"C"

C language :- 'C' language is a ~~structured~~ procedural programming language and developed by Dennis Ritchi in 19 72 B.C.. It is a mixture of Algol, BCPL and B languages.

Its history :-

Year	Language	Founder
1960	Algol	International community
1967	BCPL (Basic Computer Programming Language)	- Martin Ritchard
1970	B	Ken Thamsan
1972 at Bell labs	C	Dennis Ritchi
1978	"The C Programming language" K & R	Kanidham and Ritchi
1989	ANSI C	ANSI community
1990	ANSI C/ISO C	ISO (International Standard organisation)

Importance -

- Efficient and high speed
- Portable - not forced by any O/S or any machine
- Structured Language
- Wide Acceptable.

Used for writing O/S as well as application level programm

Hello guys

welcome to our new channel

C language =>
C is a procedural programming language, developed by Dennis Ritchie in 1972 at Bell laboratory. It can be used to develop software like operating system, data-bases, compilers, and so on.

Its importance

*1- Efficient and high speed.
*2- Portable- not forced by any operating system or any machine.
*3- Structured language.
*4- Wide acceptable
*5- used for writing operating system as well as application level language.

Compiler- It is a translator which is used to translate the c language into machine
language(binary language).
ex- turbo c, visual c, borland c etc.

Tokens(basic unit) in C language =>

*1-**Keywords=>**These are the reserved words in C which is not used as user define word. These are used to write statement or instruction. There are 32 keywords in C.
Ex-char, int, if, for, void, else, float etc.

*2-**Identifiers**=>It is a name given to variable,function or class by user.

Rules to give the name of identifier=>

*1-There can be use of alphabet, digits and underscore(_).
*2-Don't started with digit.
*3-Don't use white space while naming.
*4-There is difference of lowercase and uppercase (case sensitive).
ex- ram and Ram is different for C language as it is a case sensitive language.
*5-Don't use the name of keyword.

*3-**String=>**It is continuous group of character.
ex-Ram, Raj, Anoop and so on.

*4-**Constant**=>Constant means such a value which does not change. It is such a word or value which does not change in whole program. It is declared by word "const".

*5-**Operators**=>Operators are the signs which is used to work with data such as solvivg numerical problems and others.
Ex- +, -, *, /, %, && etc.

Types of operators=>

***1-Arithmetic operator =>**

	operator name	symbol	example
(i)-	Addition	+	a+b
(ii)-	Substraction	-	a-b
(iii)-	Multiplication	*(asterisk)	a*b
(iv)-	Division	/	a/b,5/2=2
(v)-	Module	%	a%b,5%2=1

***2-Relational operator**=>

operator symbol	meaning	example
<	is less than	a<b
>	is greater than	a>b
<=	is less than or equal to	a<=b
>=	is greater than or equal to	a>=b
==	is equal to	a==b
!=	is not equal to	a!=b

***3-Logical operator=>**

operator symbol	meaning/name	example
&&	logical and	a>5&&a<99
||	logical or	a>5||a<7
!	logical not	a=!5

***4-Assignment operator=>**

(i)- simple assignment operator=> The operator is used to substitute a value into a variable is known as simple assignment operator. its symbol is "=".
ex- a=5,b=7

(ii)- shorthand assignment operator=> The format of shorthand assignment operator is "variable(operator)=expression".
It is also known as compound assignment operator.

```
ex- a+=5  => a=a+5
    b-=3  => b=b-3
    a*=3  => a=a*5
    a/=d  => a=a/d
    a%=d  => a=a%d
```

***5- Conditional or Ternary operator**=>
It is a combination of "?" and ":". It is used with 3 operand. Its syntax is-
variable=(expression)?true_value:false_value;

ex- If a and b are two numbers and we want to store a greater value in x, for it-

```
it is written in if statement-
    if(a>b)
     x=a;
    else
     x=b;

it is written with ternary operator-
    x =(a>b)?a:b;
```

***6=>Increment and Decrement operator=>**

++ :It is known as increment operator. It increses the value of an operand by 1.

```
ex- x=5;
    x++;
    x=6;
```

-- : it is known as decrement operator. It decreses the value of an operand by 1.

```
ex- x=4;
    x--;
     x=3;
```

These operator can be used by two types-

(1)-Pre-increment and Pre-decrement operator-

In this,we use the operator before an operand. As- ++a,--b;

```
  ex-  x=5;
       y=++x;
       y=6;
similarly,
       a=4;
       b=--a;
       b=3;
```

(2)-Post-increment and post-decrement operator-

In this,we use the operator after the operand. As- x++; y--;

```
  ex-   x=5;
        y=x++;
        y=5 and then x=x+1=6;
similarly,
        x=3;
        y=x--;
        y=3 and then x=2;
```

***7- size-of operator=>**

ex- sizeof(int);

Comment=>Comments allows others to better understand the code.

single line comment- //

more than one line- /*____________*/

file extension- .c

compile- alt+F9

```
run-ctrl+F9
```

Data types in C

Data types is used to store and process the information. To define the type of data that a variable can store

Types of data type
- Built-in / Primary data type (Pre-defined
- Derived data type (maked by using Primary d
- User defined data type

Derived data type:
- Array ex- int X[5];
- Function ex- main( )
- Pointer ex- int *p;

Built-in / Primary data type:
- char (1 bytes)
- int (2 bytes)
- float (4 bytes)
- double (8 bytes)

User defined data type: how to use
- Structure struct ex- struct stu
- Union union ex- union emp
- Enumeration enum ← Its takes integer value
  - ↳ ex- enum weekday

```
{
  sun, mon, tue, wed, thus, fri, sat
   0    1    2    3    4     5    6
}
```

Formatting Symbol -	Format Specifier
%c	char
%d	int
%i	long
%f	float
%s	string

ASCII values:-

1)	A to Z	65 to 90	ex- A=65, B=66, Y=89, Z=90
2)	a to z	97 to 122	ex- a=97, b=98, y=121
3)	0 to 9	48 to 57	ex- 0=48, 1=49, 2=50

4) Special character, 60 to 64, 91 to 96, 30 to 45

Escape Sequences -

\n	for new line
\t	for tabs or white space
\a	for sound ,,
\\"	for " ,,
\" "	for
\?	for ?
\\	for printing the \

Input/ Output functions in C :

In C language we have 3 below types of I/O functions -

1) Formatted I/O Function
The function that have fixed format for I/O is known as formatted I/O function.
=> pring () , scanf()

2) String I/O Function.
=> puts () , gets()

3) Character I/O Function.
=> putch (), getch(), getchar() getche()

A-Print f() :
- Printf stands for print format.
- It is used to display the instruction

Syntanx:

```
Printf ("List of specifiers", variable name);
Ex- printf (" %d, %f ", x , y);
Pring (" Enter a number: %f ", variable name);
Printf (" String text ");
```

B-Scanf():
- Stands for scan format
- used for taken input given by user or program

Syntax:
scanf(" List of specifiers", List of address of variable

Ex- scanf ("%d %f", &num, & mun);

3)- puts(): **send the cursor in new line**
- stands for put string
- used to print the string.

Syntax: puts ("String value"/Variable);
Ex- puts ("Enter a no.");
puts (name); //name is variable

4) gets():
- stands for get string
- used to read the string
- we can not read a string with white space by using scanf() but we can read a string with white space by using gets() function

Syntax: gets (String Variable);
gets (name);

5) putch():
- Stands for put character
- used to print a character

Syntax- putch(char Variable);
putch(x);

6) getch(): Ex- char x=getch();
- Stands for get character
- Can read any key including enter button.
- Does not display the entered character.

Syntax- ~~getch()~~; char getch();

Contral statements.
- If statements
- Switch statement

structure
```
Switch (va
{
case variable-value:
    statement;
break;
case variable-values:
    statement;
break;
default:
    statements;
}
```

- Simple if statement
- If — else statement
- Nested if statement
- If — else-if statement

structure
```
If (condition)
{
 Statement 1;
}
else
{Statement 2;
}
```

structure
```
If (condition)
{
statement 1;
statement 2;
}
```

structure
```
If (condition)
{
_ _ _ _
_ _ _ _
}
else if (condition)
{
_ _ _ _
_ _ _ _
}
else
{
_ _ _ _
}
```

structure
```
If (condition)
{
If (condition)
{
_ _ _ _
}
else
{
_ _ _ _
}
}
```

getchar()	getch()	getche()
display character	do not display character	display character
needed enter	do't needed enter	do't needed enter

rewrite
Loops: to revised the program

- entry control loop : which condition written in st
  - For loop
  - while loop
- exit control loop: whose condition written in en
  - Do while loop

While loop -

```
initialization;              // Declaration of initializati
   while (condition )        // Declatation of condition
{
   ______
   ______                    // can be declared inc. ord
   ______
}
```

For loop -

```
   for (initialization; expression ; increment order);
{
   ______
   ______
   ______
   ______
}
```

Note ';' is used between among all three statements

Do ——— while loop.

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

$\boxed{;}$ → Remind this one

<u>Continue, Break and Exit Commonds</u>

Break statement is used to exit the statement.

```
while ( condition ) :
{
   statement 1;
     break ;     >>>>>>>>
   statement 2;
}
   getch ();   <<<<<<<<<
```

Continue statement is used to starlib the statement again and again.

```
   a = 1
for (i = 1 ; i <= 10 ; i++) <<
{
   if ( i < 3  ||  i > 7)
        continue; >>>>>>>>>
   else a = a*i;
}
```

exit statement is used to exit from the program
It is a function of stdlib. (so use it in header file)

Ex-

```
while (i<5)
{
printf (" smaller than 5");

if (i= 3)
exit;
else
printf (" Not 3");
}
```

Arrays:- Arrays are used to store multiple value in a single variable instead of declaring seperate variable for each value.

~~Declaraction of Array~~:-

index/size of array

Data type array_name [size];

Ex-

int raj [10]; int age [5] = {16, 17, 18, 19, 20};

Types of Array -
- One dimensional array
- Two dimensional array
- Multi dimensional array

Initialization of array –
- With Declaration
- Inside the program

```
int age[5];

age[0] = 16;
age[1] = 18;
age[2] = 20;
age[3] = 21;
age[4] = 25;
```

```
int age[5] = {16, 18, 20, 21, 25}
```

Intialization of two-D array; –
- With Declaration
- Inside the program

```
int age[2][3];
age[0][0] = 1;
age[0][1] = 2;
age[0][2] = 3;
age[1][0] = 4;
age[1][1] = 5;
age[1][2] = 6;
```

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

String (Use string.h header file).

String is a collection of character. Strings are used for storing text.

Intialization of string –

1, char name[20] = "prateek";

2,
```
name[0] = 'p';
name[1] = 'r';
name[2] = 'a';
  .
name[7] = '\0';
```
$\longrightarrow$ Null character

Some String function -

(String.h) header file

- strcpy (Destination, source); - but deleted already written text in destination file
- strcat (S1, S2);
- strlen(variable_name);
- strcmp (S1, S2); - difference of ASCII values
- strcmpi (S1, S2);
- strrev (v_name);
- strlwr (v_n);
- strupr (v_n);

$\Big\}$ gives result in same variable

(ctype.h) header file

- isalnum (v_n); - check whether a character is alphanumeric or no
- isalpha (v_n);
- isdigit (v_n);
- islower (v_n);
- isupper (v_n);
- tolower (v_n);
- toupper (v_n);

(math.h) headerfile

- pow (x,y); $\Rightarrow x^y$
- sqrt (v_n);
- sin ( );
- cos ( );
- abs( );
- fabs( );

(stdlib.h) header file
- randomize();
- ran ( ) ; } Executes a random number.

Pointer! - Pointer is the memory adderess of a variable where the value of variable stores.

Declaration of Pointer! -

```
int * p;
char * p;
float * p;
```

Ex -

```
# include <stdio.h>
# include <conio.h>
void main ()
{
    int p = 5;
    int *t;
    t = &p;
                                        //output is as 06Xpy6.
printf ("The memory adderess of p is %d", t);
                              //Output is 5.
printf ("\n Thevalue of p is %d", *t);

getch();
}
```

Function:- A function is a block (set of instruction that can perform a specific task.

Types of function-
- Library or Predefined function
- User define function

Need of functions:-
To reduce the length of source code
To find the errors easily.
To call the function one or many time
To help make the program more understandable
To modularize the task of the program

Main points to create a function:

1. Function Declaration:
2. Function Definition
3. Function calling
   - Function name (arguments);
   - Ex- sum (20, 30)

Syntax:
- Return type Function name (Parameters);
- Ex- int sum (int x, int y);

Syntax:
```
Return type Function name (Parameters)
{
    //Statements

}
```

```
Ex- int sum (int x, int y)
    { int c;
      c= x+y;
      return c;
    }
```

Types of User define Function (UDF)

1, No return type and no parameters
2, Return type and no parameters
3, Return type and parameters
4, No return type and parameters

Types of Function Calling
- Call by value
- Call by Reference

__Call by value:__

Ex-

```
. #include <stdio.h>
  #include <conio.h> void swap(int a, intb);
void main()
{
  int a, b;
  a= 10;
  b = 20;
  swap (a, b);   // call by value
printf ("value of variable in main function %d%d", a,b);
  getch();
 }
void swap (int a, intb)
{
  int c
  c = a;
  a = b;
  b = c;
Printf (" value of variable in swap function %d%d", a,b);
 }
```

Output:

```
Value of variable in swap ~~main~~ function  20 10
Value of variable in main function  10,20.
```

Call by reference:

Ex-

```
# include <stdio.h>
# include <conio.h>
void swap ~~main~~ (int*x, int*y);
void main()
{
    int a=10, b=20;
    swap (&a, &b);   // call by reference
Printf (" value of variable in main function %d,%d", a,b);

}
void swap ( int*x, int*y);
{
    int c;
      c=*x;
    *x=*y;
    *y=c;
    printf ("value of variable in swap function %d,%d",*x,*y);
}
```

Output:

```
value of variable in swap function  20,10
value of variable in main function  20,10
```

Types of parameters / arguments:
- Actual parameter (at calling function)
- Formal parameter (at defining function)

Parameters: as variables

Ex- int sum (x, y)

$\quad\quad\quad\quad\quad\quad\quad\quad$ parameters

Argument: as constants.

Ex- int sum (20, 90)

$\quad\quad\quad\quad\quad\quad\quad\quad$ Arguments

Recursion: When a function calls itself, that is known as recursion.

Ex-
```
main()
{
  printf ("hi");
  main();
}
```

Output:
```
hi
hi
hi
|  |
|  |
```

Structure :- Structure is collection of different variables or datatypes. It is opposite of array because array is a collection of same variables

Syntax of defining structure -

```
struct structure_name
{
   data type variable1;
   data type variable2;
   .
   .
}
```

$\}\boxed{;}$ → Remember it

Syntax of declaring structure -

struct structure_name structure variable1, structure_variable2;

Ex-

```
    # include <stdio.h>
    # include <conio.h>
 Struct Rectangle
 {
    int length;
    int width;
 };
 void main()
 {
  struct Rectangle rect1;
  Struct Rectangle rect2;
  rect1. length = 12;
  rect1 width = 8;
  rect2. length = 5;
  rect2 width = 3;
```

```
Printf (" Rectangle 1: %d %d", rect1.length, rect1.width);
printf (" Rectangle2: %d %d", rect2.length, rect2.width);
getch();
}
```

Accessing structure Members

Syntax.

```
structure_variable. accessing members;
```

Ex- rect1 . length;   ,   rect 1 . c = 42;

↓

Dot operator.

Combining definition and declaring of structure

Syntax -

```
Struct
{
data type variable 1;
data type variable 2;


} structure_variable 1;
```

Ex -

```
Struct
{
 char name [20];
 int age;
 float salary;
} e1 , e2;
```

Initializing Structure members

Structure_name structure_variable = {value1, value2 ... };

Ex-

```
struct employee
{
  char name[20];
  int age;
  int salary;
};
employee e1 = {"Ram", 41, 35000};
employee e2 = {"Shyam", 32, 21000};
```

Nested Structure:-

Syntax-

```
struct structure_variable1
{
  datatype var1;
  datatype var2;
  :    :
};

struct structure_variable2
{
   data type var1;

   structure_variable1  structure variable;

};
```

Ex:-

```
Struct   employee
{
   char  name[20];
   int   age ;
   int   salary;
};

Struct company
{
   char company-name[20];
   employee   e1, e2 ;
};
```

Note- Struct is a keyword that is used to make a datatype in which we define that how many variable and what type variable we want to store. It is same as int, char, float etc The main difference among them is that they are already define but struct datatype is not already define

File- File is a place of one type on disk wher
related data is collected.

Types of File-
- Sequential File
- Random File

Operation related with file -

① Opening and closing the file.
② Reading and writing the file.
③ Manipulating in files
④ Searching in file.

Defining and opening a file |-

Syntax-
```
FILE * v.name (file pointer);

file_pointer = fopen("file_name", "mode");
```

For reading a file.	← r	reading mode
For writing in file / a file.	← w	writing mode
adding new data in old/new file	← a	append mode
Both reading and writing in old file	← r+	
Both reading and writing in new file	← w+	

Ex- ① File *fptr;
fptr = fopen ("demo1.c", "r");
② File * fopen (" demo1.c", "r");

Reading / Output from a file

- getc (); — reading a character
- gets (); — syntax- reading a string
- getw ();

reading a integer
getw(file pointer);

Syntax-
getc (file pointer);

gets (character array, size, file pointer);

read data store in it

how many characters you want to read

pointer / name given after the FILE

Writing / Input to a file

- putc ();
- puts ( );
- putw();

putc (variable, file pointer);

puts (entering string , file pointer);
Ex- puts ("hi", fptr);

putw (integer , file pointer);
Ex- putw (5, fptr);

Closing a file -
Syntax-

fclose (file_pointer);

Function-

① fscanf():-

Syntax-

fscanf(file pointer, "format specifier", adderess of variables);

Ex- fscanf (fptr, "%d %d %d", &a, &b, &c);

② fprintf():-

Syntax-

fprintf (file pointer, "format string ~~text~~", address of variables);

Ex- ~~fprintf (fptr, "hi bhai"~~,

fprintf(fptr, "%d %d %c", a, b, c);

③ Rewind():- This function is used to take the file pointer in the starting of file.

rewind (file_pointer);

④ ftell():- This function tells the no. of bytes used between from starting of file to current position of file

n = ftell(file pointer);
↓
variable

⑤ fseek():- This function is used to move the current position of file pointer to a valentary position.

fseek(file pointer, offset, position);

```
                        integer   integer
                                  |
                                  |-0    starting of file
                                  |-1    current position
                                  |-2    ending of file
                                  |
```

⑥ ferror():- This function is used to telling the errors in file

ferror(file-pointer);

It gives integer value if there is an error otherwise zero.