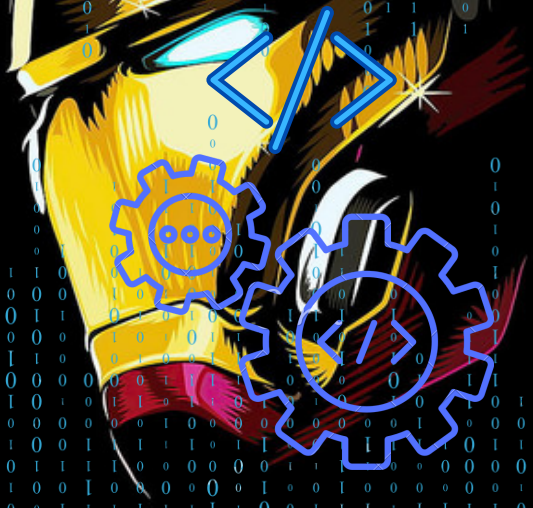
2023

C

BASIC

CYBER TRON

C-PROGRAMMING



(FIRST PART)

T.I.M. HAMIM

Index:

There are 32 keywords in C which are given bellow . Keywords. Keywords are all lowercase.

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

Escape Sequences :

Escape Sequences	Character
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\?	Question mark
\0	Null character

1. printf()

printf is a function which work is to print/show the output of the program.

Hole	Description	Example	Result
%d, %i	Print any integer	printf "%d" 5	5
%x, %o	Print any integer in Hex or Octal format	printf "%x" 255	ff
%s	Print any string	printf "%s" "ABC"	ABC
%f	Print any floating-point number	printf "%f" 1.1M	1.100000
%c	Print any character	printf "%c" '\097'	a
%b	Print any Boolean	printf "%b" false	False
%O	Print any object	printf "%0" (1, 2)	(1, 2)
%A	Print anything	printf "%A" (1, [])	(1, [])

- **Make a program that will print your name.**

```
#include<stdio.h>
int main()
{
  printf("I am Hamim");
  return 0;
}
```

Here,
Statement = printf("I am Hamim"),
Library function = #include<stdio.h>
Header file = stdio.h,
Extention = .h

- **Make a program that will show sum of 50 and 60.**

```
#include<stdio.h>
int main()
{
  int a;
  int b;
  int sum;
  a = 50;
  b = 60;
  sum = a+b;
  printf("Sum is %d",sum);
  return 0;
}
```

Here,
Variable = a,b,sum,
Data type = int (for integer number) = %d,
float (for fraction number) = %f,
double (for real number) = %lf,
or,

```
#include<stdio.h>
int main()
{
  int a=50;
  int b=60;
  int sum;
  sum = a+b;
  printf("Sum is %d",sum);
  return 0;
}
```

or,

```
#include<stdio.h>
int main()
{
  int a=50,b=60,sum;
  sum = a+b;
  printf("Sum is %d",sum);
  return 0;
}
```

or,

```
#include<stdio.h>
int main()
{
 int a=50,b=60;
 printf("Sum is %d",a+b);
 return 0;
}
```

or,

```
#include<stdio.h>
int main()
{
  int a=50,b=60;
  printf("%d+%d=%d",a,b,a+b);
  return 0;
}
```

- **What will be the output of the program given bellow?**

```
#include<stdio.h>
int main()
{
  int x,y;
  x=1;
  y=x;
  x=2;
  printf("%d",y);
  return 0;
}
```

Here,
The output will be 1. Now think about why it happend ? The reason is y=x is not a equation in program and here we are just assigning the value of y variable.

- **Make a program that will show sum of 50.45 and 60.**

```
#include<stdio.h>
int main()
{
  int a=50.45,b=60;
  printf("%d+%d=%d",a,b,a+b);
  return 0;
}
```

Here,
The output will be 110. Now think about why it happend ? Because a is an integer number and we have assigned 50.45. But C compiler take it as 50 using type cast.

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int a=277;
  printf("a = %d\n",a);
  printf("a = %5d\n",a);
  printf("a = %05d\n",a);
  printf("a = %5d\n",a);
  printf("a = %05.5d\n\n",a);

  float b=277.1, c=277, d=277.10, e=277.0;
  printf("b = %f\n",b);
  printf("b = %09.3f\n",b);
  printf("b = %9.3f\n",b);
  printf("b = %9f\n",b);
  printf("c = %f\n",c);
  printf("c = %09.3f\n",c);
  printf("c = %9f\n",c);
  printf("d = %09.3f\n",d);
  printf("e = %9f\n",e);

  return 0;
}
```

Here,
The output will be :

a = 277
a =   277
a = 277
a =   277
a = 00277

b = 277.100006
b = 00277.100
b =   277.100
b = 277.100006
c = 277.000000
c = 00277.000
c = 277.000000
d = 00277.100
e = 277.000000

- **Range of data type :**

Type	size	Range	Format
unsigned char	1 byte	0 to 255	%c
signed char or char	1 byte	-128 to +127	%c
unsigned int	2 byte	0 to 65535	%u
signed int or int	2 byte	-32,768 to +32767	%d/%i
unsigned short int	2 byte	0 to 65535	%u
signed short int or short int	2 byte	-32,768 to +32767	%d/%i
unsigned long int	4 byte	0 to +4,294,967,295	
signed long int or long int	4 byte	-2,147,483,648 to +2,147,483,647	
long double	10 byte	3.4E-4932 to 1.1E+4932	%ld
double	8 byte	1.7E-308 to 1.7E+308	%lf
float	4 byte	3.4E-38 to 3.4E+38	%f

Here,
1 byte = 8 bit,
1 bit = 0 and 1,
2 byte = 16 bit = 2^32,
signed = +value/-value,
unsigned = +value.

- **Make a program that will show sum of two real number 4.9 and 7.3.**

```
# include<stdio.h>
int main()
{
  double a=4.9, b=7.3, sum;
  sum=a+b;
  printf("Sum is %lf\n",sum);
  printf("Sum is %.1f\n",sum);
  printf("Sum is %.2lf\n",sum);
  printf("Sum is %.3lf\n",sum);
  printf("Sum is %.4lf\n",sum);
  printf("Sum is %.5lf\n",sum);
  printf("Sum is %.9lf\n",sum);
  return 0;
}
```

Here,

Think about why I have written so many printf() statement.

- **Make a double number into integer number using type cast.**

```
#include<stdio.h>
int main()
{
  double x=10.5;
  int n;
  n=(int) x;
  printf("Value of n is %d\n",n);
  printf("Value of x is %lf\n",x);
  return 0;
}
```

2. scanf()

In the C programming language, scanf is a function that reads formatted data from stdin (i.e, the standard input stream, which is usually the keyboard, unless redirected) and then writes the results into the arguments given.
This function belongs to a family of functions that have the same functionality but differ only in their source of data. For example, fscanf gets its input from a file stream, whereas sscanf gets its input from a string.

- **Make a program that can show sum of any two integer numbers.**

```
# include<stdio.h>
int main()
{
  int a,b,sum;
  scanf("%d%d",&a,&b);
  sum=a+b;
  printf("Sum is %d",sum);
  return 0;
}
```

- **Make a program that can show sum of any two numbers.**

```
# include<stdio.h>
int main()
{
  double a,b,sum;
  scanf("%lf",&a);
  scanf("%lf",&b);
  sum=a+b;
  printf("Sum is %.3lf",sum);
  return 0;
}
```

- **Make a program that will convert decimal number into octal and hexadecimal number.**

```
#include<stdio.h>
int main()
{
    int number;
    printf("Decimal number = ");
    scanf("%d",&number);

    printf("Octal number = %o\n",number);
    printf("Hexadecimal number = %x\n",number);
    return 0;
}
```

- **Make a program that will show the first letter of your name.**

```
# include<stdio.h>
int main()
{
  char ch;
  printf("Enter the first letter of your name :");
  scanf("%c",&ch);
  printf("The first letter of your name is : %c\n",ch);
  return 0;
}
```

Here,
Data type = char (for any character) = %c,
It can be any single English letter or sign.
or,

```
# include<stdio.h>
int main()
{
 char ch;
 printf("Enter the first letter of your name :");
 ch = getchar();
 printf("The first letter of your name is : %c\n",ch);
 return 0;
}
```

Here,
Getchar is also a function as like char data type .It reads a character then assign into variable as like scanf function do.

- **Make a program that can summation, subtraction, multiplication and division.**

```
# include<stdio.h>
int main()
{
 int num1, num2;
 printf("Enter a number :");
 scanf("%d",&num1);
 printf("Enter another number :");
 scanf("%d",&num2);
 printf("%d+%d=%d\n",num1,num2,num1+num2);
 printf("%d-%d=%d\n",num1,num2,num1-num2);
 printf("%d*%d=%d\n",num1,num2,num1*num2);
 printf("%d/%d=%d\n",num1,num2,num1/num2);
 return 0;
}
```

or,

```
# include<stdio.h>
int main()
{
 double num1, num2;
 printf("Enter a number :");
 scanf("%lf",&num1);
 printf("Enter another number :");
 scanf("%lf",&num2);
 printf("%lf+%lf=%.3lf\n",num1,num2,num1+num2);
 printf("%lf-%lf=%.3lf\n",num1,num2,num1-num2);
 printf("%lf*%lf=%.3lf\n",num1,num2,num1*num2);
 printf("%lf/%lf=%.3lf\n",num1,num2,num1/num2);
 return 0;
}
```

or,

```
# include<stdio.h>
int main()
{
  //I am Hamim. I love Allah infinity.
  /*I love my country and my countries
  people. I want to do something for them.*/
  double a,b,value;
  char sign;
  printf("Enter a number :");
  scanf("%lf",&a);
  printf("Enter another number :");
  scanf("%lf",&b);
  value=a+b;
  sign='+';
  printf("%lf%c%lf=%.2lf\n",a,sign,b,value);
  value=a-b;
  sign='-';
  printf("%lf%c%lf=%.2lf\n",a,sign,b,value);
  value=a*b;
  sign='*';
  printf("%lf%c%lf=%.2lf\n",a,sign,b,value);
  value=a/b;
  sign='/';
  printf("%lf%c%lf=%.2lf\n",a,sign,b,value);
  return 0;
}
```

Here,
// and /* */ are comment of coding . Using these we can write anything in program but these will not effect in program because compiler do not take it as a part of a program. // this sign use to write one line of comment and /* */ this sign use to determine many lines of comment and it start from /* and end at */.

```
#include <stdio.h>
int main()
{
  // test program - comment 1
  printf("Hello ");
  /* We have printed Hello,
  now we shall print World.
  Note that this is a multi-line comment */
  printf("World"); // printed world
  return 0;
}
```

3. Operator And Expression :

- Consider the expression **A + B * 5** , where,

+ , *	are **operators**,
A, B	are **variables**,
5	is **constant**,
A, B and 5	are called **operand**, and
A + B * 5	is an **expression**.

`(a+b)*c`	Operator is `*`, operands are `(a+b)` and `c`
`(a+b)`	Operator is `()`, operand is `a+b`
`a+b`	Operator is `+`, operands are `a` and `b`

Why we are learning these things:

15 * 10 ⟶ Arithmetic operator

300000000 >= 500000000 ⟶ Relational operator

if ( && ) ⟶ Logical operator

Types of operators in C:

NAME OF OPERATORS	OPERATORS
ARITHMETIC OPERATORS	**+ , - , * , / , %**
INCREMENT/DECREMENT OPERATORS	**++ , --**
RELATIONAL OPERATORS	**== , != , <= , >= , < , >**
LOGICAL OPERATORS	**&& , || , !**
BITWISE OPERATORS	**& , ^ , | , ~ , >> , <<**
ASSIGNMENT OPERATORS	**= , += , -= , *= , /= , %= , <<= , >>= , &= , ^= , |=**
OTHER OPERATORS	?: , & , * , SIZEOF() , ,

ARITHMETIC OPERATORS

+ , - , * , / , %

addition Subtraction Multiplication Division Modulus

All are binary operators → means two operands are required to perform the operation.

For example: A + B

Op1 Op2

Operator Precedence and Associativity:

PRECEDENCE ↓	OPERATORS	ASSOCIATIVITY
HIGHEST	+ , - , * , / , %	LEFT TO RIGHT
LOWEST	+ , -	LEFT TO RIGHT

Note: Associativity is used only when two or more operators are of same precedence.

For example: + , -

Same precedence therefore we use associativity

+ , - | left to right

Coding Example:

```
#include<stdio.h>
int main()
{
  int a=2,b=3,c=4,d=5;
  printf("a*b/c=%d\n",a*b/c);
  printf("a+b-c=%d\n",a+b-c);
  printf("a+b*d-c%%a=%d",a+b*d/b-c%a);
  return 0;
}
```

Output:
a*b/c=1
a+b-c=1
a+b*d-c%a=7

Here:
a+b*d/b-c%a
= a+(b*d)/b-(c%a)
= 2+(3*5)/3-(4%2)
= 2+15/3-0
= 2+5-0
= 7-0
= 7

Increment and Decrement operators:

Increment operator is used to increment the value of a variable by one. Similarly, decrement operator is used to decrement the value of a variable by one.

Increment	Decrement
int a=5	int a=5;
a++;	a-- ;
a=6	a=4

a++; is same as a = a + 1;
a-- ; is same as a = a - 1;

Both are unary operators.
- because they are applied on single operand.

a++; ✓ a ++ a; ✗

Pre-increment operator
++a;

post-increment operator
a++;

Pre-decrement operator
--a;

post-decrement operator
a--;

(a+b)++; error!
++(a+b); error!

error: lvalue required as increment operand

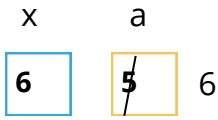
Compiler is expecting a variable as an increment operand but we are providing an expression (a+b) which dose not have the capability to store data.

Question: What is the difference between pre-increment and post-increment operator or pre-decrement and post-decrement operator?

Pre - means first increment/decrement then assign it to another variable.

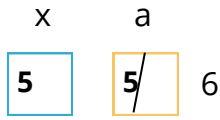
Post - means first assign it to another variable then increment/decrement.

x = ++a;



x = 6

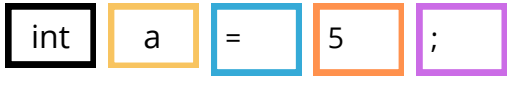
x = a++;



x = 5

Token Generation:

- Lexical analysis is the first phase in the compilation process.
- Lexical analyzer(scanner) scans the whole source program and when it finds the meaningful sequence of characters(lexemes) then it converts it into a token.
- Token: Lexemes mapped into token-name and attribute-value.
  Example: int → <keyword, int>
- It always matches the longest character sequence.

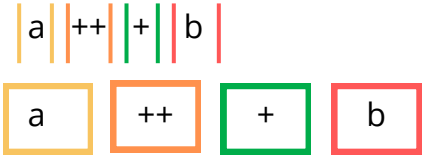


- **What is the output of the following C program fragment:**

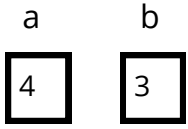
```
#include<stdio.h>
int main()
{
  int a=4,b=3;
  printf("%d\n",a+++b);
  return 0;
}
```

Output:
7

Here,



a++ + b
= 4 + 3
= 7



Post-increment/ decrement in context of equation:
First use the value in the equation and then increment the value.

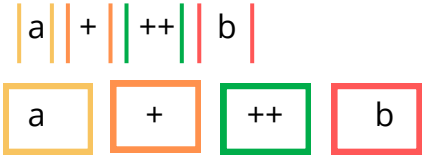
Pre-increment/ decrement in context of equation:
First increment the value and then use in the equation after completion of the equation.

- **What is the output of the following C program fragment:**

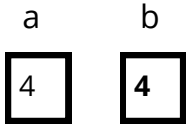
```
#include<stdio.h>
int main()
{
  int a=4,b=3;
  printf("%d\n",a + ++b);
  return 0;
}
```

Output:
8

Here,



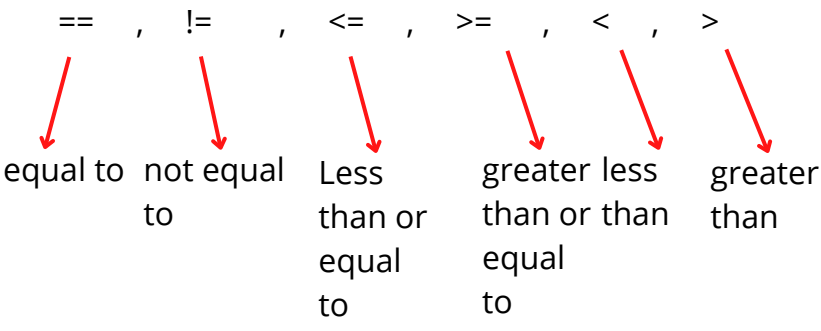
a+ ++ b
= 4 + 4
= 8



Post-increment/ decrement in context of equation:
First use the value in the equation and then increment the value.

Pre-increment/ decrement in context of equation:
First increment the value and then use in the equation after completion of the equation.

Relation operation:



Used for comparing two values

- all relational operators will return either true or False.

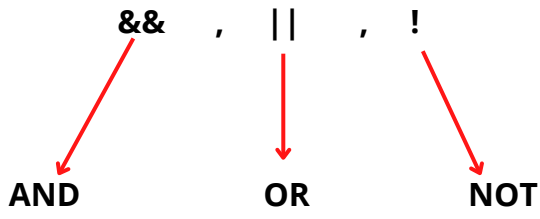
4==5 is equivalent to is 4 == 5 ?
Answer: False
4!=5 is equivalent to is 4!==5 ?
Answer: True

- **What is the output of the following C program fragment:**

```
#include<stdio.h>
int main()
{
  int a=3,b=4;
  if(b>=a){
    printf("Bingo! You are in");
  }
  else{
    printf("OOPS! You are out");
  }
  return 0;
}
```

Output:
Bingo! You are in

Logical operators:



- && and || are used to combine two conditions. && - returns TRUE when all the conditions under consideration are true and returns FALSE when any one or more than one condition is false.

For example:

- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
    int a = 5;
    if (a == 5 && a != 6 && a <= 56 && a > 4)
       {
    printf("Welcome to this beautiful world of
operators");
        }
     return 0;
}
```

Output:
Welcome to this beautiful world of operators

- || - returns TRUE when one or more than one condition under consideration is true and returns FALSE when all conditions are false.

For example:

- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
  int a = 5;
  if (a == 5 || a != 6 || a <= 56 || a > 4)
  {
    printf("Welcome to this beautiful world of
operators");
  }
  return 0;
}
```

Output:
Welcome to this beautiful world of operators

- ! operator is used to complement the condition under consideration.
  ! - returns TRUE when condition is FALSE and returns FALSE and False when condition is TRUE.

For example:

- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
  int a = 5;
  if (a == 5 || a != 6 || a <= 56 || a > 4)
  {
    printf("Welcome to this beautiful world of operators");
  }
  return 0;
}
```

Output:
Welcome to this beautiful world of operators

Concept of short circuit in logical operators:

Short circuit in case of &&: Simply means if there is a condition anywhere in the expression that returns false, then the rest of the conditions after that will not be evaluated.
For example:

- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
  int a = 5,b=3;
  int incr;
  incr = (a<b) && (b++);
  printf("%d\n",incr);
  printf("%d\n",b);
  return 0;
}
```

Output:
0
3

- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
  int a = 5,b=3;
  int incr;
  incr = (a>b) && (b++);
  printf("%d\n",incr);
  printf("%d\n",b);
  return 0;
}
```

Output:
1
4

Short circuit in case of ||: Simply means if there is a condition anywhere in the expression that returns True, then the rest of the conditions after that will not be evaluated.
For example:

- **What is the output of the following C program fragment:**

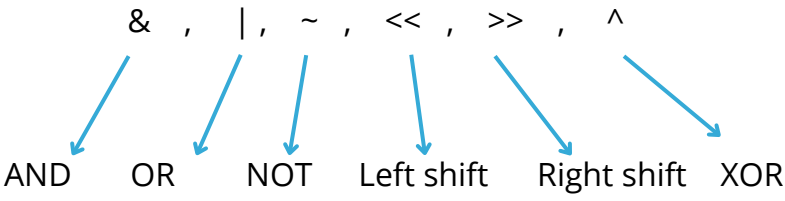
```
#include <stdio.h>
int main()
{
  int a = 5,b=3;
  int incr;
  incr = (a>b) || (b++);
  printf("%d\n",incr);
  printf("%d\n",b);
  return 0;
}
```

Output:

```
1
3
```

Bitwise operators:

As name suggests - it dose bitwise manipulation.



BIT BY BIT

Bitwise AND (&) operator:

- It takes two bits at a time and perform AND operation.
- And (&) is binary operator. It takes two numbers and perform bitwise AND.
- Result of AND is 1 when both bits are 1.

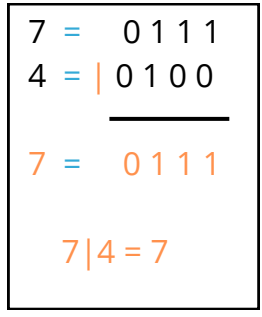
```
7 =    0 1 1 1
4 = &  0 1 0 0
-------------
4 =    0 1 0 0

  7&4 = 4
```

Truth Table

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise OR (|) operator:

- It takes two bits at a time and perform OR operation.
- OR (|) is binary operator. It takes two numbers and perform bitwise OR.
- Result of OR is 0 when both bits are 0.

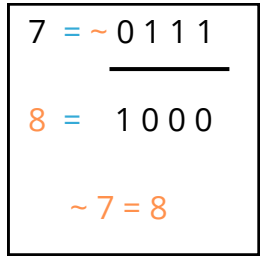


Truth Table

A	B	A&B
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise NOT (~) operator:

- NOT is a unary operator.
- It's job is to complement each bit one by one.
- Result of NOT is 0 when bit is 1 and 1 when bit is 0.



Truth Table

A	~A
0	1
1	0

Difference between bitwise and logical operators:

```
#include <stdio.h>
int main()
{
  char x=1,y=2; /* x=1(0000 0001),
           y=2(0000 0010)
          1&2=0(0000 0000)
  1&&2 = TRUE && TRUE = TRUE =1 */

  if(x&y)
  printf("Result of x&y is 1\n");
  if(x&&y)
  printf("Result of x&&y is 1\n");
  return 0;
}
```

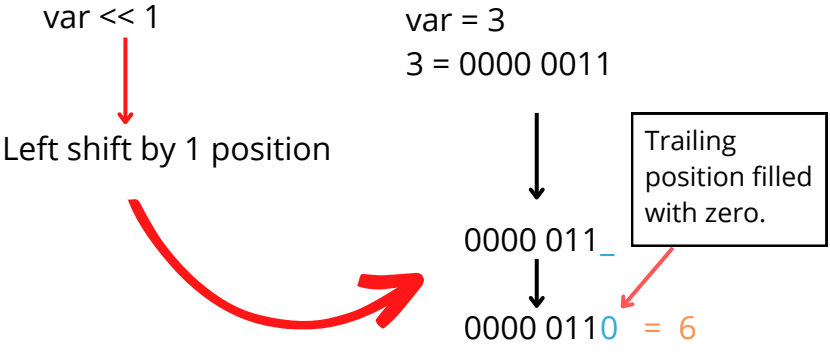
output:
Result of x&&y is 1

Left shift operator:

First operand << Second operand

Whose bits get left shifted

- How left shift works?



Important points:

Left shifting is equivalent to multiplication by 2^(right Operator) .

Example: var = 3;
var << 1 Output: 6 [3 x 2^1]
var << 4 Output: 48 [3 x 2^4]

Important points:

- When bits are shifted left then trailing positions are filled with zeros.

```
#include <stdio.h>
int main()
{
  char var=3; // 3 in binary = 0000 0011
  printf("%d\n",var<<1);
  return 0;
}
```

char = 1 byte = 8 bits

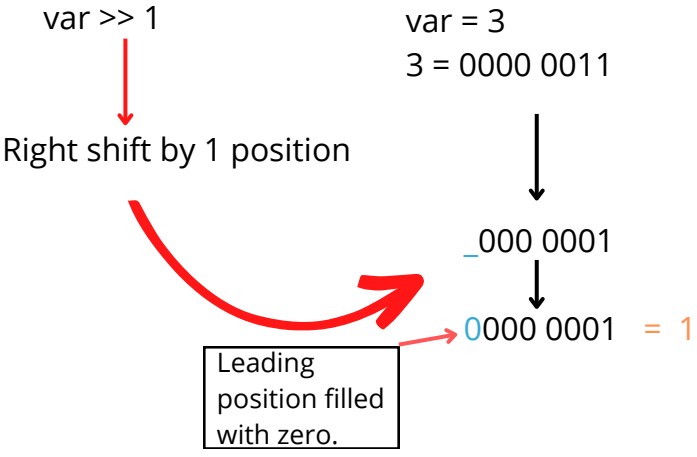
output:
6

Right shift operator:

First operand >> Second operand

Whose bits get right shifted

- How right shift works?



Important points:

Right shifting is equivalent to division by 2^(right Operator) .

Example:

var = 3;
var >> 1 Output: 1 [3 / 2^1]

var = 32
var >> 4 Output: 2 [32 / 2^4]

Important points:

- When bits are shifted right then leading positions are filled with zeros.

```
#include <stdio.h>
int main()
{
  char var=3; // 3 in binary = 0000 0011
  printf("%d\n",var>>1);
  return 0;
}
```

output:
1

Bitwise XOR operator:

X - OR

Inclusive OR:

- Either A is 1 or B is 1 or Both are 1, then the output is 1.
- Including BOTH.

Exclusive OR:

- Either A is 1 or B is 1 then the output is 1 but when both A and B are 1 then Output is 0.
- Excluding BOTH

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Only difference

- Bitwise XOR(^) is binary operator. It takes two numbers and perform bitwise XOR.
- Result of XOR is 1 when two bits are different otherwise the result is 0.

```
7 =    0 1 1 1
4 = ^  0 1 0 0
       -------
7 =    0 0 1 1

7^4 = 3
```

Truth Table

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

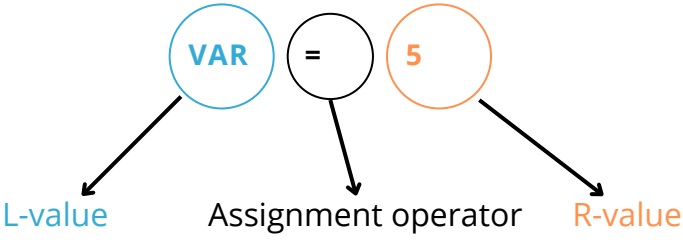
- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
  int a = 4,b=3;
  a=a^b;
  b=a^b;
  a=a^b;
  printf("After XOR, a=%d and b=%d\n",a,b);
  return 0;
}
```

Output:
After XOR, a=3 and b=4

Assignment operator:

Values to a variable can be assigned using assignment operator.
Requires two values - L-value and R-value.
This operator copies R-value to L-value.



Shorthand assignment operators:

+=	First addition than assignment
-=	First subtraction than assignment
*=	First multiplication than assignment
/=	First division than assignment

Example: a += 1 is equivalent to a = a +1
Similar concept for other shorthand assignment operators as well.

- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
  char a=7;
  a^=5;
  printf("%d", printf("%d", a+=3));
  return 0;
}
```

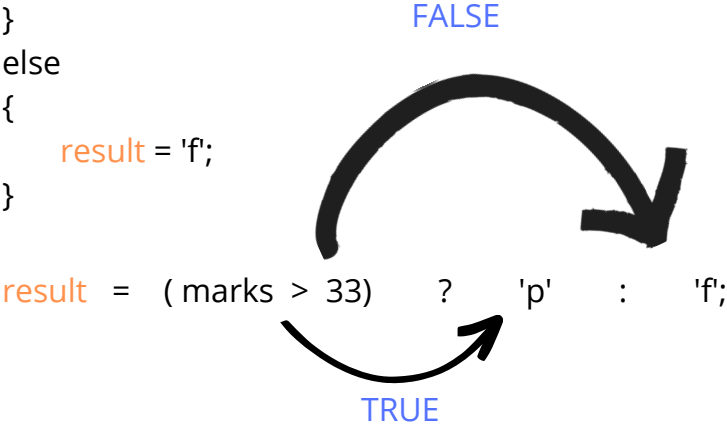
Output:
51

Conditional operator:

Look and feel: ? :

```
char result;
int marks;
if(marks > 33)
{
      result = 'p';
}
else
{
     result = 'f';
}
```

```
char result;
int marks;
result = (marks > 33) ? 'p' :  'f';
```



(marks > 33) is a boolean expression, therefore it will return either true or false.

(marks > 33) ? 'p ' : 'f' is a conditional expression, which is after all expression, therefore it is an r-value and result is l-value.

Quick facts checklist:

- Conditional operator is the only ternary operator available in the list of operators in C language.
- As in Expression1 ? Expression2 : Expression3 , expression1 is the boolean expression. If we simply write 0 instead of some boolean expression than that simply means FALSE and therefore Expression3 will getevaluated.

Example: int result;
result = 0 ? 2 : 1

result

1

Comma (.) opeartor:

Comma operator can be used as an "operator".

```
int a= (3,4,8);
printf("%d",a);
```

Output:
8

Comma operator returns the rightmost operand in the expression and it simply evaluates the rest of the operands and finally reject them.

Example:

```
int var = (print("%d\n", "HELLO"),  5  ;
print("%D"< var);
```

This value will be returned to var after evaluating the first operand

OUTPUT:
HELLO!
5

It will simply not rejected. First evaluated and then rejected

- Comma operator is having least precedence among all the operators available in C language.

Example 1:

```
int a;
a = 3, 4, 8;
printf("%d", a);
```

≡

```
int a;
(a = 3), 4, 8;
printf("%d", a);
```

Output: 3

Example 2:

```
int a = 3, 4, 8;
printf("%d",a);
```

Here,
Comma is behaving like a separator.

int a = 3, 4, 8;
is equivalent to
int a = 3; int 4; int 8;

ERROR!

Example 3:

```
int a;
a = (3,4,8);
printf("%d", a);
```

or

```
int a = (3,4,8);
printf("%d", a);
```

Here,
Bracket has the highest precedence than only other operator.

Output: 8

- **What is the output of the following C program fragment:**

```
#include <stdio.h>
int main()
{
  int var;
  int num;
  num = (var = 15, var+35);
  printf("%d",num);
  return 0;
}
```

Output:
50

Precedence and Associativity of Operators::

Precedence of operators come into picture when in an expression we need to decide which operator will be evaluated first. Operator with higher precedence will be evaluated first.

Example: 2 + 3 * 5

Precedence of multiplication is greater than addition.

2 + (3 * 5) = 17

(2 + 3) * 5 = 25

Associativity of operators come into picture when precedence of operators are some need to decide which operator will be evaluated first.

Example: 10 / 2 * 5

Left to right: (10 / 2) * 5 = 25 

Right to left: 10 / (2 * 5) = 1 

Associativity can be either :

1. Left to right
2. Right to left

() - parenthesis in function calls:

Example: int var = fun();

operand

= operator is having less precedence as co,pared to () therefore, () belongs to fun will be treated as a function.

int var = (fun());

If suppose = operator is having greater precedence then, fun will belong to = operator and therefore it will be treated as a variable.

int (var = fun) ();

Member access operators (-> .) :

- They are used to access members of structures.
- We will talk about structures later in this course.

Postfix Increment/Decrement (++, --)

- Precedence of Postfix Increment/Decrement operator is greater than Prefix Increment/ Decrement.
- Associativity of Postfix is also different from Prefix. Associativity of postfix operators is from left to right and that of prefix operators is from right to left.

- Associativity can only help if there are two or more operators of some precedence and not when there is just one operator.
- Operators with some precedence have some associativity as well.

Example:

```
int main()
{
    int a;
    a = fun1() + fun2();
    printf("%d", a);
    return 0;
}
```

```
int fun1()
{   printf("Neso");
    return 1;
}

int fun2()
{   printf("Academy");
    return 1;
}
```

Which function is called first ?
fun1() or fun2()?

It is not defined whether fun1() will be called first or whether fun2() will be called. Behaviour is undefined and output is compiler dependent.

NOTE: Here associativity will not come into picture as we have just one operator and which function will be called first is undefined. Associativity will only work when we have more than one operators of same precedence

Example:

```
int main()
{
    int a;
    a = fun1() + fun2();
    printf("%d", a);
    return 0;
}
```

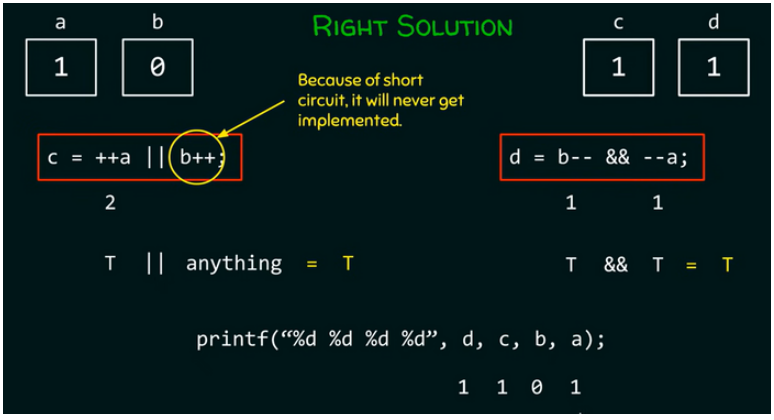
```
int fun1()
{   printf("Neso");
    return 1;
}

int fun2()
{   printf("Academy");
    return 1;
}
```

Output: NesoAcademy2
OR
AcademyNeso2

What is the output of the following C program fragment?

```
int a = 1;
int b = 1;
int c = ++a || b++;
int d = b-- && --a;

printf("%d %d %d %d", d, c, b, a);
```

a) 1 1 1 1
b) 0 1 0 0
c) 1 0 0 1
d) 1 1 0 1



- The following table lists the precedence and associativity of C operators.

Operators are listed top to bottom, in descending precedence.

Precedence	Operator	Description	Associativity
1	`( )`	Function call	
	`[ ]`	Array subscripting	
	`.`	Structure and union member access	
	`->`	Structure and union member access through pointer	
2	`++ --`	Prefix increment and decrement	Right-to-left
	`+ -`	Unary plus and minus	
	`! ~`	Logical NOT and bitwise NOT	
	`(type)`	Type cast	
	`*`	Indirection (dereference)	
	`&`	Address-of	
	`sizeof`	Size-of	
3	`* / %`	Multiplication, division, and remainder	Left-to-right
4	`+ -`	Addition and subtraction	
5	`<< >>`	Bitwise left shift and right shift	
6	`< <=`	For relational operators < and ≤ respectively	
	`> >=`	For relational operators > and ≥ respectively	
7	`== !=`	For relational = and ≠ respectively	
8	`&`	Bitwise AND	
9	`^`	Bitwise XOR (exclusive or)	
10	`|`	Bitwise OR (inclusive or)	
11	`&&`	Logical AND	
12	`||`	Logical OR	
13	`?:`	Ternary conditional	Right-to-Left
14	`=`	Simple assignment	
	`+= -=`	Assignment by sum and difference	
	`*= /= %=`	Assignment by product, quotient, and remainder	
	`<<= >>=`	Assignment by bitwise left shift and right shift	
	`&= ^= |=`	Assignment by bitwise AND, XOR, and OR	
15	`,`	Comma	Left-to-right

4.if-else

The if-else statement is used to perform two operations for a single condition. The if-else statement is an extension to the if statement using which, we can perform two different operations, i.e., one is for the correctness of that condition, and the other is for the incorrectness of the condition. Here, we must notice that if and else block cannot be executed simiulteneously. Using if-else statement is always preferable since it always invokes an otherwise case with every if condition.

Here,
for greater >,
for smaller <,
for equal ==,
for greater and equal >=,
for smaller and equal <=,
for not equal !=

- **Make a program that will show positive and negative number.**

```
#include <stdio.h>
int main()
{
  int n;
  n = 10;
  if(n >= 0) {
  printf("The number is positive\n");
  }
  else {
  printf("The number is negative\n");
  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int n;
 n = 10;
 int a=4,b=2;
 char sign;
 if(n < 0) {
 printf("The number is negative\n");
 }
 else {
 int value=a/b;
 sign='/';
 printf("%d%c%d=%d\n",a,sign,b,value);
 }
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int n = 10;
  if(n < 0) {
  printf("The number is negative\n");
  }
  else if (n > 0) {
  printf("The number is positive\n");
  }
  else if (n == 0) {
  printf("The number is zero!\n");
  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int n = 10;
  if(n < 0) {
  printf("The number is negative\n");
  }
  else if (n > 0) {
  printf("The number is positive\n");
  }
  else {
  printf("The number is zero!\n");
  }
  return 0;
}
```

- **Make a program that will show 12 is greater than 10.**

```
#include <stdio.h>
int main()
{
  int number = 12;
  if(number > 10) {
  printf("The number is greater than ten\n");
  }
  return 0;
}
```

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int n = 10;
  if (n < 30) {
  printf("n is less than 30.\n");
  }
  else if(n < 50) {
  printf("n is less than 50.\n");
  }
  return 0;
}
```

Here,
The output will be n is less than 30 although else if (n<50) is true. Since if (n<30) have become true that's why any other operation will not be accepted.

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int n = 10;
 if (n < 30) {
 printf("n is less than 30.\n");
 }
 if(n < 50) {
 printf("n is less than 50.\n");
 }
 return 0;
}
```

Here,
The output will be n is less than 30 and
n is less than 50.

- **What will be the output of the program given bellow.**

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

   int a=100;
   int b=200;
   if(a==100){
      if(b==200){
         printf("Value of a is 100 and b is 200\n");
      }
   }
   printf("Exact value of a is : %d\n",a);
   printf("Exact value of b is : %d\n",b);
   return 0;
}
```

Here,
The output will be :

```
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

We have use nested if statements. It is always legal in C to nest if-else statements, which means you can use one if or else if statement inside another if or else if statements.

- **What will be the output of the program given bellow.**

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

  int a=100;
  int b=200;
  if(a==100){
    if(b==200){
      printf("Value of a is 100 and b is 200\n");
      if(a<b){
        printf("%d is smaller than %d\n",a,b);
      }
      else{
        printf("%d is bigger than %d\n",a,b);
      }
    }
  }
  printf("Exact value of a is : %d\n",a);
  printf("Exact value of b is : %d\n",b);
  return 0;
}
```

Here,
The output will be :
Value of a is 100 and b is 200
100 is smaller than 200
Exact value of a is : 100
Exact value of b is : 200

- **Make a program that will show 5 is a odd number.**

```
#include <stdio.h>
int main()
{
  int number, remainder;
  number = 5;
  remainder = number % 2;
  if(remainder == 0) {
  printf("The number is even\n");
  }
  else {
  printf("The number is odd\n");
  }
  return 0;
}
```

Here,
'%' is a modulus operator. It show us remainder of any division.

- **Make a program that will show even and odd number.**

```
#include <stdio.h>
int main()
{
  int number;
```

```
  printf("Enter the number :\n");
  scanf("%d",& number);
  if(number % 2 == 0) {
  printf("The number is even\n");
  }
  else {
  printf("The number is odd\n");
  }
  return 0;
}
```

- **Make a program that will show small/lower case letter and capital /upper case letter.**

```
#include<stdio.h>
int main()
{
  char ch = 'B';
  if (ch == 'a')
  {
  printf("%c is lower case\n", ch);
  }
  else if (ch == 'A')
  {
  printf("%c is upper case\n", ch);
  }
  else if (ch == 'b)
```

```
  {
  printf("%c is lower case\n", ch);
  }
  else if (ch == 'B')
  {
  printf("%c is upper case\n", ch);
  }
  else if (ch == 'c')
  {
  printf("%c is lower case\n", ch);
  }
  else if (ch == 'C')
  {
  printf("%c is upper case\n", ch);
  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  char ch = 'W';
  if(ch >= 'a' && ch <= 'z') {
  printf("%c is lower case\n", ch);
  }
  else if(ch >= 'A' && ch <= 'Z') {
  printf("%c is upper case\n", ch);
  }
 return 0;
}
```

Here,
'&&' is an AND operator.

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int num = 5;
  if(num >= 1 || num <= 10) {
  printf("yes\n");
  }
  else {
  printf("no\n");
  }
  return 0;
}
```

Here,
The output will be Yes.

Here,
'||' is OR operator.

- **Make a program that will show vowel and consonant.**

```
#include <stdio.h>
int main()
{
 char ch;
 printf("Enter the letter :");
 scanf("%c",& ch);
 if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u'
 || ch=='A' || ch=='E' || ch=='I' || ch=='O' ||
ch=='U' )
 {
  printf("%c is vowel\n",ch);
 }
 else{
  printf("%c is consonant",ch);
 }
 return 0;
}
```

5.switch-case

Switch is a multiple choice selection statement. It will be executed according to user choice.
Each switch-case must include break keyword.
Sometime time we should use default keyword.

- **Write a program that read a digit and display its spelling.**

```
#include <stdio.h>
int main()
{
  int digit;
  printf("Enter a digit :");
  scanf("%d",&digit);
  switch(digit)
  {
    case 0:
    printf("Zero\n");
    break;

    case 1:
    printf("One\n");
    break;

    case 2:
    printf("Two\n");
    break;
    case 3:
    printf("Three\n");
    break;
```

```
    case 4:
    printf("Four\n");
    break;

    case 5:
    printf("Five\n");
    break;

    case 6:
    printf("Six\n");
    break;

    case 7:
    printf("Seven\n");
    break;

    case 8:
    printf("Eight\n");
   break;

    case 9:
    printf("Nine\n");
    break;

    default:
    printf("Not a valid digit\n");
    }
    return 0;
}
```

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int a = 100;
  int b = 200;
  switch (a)
  {
  case 100:
   printf("Value of a is 100\n");
   switch (b)
   {
   case 200:
     printf("Value of b is 200\n");
   }
  }
  printf("Exact value of a is : %d\n", a);
  printf("Exact value of b is : %d\n", b);
  return 0;
}
```

Here,
The output will be:
Value of a is 100
Value of b is 200
Exact value of a is : 100
Exact value of b is : 200
It is possible to have a switch as part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

6.Ternary opereator

Ternary operator (? :) is used to replace if-else statements. It has the following general from.

Exp1 ? Exp2 : Exp3 ;

For example, consider the following code:

```
if(condition){
    var = 30;
}
else{
    var = 40;
}
```

Above code can be rewritten like this:

```
var = (Y<10) ? 30 : 40;
```

Here,
X is assigned the value of 30 if Y is less than 10 and 40 if it is not .

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int var,y=7;
  if(y<10){
      var = 30;
}
else{
      var = 40;
}
printf("%d\n",var);
  return 0;
}
```

Here,
The output will be 30.
Now we will do this program using ternary operator.

or,

```
#include <stdio.h>
int main()
{
  int var,y=7;
  var = (y<10) ? 30 : 40;
  printf("%d\n",var);
  return 0;
}
```

7.while

A while loop in C programming repeatedly executes a target statement as long as a given condition is true.

- **Make a program that will show 1 to 10.**

```
#include <stdio.h>
int main()
{
  int n = 1; // initialization
  while (n <= 10) // condition
  {
    printf("%d\n", n); // statements
    n++; // increment/decrement
  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int n=1;
  while (n<=100)
  {
    printf("%d\n",n);
    n++;
    if (n>10)
```

```
   {
     break;
   }

  }

  return 0;
}
```

Here,
Break is a statement which is used to left the loop.

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int n=1;
  while(n<=10){
    printf("%d\n",n);
  }
    n++;
    return 0;
}
```

Here,
We can see on the output that 1 is printing again and again.

- **Make a program that will show your name again and again, I mean infinity times.**

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

- **Make a program that will show odd number 1 to 10.**

```
#include <stdio.h>
int main()
{
  int n=0;
  while (n<10)
  {
    n++;
    if (n%2==0)
    {
      continue;
    }
    printf("%d\n",n);

    }

    return 0;
}
```

Here,
Continue is a statement which is used to omit any of elements from the loop.

- **Make a program that will show multiplication table of 5.**

```
#include <stdio.h>
int main()
{
  int n=5,i=1;
  while(i<=10)
  {
    printf("%dX%d=%d\n",n,i,n*i);
    i=i+1;
  }
  return 0;
}
```

Here
Increment : i=i+1/i++/++i/i+=1
Decrement : i=i-1/i--/--i/i-=1

- **Make a program that will show given bellow.**
  When i = 1
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 2
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 3
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 4
  j = 0, j = 1, j = 2, j = 3, j = 4,

```
#include <stdio.h>
void main()
{
  int i = 0;
  while (i < 5) /* Outer loop */
  {
    printf("When i = %d\n", i);
    int j = 0;
    while (j < 5) /* Inner loop */
    {
      printf("j = %d, ", j);
      j++;
    }
    printf("\n\n");
    i++;
  }
}
```

8.do while

Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming checks its condition at the bottom of the loop.

A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.
The syntax of a do...while loop in C programming language is −

```
do {
   statement(s);
} while( condition );
```

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.
If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.

- **Make a program that will print 1 to 10 using do while loop.**

```
#include <stdio.h>
int main()
{
   int i=1;  // initialization
   do{
      printf("%d\n",i); //statements
      i++;  // increment/decrement
   }while(i<=10);  // condition
   return 0;
}
```

- **Make a program that will print 10 to 19 using do while loop.**

```
#include <stdio.h>
int main()
{
  int a = 10;
  do
  {
    printf("value of a: %d\n", a);
    a = a + 1;
  }

  while (a < 20);

  return 0;
}
```

- **Make a program that will show given bellow.**
  When i = 1
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 2
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 3
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 4
  j = 0, j = 1, j = 2, j = 3, j = 4,

```
#include <stdio.h>
void main()
{
  int i = 0;
  do
  {
    printf("When i = %d\n", i);
    int j=0;
    do
    {
      printf("j = %d, ", j);
      j++;
    } while (j < 5);/* Inner loop */
    printf("\n\n");
    i++;
  } while (i < 5);/* Outer loop */
}
```

9.for

The **for loop in C language** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

- **Make a program that will show multiplication table of 5 using for loop.**

```
#include <stdio.h>
int main()
{
 int n=5,i;
 for(i=1; i<=10; i=i+1) // initialization; condition;
increment/decrement
 {
  printf("%dX%d=%d\n",n,i,n*i); // statements
 }
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int n=5,i=1;
  for(; i<=10; i=i+1)
  {
    printf("%dX%d=%d\n",n,i,n*i);
  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int n=5,i=1;
  for( ; ; )
  {
    printf("%dX%d=%d\n",n,i,n*i);
    i++;
    if (i>10)
    {
      break;
    }

  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int m=0,n=5,i;
  for( i=1; i<=10 ; i=i+1 )
  {
    m=m+n;
    printf("%dX%d=%d\n",n,i,m);
  }
  return 0;
}
```

- **Make a program that will show multiplication table of 1 to 10.**

```
#include <stdio.h>
int main()
{
  int i;
  for (int n=1;n<=10;n++)
  {
    for ( i = 1; i <10; i++)
    {
      printf("%dX%d=%d\n",n,i,n*i);
    }

  }
  return 0;
}
```

- **Make a program that will show given bellow.**

  1 1 1
  2 3 4
  3 5 7
  4 7 10
  5 9 13
  6 11 16
  7 13 19
  8 15 22
  9 17 25
  10 19 28

```
#include <stdio.h>
int main()
{
   int i,j,k;
   for(i=1,j=1,k=1;i<=10;i=i+1,j=j+2,k=k+3){
      printf("%d %d %d\n",i,j,k);
   }
   return 0;
}
```

- **Make a program that will show permutation of 1,2,3 and the output will be**
  1, 2, 3
  1, 3, 2
  2, 1, 3
  2, 3, 1
  3, 1, 2
  3, 2, 1

```
#include <stdio.h>
int main()
{
  int a, b, c;
  for (a = 1; a <= 3; a++) {
   for (b = 1; b <= 3; b++) {
    for (c = 1; c <= 3; c++) {
  printf ("%d, %d, %d\n", a, b, c);
      }
    }
  }
  return 0;
}
```

here,
The output will be :
1, 1, 1
1, 1, 2
1, 1, 3
1, 2, 1
1, 2, 2
1, 2, 3
1, 3, 1

1, 3, 2
1, 3, 3
2, 1, 1
2, 1, 2
2, 1, 3
2, 2, 1
2, 2, 2
2, 2, 3
2, 3, 1
2, 3, 2
2, 3, 3
3, 1, 1
3, 1, 2
3, 1, 3
3, 2, 1
3, 2, 2
3, 2, 3
3, 3, 1
3, 3, 2
3, 3, 3

Here,
Something is wrong. Think about it and try again.

```
#include <stdio.h>
int main()
{
  int a, b, c;
  for (a = 1; a <= 3; a++) {
   for (b = 1; b <= 3 && b != a; b++) {
     for (c = 1; c <= 3 && c != a && c != b; c++) {
```

```
    printf ("%d, %d, %d\n", a, b, c);
        }
     }
  }
  return 0;
}
```

Here,
Output will be 3,2,1.
Something is wrong. So, lets try again.

```
#include <stdio.h>
int main()
{
  int a, b, c;
  for (a = 1; a <= 3; a++) {
    for (b = 1; b <= 3; b++) {
    if (b != a) {
      for (c = 1; c <= 3; c++) {
      if (c != b && c != a){
      printf ("%d, %d, %d\n", a, b, c);
              }
          }
        }
     }
  }
  return 0;
}
```

Here;
The output is 1, 2, 3
1, 3, 2
2, 1, 3
2, 3, 1
3, 1, 2
3, 2, 1
So, at last our problem has solved.

or,

```
#include <stdio.h>
int main()
{
  int a, b, c;
  for (a = 1; a <= 3; a++) {
   for (b = 1; b <= 3; b++) {
    for (c = 1; c <= 3; c++) {
    if(b != a && c != a && c != b) {
    printf ("%d, %d, %d\n", a, b, c);
      }
     }
    }
  }
  return 0;
}
```

- **Make a program that will show given bellow.**
  When i = 1
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 2
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 3
  j = 0, j = 1, j = 2, j = 3, j = 4,

  When i = 4
  j = 0, j = 1, j = 2, j = 3, j = 4,

```
#include <stdio.h>
void main()
{
  int i, j;
  for (i = 0; i < 5; i++) /* Outer loop */
  {
    printf("When i = %d\n", i);
    for (j = 0; j < 5; j++) /* Inner loop */
    {
      printf("j = %d, ", j);
    }
    printf("\n\n");
  }
}
```

- **Make a program that will calculate power(base, exponent) using for loop.**

```
#include<stdio.h>
int main()
{
  double base,exp,result=1,i;
  printf("Enter base : ");
  scanf("%lf",&base);
  printf("Enter exponent : ");
  scanf("%lf",&exp);
  for(i=1;i<=exp;i++){
    result=result*base;
  }
  printf("%.1lf\n",result);
  return 0;
}
```

- **1+2+3+4+.....+100 solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,s=0;
   for(i=1;i<=100;i++){
      s=s+i;
   }
   printf("%d",s);
   getch();
}
```

- **1+2+3+4+.....+n solve the problem.**

```
#include <stdio.h>
main()
{
   int i,n,s=0;
   scanf("%d",&n);
   for(i=1;i<=n;i++){
      s=s+i;
   }
   printf("%d",s);
   getch();
}
```

- 1+3+5+7+.....+n solve it.

```
#include <stdio.h>
main()
{
   int i,n,s=0;
   scanf("%d",&n);
   for(i=1;i<=n;i=i+2){
      s=s+i;
   }
   printf("%d",s);
   getch();
}
```

- **2+4+8+16+.....+n solve it.**

```
#include <stdio.h>
main()
{
   int i,n,s=0;
   scanf("%d",&n);
   for(i=2;i<=n;i=i*2){
      s=s+i;
   }
   printf("%d",s);
   getch();
}
```

- **2^1+2^2+2^3+2^4+.....+2^n solve it.**

```
#include <stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
   int i,n;
   long int s=0;
   scanf("%d",&n);
   for(i=1;i<=n;i++){
      s=s+pow(2,i);
   }
   printf("%ld",s);
   getch();
}
```

Here,
Library function = pow(a,b) = a^b

- **1^2+2^2+3^2+4^2+.....+n^2 solve it.**

```
#include <stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
   int i,n;
   long int s=0;
   scanf("%d",&n);
   for(i=1;i<=n;i++){
      s=s+pow(i,2);
   }
   printf("%ld",s);
   getch();
}
```

- **1^2+3^2+5^2+7^2+.....+n^2 solve it.**

```
#include <stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
   int i,n;
   long int s=0;
   scanf("%d",&n);
   for(i=1;i<=n;i+=2){
      s=s+pow(i,2);
   }
   printf("%ld",s);
   getch();
}
```

- **100+99+98+.....+n solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,n,s=0;
   scanf("%d",&n);
   for(i=100;i>=n;i--){
      s=s+i;
   }
   printf("%d",s);
   getch();
}
```

- **n+(n-1)+(n-2)+.....+1 solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,n,s=0;
   scanf("%d",&n);
   for(i=n;i>=1;i--){
      s=s+i;
   }
   printf("%d",s);
   getch();
}
```

- **1-2+3-4+5-……..+n solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,n,s=0;
   scanf("%d",&n);
   for(i=1;i<=n;i++){
      s=s+pow(-1,i+1)*i;
   }
   printf("%d",s);
   getch();
}
```

- **-1+2-3+4-5+……..+n solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,n,s=0;
   scanf("%d",&n);
   for(i=1;i<=n;i++){
      s=s+pow(-1,i)*i;
   }
   printf("%d",s);
   getch();
}
```

- **5! = 1*2*3*4*5 solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,f=1;
   for(i=1;i<=5;i++){
      f=f*i;
   }
   printf("%d",f);
   getch();
}
```

- **5! = 5*4*3*2*1 solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,f=1;
   for(i=5;i>=1;i--){
      f=f*i;
   }
   printf("%d",f);
   getch();
}
```

- **n! = 1*2*3*.....*(n-1)*n solve it.**

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,n,f=1;
   scanf("%d",&n);
   for(i=1;i<=n;i++){
      f=f*i;
   }
   printf("%d",f);
   getch();
}
```

or,

```
#include <stdio.h>
#include<conio.h>
int main()
{
   int i,n,f=1;
   scanf("%d",&n);
   if(n>=0){
   for(i=1;i<=n;i++){
      f=f*i;
   }
   printf("%d",f);
   }
   else{
      printf("Invalid Input");
   }
   getch();
}
```

10.goto

A goto statement in C language provides an unconditional jump from the goto to a labeled statement in the function.
The given label must be reside in the same function.
goto label;
..
.
label : statement;

- **What will be the output of the program given bellow.**

```
#include <stdio.h>
int main()
{
  int a=10;
  loop : do
  {
    if(a==15)
    {
      a=a+1;
      goto loop;
    }
    printf("Value of a: %d\n",a);
    a++;
  }
  while(a<20);
  return 0;
}
```

Here,
The output will be :

Value of a: 10
Value of a: 11
Value of a: 12
Value of a: 13
Value of a: 14
Value of a: 16
Value of a: 17
Value of a: 18
Value of a: 19

11.array

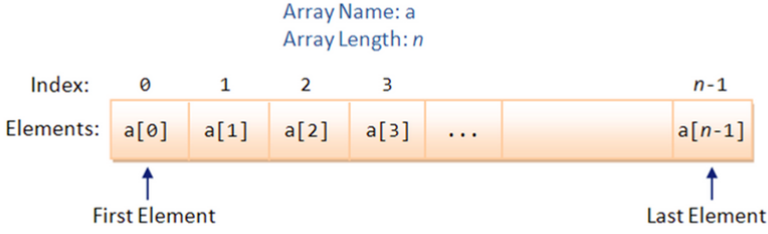
An array is a collection of variables of same data type.

Arrays can following types :

1.One dimensional (1-D) arrays Linear arrays.
Example : int marks[10];
2. Multi dimensinal arrays
(a) Two dimensional (2-D) arrays or Matrix arrays
Example : int marks[2][3];
(b) Three dimensional (3-D) arrays
Example : int marks[2][3][2];

- C programming language provides a data structure called the array, which can store a fixed-size sequential collection of elements of the 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 number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.
- A specific element in an array is accessed by an index.

- All arrays consist of contiguous memory locations.
- The lowest address corresponds to the first element and the highest address to the last element.



1-D/One dimensional array :

Decliaring Arrays :

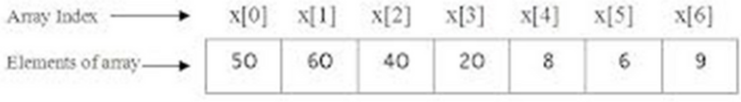
- To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows :

```
type arrayName [ arraySize ];
```

- This is called a single-dimensional array. The arraySize must be an integer constant greater than zero and type can be any valid C data type.
- For example, to declare a 10-element array called balance of type double, use this statement:

```
double balance[10];
```

- Now balance is a variable array which is sufficient to hold up to 10 double numbers.



Initializing Arrays :

- You can initialize array in C either one by one or using a single statement as follows:

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

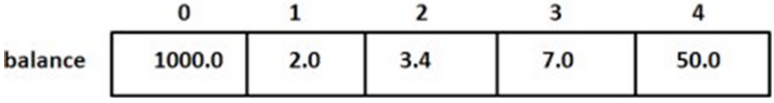
- The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ].
- If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write:

```
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

- You will create exactly the same array as you did in the previous example.
- Following is an example to assign a single element of the array:

```
balance[4] = 50.0;
```

- The above statement assigns element number 5th in the array with a value of 50.0.
- All arrays have 0 as the index of their first element which is also called base index and last index of an array will be total size of the array minus 1.
- Following is the pictorial representation of the same array we discussed above:



Accessing Array Elements :

- An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array.
- For example:

```
double salary = balance[9];
```

- The above statement will take 10th element from the array and assign the value to salary variable.

Code Example: Array :

- Following is an example which will use all the above mentioned three concepts declaration, assignment and accessing arrays:

```c
#include <stdio.h>

int main ()
{
   int n[ 10 ]; /* n is an array of 10 integers */
   int i,j;

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ )
   {
      n[ i ] = i + 100; /* set element at location i to i + 100 */
   }

   /* output each array element's value */
   for (j = 0; j < 10; j++ )
   {
      printf("Element[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```

Output :

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

- **Print 10,20,30,40,50 as integer type using array.**

```
#include <stdio.h>
int main()
{
  int marks[5];
  marks[0]=10;
  marks[1]=20;
  marks[2]=30;
  marks[3]=40;
  marks[4]=50;
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
}
```

0r,

```
#include <stdio.h>
int main()
{
  int marks[5]={10,20,30,40,50};
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
}
```

or,

```
#include <stdio.h>
int main()
{
  int marks[5]={10,20,30,40,50},i;
  for (i = 0 ; i <=4 ; i++)
  {
    printf("%d\n",marks[i]);
  }

}
```

or,

```
#include <stdio.h>
int main()
{
  int marks[5];
  printf("marks[0]=");
  scanf("%d",&marks[0]);
  printf("marks[1]=");
  scanf("%d",&marks[1]);
  printf("marks[2]=");
  scanf("%d",&marks[2]);
  printf("marks[3]=");
  scanf("%d",&marks[3]);
  printf("marks[4]=");
  scanf("%d",&marks[4]);
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int marks[5];
  scanf("%d",&marks[0]);
  scanf("%d",&marks[1]);
  scanf("%d",&marks[2]);
  scanf("%d",&marks[3]);
  scanf("%d",&marks[4]);
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int marks[5]={10,20,30,40,50},i;
 for (i = 0 ; i <=4 ; i++)
 {
  printf("marks[%d]=%d\n",i,marks[i]);
 }

}
```

or,

```
#include <stdio.h>
int main()
{
  int marks[5],i;
  for(i=0;i<=4;i++){
    scanf("%d",&marks[i]);
  }
  printf("%d %d %d %d %d",marks[0],marks[1],
  marks[2],marks[3],marks[4]);
}
```

- **Do the program given bellow using loop.**

```
#include <stdio.h>
int main()
{
 int marks[5];
 printf("marks[0]=");
 scanf("%d",&marks[0]);
 printf("marks[1]=");
 scanf("%d",&marks[1]);
 printf("marks[2]=");
 scanf("%d",&marks[2]);
 printf("marks[3]=");
 scanf("%d",&marks[3]);
 printf("marks[4]=");
 scanf("%d",&marks[4]);
 return 0;
}
```

```
#include <stdio.h>
int main()
{
 int marks[5],i;
 for(i=0;i<5;i++){
  printf("marks[%d]=",i);
  scanf("%d",&marks[i]);
 }
  return 0;
 }
```

- **Make a program that will revers the elements of a[ ] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100} as integer type using array.**

```
#include <stdio.h>
int main()
{
 int a[]={10,20,30,40,50,60,70,80,90,100};
 int i,j, b[10];

 for(i=0,j=9;i<10;i++,j--){
  b[j]=a[i];

 }
 for(i=0;i<10;i++){
 a[i]=b[i];
 }
 for(i=0;i<10;i++){
 printf("%d\n",a[i]);
 }
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int a[]={10,20,30,40,50,60,70,80,90,100};
 int i,j,temp;

 for(i=0,j=9;i<5;i++,j--){
  temp = a[j];
  a[j] = a[i];
  a[i] = temp;
 }
 for(i=0;i<10;i++){
 printf("%d\n",a[i]);
 }
 return 0;
}
```

- **Make a program that will show sum of two numbers using array.**

```
#include<stdio.h>
int main()
{
  int ara[9];
  char x[10]={'h','h'};
  int i,j;
  printf("Plase enter two value:\n");
  scanf("%d%d",&ara[0],&ara[1]);
  printf("Your result is = %d\n",ara[0]+ara[1]);
  return 0;
}
```

- **Specifying the length of an arry using macro is considered to be an excellent practice.**
  For example :

```
#include<stdio.h>
#define n 10
int main()
{
   int a[n], i;
   for(i=0; i<n; i++){
      printf("Enter the input for index %d: ",i);
      scanf("%d",&a[i]);
   }
   printf("\n Array elements are as follows:\n");
   for(i=0; i<n; i++){
      printf("%d ",a[i]);
   }
   return 0;
}
```

- **What if number of elements are lesser than the length specified?**
  int arr[10] = {45, 6, 2, 78, 5, 6};

Here,
The remaining locations of the array are filled by value 0.

```
int arr[10] = {45, 6, 2, 78, 5, 6, 0, 0, 0, 0};
```

- **Some facts about 1D array:**

1. If the number of elements are lesser than the length of the array than the rest of the locations are automatically filled by value 0.
2. Easy way to initialize an array with all zeros is given by:

   int arr[10] = {0}
3. At the time of initialization, never leave these flower brakets {} empty and also never exceed the limit of number of elements specified by the length of an array.

   int arr[5] = {}; ❌

   int arr[5] = {1, 2, 3, 4, 5}; ❌

- **Designated Initializers:**

int arr[10] = {1,0,0,0,0,2,3,0,0,0};
We want:
- 1 in position 0
- 2 in position 5
- 3 in position 6

int arr[10] = {[0] = 1, [5] = 2, [6] =3};

This way of initilization is called designated initialization. And each number in the square brackets is said to be a designator.

. No need to bother about the entries containing zeros.
int a[15] = {1,0,0,0,0,2,0,0,0,0,0,0,0,0,0}
int a[15] = {[0] = 1, [5] = 2};

.No need to bother about the order at all.

```
int a[15] = {[0] = 1, [5] = 2};

int a[15] = {[5] = 2, [0] = 1}
```

Both are same

- Designators could be any non-negative integer.
- Compiler will deduce the length of the array from the largest designator in the list.

```
int a[] = {[5] = 90, [20] = 4, [1] = 45, [49] = 78};
```

Because of this designator, maximum length of this array would be 50.,

Finally, no one can stip you from doing this:

```
int a[] = {1, 7, 5, [5] = 90, 6, [8] = 4};
```

=

```
int a[] ={1, 7, 5, 0, 0, 90, 6, 0, 4};
```

```
#include<stdio.h>
int main()
{
  int a[] = {1, 7, 5, [5] = 90, 6, [8] = 4};
  for(int i=0;i<9;i++){
    printf("%d ",a[i]);
  }
  return 0;
}
```

sizeof operator with arrays:

sizeof(name_of_arr) / sizeof (name_of_arr[0])

sizeof(name_of_arr[0])

int a[] = {1,2,3,4,5,6,7,8,9,10}

sizeof(a[0]);

sizeof(a[0]) = 4 bytes

40 / sizeof(a[0])

= 40 / 4

= 10 → number of elements

size of 1 array element x number of elements = size of whole array.

$$\text{number of elements} = \frac{\text{size of whole array}}{\text{size of 1array element}}$$

- **What is the output of the following C program fragment:**

```
#include<stdio.h>
int main()
{
  int a[] = {1,2,3,4,5,6,7,8,9,10};
  printf("%d",sizeof (a)/sizeof(a[0]));
  return 0;
}
```

Here,
The output will be : 10

Constant arrays in C:

Either one dimensional or multi-dimensional arrays can be made constant by starting the declaration with the keyword const.

For example:

```
#include<stdio.h>
int main()
{
 const int a[] = {1,2,3,4,5,6,7,8,9,10};
 a[1]=45;
 printf("%d",a[1]);
 return 0;
}
```

Here, the output will be : ERROR!

- It assures us that the program will not modify the array which may contain some valuable information.
- It also helps the compiler to catch errors by informing that there is no intention to modify this array.
- **Write a program that can store a constant value of pi.**

```
#include<stdio.h>
const float pi=3.1416;
int main()
{
  printf("%.4f",pi);
  return 0;
}
```

- **Make a program that will show sum and average of 10,20,30,40,50 using array.**

```
#include <stdio.h>
int main()
{
int marks[5]={10,20,30,40,50},i,sum=0;
  for(i=0;i<=4;i++){
    sum=sum+marks[i];
  }
  printf("Sum is: %d\n",sum);
  printf("Average is %d",sum/5);
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
int marks[5]={10,20,30,40,51},i,sum=0;
 for(i=0;i<=4;i++){
  sum=sum+marks[i];
 }
 printf("Sum is: %d\n",sum);
 printf("Average is %.2f",(float)sum/5);
 return 0;
}
```

Here,
Type cast : (float)sum/5
Using type cast we make 'sum/5' which into float was in integer type.

- **Make a program that will show sum and average of any five numbers using array.**

```
#include <stdio.h>
int main()
{
  int marks[5],i,sum=0;
  printf("marks[0]=");
  scanf("%d",&marks[0]);
  printf("marks[1]=");
  scanf("%d",&marks[1]);
  printf("marks[2]=");
  scanf("%d",&marks[2]);
  printf("marks[3]=");
  scanf("%d",&marks[3]);
  printf("marks[4]=");
  scanf("%d",&marks[4]);
  for(i=0;i<5;i++){
  sum=sum+marks[i];
  }
  printf("Sum is: %d\n",sum);
  printf("Average is %.2f",(float)sum/5);
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int marks[5],i,sum=0;
 for(i=0;i<5;i++){
  printf("marks[%d]=",i);
  scanf("%d",&marks[i]);
 }
 for(i=0;i<5;i++){
 sum=sum+marks[i];
 }
 printf("Sum is: %d\n",sum);
 printf("Average is %.2f",(float)sum/5);
 return 0;
}
```

- **Make a program that will show multiplication table of 1 to 10.**

- **Make a program that will show sum and average of any numbers using array.**

```
#include <stdio.h>
int main()
{
 int marks[100],i,sum=0,n;
 printf("How many numbers do you want to add :");
 scanf("%d",&n);
 for(i=0;i<n;i++){
 scanf("%d",&marks[i]);
 }
 for(i=0;i<5;i++){
 sum=sum+marks[i];
 }
 printf("Sum is: %d\n",sum);
 printf("Average is %.2f",(float)sum/5);
 return 0;
}
```

- **Write a program that can take some numbers and display maximum.**

```
#include <stdio.h>
int main()
{
  int a[100],i,n;
  printf("How many numbers do you want to enter
:");
```

```
scanf("%d",&n);
  for(i=0;i<n;i++){
  scanf("%d",&a[i]);
  }
  int max=a[0];
  for(i=1;i<n;i++){
    if(max<a[i]){
      max=a[i];
    }
  }
  printf("%d is maximum value.\n",max);
  return 0;
}
```

- **Write a program that can take some numbers and display minimum.**

```
#include <stdio.h>
int main()
{
  int a[100],i,n;
  printf("How many numbers do you want to enter
:");
  scanf("%d",&n);
  for(i=0;i<n;i++){
  scanf("%d",&a[i]);
  }
  int min=a[0];
```

```
 for(i=1;i<n;i++){
    if(min>a[i]){
      min=a[i];
    }
  }
  printf("%d is minimum value.\n",min);
  return 0;
}
```

- **Make a program that will show Fibonacci series using array.**

```
#include <stdio.h>
int main()
{
  int a[100],i,n;
  printf("How many Fibonacci numbers do you want
to see:");
  scanf("%d",&n);
  a[0]=0;
  a[1]=1;
  for(i=2;i<n;i++){
    a[i]=a[i-1]+a[i-2];
  }
  printf("\n");
  printf("space\n");
  printf("Hridi\n");
```

```
printf("\n");
  for(i=0;i<n;i++){
    printf("%d   ",a[i]);
  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  int i,n;
  printf("How many Fibonacci numbers do you want
to see:");
  scanf("%d",&n);
  int a[n];
  a[0]=0;
  a[1]=1;
  for(i=2;i<n;i++){
    a[i]=a[i-1]+a[i-2];
  }
  printf("\n");
  printf("space\n");
  printf("Hridi\n");
  printf("\n");
  for(i=0;i<n;i++){
    printf("%d   ",a[i]);
  }
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int a[100],i,n;
 printf("How many Fibonacci numbers do you want
to see:");
 scanf("%d",&n);
 a[0]=0;
 a[1]=1;
 if(n<101){
 for(i=2;i<n;i++){
  a[i]=a[i-1]+a[i-2];
 }
 printf("\n");
 printf("space\n");
 printf("Hridi\n");
 printf("\n");
 for(i=0;i<n;i++){
  printf("%d ",a[i]);
 }
 }
 else{
  printf("Hridi\n");
 }
 return 0;
}
```

- **Amar bondhu Masum.**

```
#include <stdio.h>
int main()
{
 int a, b, c;
  for (b = 1; b <= 3; b=b*3) {
  for (c = 7; c >= 5; c--) {
 printf ("I=%d J=%d\n",b, c);
   }
  }
  printf(". . .\n");
   while ( 1 )

   {
    b=9;
   for (c = 7; c >= 5; c--) {
    printf ("I=%d J=%d\n",b, c);
   } b++;
   if (b>9) break;
  }
 return 0;
}
```

- **Make a program that will take any of integer numbers you want into your database and if you want to find any number in your database it will show you using Linear Search.**

```
//Linear search
#include <stdio.h>
int main()
{
  int value,position=-1,i,n;
  printf("Sir, how many numbers do you want to
enter into your database :");
  scanf("%d",&n);
  int a[n];
  printf("Boss, please enter your all numbers into
your database :\n");
  for(i=0;i<n;i++){
    printf("%d no. number =",i+1);
    scanf("%d",&a[i]);
  }
  printf("Please enter your value to find in your
database :");
  scanf("%d",&value);
  for(i=0;i<n;i++){
    if(value==a[i]){
      position=i+1;
      break;
    }
  }
  if(position==-1){
    printf("The number is not found\n");
  }
```

```
 else{
 printf("The value is found at %d no.\n",position);
 }
 return 0;
}
```

0r,

```
//Linear search
#include <stdio.h>
int main()
{
  int value,position,i,n;
  printf("Sir, how many numbers do you want to
enter into your database :");
  scanf("%d",&n);
  int a[n];
  printf("Boss, please enter your all numbers into
your database :\n");
  for(i=0;i<n;i++){
    printf("%d no. number =",i+1);
    scanf("%d",&a[i]);
  }
  printf("Please enter your value to find in your
database :");
  scanf("%d",&value);
  for(i=0;i<n;i++){
    if(value==a[i]){
      position=i+1;
      break;
    }
  }
  if(position>0){
```

```
 printf("The value is found at %d no.\n",position);
}
 else{
 printf("The number is not found\n");
 }
 return 0;
}
```
or,
```
//Linear search
#include <stdio.h>
int main()
{
 int value,position,i,n,j;
 printf("Sir, how many numbers do you want to
enter into your database :");
 scanf("%d",&n);
 int a[n];
 printf("Boss, please enter your all numbers into
your database :\n");
 for(i=0;i<n;i++){
  printf("%d no. number =",i+1);
  scanf("%d",&a[i]);
 }
 printf("Please enter your value to find in your
database :");
 scanf("%d",&value);
 for(i=0;i<n;i++){
  if(value==a[i]){
   position=i+1;
   printf("The value is found at %d no.\n",position);
  }
```

```
}
for(i=0;i<n;i++){
 if(value!=a[i]){
 j=i+1;
 }
 }
 if(j>0){
 printf("The number is not found.\n");
 }
 return 0;
}
```
or,
```
//Linear search
#include <stdio.h>
int main()
{
 int value,position,i,n,j;
 printf("Sir, how many numbers do you want to
enter into your database :");
 scanf("%d",&n);
 int a[n];
 printf("Boss, please enter your all numbers into
your database :\n");
 for(i=0;i<n;i++){
 printf("%d no. number =",i+1);
 scanf("%d",&a[i]);
 }
 printf("Please enter your value to find in your
database :");
 scanf("%d",&value);
 for(i=0;i<n;i++){
```

```
 if(value==a[i]){
 position=i+1;
 printf("The value is found at %d no.\n",position);
 }
 }
 if(position==-1){
 printf("The number is not found.\n");
 }
 return 0;
}
```

- **Make a program that will find the position of any capital letter you want .**

```
#include <stdio.h>
int main()
{
  int position=-1,n,i;
  char value;
  char a[26]=
{'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R'
,'S','T','U','V','W','X','Y','Z'};
  printf("Please enter your letter in capital :");
  scanf("%c",&value);
  int j=value;
  for(i=0;i<26;i++){
  if(j==a[i]){
   position=i+1;
   break;
  }
 }
```

```
if(position==-1){
 printf("The number is not found\n");
 }
 else{
 printf("The letter is found at %d no.\n",position);
 }
 return 0;
}
```

- **Make a program that will copy all the elements from a array to another array variable .**

```
#include<stdio.h>
int main()
{
  int array1[5]={10,20,30,40,50},array2[5],i;
  printf("Array1 :");
  for(i=0;i<5;i++){
    printf("%d  ",array1[i]);
  }
  // copying all the elements from array1 to array2.
  for(i=0;i<5;i++){
    array2[i]=array1[i];
  }
  printf("\nArray2 :");
  for(i=0;i<5;i++){
    printf("%d  ",array2[i]);
  }
  return 0;
}
```

```
#include<stdio.h>
int main()
{
   int i,n;
   printf("How many numbers do you want to enter :");
   scanf("%d",&n);
   int array1[n],array2[n];
   printf("\nEnter all the numbers :\n");
   for(i=0;i<n;i++){
      printf("%d no. number = ",i+1);
      scanf("%d",&array1[i]);
   }
   printf("Array1 :");
   for(i=0;i<n;i++){
      printf("%d    ",array1[i]);
   }
   // copying all the elements from array1 to array2.
   for(i=0;i<n;i++){
      array2[i]=array1[i];
   }
   printf("\nArray2 :");
   for(i=0;i<n;i++){
      printf("%d    ",array2[i]);
   }
   return 0;

}
```

- **Make a program that will sort elements of array in ascending order.**

```
#include <stdio.h>

void main()
{
   int n;

      printf("\n\nsort elements of array in ascending
order :\n ");
      printf("-----------------------------------------\n");

   printf("Input the size of array : ");
   scanf("%d", &n);

   int arr1[n];
   int i, j, tmp;

      printf("Input %d elements in the array :\n",n);
      for(i=0;i<n;i++)
          {
      printf("element - %d : ",i);
      scanf("%d",&arr1[i]);
    }

   for(i=0; i<n; i++)
   {
      for(j=i+1; j<n; j++)
      {
         if(arr1[i] >arr1[j])
```

```
      {
            tmp = arr1[i];
            arr1[i] = arr1[j];
            arr1[j] = tmp;
         }
       }
   }
   printf("\nElements of array in sorted ascending
order:\n");
   for(i=0; i<n; i++)
   {
      printf("%d  ", arr1[i]);
   }
       printf("\n\n");
}
```

2-D/Two dimensional array :

A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x,y you would write as follows:

```
type arrayName [ x ][ y ];
```

- A two-dimensional array can be think as a table which will have x number of rows and y number of columns.
- A 2-dimensional array a, which contains three rows and four columns can be shown as below:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

- Thus, every element in array a is identified by an element name of the form a[i][j], where a is the name of the array, and i and j are the subscripts that uniquely identify each element in a.

Initializing Two-Dimensional Arrays:

- Multidimensional arrays may be initialized by specifying bracketed values for each row.
- Following is an array with 3 rows and each row has 4 columns.

```
int a[3][4] = {
 {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
 {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
 {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
```

- The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to previous example:

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Accessing Two-Dimensional Array Elements ;

An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example:

```
int val = a[2][3];
```

- The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

Example :

Let us check below program where we have used nested loop to handle a two dimensional array:

```
#include <stdio.h>

int main ()
{
   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

   /* output each array element's value */
   for ( i = 0; i < 5; i++ )
   {
      for ( j = 0; j < 2; j++ )
      {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }
   return 0;
}
```

Output :

```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

- **Make a program that will show given bellow using 2-D array.**

  1 2 3 4
  5 6 7 8
  9 10 11 12

```
#include <stdio.h>
int main()
{
 int a[3][4]={
  {1,2,3,4},
  {5,6,7,8},
  {9,10,11,12}
 };
 printf("%d ",a[0][0]);
 printf("%d ",a[0][1]);
 printf("%d ",a[0][2]);
 printf("%d ",a[0][3]);
 printf("\n");

 printf("%d ",a[1][0]);
 printf("%d ",a[1][1]);
 printf("%d ",a[1][2]);
 printf("%d ",a[1][3]);
 printf("\n");

 printf("%d ",a[2][0]);
 printf("%d ",a[2][1]);
 printf("%d ",a[2][2]);
 printf("%d ",a[2][3]);
 printf("\n");
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}},i,j;
 for(i=0;i<=2;i++){
  for(j=0;j<=3;j++){
   printf("%d ",a[i][j]);
  }
  printf("\n");
 }
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int n, m, i, j;
 printf("Enter the Raw :");
 scanf("%d", &n);
 printf("Enter the Column :");
 scanf("%d", &m);
 int a[n][m];
 for (i = 0; i < n; i++)
 {
  for (j = 0; j < m; j++)
  {
   printf("a[%d][%d]=",i,j);
   scanf("%d", &a[i][j]);
   if (j == m - 1)
   {
```

```
 printf("\n");
 }
 }
 }
 for (i = 0; i < n; i++)
 {
 for (j = 0; j < m; j++)
 {
 printf("%d ", a[i][j]);
 }
 printf("\n");
 }
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 int n, m, i, j;
 printf("Enter the RAw :");
 scanf("%d", &n);
 printf("Enter the Column :");
 scanf("%d", &m);
 int A[n][m];
 //Scaning A matrix
 printf("Enter elements for A matrix :\n");
 for (i = 0; i < n; i++)
 {
  for (j = 0; j < m; j++)
  {
```

```
printf("A[%d][%d]=",i,j);
 scanf("%d", &A[i][j]);
 }
 printf("\n");
 }
 //Printing A matrix
 printf("A=");
 for (i = 0; i < n; i++)
 {
 printf("\t");
 for (j = 0; j < m; j++)
 {
 printf("%d ", A[i][j]);
 }
 printf("\n");
 }
 return 0;
}
```

- **Make a program that will show sum of A matrix and B matrix using array.**

```
#include <stdio.h>
int main()
{
 int n, m, i, j;
 printf("\nEnter the RAw :");
 scanf("%d", &n);
 printf("Enter the Column :");
 scanf("%d", &m);
 int A[n][m],B[n][m],C[n][m];
 printf("\n");
```

```
 //Scaning A matrix
 printf("Enter elements for A matrix :\n\n");
 for (i = 0; i < n; i++)
 {
 for (j = 0; j < m; j++)
 {
  printf("A[%d][%d]=",i,j);
  scanf("%d", &A[i][j]);
 }
 printf("\n");
 }
//Printing A matrix
 printf("A=");
 for (i = 0; i < n; i++)
 {
 printf("\t");
 for (j = 0; j < m; j++)
 {
  printf("%d ", A[i][j]);
 }
 printf("\n\n");
 }
 //Scaning B matrix
 printf("\nEnter elements for B matrix :\n\n");
 for (i = 0; i < n; i++)
 {
  for (j = 0; j < m; j++)
  {
   printf("B[%d][%d]=",i,j);
   scanf("%d", &B[i][j]);
```

```
 }
 printf("\n");
 }

 //Printing B matrix
 printf("\nB=");
 for (i = 0; i < n; i++)
 {
  printf("\t");
  for (j = 0; j < m; j++)
  {
   printf("%d ", B[i][j]);
  }
  printf("\n\n");
 }
 //Adding the matrix
 printf("A+B = ");
 for (i = 0; i < n; i++)
 {
  printf("\t");
  for (j = 0; j < m; j++)
  {
   C[i][j]=A[i][j]+B[i][j];
   printf("%d ", C[i][j]);
  }
  printf("\n\n");
 }
 return 0;
}
```

- **Make a program that will show sum and subtraction of A and B matrix using array.**

```
#include <stdio.h>
int main()
{
 int n, m, i, j;
 printf("\nEnter the RAw :");
 scanf("%d", &n);
 printf("Enter the Column :");
 scanf("%d", &m);
 int A[n][m],B[n][m],C[n][m];
 printf("\n");
 //Scaning A matrix
 printf("Enter elements for A matrix :\n\n");
 for (i = 0; i < n; i++)
 {
  for (j = 0; j < m; j++)
  {
   printf("A[%d][%d]=",i,j);
   scanf("%d", &A[i][j]);
  }
  printf("\n");
 }
 //Scaning B matrix
 printf("\nEnter elements for B matrix :\n\n");
```

```
for (i = 0; i < n; i++)
 {
 for (j = 0; j < m; j++)
 {
 printf("B[%d][%d]=",i,j);
 scanf("%d", &B[i][j]);
 }
printf("\n");
 }
 //Printing A matrix
 printf("A=");
 for (i = 0; i < n; i++)
 {
 printf("\t");
 for (j = 0; j < m; j++)
 {
 printf("%d ", A[i][j]);
 }
 printf("\n\n");
 }
 //Printing B matrix
 printf("\nB=");
 for (i = 0; i < n; i++)
 {
 printf("\t");
 for (j = 0; j < m; j++)
 {
 printf("%d ", B[i][j]);
 }
 printf("\n\n");
 }
```

```
//Adding the matrix
 printf("A+B = ");
 for (i = 0; i < n; i++)
 {
 printf("\t");
 for (j = 0; j < m; j++)
 {

 C[i][j]=A[i][j]+B[i][j];
 printf("%d ", C[i][j]);
 }
 printf("\n\n");
 }
 //Substraction the matrix
 printf("A-B = ");
 for (i = 0; i < n; i++)
 {
 printf("\t");
 for (j = 0; j < m; j++)
 {
 C[i][j]=A[i][j]-B[i][j];
 printf("%d ", C[i][j]);
 }
 printf("\n\n");
 }
 return 0;
}
```

- **Make a program that will show multiplication of two matrix A and B using array.**

```
#include <stdio.h>
int main()
{
 int r1, c1, r2, c2, i, j,k,sum=0;
 printf("\nEnter the Raw and Column for A matrix
:");
 scanf("%d%d", &r1, &c1);
 printf("\nEnter the Raw and Column for B matrix
:");
 scanf("%d%d", &r2, &c2);
 while (c1!= r2)
 {
  printf("\nError!! column of first matrix isd not
equal to raw of second matrix\n");

  printf("\nEnter the Raw and Column for A matrix
:");
 scanf("%d%d", &r1, &c1);
 printf("\nEnter the Raw and Column for B matrix
:");
 scanf("%d%d", &r2, &c2);
 }
 int A[r1][c1], B[r2][c2],C[r1][c2];
 printf("\n");
// Scaning A matrix
 printf("Enter elements for A matrix :\n\n");
```

```
for (i = 0; i < r1; i++)
{
for (j = 0; j < c1; j++)
{
printf("A[%d][%d]=", i, j);
scanf("%d", &A[i][j]);
}
printf("\n");
}
// Scaning B matrix
printf("\nEnter elements for B matrix :\n\n");
for (i = 0; i < r2; i++)
{
for (j = 0; j < c2; j++)
{
printf("B[%d][%d]=", i, j);
scanf("%d", &B[i][j]);
}
printf("\n");
}
// Printing A matrix
printf("A=");
for (i = 0; i < r1; i++)
{
printf("\t");
for (j = 0; j < c1; j++)
{
printf("%d ", A[i][j]);
}
printf("\n\n");
}
```

```
// Printing B matrix
 printf("\nB=");
for (i = 0; i < r2; i++)
 {
 printf("\t");
 for (j = 0; j < c2; j++)
 {
 printf("%d ", B[i][j]);
 }
 printf("\n\n");
 }
 //Multiplying matrix
 printf("A*B = ");
 for (i = 0; i < r1; i++)
 {
 printf("\t");
 for (j = 0; j < c2; j++)
 {
 for(k=0;k<c1;k++){
 sum=sum+A[i][k]*B[k][j];
 }
 C[i][j]=sum;
 sum=0;
 printf("%d ", C[i][j]);
 }
 printf("\n\n");
 }
 return 0;
}
```

- **Make a program that will show transpose matrix of A matrix using array.**

```
#include <stdio.h>
int main()
{
 int r, c, i, j;
 printf("\nEnter the Raw of A matrix :");
 scanf("%d", &r);
 printf("Enter the Column of A matrix :");
 scanf("%d", &c);
 int A[r][c],T[c][r];
 printf("\n");
 //Scaning A matrix
 printf("Enter elements for A matrix :\n\n");
 for (i = 0; i < r; i++)
 {
  for (j = 0; j < c; j++)
  {
   printf("A[%d][%d]=",i,j);
   scanf("%d", &A[i][j]);
  }
  printf("\n");
 }
 //Printing A matrix
 printf("A=");
 for (i = 0; i < r; i++)
 {
  printf("\t");
```

```
 for (j = 0; j < c; j++)
 {
 printf("%d ", A[i][j]);
 }
 printf("\n\n");
 }
 //Transpose matrix
 for (i = 0; i < r; i++)
 {
 for (j = 0; j < c; j++)
 {
 T[j][i]=A[i][j];
 }
 }
 //Printing Transpose matrix
 printf("\nT=");
 for (i = 0; i < c; i++)
 {
 printf("\t");
 for (j = 0; j < r; j++)
 {
 printf("%d ", T[i][j]);
 }
 printf("\n\n");
 }
 return 0;
}
```

- **Make a program that will show sum of diagonal of A matrix using array.**

```
#include <stdio.h>
int main()
{
 int r, c, i, j,sum=0;
 printf("\nEnter the Raw of A matrix :");
 scanf("%d", &r);
 printf("Enter the Column of A matrix :");
 scanf("%d", &c);
 int A[r][c];
 printf("\n");
 //Scaning A matrix
 printf("Enter elements for A matrix :\n\n");
 for (i = 0; i < r; i++)
 {
  for (j = 0; j < c; j++)
  {
   printf("A[%d][%d]=",i,j);
   scanf("%d", &A[i][j]);
  }
  printf("\n");
 }
 //Printing A matrix
 printf("A=");
 for (i = 0; i < r; i++)
 {
  printf("\t");
```

```
 for (j = 0; j < c; j++)
 {
 printf("%d ", A[i][j]);
 }
  printf("\n\n");
 }
 //sum of diagonal elements
 printf("Diagonal elements : ");
 for (i = 0; i < r; i++)
 {
 for (j = 0; j < c; j++)
 {
 if(i==j){
 printf("%d ",A[i][j]);
 sum=sum+A[i][j];
 }
 }
 }
 printf("\nSum of diagonal is %d",sum);
 return 0;
}
```

- **Make a program that will show upper and lower triangle elements of A matrix using array.**

```
#include <stdio.h>
int main()
{
 int r, c, i, j,uppersum=0,lowersum=0;
 printf("\nEnter the Raw of A matrix :");
 scanf("%d", &r);
 printf("Enter the Column of A matrix :");
 scanf("%d", &c);
 int A[r][c];
 printf("\n");
 //Scaning A matrix
 printf("Enter elements for A matrix :\n\n");
 for (i = 0; i < r; i++)
 {
  for (j = 0; j < c; j++)
  {
   printf("A[%d][%d]=",i,j);
   scanf("%d", &A[i][j]);
  }
  printf("\n");
 }
 //Printing A matrix
 printf("A=");
 for (i = 0; i < r; i++)
 {
  printf("\t");
```

```
 for (j = 0; j < c; j++)
 {
 printf("%d ", A[i][j]);
 }
 printf("\n\n");
 }
 //sum of upper and lower triangle matrix
 for (i = 0; i < r; i++)
 {
 for (j = 0; j < c; j++)
 {
 if(i<j){
 uppersum=uppersum+A[i][j];
 }
 if(i>j){
 lowersum=lowersum+A[i][j];
 }
 }
 }
 printf("Sum of upper triangle elements is
%d",uppersum);
 printf("\nSum of lower triangle elements is
%d",lowersum);
 return 0;
}
```

3-D/Three dimensional array :

A 3D array is a multi-dimensional array(array of arrays). A 3D array is a collection of 2D arrays. It is specified by using three subscripts:Block size, row size and column size. More dimensions in an array means more data can be stored in that array.

C programming language allows multidimensional arrays. Here is the general form of a multidimensional array declaration:

```
type name[size1][size2]...[sizeN];
```

- For example, the following declaration creates a three dimensional 5 . 10 . 4 integer array:

```
int threedim[5][10][4];
```

- As explained above, you can have arrays with any number of dimensions, although it is likely that most of the arrays you create will be of one or two dimensions.

Visualizing 3D array:

If we want to visualize a 2D array, we can visualize it in this way:
int arr[3][3], it means a 2D array of type integer having 3 rows and 3 columns.It is just a simple matrix

int arr[3][3]; //2D array contaning 3 rows and 3 columns

1 2 3
4 5 6
7 8 9
3x3

But, what happen if we add one more dimension here,
i.e, int arr[3][3][3], now it becomes a 3D array.

- int shows that the 3D array is an array of type integer.
- arr is the name of array.
- first dimension represents the block size(total number of 2D arrays).
- second dimension represents the rows of 2D arrays.
- third dimension represents the columns of 2D arrays.
- i.e; int arr[3][3][3], so the statement says that we want three such 2D arrays which consists of 3 rows and 3 columns.

int arr[3][3][3]; //3D array

block(1)	block(2)	block(3)
1 2 3	10 11 12	19 20 21
4 5 6	13 14 15	22 23 24
7 8 9	16 17 18	25 25 27
3x3	3x3	3x3

Declaring a 3D array:

To declare 3D array:

- Specify data type, array name, block size, row size and column size.
- Each subscript can be written within its own separate pair of brackets.
- Syntax: data_type array_name[block_size][row_size][column_size];
  Example:

```
int arr[2][3][3];      //array of type integer
                       //number of blocks of 2D arrays:2
                         |rows:3 |columns:3
                       //number of elements:2*3*3=18
```

```
block(1)  11 22 33        block(2)  12 13 14
          44 55 66                  21 31 41
          77 88 99                  12 13 14
                3x3                       3x3
```

Ways to declare 3D array:

1). int arr[2][3][3];
In this type of declaration, we have an array of type integer, block size is 2, row size is 3 and column size is 3.Here we have not stored any values/elements in the array.So the array will hold the garbage values.

int arr[2][3][3]; //no elements are stored

```
 block(1)                block(2)
1221 -543 3421       654 5467 -878     //all values are
3342 6543 4221       456 1567 7890     //garbage values
-564  4566 -345      567 6561 2433
                  3x3                        3x3
```

2). int arr[2][3][3]={};
In this type of declaration, we have an array of type integer, block size is 2, row size is 3 and column size is 3 and we have put curly braces after assignment operator.So the array will hold 0 in each cells of array.

int arr[2][3][3]={}; //0 will be stored

```
block(1)  0 0 0       block(2)  0 0 0
          0 0 0                 0 0 0
          0 0 0                 0 0 0
            3x3                     3x3
```

3). int arr[3][2][2]={0,1,2,3,4,5,6,7,8,9,3,2}
In this type of declaration, we have an array of type integer, block size is 3,
row size is 2, column size is 2 and we have mentioned the values inside the curly braces during the declaration of array. So all the values will be stored one by one in the array cells.

int arr[3][2][2]={0,1,2,3,4,5,6,7,8,9,3,2}

block(1) 0 1 block(2) 4 5 block(3) 8 9
2 3 6 7 3 2
2x2 2x2 2x2

4). int arr[3][3][3]=
{ {{10,20,30},{40,50,60},{70,80,90}},
{{11,22,33},{44,55,66},{77,88,99}},
{{12,23,34},{45,56,67},{78,89,90}}
};
In this type of declaration, we have an array of type integer, block size is 3,
row size is 3, column size is 3 and the values of each blocks are assigned during its declaration.

int arr[3][3][3]=
{ {{10,20,30},{40,50,60},{70,80,90}}, // block 1
{{11,22,33},{44,55,66},{77,88,99}}, //block 2
{{12,23,34},{45,56,67},{78,89,90}} //block 3
};

block(1)10 20 30 block(2)11 22 33 block(3)12 23 34
40 50 60 44 55 66 45 56 67
70 80 90 77 88 99 78 89 90
3x3 3x3 3x3

Inserting values in 3D array:

In 3D array, if a user want to enter the values then three for loops are used.

- First for loop represents the blocks of a 3D array.
- Second for loop represents the number of rows.
- Third for loop represents the number of columns.

Example:

Following is the implementation in C:

```
#include<stdio.h>
int i,j,k;                    //variables for nested for
loops
int main()
{
 int arr[2][3][3];            //array declaration
 printf("enter the values in the array: \n");
 for(i=1;i<=2;i++)            //represents block
 {
  for(j=1;j<=3;j++)          //represents rows
  {
   for(k=1;k<=3;k++)        //represents columns
   {
    printf("the value at arr[%d][%d][%d]: ",i,j,k);
    scanf("%d",&arr[i][j][k]);
   }
  }
 }
 printf("printing the values in array: \n");
```

```
for(i=1;i<=2;i++)
 {
  for(j=1;j<=3;j++)
  {
   for(k=1;k<=3;k++)
   {
    printf("%d ",arr[i][j][k]);
    if(k==3)
    {
     printf("\n");
    }
   }
  }
  printf("\n");
 }
 return 0;
}
```

Here,
The output will be

```
enter the values in the array:

the value at arr[1][1][1]: 23
the value at arr[1][1][2]: 34
the value at arr[1][1][3]: 45
the value at arr[1][2][1]: 76
the value at arr[1][2][2]: 78
the value at arr[1][2][3]: 98
the value at arr[1][3][1]: 87
```

the value at arr[1][3][2]: 67
the value at arr[1][3][3]: 98
the value at arr[2][1][1]: 34
the value at arr[2][1][2]: 23
the value at arr[2][1][3]: 67
the value at arr[2][2][1]: 58
the value at arr[2][2][2]: 19
the value at arr[2][2][3]: 84
the value at arr[2][3][1]: 39
the value at arr[2][3][2]: 82
the value at arr[2][3][3]: 44

printing the values in array:
23 34 45
76 78 98
87 67 98

34 23 67
58 19 84
39 82 44

In the above program;

- We have declared three variables i,j,k for three for loops.
- we have declared an array of type integer int arr[2][3][3];(blocks:2 rows:3 columns:3)
- First section of nested for loops ask the user to insert the values.
- second section of nested for loops will print the inserted values in the matrix form.

Updating the 3D array:

We can update the elements of 3D array either by specifying the element to be replaced or by specifying the position where replacment has to be done.
For updating the array we require,

- Elements of an array
- Element or position, where it has to be inserted
- The value to be inserted

Example

Following is the implementation in C:

```
#include<stdio.h>
int i,j,k;                    //variables for nested for loop
int num;                      //will hold the value to be
replaced
int main()
{
 int arr[2][3][3];           //3D array declaration
 printf("enter the values in the array: \n\n");
 for(i=1;i<=2;i++)           //represents block
 {
  for(j=1;j<=3;j++)        //represents rows
  {
   for(k=1;k<=3;k++)    //represents columns
   {
    printf("the value at arr[%d][%d][%d]: ",i,j,k);
    scanf("%d",&arr[i][j][k]);
```

```
 }
 }
}
 printf("\nprinting the values in array: \n");
 for(i=1;i<=2;i++)
 {
  for(j=1;j<=3;j++)
  {
   for(k=1;k<=3;k++)
   {
    printf("%d ",arr[i][j][k]);
    if(k==3)
    {
     printf("\n");
    }
   }
  }
  printf("\n");
 }
 printf("\nenter the block row and column number:
\n");
 scanf("%d %d %d ",&i,&j,&k);         //position where
we want to update the element
 printf("enter the new number you want to update
with: ");
 scanf("%d",&num);                 //element to be
replaced
 arr[i][j][k]=num;                 //element assigned to
array position
 printf("\narray after updating: \n");
```

```
for(i=1;i<=2;i++)
 {
  for(j=1;j<=3;j++)
  {
   for(k=1;k<=3;k++)
   {
    printf("%d ",arr[i][j][k]);
    if(k==3)
    {
     printf("\n");
    }
   }
  }
  printf("\n");
 }
 return 0;
}
```

Here,
The output will be :
enter the values in the array:

the value at arr[1][1][1]: 11
the value at arr[1][1][2]: 22
the value at arr[1][1][3]: 33
the value at arr[1][2][1]: 44
the value at arr[1][2][2]: 55
the value at arr[1][2][3]: 66
the value at arr[1][3][1]: 77
the value at arr[1][3][2]: 88

the value at arr[1][3][3]: 99
the value at arr[2][1][1]: 10
the value at arr[2][1][2]: 20
the value at arr[2][1][3]: 30
the value at arr[2][2][1]: 40
the value at arr[2][2][2]: 50
the value at arr[2][2][3]: 60
the value at arr[2][3][1]: 70
the value at arr[2][3][2]: 80
the value at arr[2][3][3]: 90

printing the values in array:
11 22 33
44 55 66
77 88 99

10 20 30
40 50 60
70 80 90
enter the block row and column number:2 1 1
enter the new number you want to update with: 15

array after updating:
11 22 33
44 55 66
77 88 99

15 20 30
40 50 60
70 80 90

In the above program;

- We have declared three variables i,j,k for three for loops and num variable which will hold the value/element to be updated.
- we have declared an array of type integer int arr[2][3][3];(blocks:2 rows:3 columns:3)
- First section of nested for loops ask the user to insert the values.
- second section of nested for loops will print the inserted values in the matrix form.
- Position/value will be updated.
- Third section of nested for loop will print the updated 3D array.

- **Make a program that will Convert 3D array into 2D array.**

```
#include<stdio.h>
int main()
{
 int i,j,k;              //variables for nested for loop
 int arr[2][3][3];       //declaration of 3D array
 printf("enter the elements: \n");
 for(i=0;i<2;i++)
 {
  for(j=0;j<3;j++)
  {
   for(k=0;k<3;k++)
   {
```

```
 printf("element at [%d][%d][%d]: ",i,j,k);
   scanf("%d",&arr[i][j][k]);
   //arr[i][j][k]=++ctr;
  }
 }
}
printf("\nprinting 3D array\n");
for(i=0;i<2;i++)                    //printing 3d array
{
 for(j=0;j<3;j++)
 {
  for(k=0;k<3;k++)
  {
   printf("%d ",arr[i][j][k]);
  }
  printf("\n");
 }
 printf("\n");
}
int a[3][3];                  //2D array declaration
int b[3][3];
printf("\ncopying values in new 2D array: \n");
for(i=0;i<2;i++)
{
 for(j=0;j<3;j++)
 {
  for(k=0;k<3;k++)
  {
```

```
if(i==0)
   {
    a[j][k]=arr[i][j][k];    //copying values in new 2d
array
   }
   else
   {
    b[j][k]=arr[i][j][k];
   }
  }
 }
}
 printf("\nprinting elements in first 2D array: \n");
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)                //printing first 2d array
  {
   printf("%d ",a[i][j]);
  }
  printf("\n");
 }
 printf("\n");
 printf("\nprinting elements in second 2D array: ");
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)                //printing 2nd 2d array
  {
   printf("%d ",b[i][j]);
  }
```

```
printf("\n");
 }
 return 0;
}
```

- **Make a program that will convert 2D array into 3D array**

```
#include<stdio.h>
int main()
{
 int i,j,k;
 int a[3][3];                  //2D array declaration
 int b[3][3];
 printf("enter the elements in array a: \n");
//entering elements in array 'a'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("element at [%d][%d]: ",i,j);
   scanf("%d",&a[i][j]);
  }
 }
 printf("\nprinting 2D array a\n");                //printing
elements of array 'a'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("%d ",a[i][j]);
  }
```

```
 printf("\n");
 }
 printf("\nenter the elements in array b: \n");
//entering elements in array 'b'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("element at [%d][%d]: ",i,j);
   scanf("%d",&b[i][j]);
  }
 }
 printf("\nprinting 2D array b\n");                //printing
elements of array 'b'
 for(i=0;i<3;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("%d ",b[i][j]);
  }
  printf("\n");
 }
 int arr[3][3][3];                              //3D array
declaration
 printf("\ncopying values in 3D array: \n");
 for(i=0;i<2;i++)
 {
  for(j=0;j<3;j++)
  {
   for(k=0;k<3;k++)
```

```
{
   if(i==0)
   {
    arr[i][j][k]=a[j][k];              //copying elements of
2D array into 3D array
   }
   else
   {
    arr[i][j][k]=b[j][k];
   }
  }
 }
}
 printf("\nprinting elements in 3D array: \n");
//printing elements of 3D array
 for(i=0;i<2;i++)
 {
  for(j=0;j<3;j++)
  {
   for(k=0;k<3;k++)
   {
   printf("%d ",arr[i][j][k]);
   if(k==2)
   {
   printf("\n");
   }
   }
  }
```

```
 printf("\n");
 }
 return 0;
}
```

- **Dynamically allocating memory using malloc in 3D array.**

As we know that static array variables are fixed in size and can't be changed(enlarged or shrinked).To remove this drawback we use dynamic memory allocation.dynamic array is nothing but it is allocated during runtime with malloc or calloc.
*syntax: int *array=int(int *)malloc(sizeof(int)element-count);
Example :

```
#include <stdio.h>
#include <malloc.h>              //malloc library
int main(int argc, char* argv[]) //command line arguments
{
  int ***arr;                    //triple pointer
  int block,row,column;          //variables for block, rows and columns
  int i,j,k;                     //nested for loop
  printf("enter the blocks, rows and columns: ");
  scanf("%d %d %d",&block,&row,&column);
  arr=(int ***)malloc(sizeof(int ***)*block);
  for(i=0;i<block;i++) {
  arr[i]=(int **)malloc(sizeof(int*)*row);
```

```
 for(j=0;j<row;j++)
{
    arr[i][j]=(int *)malloc(sizeof(int)*column);
  }
 }
 for(i=0;i<block;i++)
 {
  for(j=0;j<row;j++)
{
    for(k=0;k<column;k++)
 {
      printf("element at [%d][%d][%d] : ",i,j,k);
      scanf("%d",&arr[i][j][k]);
    }
  }
 }
 printf("Printing 3D Array:\n");
 for(i=0;i<block;i++)
 {
  for(j=0;j<row;j++)
{
    for(k=0;k<column;k++)
 {
      printf("%.2d ",arr[i][j][k]);
    }
    printf("\n");
  }
  printf("\n");
 }
 return 0;
}
```

12.strings

Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

- **Make a program that will show your name using strings.**

```
#include <stdio.h>
int main()
{
 char c[6];
 c[0]='H';
 c[1]='a';
 c[2]='m';
 c[3]='i';
 c[4]='m';
 c[5]='\0';
 printf("c = %s\n",c);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char c[6]={'H','a','m','i','m','\0'};
 printf("c = %s\n",c);
 return 0;
}
```

- **What will be the output of given bellow.**

```
#include <stdio.h>
int main()
{
   char c[6]={'H','a','m','i','m','\0'};
   printf("c = %c\n",c[0]);
   return 0;
 }
```

Here,
The output will be c = H

- **Make a program that will show your short name using strings.**

```
#include <stdio.h>
int main()
{
 char c[30];
 printf("Enter your name : ");
 scanf("%s",&c);
 printf("Your name is %s\n",c);
 return 0;
}
```

- **Make a program that will show your full name using strings.**

```
#include <stdio.h>
int main()
{
 char c[ ]="Hamim Talukder";
 printf("c = %s\n",c);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char c[15]="Hamim Talukder";
 printf("c = %s\n",c);
 return 0;
}
```

Here,
We also need a character variable for giving space between Hamim and Talukder.

or,

```
#include <stdio.h>
int main()
{
 char c[30];
 printf("Enter your full name : ");
 gets(c);
 printf("Your full name is %s\n",c);
 return 0;
}
```

or,

```
#include <stdio.h>
#include<string.h>
int main()
{
 char c[]="Hamim Talukder";
 printf("Enter your full name : ");
 gets(c);
 printf("Your full name is %s\n",c);
 return 0;
}
```

Here,
gets() function works as scanf() function do. Here scanf() function can not diagnosis the separate word after space but gets() function can do.

- **Make a program that will show your full name and the first letter of your name using strings.**

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[100],str2[100],str3[100];
  scanf(" %[^\n]",&str1);
  printf("%s\n",str1);
  printf("%c\n",str1[0]);
  return 0;
}
```

- **Make a program that will show multiple strings.**

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str1[100],str2[100],str3[100];
  printf("Enter: ");
  scanf(" %[^\n]",&str1);
  printf("Enter: ");
  scanf(" %[^\n]",&str2);
  printf("Enter: ");
  scanf(" %[^\n]",&str3);
```

```
 printf("%s\n",str1);
  printf("%s\n",str2);
  printf("%s\n",str3);
  return 0;
}
```

- **Make a program that will show the full name of yours and your brother and sister using strings.**

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[100],str2[100],str3[100];
   printf("Enter your full name: ");
   fflush(stdin);
   gets(str1);
   printf("Enter your brother's full name: ");
   gets(str2);
   printf("Enter your sister's full name: ");
   gets(str3);
   printf("\n");
   printf("Your name is: %s\n",str1);
   printf("Your brother name is: %s\n",str2);
   printf("Your sister name is: %s\n",str3);
   return 0;
}
```

Here,
fflush(stdin) is a library function that is used to solve the string assigning problem.

- **Make a program that will show multiple line using strings.**

```
#include <stdio.h>
int main()
{
 char c[]="Hamim Talukder \
 Taseen";
 printf("c = %s\n",c);
 return 0;
}
```

- **Make a program that will show all the character/letter Of 'Hamim'.**

```
#include <stdio.h>
int main()
{
 char c[]="Hamim";
 printf("%c\n",c[0]);
 printf("%c\n",c[1]);
 printf("%c\n",c[2]);
 printf("%c\n",c[3]);
 printf("%c\n",c[4 ]);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char c[]="Hamim";
 int i;
 for(i=0;c[i]!='\0';i++){
 printf("%c\n",c[i]);
 }
 return 0;
}
```

2D character arrays of string :

Array of Character arrays

V	I	H	A	N	A	\0				
A	D	H	R	I	T	T	\0			
M	E	E	R	A	\0					
D	I	Y	A	\0						

- Ex:

```
char names[4][10];
```

No of Items Size of Item

- **Make a program that will show four names Hamim, Jim, Rahim, Mithu using 2D character arrays of string.**

```
#include<stdio.h>
#include<string.h>
int main()
{
  int n,i,j,k;
  char arr[4][10]={"Hamim","Jim","Rahim","Mithu"};
  printf("The names are :\n",n);
  for(i=0;i<4;i++){
    printf("%s\n",arr[i]);
  }
  return 0;
}
```

- **Make a program that will show some words depending on user input using 2D character arrays of string.**

```
#include<stdio.h>
#include<string.h>
int main()
{
  int n,i,j,k;
  printf("How many words do you want to enter : ");
  scanf("%d",&n);
  char arr[n][100];
  printf("Enter %d words :\n",n);
  for(i=0;i<n;i++){
    scanf("%s",&arr[i]);
  }
  printf("The %d words are :\n",n);
  for(i=0;i<n;i++){
    printf("%s\n",arr[i]);
  }
  return 0;
}
```

- **Write a program that will show given bellow:**
  How many words do you want to enter : 2
  Enter 2 words :
  Asdfghjklyryt
  Hamim
  The 2 modified words are :
  A11t
  Hamim

```
#include<stdio.h>
#include<string.h>
int main()
{
  int n,i,j,k;
  printf("How many words do you want to enter : ");
  scanf("%d",&n);
  char arr[n][100];
  printf("Enter %d words :\n",n);
  for(i=0;i<n;i++){
    scanf("%s",&arr[i]);
  }
  printf("The %d modified words are :\n",n);
  for(i=0;i<n;i++){
    if(strlen(arr[i])>10){
      printf("%c%d%c\n",arr[i][0],strlen(arr[i])-2,arr[i]
[strlen(arr[i])-1]);
    }
    else{
      printf("%s\n",arr[i]);
    }
  }
  return 0;
}
```

- **Make a program that will find the string length of 'Hamim Talukder'.**

```
#include <stdio.h>
int main()
{
  char c[] = "Hamim Talukder";
  int length;
  length = strlen(c);
  printf("String length is %d\n", length);
  return 0;
}
```

Here,
strlen() is a library function that shows us the string length of any strings.

```
#include <stdio.h>
int main()
{
  char c[] = "Hamim Talukder";
  int i = 0, j = 0;
  while (c[i] != '\0')
  {
    i++;
    j++;
  }
  printf("String length is %d", j);
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  char c[] = "Hamim Talukder";
  int i = 0;
  while (c[i] != '\0')
  {
    i++;
  }
  printf("String length is %d", i);
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char c[]="Hamim Talukder";
 int i,j=0;
 for(i=0;c[i]!='\0';i++){
 j++;
 }
 printf("String length is %d",j);
 return 0;
}
```

- **Make a program that can copy any strings.**

```
#include <stdio.h>
int main()
{
 char source[ ] = "C Programming";
 char target[20];
 strcpy(target, source);
 printf("Source string is = %s \n", source);
 printf("Target string is = %s \n", target);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char source[15]="Hamim Talukder";
 printf("Source string = %s\n",source);
 int i,k=0;
 for(i=0;source[i]!='\0';i++){
  k++;
 }
 printf("String length is %d\n",k);
 int j=k+1;
 char target[j];
 for(i=0;source[i]!='\0';i++){
  target[i]=source[i];
 }
 printf("Target string = %s\n",target);
 return 0;
}
```

Here,
strcpy() is a library function that can copy characters from a string and paste it in another string .

or,

```
#include <stdio.h>
int main()
{
 char source[15]="Hamim Talukder";
 printf("Source string = %s\n",source);
 int i,k=0;
 for(i=0;source[i]!='\0';i++){
  k++;
 }
 printf("String length is %d\n",k);
 int j=k+1;
 char target[j];
 for(i=0;source[i]!='\0';i++){
  target[i]=source[i];
 }
 printf("Target string = ");
 for(j=0;target[j]!='\0';j++){
 printf("%c",target[j]);
 }
 return 0;
}
```

- **Make a program that can strings concatenation .**

```
#include <stdio.h>
#include<string.h>
int main()
{
 char str1[]="My name is ";
 char str2[]="Hamim Talukder";
 strcat(str1,str2);
 printf("%s\n",str1);
 return 0;
}
```

Here,
strcat() is a library function that can concat/add two strings.

or,

```
#include <stdio.h>
int main()
{
 char str1[]="My name is ";
 //char str2[]="Hamim Talukder";
 strcat(str1,"Hamim Talukder");
 printf("%s\n",str1);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char str1[]="My name is_";
 char str2[]="Hamim Talukder";
 strcat(str1,str2);
 printf("%s\n",str1);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char str1[50]="My name is ";
 char str2[]="Hamim Talukder";
 int i,len1=0,len2=0;
 for(i=0;str1[i]!='\0';i++){
 len1++;
 }
 for(i=0;str1[i]!='\0';i++){
 str1[len1+i]=str2[i];
 }
 printf("%s\n",str1);
}
```

- **Make a program that can compare stringts.**

```
#include<stdio.h>
int main()
{
 char str1[]="Hamim Talukder";
 char str2[]="Hamim Talukder";
 int d=strcmp(str1,str2);
 if(d==0){
  printf("Strings are equal.");
 }
 else
 printf("Strings are not equal.");
 return 0;
}
```

Here,
strcmp() is a library function that can compare among strings.

or,

```
#include<stdio.h>
int main()
{
 char str1[]="Hamim Talukder";
 char str2[]="Hamim Talukder";
 int i,d=0,k,s;
 k= strlen(str1);
 s= strlen(str2);
 printf("k=%d\n",k);
 printf("s=%d\n",s);
 for(i=0;str1[i]!='\0';i++){
```

```
 if(str1[i]!=str2[i]){
    d=d+1;
 }
 }
 printf("d=%d\n",d);
 if(d==0){
  printf("Strings are equal.\n");
 }
 else
 printf("Strings are not equal.\n");
 return 0;
}
```

or,

```
#include<stdio.h>
int main()
{
 char str1[]="Hamim Talukder";
 char str2[]="Hamim Talukder";
 int i,j=0,k,s;
 k= strlen(str1);
 s= strlen(str2);
 printf("%d\n",k);
 printf("%d\n",s);
 for(i=0;str1[i]!='\0';i++){
if(str1[i]==str2[i]){
 j=j+1;
 }
 }
 printf("%d\n",j);
 if(j==k&&j==s){
```

```
  printf("Strings are equal.\n");
 }
 else
 printf("Strings are not equal.\n");
 return 0;
}
```

- **Make a program that will be able to reverse a string.**

```
#include<stdio.h>
int main()
{
 char str[]="Hamim Talukder";
 strrev(str);
 printf("Reverse string is : %s",str);
 return 0;
}
```

Here,
strrev() is a library function that can reverse a string

or,

```
#include <stdio.h>
int main()
{
 char str1[] = "Hamim Talukder";
 int i, len1 = 0;
 for (i = 0; str1[i] != '\0'; i++)
 {
  len1++;
 }
```

```
printf("String length of str is %d\n",len1);
 int j=len1+1;
 char str2[j];

 for(j=0,i=len1-1;i>=0;i--,j++){
   str2[j]=str1[i];
 }
 str2!='\0';
 printf("Reverse string is : %s\n",str2);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char str[] = "Hamim Talukder";
 int i, len = 0;
 for (i = 0; str[i] != '\0'; i++)
 {
  len++;
 }
 printf("String length of str is %d\n",len);
 printf("Reverse string is : ");
 for(i=len;i>=0;i--){
  printf("%c",str[i]);
 }
 return 0;
}
```

- **Make a program that will show you palindrome strings.**

```
#include <stdio.h>
int main()
{
 char str1[30] = "madam";
 printf("string is : %s\n",str1);
 int i,j, len1 = 0;
 for (i = 0; str1[i] != '\0'; i++)
 {
 len1++;
 }
 printf("String length is %d\n",len1);
 char str2[30];
 for(j=0,i=len1-1;i>=0;i--,j++){
  str2[j]=str1[i];
 }
 printf("Reverse string is : %s\n\n",str2);
 int d=strcmp(str1,str2);
 if(d==0){
 printf("String is palindrome.\n");
 }
 else
 printf("String is not palindrome.\n");
 return 0;
}
```

- **Make a program that will be able to swapping strings.**

```
#include <stdio.h>
int main()
{
  char str1[15]="Bangladesh";
  char str2[15]="India";
  char temp[15];

  printf("Before swapping :\n");
  printf("str1 = %s\n",str1);
  printf("str2 = %s\n",str2);

  strcpy(temp,str1);
  strcpy(str1,str2);
  strcpy(str2,temp);

  printf("\nAfter swapping :\n\n");
  printf("str1 = %s\n",str1);
  printf("str2 = %s\n",str2);
}
```

- **Make a program that will convert a string into upper case and lower.**

```
#include <stdio.h>
int main()
{
  char str[ ]="Hamim Talukder";

  strupr(str);
  printf("Upper case of str = %s\n",str);
  strlwr(str);
  printf("Lower case of str = %s\n",str);
  return 0;
}
```

Here,
strupr() is a library function that can convert a string into upper case and strlwr() is also a library function that can convert a string into lower case.

- **Make a program that will show number of vowels, consonants, digits, words and others.**

```
#include <stdio.h>
int main()
{
  char str[100],ch;
  int i,vowel,consonant,digit,word,other;

  printf("Enter a string : ");
  gets(str);
  i=vowel=consonant=digit=word=other=0;
```

```
  while((ch=str[i])!='\0'){
if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||ch
=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
  vowel++;
  else if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
  consonant++;
  else if(ch>='0' && ch<='9')
  digit++;
  else if(ch==' ')
  word++;
  else
  other++;
  i++;
 }
 word++;

 printf("Number of vowels : %d\n",vowel);
 printf("Number of consonants : %d\n",consonant);
 printf("Number of digits : %d\n",digit);
 printf("Number of words : %d\n",word);
 printf("Number of others : %d\n",other);
 return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
 char str[100],ch;
 int i,vowel,consonant,digit,word,other;
```

```
printf("Enter a string : ");
 gets(str);

  for(i=vowel=consonant=digit=word=other=0;
(ch=str[i])!='\0';i++){

if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'||ch
=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
   vowel++;
   else if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
   consonant++;
   else if(ch>='0' && ch<='9')
   digit++;
   else if(ch==' ')
   word++;
   else
   other++;
  }
  word++;

printf("Number of vowels : %d\n",vowel);
 printf("Number of consonants : %d\n",consonant);
 printf("Number of digits : %d\n",digit);
 printf("Number of words : %d\n",word);
 printf("Number of others : %d\n",other);
 return 0;
}
```

- **Make a program that will show number of capital letters, small letters and digit.**

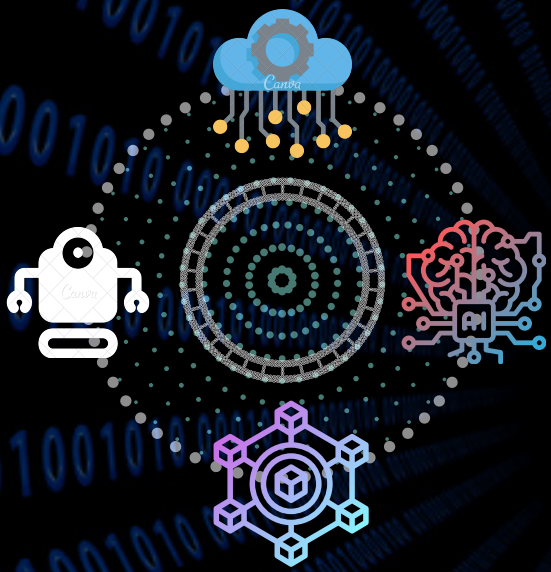
```
#include <stdio.h>
int main()
{
  char str[100];
  int i,capital,small,digit;

  printf("Enter a string : ");
  gets(str);

  for(i=capital=small=digit=0; str[i]!='\0';i++){
   if(str[i]>=65 && str[i]<=90)
   capital++;
   else if(str[i]>=97 && str[i]<=122)
   small++;
   if(str[i]>=48 && str[i]<=57)
   digit++;
  }

  printf("Number of capital letters : %d\n",capital);
  printf("Number of small letters : %d\n",small);
  printf("Number of digits : %d\n",digit);
  return 0;
}
```

or,

```
#include <stdio.h>
int main()
{
  char str[100];
  int i,capital,small,digit;

  printf("Enter a string : ");
  gets(str);

  i=capital=small=digit=0;
  while(str[i]!='\0'){
   if(str[i]>=65 && str[i]<=90)
   capital++;
   else if(str[i]>=97 && str[i]<=122)
   small++;
   if(str[i]>=48 && str[i]<=57)
   digit++;
   i++;
  }

  printf("Number of capital letters : %d\n",capital);
  printf("Number of small letters : %d\n",small);
  printf("Number of digits : %d\n",digit);
  return 0;
}
```

- **Make a program that will show given bellow.**
  1
  6 2
  okfine
  f e
  okin

```
#include<stdio.h>
#include<string.h>
int main()
{
  int n,i,a,b,t,k,count=0;
  scanf("%d",&n);
  for(i=1;i<=n;i++){
    scanf("%d %d ",&a,&b);
   char str1[a+1], str2[2*b+2];
    scanf("%s\n",str1);
    gets(str2);
    printf("Case %d: ",i);
    for(t=0;str1[t]!='\0';t++){
      for(k=0;str2[k]!='\0';k++){
        if(str1[t]==str2[k]){
          count++;
        }
      }
      if(count==0){
        printf("%c",str1[t]);
      }
      count=0;
    }
  printf("\n");
  }
  return 0;
}
```



C-PROGRAMMING (FIRST PART)

T.I.M. HA-MEAM

ABC PROKASHONI