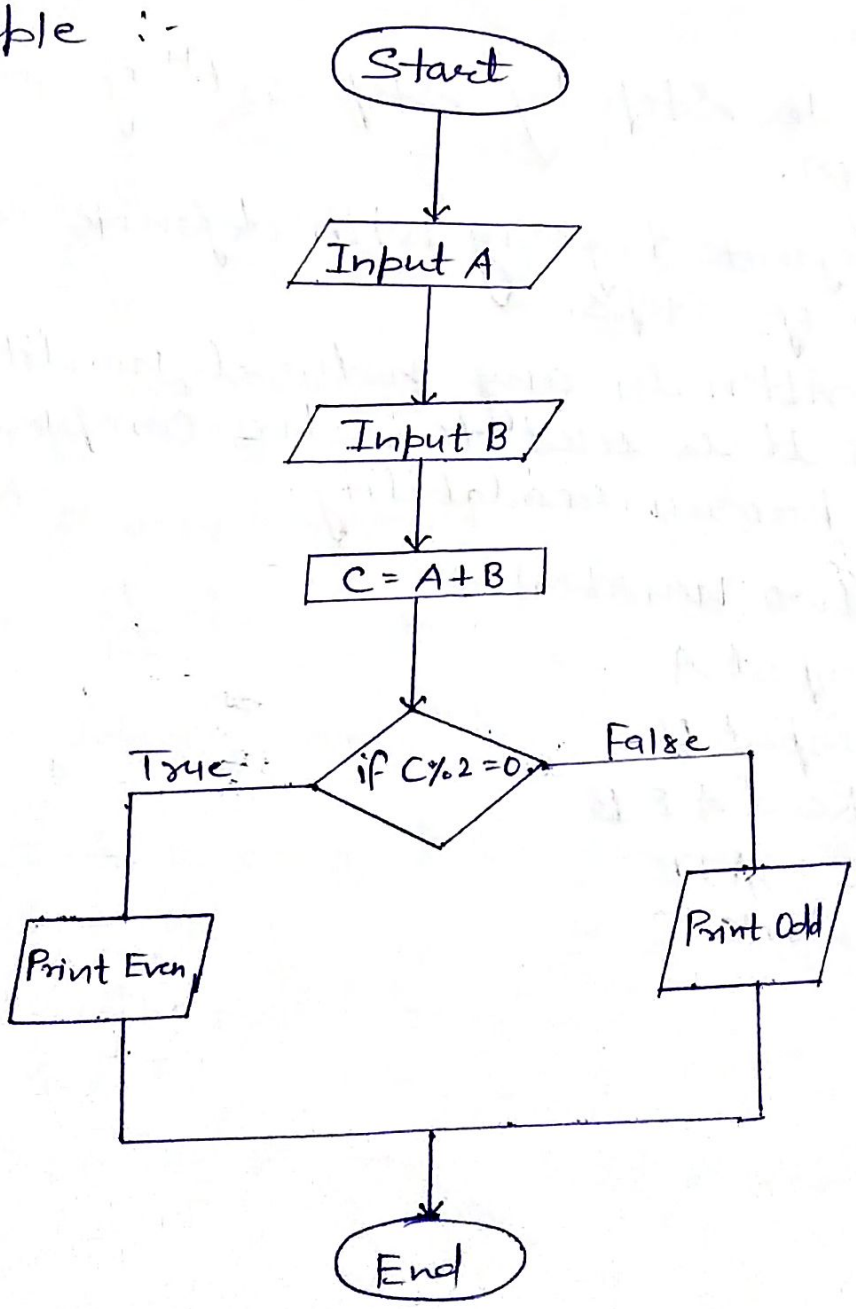
③ Process 

④ Condition 

⑤ Flowline 

⑥ Connector (o)→ for Connecting o

Example :-



GoTo STATEMENT

The goto statement is a jump statement which is something also reffered to as unconditional jump statement.

Goto Statement can be used to jump from anywhere to anywhere within a function.

Ex:-

```
# include <Stdio.h>
# include < Conio.h>
int main ()
{
Printf ("Hello World \n");
goto Label;
Printf ("How are you ?");
Printf ("Are you okay ?");
Label :
        Printf ("Hope you are fine");
getch ();
}
```

Jumped

Skipped

⑨ Conditional / Ternary Operator:- (?:)

Syntax:- `Condition ? Statement : Statement ;`

Ex:-
```
main()
{int a;
 clrscr();
 Printf("Enter any no. to check!");
 Scanf("%d", &a);
 a>0 ? Printf("Positive") : Printf("Negative");
 getch(); }
```

→ Conditional Operator also uses as selective operator

LOOP in 'C' Language

They also known as iterative control instructions.
Loop is used to repeatation of a set of instructions.

3 Types → While
→ Do-while
→ For

Ex:-
```
main()
① { int i=1;
  clrscr();
  while(i<=5) //(Condition to print my name 5 times)
  { Printf("My name is Kapil Gupta");
    i++
  }
  getch(); }
```

②
```
main()
{ int i=1;
 do
 { Printf("My name is kapil");
   i++
 } while (i<=5);
   getch();
 }
```

③
```
main()
{ int i;
for (i=1; i<=5; i++)
 { Printf("My name is kapil");
 }
getch();
}
```

LOOP :- It is a sequence of instructions that is continiously repeated until a certain condition is reached.

① **While**

→ While is a top tested loop.

→ It is an entry controlled loop statement.

```
While (test condition)
{
 Body of the loop
}
```

The condition is evaluated and if the condition is true then body of the loop is executed. This process repeated again and again the condition execute of the body continues untill the test condition finally becomes false and control is transfered out of the loop.

② DO STATEMENT

In while loop it may not be executed at all if the condition is not satisfied at very first attempt.
On some occassions it might be necessary to execute the body of loop before the test is performed.

→ Such Conditions can be handled with 'do' statement.

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

③ FOR STATEMENT

The for loop is another entry-controlled that provides a more concise loop control structure.

```
for (initiation; test-Cond"; increment/Deremment)
{
body of loop
}
```

break keyword :- The breake keyword is used to terminate loop's execution immediately as it encounters.

→ It can be use in loop as well as switch body.

→ It always written in small letter.

→ Normally break is used just after if statement.

Ex:-
```
main()
{int i=1, n;
 clrscr();
 while(i<=7)
 { Printf("Enter any number");
  Scanf("%d", &n);
   if (n>0)
       break;
    i++;
 }
if i==8 ? Printf("Normally loop ends"): Printf("break is applied");
getch();
}
```

Switch-Case Statements :-

Syntax

```
Switch(Expression)
{ Case Constant : Code; break;
  Case Constant : Code; break;
    :       :       :    :
  default : Code;
}
```

Ex:- Write a Menu-driven program with following options:-

1. Additions
2. Odd-Even
3. Printing First n-natural numbers.

```
main()
{ int n;
 clrscr();
 Printf("Enter 1 for Sum, 2 for odd-Even, 3 for Natural Number");
 Scanf("%d", &n);
 Switch(n)
 { Case 1:
       Printf("Enter two no. two add");
       Scanf("%d%d", &a, &b);
       Printf("The Sum is %d", a+b);
        break;
   Case 2:
       Printf"(Enter a no. to check odd-Even");
       Scanf("%d", &a);
       if (a%2==0)
            Printf("The no. is Even");
        else
            Printf("No. is Odd");
        break;
    Case 3:
       Printf("Enter limit of n-numbers");
       Scanf("%d", &a)
         for(i=1; i<=a; i++)
         Printf("%d", i);
       break;
     default:
        Printf("Invalid Choice of n"); }
        getch();
       }
```

FUNCTION IN C

Function is piece of code to accomplish certain operations.
It has a name for identification.

- Predefined functions
- User defined functions.

Q 2 Explain declaration of function?

Ans 2 Like variables, all functions in C program must be declared, before they are invoked. A function declaration consists of four parts:-

1. Function type (Return type)
2. Function Name
3. Parameter list
4. Terminating semicolon.

They are coded in the following format:-

< function-type > < function-name > (Parameter list);

1. Return type → A function may return a value. The "return-type" is a datatype of the value the function return.

eg- int < fun.-name > (Parameter list) /* function-prototype */

2. Function-Name →
This is the actual name of function. The function name and the parameter list together constitute the function signature.

3. Parameter →
A parameter is like a place-holders. These value is referred to as actual parameter or arguments. The parameter refers to the order, type and no of the parameter of a function.

Q = 3 Define Structure? With example?

Ans 3 1. C supports a constructed datatype known as structures.
- A structures is a convient tool for handling a group of logically related data items.

2. To create structure we use "struct" keyword.
- In structure, memory is allocated to each member individually.

- No share of memory, each member has its own
- Individual member can be accessed at a time.
- All member can be initialized while declaring the variable of structure.

e.g.
```
struct student
{
int no;
int age;
float Marks;
};
```

DECLARING STRUCTURE: 1. The keyword struct
2. The structure tag name
3. List of variables names seperated by comma
4. A terminating semicolon.

for exa - the stat.
struct book bank, books, book<2;

Q: DEFINE UNION WITH EXAMPLE?

Ans: Union follows the same syntax as structures.
- To create union we use "union" keyword.
- The union memory is allocated to only the largest member of union only.
- The union memory allocated to only the largest member of union only.
- Memory shared by individual member of union
- Only one member can accessed at a time.
- Only 1st member can be initialized while declaring the variable of union.

Exa -
```
union student
{
int rno;
float Marks;
float age;
};
```

ARRAY

Q=5 ARRAY :- IN DETAIL ?

Advantages → ① An array can be used to represent a list of nos, or a list of names.

② Array can be used to handle the large volume of data in terms of reading, processing and printing.

③ Advantage is it can store a fixed size sequential collection of elements of the same types.

④ It is more useful to think of an array as a collection of the same type.

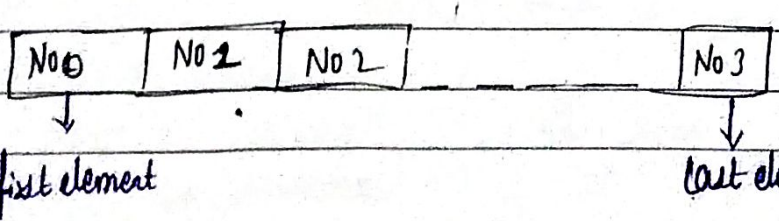
⑤ Array has continue allocation of memory at compile time.

⑥ Every single data item can be access by their index no. It's index no begin from 0-(n-1).

Defination → ARRAY is a kind of data structure that can store a fixed size sequential collection of elements of same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type

Instead of declaring individual variables, such as no.0, no.1, you declare one array variable such as nos and use no1, no2, ... no-n to represent individual variables. A specific element in an array is accessed by an index.

All array consists of continuous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Following types of array:-

1. One-dimensional array
2. Two dimen —
3. Multidimensional array

→ One dimensional array :⇒ A list of items can be given one variable name using only one subscript and such a variable is called one-dimensional array.

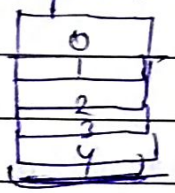
The subscript can begin with no. 0. That is:

x[0]

We may declare the variable no. as follows

int no[5];

The compiler reserve five storage as shown:-



Declaration:- Like any other variable, arrays must be declared after they are used so that are compiler can allocate space for them in memory.

The general form of array declaration is:

type variable-name[size];

The type specie to datatype such as int, float etc.

eg float height[50].

Initialisation:-

After an declared, its elements must be initialised. Otherwise, they will contain "garbage".

An can be initialised at following stages:

1. At Compiler time
2. At run time.
3. Compiler time initialisation :-

The general form of initialisation of arrays is:-

type array-name[size] = { list of values };

The values in the list are seprated by commas ;

for eg-

```
int NO[3] = {0,1,5};
```

(2) Run time initialization → An array can be explicity initialized at run time. This approach is usually applied for initializing large arrays. for eg :-

```
for(i=0; i<=100; i++)
{
if (i<50)
 sum[i] = 0.0;
 else
 sum[i] = 1.0;
 }
```

2-D-ARRAY

2D array is a group of rows and columns. It is also known as Matrix.

C allows us to define such tables of items by using 2D array. The table discussed above can be defined in C as :-

```
V[4][3]
```

In mathematics, we represent a particular value in a matrix by using 2 subscripts as V_{ij}.

V_{ij} refers to value in the i^{th} rows and j^{th} column.

declaration :-

2-D array declaration is as follows :-

```
type array-Name [row size] [Column size]
```

Initialization :-

Like 1-D array, 2-D array may be initialized by following their declaration with a list of initial values enclosed in braces.

for eg. `int table[2][3] = {0,1,1,0,1,1};`

Dynamic Array → We create array at run time.

Page No. Date

* Dynamic Array :- Static Array -

We created array at Compile time. An array created at Compile time by specific size in the source code has a fixed size and cannot be modified at run time. The process of allocating memory at Compile time is known as static memory allocation and the arrays that receive static memory allocation are called ~~dynamic~~ / static array.

function Malloc(), calloc() and realloc() are included in <Alloc.h> headerfiles.

① Malloc() :-

① Malloc() function is used for allocating block of memory at runtime. This function reserves a block of memory of given size and returns a pointer of type void. This means that we can assign it to any type of pointer using typecasting. If it fails to locate enough space it returns a NULL pointer.

② • It takes a single argument. that is No of bytes.

③ • It does not initialize the memory allocated.

Eg→
```
int *x;
x = (int*) Malloc(50*size of int);
```

4 • Malloc is much faster than Calloc.

5 • Malloc is stand for Memory allocation.

6 •

Calloc() :=

- The name Calloc stand for Contiguous/Continuous allocation
- Calloc takes 2 argument those are: No of blocks and ~~No of~~ size of each block.
- A Continuous block of memory.
- Calloc takes little longer than Malloc because of extra steps of initializing the allocated memory by zero.

int *Calloc(50*size of int);

Character ARRAY And STRING

quotation marks is a string constant.

Q. **STRING IN DETAIL ?**

① A string is sequence of character known as string.

② A string is terminated by null character \0.

③ Any group of characters defined b/w double

eg → "I C string tutorial".

Here "I C string tutorial" is a string. A string when, compiler encounters strings, it appends a null character \0 at the end of string

C		S	T	R	I	N	G		T	U	T	O	R	I	A	L	\0

Read/write strings :-

One possible way to read in a string is by using scanf. However the problem with this - - -

② gets() ③ getchar. (Refers to P.N - 238,242)

declaration/initialisation :-

C allows us to represent string as characters array.

The general form of declaration of a string variable is.

Char string name [size];

eg- Char string1[10];

Char city[40] = "NEW YORK".

✱ STRING HANDLING FUNCTION :-

Fortunately, the C-library supports a large no of string H. F. that can be used to carry out many of the string manipu. discussed so far.

Following are the most commonly used string handling functions

	function	Action
①	strlen()	Cal. the length of string.
②	strcpy()	Copies a string to another string
③	strcat()	Joins to string
④	strcmp	Compares two strings
⑤	strlwr	Converts string to lowercase.
⑥	strupr	Converts string to uppercase.

Page No. Date

Q-1 What is function, its types and elements ?

function :-

(1) A function is a group of statements that together perform a particular task. Or self-contained block of code

A function definition in C-programming consistst of a header and function body.

element related to the function:

1. function declaration / Prototype
2. Calling of fun
3. defination of function.

(1) PROTOTYPE :- The declaration part of function.

Syntax →

Void < functionname > (datatype argument) ;

Eg → Void function swap (datatype argument)

(2) CALLING OF FUNCTION :-

While creating a C-function, you give a defination of what the function has to do.

To use function you have to call that function to perform desired task.

When a program calls a function the program is controlled transfered ~~called~~ to the called function.

To call a function, you simply need to pass the required parameter along with the function name and if function return a value then you can store the returned value then you can store the returned value.

(3) DEFINATION OF FUNCTION :-

When we define body of function below the main function is known as defination of function.

Syntax - Void (datatype formal argument)

{

Qty

}

Q 2. Explain declaration of function?

Ans 2 like Variables, all functions in C program must be declared before they are invoked. A function declaration consists of four parts:-

1. Function type (Return type)
2. Function Name
3. Parameter list
4. Terminating Semicolon.

They are coded in the following format:-

< function-type > < function-name > (Parameter list);

① Return type → A function may return a value. the "return-type" is a datatype of the value the function return.

Eg- int < fun.-name > (Parameter list) /* function-prototype */

② Function-Name →

This is the actual-name of function. The function name and the parameter list together constitute the function signature.

③ Parameter →

A parameter is like a place-holders. Their value is referred to as actual parameter or arguments. The parameter refers to the order, type and no of the parameter of a function.

Q = 3 Define Structure? with example?

Ans 3 1. C supports a constructed datatype known as structures.
- A structures is a convient tool for handling a group of logically related data items.

2. To create structure we use "struct" keyword.
- In structure, memory is allocated to each member individually.

POINTER

* Like any variable or Constant, you must declare a pointer before using it to store any variable address.
The general form of a pointer variable declaration is —
< data Type * var_name; >

Here type is the pointer's base type; it must be a valid C data type and variable name is the name of the pointer variable.
The asterisk (*) used to declare a pointer is the same asterisk for multiplication.

Initialization / Declaration →

Since pointer variables Contains addresses that belong to the seperate data type.

```
datatype *Pt_name;
```

Eg- int *P; /* integer pointer */

```
int = quantity ;
int *P ;
P = &quantity ;
```

Q.6 Define file Modes And file handling ?

Ans 6 (1) A file represents a sequence of bytes, regardless it being a text of file.

(2) Filename is a string of characters that makes up a valid filename for the operating system.

(3) Data structure of a file is defined as file in the library of standerd I/O function definitions. So all file should be declared as type FILE before they are used. FILE is a defined datatype.

① Opening of file:-

The general format for declaring and opening a datatype.

```
FILE *fp;
fp= fopen("filename","mode");
```

The first statement declares the variable fp as pointer type "FILE". As stated The second statement Opens the the file named filename. and assigned an identifier to FILE type pointer fp.

Mode Can be One of the following:

(1) r : Open the file for reading only.

(2) w : Open the file for writing only. It does not ~~Open the file for adding data to it.~~ exits then a new file is created. Here your program will start writing your content from the begging of file.

(3) a = Opens a text file for appending data to it. If it does not exits, then a new file is created. Here your program will start appending content in the existing file content.

(4) r+ → Opens the text file for both reading and writing

(5) w+ → Same as w except both for reading & writing.

(6) a+ → Same as a except both for rea. & writing.

② Closing of file:-

To close a file fclose() function is used. The prototype of this function is—

The I/O library supports a function, it takes following forms—

```
fclose(file_pointer);
```

This would close the file associated with the file pointer file_pointer. Look at the following segment of a program:

```
FILE *P1, *P2;
P1 = fopen("input", "w");
P2 = fopen("output", "r");
fclose(P1);
fclose(P2);
```

This program opens the two file and close them after all operations on them completed.

(3) Input/output operations on files :-

Following is the simplest function to write or reading a single character to the file :-

→ the getc and putc functions →

The simplest file I/O functions are getc and putc. these are analogous to getchar and putchar functions and handle one character at a time.

Assume that a file is opened with mode w and ~~filename~~ file pointer fp2. Then the statement -

```
putc(c, fp2);
```

Putc → It take one character at a time and display on output and write into file.

(2) getc →

Similarly getc is used to read a character from a file that has been opened in read mode. for example, the statement -

```
c = getc(fp2);
```

It would read a character from the file whose file pointer is fp2.

The file pointer moves by one character positions for every operation of getc or putc. The getc will return an end-of-file marker EOF, when end of the file has been reached. Therefore the reading should be terminated when EOF is encountered.

Extra Gyan

Classfriend
Page No. Date

(6) ERROR HANDLING DURING I/O OPERATIONS :-

If we fail to check such read and write errors in a program may behave abnormally when an error occurs. An unchecked error may result in a premature termination of the program or incorrect output.

Fortunately we have to 2 status-inquiry library functions feof and ferror that can help us detect I/O errors in the files.

The feof function can be used to test for an end of file condition. It takes a file pointer as its argument and returns a non zero integer value if all the data from the specified file has been read and return 0 otherwise.

If fp is a pointer to file that has just been opened for reading, then the statement —

```
if(feof(fp))
printf("end of data.\n");
```

The ferror function reports the status of the file, indicated. It also takes the file-pointer as its argument. And returns a nonzero integer if an error has been detected upto that point, during processing. It returns 0 otherwise.

The statement —

```
if(error(fp)!=0)
Printf("An error has occured"\n");
```

* RANDOM ACCESS TO FILES :-

If we are interested in accessing only a particular part of file and not in reading the other parts, this can be achived with the help of functions fseek, ftell and rewind available in the I/O library.

ftell →

ftell takes a file pointer and return of type long, that corresponds to the current position. This function is useful to saving the current position of a file, which can be used later in the program. It takes following form —

```
n= ftell(fp);
```

[Codingninjas.in] HTML, CSS, Java.

n would give the relative offset (in bytes) of the current position. it means n bytes already been read.

② Rewind :- Rewind takes a file pointer and resets the position to the start of the file. for eg. the statement :

```
rewind(fp);
n = ftell(fp);
```

③ fseek — fseek function is used to move the file position to a desired location within the file. It takes the following form —

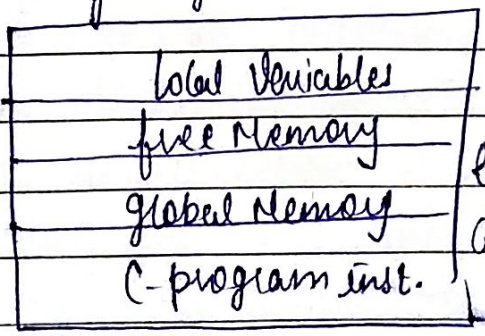
```
fseek(file ptr, offset, position);
```

④ ftell :- ftell takes a pointer and returns a no of type

Q.1 what do you mean by direct memory allocation process.?

Ans.1 The program instruction and global and static variables are stored in a region known as permanent storage area and the local variables are stored in another area called stack. The memory space that is located b/w these two regions is available for dynamic allocation during execution of the program.

The free memory region is called the heap.



The size of heap keeps changing when the prog. is executed due to creation and death of variables that are local to functions and blocks.

Therefore, it is possible to encounter memory "overflow" during dynamic allocation process. so it will return a null pointer.

Q-2 Define Realloc() and free in ~~dynamic~~ Memory allocation

Ans 2 Realloc() →

If previously allocated memory is not sufficient. And we need additional memory space for more element.

But it is also possible that memory allocated is much larger than necessary and we want to reduce it.

In both case, we can change the memory size already allocated with the help of the function realloc.

```
realloc (pointer_name, number * sizeof (int));
```

② FREE →

- free() function frees the allocated memory for malloc(), calloc(), realloc() function and returns the memory to the system.

```
free (pointer_name);
```

Q-1 Diff actual and formal argument?

Actual argu.	formal argu.
∘ The arg. which is used in function call only is known as actual arg.	The formal arg. is used in prototype as well as in defination of fun. known as fa. We can change the name of variable in def. of function
2. To send the arg. we can use numeric constant or variable	
eg- void add(int a, int b); void Main() { int a, b; printf("enter value"); scanf(—) add(a,b) ← }	eg ① void add(int a, int b); { void add(int x, int y); { printf(—) } }

Q2 define operator precedence?

Ans Ans first we should know about an expression, as an expression is a group of operand and operators

- ∘ Precedence in C is the rule that specifies the order in which certain operations needs to be performed in an expression.
- ∘ For a given expression containing more than two operators it determines which operation should be calculated first.
- ∘ Evaluation of an expression start from left to right.
- ∘ As an expression can be evaluate on the bases of pre-defined rules..

High priority → *, /, %.

Less priority → +, −.

for an eg- C= a*b/c. (here * has high priority then it will solve first then after b/c)

Q-3 Explain what is programming domain in c?

Ans 3. It means for which purpose we are developing any appli.

using different kinds of prog language.

(6) As far as C-language is concerned it is used in low level to high-level prog. language.

→ Some of programming domains are listed below —

- General purpose application
- Expert system
- Artificial intelligence system
- Gaming - application
- ERP - Programming application
- Text processing
- Inventory management
- Financial Management
- Educational application
- Network application.

Q-4 What is pseudocode?

Ans 1 It can be defined as a combination of algorithm and programming constructs.

P.C ⇒ alg + Prog. const

2 (a) Pseudocode is not actual programming lang. it is used to write code for programs before you are actually create it in a specific language.

(b) Once you know that the prog. is what about and how it will function, then you can use pseudo code to create statements to acheive result of your program.

3. (a) Pseudo code cannot be compiled nor executed and there are no real formatting or syntax rules.

4 (a) The benefit of P.C is that it enables the programmers to concentrate on the algorithm without worrying about all the syntactic details of a particular prog. lang.