The C Programming language is a general purpose operating system agnostic & procedural language that structure & provide low level access to the system memory

It is generally used to create hardware device, OS, drivers games etc Dennis Ritchie developed C at Bell laboratory in 1972.

→ Advantages

- Simple to comprehend
- Basic Building Block
- Powerful & efficient language
- Rich in library functions
- Simple to write
- Portable
- Procedural Prog. language
- Dynamic memory allocation
- ability to increase user functions
- implementation of algorithm & data structure
- Simple to debug
- low cost
- Reserved words
- ability to extend itself
- Structured Prog. langua
- Middle level Prog. langu
- Fastest compilation
- System Programming
- Run time performance

→ Disadvantage

- lack of OOPs Concept
- Not effecient as Python, Java etc
- Only programmer can understand
- Difficult to learn
- It required Prog. skills
- Prog. skills required
- Run time checking
- lack of open source
- Difficult to memory rese words.
- Algorithm concept
- It doesn't fit to those who doesn't understand algorithm

```
int number 1, number 2, sum;
printf ("Enter 2 integers:)
Scanf ("%d%d", &no1 &no2);
  Calculating sum
sum= no1+no2
Printf ("%d+%d=%d", no1, no2, sum);
```

A Keyword in C has a pre-defined meaning whereas an identifier is just a name assigned by a user or programmer to a memory function or constant value. It plays an imp role they are basic building block of lang.

- Constant - It is an entity that doesn't change
- Variable - It is an entity that may change

KEYWORDS	IDENTIFIER
1 NO special symbols is used	1. Except (_) no special system is used.
2. Specifies type of entity	2 Gives name to type of entity
3. always in lcase	3. It can be in lcase & Ucase
4 It has pre-defined meaning they are reserved for use by the user.	4 It has no meaning attached to it
5. Eg - void, int, char	5. Eg - test, class-abc

Constant

- Primary constant
  - Integer
  - real
  - character
- Secondary constant
  - List, pointer etc

```
#include <stdio.h>
#include <conio.h>
 int main()
 {
  int A, B sum = 0;
  printf ("Enter two no, A&B, \n");
   Scanf ("%d %d", &A&B);
     sum= A+B
   Printf ("Sum of A&B is %d", sum);
    return 0 getch()
     }
```

- Terminology

1 Main() - It is a function which is a container for a set statements all the statements belong to main() enclosed within pair of {}.

2 Printf - All the statements inside Print function is the output to screen which is achieved by using readquads libra function. It needs to be displayed on screen need use printf. It is necessary to use #include <stdio.h> at the top of prog.

It can not only print values of variables it also print the result of an expression.

3 Scanf - It is used to take input by using the scanf by which memory allocation should be supplied to store the value by user from the keyboard using & address. A new tab or line must be separated by the values supplied to scanf by using '\n' '\t'

⇒ Types of Instructions - There are 3 types
1. Type Declaration
2 Arithmetic
3 Control

- Type Declaration - It is used to declare the type of variables which is used in program any variable used in the program must be declared before using it any statement.

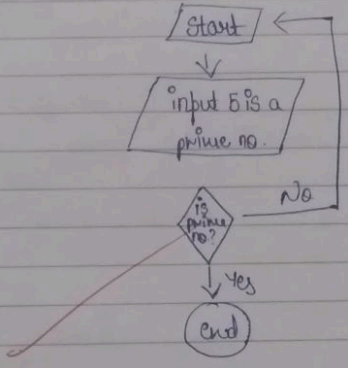
Eg int a, b;
char student, class;
float rupees, dollars

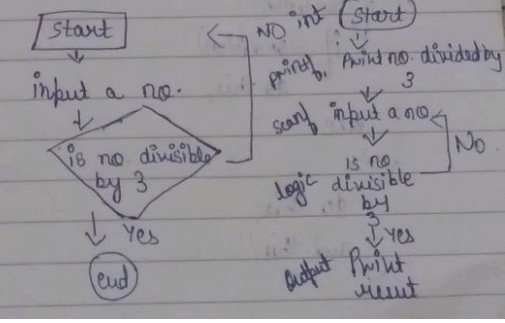
- Arithmetic - In C arithmetic instruction consists of a variable on LHS of '=' & variable names & constants on RHS. The variables & const appear on RHS of '=' are connected by arithmetic operation '+, -, ÷, X' etc

- Control instruction - It enables various instruction in a order while executing the program has 4 types

1 Sequence control instruction
2 Selection or decision control instruction
3 Repeatition or loop control instruction
4 Case Control instruction

Decision control statement
- if else statement
- if statement





*
0 is no. divisible by 3.

$\frac{n}{3} = 0$

TYPES OF IF STATEMENT

a)
```
if (condition)
    do this;
```

b)
```
if (condition)
{
    do this;
    and do this
}
```

c)
```
if (condition)
do this;
else
do this;
```

d)
```
if (condition)
{
    do this;
    and this;
}
else
{
    do this;
    and this;
}
```

e)
```
if (condition)
{ if (condition)
    do this;
else
}
    do this;
    and this;
else
    if (condition)
    do this;
    else
    {
        if (condition)
        do this,
        else
        {
            do this;
            and this;
```

```
#include <math.h>
#include<stdio.h>
#include<conio.h>
int main()
{
  int S1, S2, S3, S4, S5, Per;
  printf("Enter marks in 5 subjects");
  Scanf("%d%d%d%d%d", &S1, &S2, &S3, &S4, &S5);
  per = (S1+S2+S3+S4+S5) * 100/500
  if (per >= 60)
     printf("First Division\n");
  else
  { if (per >=50)
     printf("Second division\n");
  else
  { if per (>=40)
     printf("third division\n");
  else
    printf("fail\n");
  }
  {
   return 0;
  }
```

operators	Types
!	logical NOT
* / %	Arithmetic & modulus
+ -	Arithmetic
< > <= >=	Relational
== !=	Relational
&&	logical AND
||	logical OR
=	Assignment

? or ":" Conditional operator - The conditional operators are also known as ternary operators it is kind of forshortened if-then-else expression1 ? expression2 : expression3
It expresses that if expression1 is true i.e, (if its value non-zero) then the value returned will be expression2 otherwise the returned value will be expression 3

```
#include<stdio.h>
int main()
{
 int i=4,z=12,
 if (i=5 || z>50)
   printf("Dean of students affairs\n")
else
 printf("Dasa\n"),
 return 0,
}
```

```
#include<stdio.h>
 int main()
 {
  int x=4, y=40, z=45;
  if (x>y && x>z)
    printf("biggest = %d\n",x),
  else if (z>x && z>y)
     printf("biggest = %d\n",z);
   return 0,
  }
```

a=10
b=12
c=0

expression:-
a != 6 && b >5=
a==9||b<3=
!(a<10)=
!(a>5 &&c)=
5 &&c != 8||!c

```
#include<stdio.h>
int main()
{
  int a,b,
  printf("Enter two numbers to Add,Subtract,multiply
        & Divide\n")
  Scanf("%d%d", &a &b);
  printf("Sum= %d\n Subtract=%\n multiplication=%d\n
        Division=%d", a+b, a-b,a%b,a/b);
```

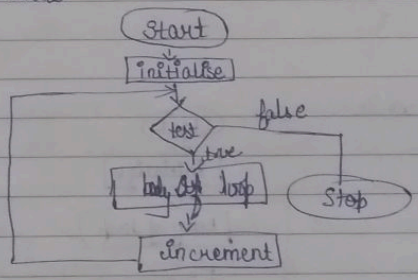
4.
```
# include <stdio.h>
int main()
{
 i, j;
 int evenodd (int);
 printf ("enter a number \n");
 scanf ("%d", &i);
 j = Evenodd (i);
 printf (j == 0 ? "odd" : "Even"
 return 0
}
 int Evenodd (p)
 {
 return (p % 2 == 0 ? 1 : 0);
 }
```

(2)
```
#include <stdio.h>
int main()
{
 int a, b, c, p;
 printf ("enter three numbers \n");
 scanf ("%d %d \n %d \n", &a, &b, &c);
 if (a > b && a > c)
{
 printf (" greatest is : %d", b);
}
 if (c > a && c > b)
{
printf (" greatest is %d", c);
}
```

Loop Control Instructions - The programs that we have developed so far used either sequential or decision control instruction. firstly it calculates the no. in fixed order secondly, an appropriate set of instructions were executed upon the outcome based on instruction given through the. The versatility of the computer lies in its ability to perform a set of instructions repeatedly it involves repeating some portion of the program either a specified no of times or until a particular condition is being satisfied. This repetitive operation is done through a loop control instruction There are 3 methods for loop by which we can repe
- For loop
- While loop
- do-while

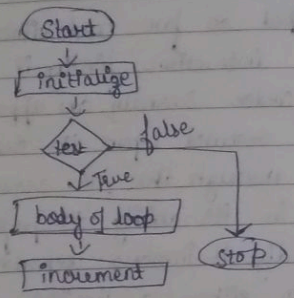
part of program using a for, while, do-while statement.

while loop



Syntax
```
initialise loop counter;
while (test loop counter using a condition)
{
   do this;
   and this;
   increment loop counter;
}
```

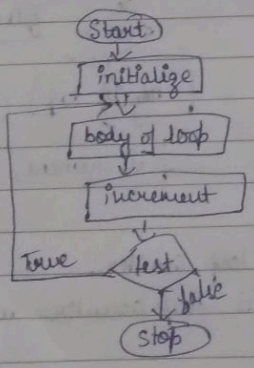
- For loop



Syntax

```
for (initialize counter; test counter; increment counter)
{
  do this
  and this;
  and this;
}
```

- Do while



Syntax

```
Do
{
  this;
  and this;
  and this;
}
while (this condition is true);
```

- The break statement It is a situation where we want to jump out of the loop instantly without waiting to get back to the condition. They keyword 'break' allows us to do this. When break is encountered inside any loop it controls automatically passing to the first statement after the loop. A break is usually associated with the if statements or 'if' keywords.

Continue Statement - In some programming situations if we want to take the control to begin the loop by each passing statement in the loop which is not been executed until now. The keyword continue allows us to do this. When 'continue' is encountered inside any loop it controls passes automatically to begin the loop. A continue is usually associated with an 'if'.

```
#include<stdio.h>
 int main()
 {
  int x, c, sum;
  for(x=1;x<=3;x++)
 {
  for(c=1;c<=2;c++)
 {
  sum=x+c;
  printf("x=%d c=%d \n",x,c,sum);
  }
 }
 }
```

* Keywords in C

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

- storage classes - It possesses a characteristics called storage class. It defines two different characteristics of variable lifetime & visibility. lifetime of a variable is the duration of time it retains a particular value, whereas visibility of a variable is also known as scope that refers to a part of a program make enable to recognise. If storage classes are important as it makes variable usability with the appropriate storage class where we can write the program which uses computer memory more efficiently, run faster & are less prone to programming errors. The most important role of storage class is that it is very useful for large programs. C language supports 4 storage classes :- auto, static, extern, register. The declaration statement of storage class is static int x; it tells the compiler that the storage class for variable x is static which signifies -

1. Variable x will be stored in the main memory of the computer.
2. Since variable x is not initialised, it will automatically will initialised with value zero.
3. Variable x will be visible to the block in which it is declared.
4. The variable x if declared in a function other than main function it will retain its value during diff. function calls.

Precedence & associativity of operators - Precedence is used to determine the order in which different operators are evaluated in an expression. Associativity is used to determine the order in which diff. operators with same precedence are evaluated in expression.

Eg: BODMAS

3+3*3÷3

Ans 6

Precedence is applied before associativity to determine the order in which expressions are evaluated. The concept of precedence is well defined by the BODMAS rule.

Write a program to convert temp from celcius to F by taking input from user.

```
#include<stdio.h>
#include<conio.h>
int main()
{
float celcius, fahrenheit;
printf("Enter temperature in celcius:\n");
scanf("%f", &celcius);
fahrenheit = (celcius * 9/5) + 32;
printf("Temperature in fahrenheit is %f\n", fahrenheit);
return 0
}
```

2. Write a program to find greatest no. among 3 no given by the user.

```
#include<stdio.h>
#include<conio.h>
int main()
{
float num1, num2, num3;
printf("Enter 3 no. :\n");
scanf("%f %f %f", &num1, &num2, &num3);
(num1 >= num2)
{
if(num1 >= num2)
{
if(num1 >= num3)
printf("The greatest number is: %2f", num1);
else
printf("The greatest number is: %f", num2);
else
printf("The greatest number is: %2f", num3);
}
return 0;
}
```

3 Write a program to check if given number is a prime number or not

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n, i, flag=0;
printf("Enter a positive integer");
scanf("%d", &n);
if(n==0||n==1)
flag=1;
for(i=2; i<=n/2; ++i)
{
if(n%i==0)
{
flag=1;
break
}
}
if(flag==0)
printf("%d is a prime no:", n);
else
printf("%d is not a prime no:", n);
return 0;
}
```

4
```
2 3
4 5 6
7 8 9 10
```
Write a program to display a following pattern up to n rows, taking the value of n from the user.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int rows, i, j, number=1;
printf("Enter the number of rows:");
scanf("%d", &rows);
for(i=1; i<=rows; i++)
{
for(j=1; j<=i; ++j)
{
printf("%d", number);
++number;
}
printf("\n");
}
return 0;
}
```

Standard Input / Output

When a C program begins execution it has access to following pre-defined files

-> Standard Input (stdin)
-> Standard Output (stdout)
-> Standard Error (stderr)

} Stdio

1. Standard Input - It is the file from which input is received this file is usually a keyboard but it can be redirected to obtain input from a disc file or other device

2. Standard Output - It is a file to which output is directed the file is usually a computer screen but it can also redirect to disc file or to a another device

3. Standard error - It is used to keep error message separate from programs output if a output is going to be a disc file then the standard error file can be a computer screen to make user for seeing the error messages coming from the output

- Standard Input / Output Functions

There are many library functions available for I/O these functions are divided into 2 categories

-> unformatted
-> formatted

There is only one difference in b/w these functions, formatted function allow input from a standard input device or sent output to the standard output device whereas unformatted I/O function performs input/output of one character at a time

* Character I/O function - For character inputs the function is getchar() which is available in all version of C language Turbo C provide 2 types of versions of character input functions in the form of getche() getch()
For character output the available function is putchar()

* String I/O function - For string input the function is gets() is used & for string output puts() is used

getche()	getch()
1. This functions return a character that has been recently typed	1. It also returns a character which has been recently typed
2. The typed character is been echoed on the output screen	2 Neither the user required to press Enter key after entering the character nor the typed character is being echoed to the monitor
3 The user is not required to press enter key after typing a character. The advantage of this function is that the user types the character & it is immediately accepted	3. This function has advantage over the other function in application where the user wants to hide a input

It reduces the user's action to single key.

gets()	puts()
• It accepts the string from the standard input device. The length of the string is limited by the declaration of the string variable. The input string may consist of multiple words typing Enter key code completes the input	The function outputs a string constant or a string variable to standard output device

Format Specifier	Used for
%d	Signed decimal integer
%ld	Signed long decimal integer
%u	Unsigned decimal integer
%f	Single precision floating real no.
%lf	double precision floating real no.
%e	floating point (exponential notation)
%x	Unsigned hexadecimal integer
%o	Unsigned octal integer
%c	Single character
%s	Single word string
%[^\n]s	multi word string

• do-while program

```
# include <stdio.h>
int main()
{
  while(4<1)
    printf("Hello there\n");
  return 0;
}
```

•

```
# include <stdio.h>
int main ()
{
  int i, j;
  for (i=1; i<=2; i++)
  {
   for (j=1; j<=2; j++)
   {
   if ( i == j)
     Continue;
   printf("%d %d \n", i, j);
   }
  }
  return 0;
}
```

```
#include <stdio.h>
int main()
{
 char another;
 int num;
 do
 {
 printf("Enter a number");
 scanf("%d", &num);
 printf("Square of %d is %d\n", num, num of num);
 printf("want to enter another number y\n");
 fflush(stdin);
 scanf("%c", &another);
 }
 while(another == 'y');
 return 0;
 }
```

- Associativity - It is applied when more than one operator of same precedency is used in an expression. It can be right to left or left to right. In left to right associativity it evaluates the expression to start on the left & moving to the right whereas right to left associativity evaluates the expression to start on the right & moving to the left

Eg: (12+5)+7) → (7+7) → 14

Symbol used	operation Performed	Precedence level
+	Addition	4
−	Subtraction	4
*	Multiplication	3
/	Division	3
%	Modulus	3

Associativity
L-2-R
L-2-R
L-2-R
L-2-R
L-2-R

Increment
i++
i--
++j
--j

There are two different operator in C language which can be supported by high level language It incorporated first time in C language these are known as special operators later it moves into the C++, Java and high level language that supports such kind of operator

- Increment operator adds one to the operand while decrement operator subtracts one from the operand both operators are unary operators which can used in

- Recursion- It is a process in which a function calls itself directly or indirectly is called recursion & the corresponding function which is known as a recursion function. By using a recursion algorithm there are certain problems which can be solved easily.

For eg:- factorial(5)

$= 5'4'3'2'1$

$3! = 3*2*1$

$5! = 5*4*3*2*1$

- There are three types of Recursions

1. Direct - It is typified by the factorial implementation where the method calls itself.

2. Mutual - It happens where one method say method a and another method say method b, then they decide whom to call

- Multi - It calls in this each recursive call has a value smaller non-negative value so that if the first call has non-negative value of last first then each calls including the first terminates

Recursion depends mainly on two types of function.

- A function calls itself from within itself.
- Or more than one function call each other mutually

- Call by value → Passes value
- Call by reference → passes address of var

⇒ Call by reference - It is the function whenever a calling function is required rather than passing the variable's values then it passes the variables address instead of function the variable has. It is thus a name that refers the references by calling the variables address of its function & also passes the value of its variables.

⇒ Call by value - While calling the function we passes the values of variables it is known as call by value.

- Pass by reference - It means passing the reference of an argument in calling function which is corresponding to the formal parameter of the call function. In this the call function has the capability to modify the arguments value by using the references it passes.

- Return by reference - It is very diff. from call by reference in this the function behaves a very important role when the pointers or variable are returned as reference.

- Malloc() & Calloc() — Malloc() function() creates a single block of memory of size specified by user whereas Calloc() function can assign multiple blocks of memory for a variable. Malloc() is used to allocate space whereas Calloc() is leaves path of data uninitialized & set the

unit fact zero Calloc() stands for contiguous allocation whereas Malloc() stands for Memory allocation

- Features of C
- Simple & efficient
- fast
- Portability
- extinsibility
- function rich libraries
- Dynamic memory mgnt
- Modularity with structured language
- Mid-level programming language

Write a prog. to input Marks of 50 students using array & display avg

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n,i;
float num[100],sum=0,avg;
printf("enter the number of students:");
scanf("%d",&n);
while(n>100||n<1)
{
printf("error! number should be in range of (1 to 100)
\n");
printf("Enter the number again ");
scanf("%d",&n);
for(i=0; i<n;++i)
{
printf("%d enter mark:" i+1);
Scanf("%f", &num[i]);
sum+=num[i]);
}
avg = sum/n;
printf("Average=%f", avg);
return 0;
}
```

- Write a program to search for no entered by user in a given array & diplay array in descending order

```
#include<stdio.h>
#include<conio.h>
void main()
{
int arr1[100]
int n,i,j,tmp;
printf("\n\n Sort elements of array in ascending order");
printf("\n ------ ");
printf("\n Enter the size of array: ");
Scanf("%d", &n);
printf("Enter %d elements in the array:\n", n);
for(i=0;i<n;i++)
{
for(j=i+1; j<n;j++)
{
```

Unit - III

- Strings - The way a group of integers can be stored in an integer array which can select a particular group of character that can exist inside an array when we store characters into the array it is called string

Char stud = {'A', 'B', 'C', 'D', 'E', '\0'};

```
#include <stdio.h>
int main()
{
 char stud[7] = "John";
 int i=0;
 while (i<=7)
 {
 printf ("%c", name[i]);
 i++;
 }
 printf ("\n");
 return 0;
}
```

```
# include <stdio.h>
 int main()
 {
 char name = "John";
 Char *ptr;
 ptr = name;
 while (*ptr != '\0')
 {
 printf ("%c", *ptr);
 ptr++;
 }
 print ("\n");
 return 0;
 }
```

- Functions

- Standard library string functions
Strlen - Finds length of string
Strlwr - Converts a string to lower case
Strupr upper case
Strcat - appends a string at the end of another
Strncat strcat appends first n characters of a string at end of another
Strcpy - copies a string into another
strncpy - copies first n characters of a string into another
Strcmp - Compare two strings
strncmp - Compare first n characters of two string
strcmpi - Compares two strings by ignoring a case
Stricmp - Compare two strings without regard to case
Strnicmp - Compares first n characters of two string without regard to case
strdup - duplicates a string
strchr - find first occurence of a given character in a string
strstr - find first occurence of string in another string
strset - sets all characters of a string to a given character
strnset - set first n
strrev - reverses a string

Initialization # user can initialize multiple members at same time

In case of union a user can initiate first member at a time.

- File handling - The output of a C program is generally deleted when program is closed sometimes we need to store output for several purposes such as data analysis, comparison of output for different situation, result presentation etc. The use of file handling is for situation exactly retrieve files to manage to recover & to restore or reevaluate. Therefore we need file handling

- Features of File Handling
- Reusability
- Portability
- efficient
- Storage capacity.

- Reusability - The file handling process to keep the track of information for creating after the program that has been ruuned.

- Portability - Without losing any data files can be transfer to another computer system the risk of flawed coding is minimised with this feature.

- Efficient - A large amount of input may be required for some programs. file handling allows user to easily access a part of code using individual commands to save a lot of time & reduces the chance of errors

- Storage Capacity - Files allow user to store data without having to worry about storing everything simultaneously in a program

Files

Text Binary (0-1)

24/11/23

- Structures & Unions - Structures is a way to group several related variables into a place in this each variable is known as structure of member unlike an array the structure can contain different type of data such as int, float, string, character etc for structures we use 'struct' keyword followed by tagname of the structure in this the body of structure is defined in which respective data members either primitive or user defined datatypes are added.

- String with pointer

```
#include <stdio.h>
int main()
char name() {
```

→ Difference between Structures & Union - It is a type of user defined data it is similar to structure but it combines various objects of different source & uses together It determines a new data type that has more than 1 member for the intended program

- Functions & similarities using Structures & Union
- They are both user defined data types to store different sorts of data together as a single unit
- Both of their members can be one of any types of object It may include different structures & unions or arrays where even its members can also contain a bitfield
A union or a structure can easily pass by value to function and also return to a value by functions

In this every argument must possess the same parameter of function parameter

- A union or structure pass by value similar to any scalar variable in form of their corresponding parameter

– Diff b/w structure & Union

Parameter	STRUCTURE	UNION
Keyword	A user can deploy the keyword struct to define a structure	A user can deploy keyword union to define a union
Internal Implementation	The implementation of structure in C++ internally they contains separate memory location allotted by every input members	In this case of union the memory allocation occurs for only 1 member with the largest size among all the input variables it shares the same location all these members/objects
Accessing Members	A user can access individual members at a given time	A user can access only 1 member at a given time
Size	A structure does not have a shared location for all of its members it makes the size of structure to be greater than or equal to the sum of the size of its data structure	When you union does not have a separate location for every member in it it makes its size equal to the size of largest member among all the data members
Value alternate	Altering the value of single member does not affect the value of other member structure	When you alter value of single member it affects the values of other members
Storage Value	In this case of a structure there is specific memory location for every input data member thus it can store multiple value of various members	There is an allocation of only shared memory for all the input data members Thus, it stores 1 value at a time for all of its members