Programing - C

TOKENS OF C

- all the element in C are called Token
- all the element in C language has been classified in five parts
- for only understanding purpose we may say that Tokens are KILPO / KILSO

K = Keyword
I = Identifier
L = Literals / Constants
PIS = Punctuators / separators
O = operators

1. Keywords ⇒ are reserved words by a programming language. these are reserved by a programing language for a specific task.
   - C has total 32 reserved words whose task is pre-defined and we can not use these keywords for our other purpose. the C keywords are:
     int, float, long, double, char, void, return, if, else, switch, goto, break, continue, while, do, for etc.
2. Identifiers ⇒ are the names given to a variable, function, array, structure, etc.

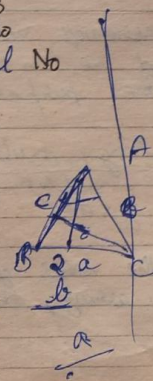
Rules of Identifiers.

1) The identifier name must contain only alphabets, digits or an underscore. symbols ✗
2) First character of an identifier must be either an alphabet or an underscore. 1st = bs - Digit ✗
3) Key words can't be used as an identifier name.
4) Upper case and lower case identifiers are different
   (A) ≠ (a) when use A than not use a in the place of A
5) Space is not allowed between identifiers.

Notes: Keep the identifier name maximum 8 characters long.

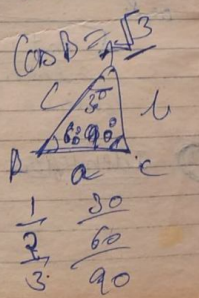
Identify correct / incorrect identifier with reason there.

1) _a Yes
2) a1 yes
3) 1a No
4) [redacted] pi val No
5) per% No
6) int No
7) Int Yes



$$\frac{a}{\sin A} = \frac{b}{\sin B}$$

$$1 - \sin^2 A + 1 - \sin^2 B = \ldots + 1 - \sin^2 C =$$



1/2	30	
√3/2	60	
1/√3	90	

L = Literals / Constants

Data Types

Data types defines the type of data a variable will store.

C provides a number of data types:
char, int, long int, float, double, long double

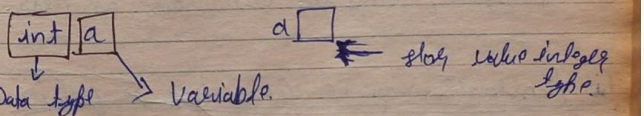
Variable

A variable is one which stores some value and its value can be changed at any point of time.

A variable is also an Identifier so rule for naming Identifiers also applies here.

Declaring a variable:

```
int a;
int a, b, c;
long int amount;
float a, b, c;
```

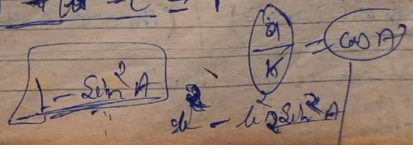


Now after declaring a variable we can put any value to it like:

```
int a;      float b;      char c;
a = 5;      b = 50.5;     c = 'M';
```

$$a^2 + b^2 - c^2 = 1$$

$$\frac{a}{\cos A} = k$$



(3) Literals / Constants

are those whose value can't be changed during the execution of the program. In other words we can say that these are fixed value.

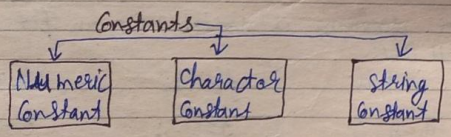
It is used where we want a fix value that can't be changed. when we try to change its value it gives an error stating constant values can't be changed.

syntax for declaring a constant.

```
Const int a = 10;        a 10
Const float b = 5.5;     a = 20 X
```

Now variable a and b has a fixed value which can't be changed during the execution of the program, doing so, will raise an error.

Lit / Constants may be categorized as follows



Const int a = 5; // Integer
Const float b = 5.5 // Real
(Both are Numeric)
	Const char gender = 'M';
	Const char a = '9';
	(Character Constant)

```
Const char a[ ] = "Ram";
Const char a[ ] = "x";
(String Constant)
```

1) Symbolic Constants

If we want to create a constant having a specified name and can be used anywhere in the program then we use to concept of symbolic constant.

```
Syntax:   # define name value
Example   # define PI 3.14
          # define MAX 100
          # define CH 'y'
          # define NAME "Himakash"
```

Note: 1) Every symbolic constant starts with # define
2) usually the constants name is in upper case.
3) No semi colon is there at the end.

(4) Punctuators / Seperators / Delimiters

Delimiters are those who have some pre-defined meaning and significance.

The difference punctuators are:

```
[ ]  - used for array declaration
{ }  - used for blocks of statement

( )  - used for expression / condition
;    - used for end of the statement
:    - used for a label
,    - used for separating variables

#    - used for declaring preprocessor directive
```

Printf() function

C language provides a function called printf() which is used to print some message / values on the screen. Its definition is found in stdio.h header file. Various forms of using printf() function are:

1. `Printf("Hello world");`
2. `Printf("Hello how are you?")`

Now we need to know one term of C programming i.e. Conversion specification:

The function printf() and scanf() uses Conversion specification to specify the type and size of data. Conversion specification starts with % sign. Most Commonly used Conversion specifications are:

1. %d – for printing / reading integer value
2. %Id – for printing / reading long integer values
3. %f – for printing / reading decimal values
4. %c – for printing / reading character values
5. %s – for printing / reading string values

Printing the values of a variable

1)
```
int a=5;
Printf("%d", a);
```
Output: 5

2)
```
Float pi=3.14;
Printf("%f", pi);
```
Output: 3.14

3)
```
char ch='M';
Printf("%c", ch);
```
Output: M

Printing the values of a variable along with message.

Consider the following code snippet:

1)
```
int a;
float b;
a=5;
b=6.5
Printf("%d \n%f", a, b);
```
Output: 5
6.5

\n → Escape sequence

2)
```
int a=5;
float b=6.5;
Printf("Value of a=%d", a);
Printf("\n Value of b=%f", b);
```
Output: Value of a=5
Value of b=6.5

Value of a=%d → message

All the values Print in one line.

3)
```
int a=5, b=6;
float c=6.6;
char ch='y';
```
Printf("Value of a=%d and value of b=%d and value of c=%f and value of ch=%c", a, b, c, ch);

Output: Value of a=5 and value of b=6 and value of c=6.6 and value of ch=y

Scanf() function

C language Provides a function called scanf() which is used to read data from the user. It holds the output screen so that the user can enter some values. [Its definition is found in stdio.h header file] It is usually associated with some print function so that a meaningful message could be displayed and the user may input appropriate values. Various forms of using scanf() function are:

Consider the following code snippet:

1)
```
int a;
Printf("Value of a = %d", a);
```

2)
```
int a;
Printf("Enter a number");
scanf("%d", &a);
Printf("Value of a = %d", a);
```

3)
```
Int a, b;
Printf("Enter two numbers!");
Scanf("%d%d", &a, &b);
Printf("Value of a = %d and value of b = %d", a, b);
```

4)
```
int a;
float b;
Printf("Enter one integer and one decimal number:");
Scanf("%d%f", &a, &b);
Printf("Value of a = %d and value of b = %f", a, b);
```

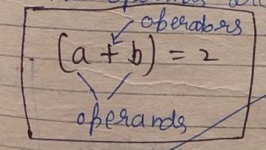
5 OPERATORS

Operators are those who performs some specific operation. C has a wide variety of operators. The various C operators are broadly categorized into.

1) Unary operator:
The operators which works on single operand are called Unary operator.
The Unary operators - Unary+ and Unary- for example:
```
int a = 5;
-a; // Here the value of a will become -5.
```

2) Binary operator
The operators which operates on two operands are called Binary operators.

3) Ternary operator
The operators which operates on three operands are called Ternary operator.



$a^2 + ab + ac + ab + b^2 + bc + ac + bc + c^2$

$a^2 + b^2 + c^2 + ac + ab + bc = 3ac$

$a^2 + b^2 + c^2 + ac + ab + bc = 2ac$

$\frac{1}{2} = a^2 + c^2 - b^2$

OPERATORS

operators are those who performs some specific operation. C has a wide variety of operators. the various C operators are categorized into:

1) Arithmetic operators
2) Comparison / Relational operators
3) Assignment operators
4) Logical operators
5) Identity operators
6) Membership operators
7) Bitwise operators
8) Sizeof operators
9) Conditional operator
10) Comma operator

1) Arithmetic operator

are those that performs mathematical calculation.
Arithmetic operators are:

Considering a=4 & b=2;

operator	Description	Example
+ (Addition)	Adds two or more numbers.	a+b = 6
- (Subtraction)	Subtracts two or more numbers.	a-b = 2
* (Multiply)	Multiplies two or more numbers.	a*b = 8
/ (Division)	Divides two number.	a/b = 2
% (Modulus)	Gives the remainder after dividing first number with second.	a%b = 0

2) Relational / Comparison operator

That performs comparison among operands. It gives true or false based on the condition.

Consider a=4 & b=2;

operator	Description	Example
> (Greater than)	check whether first number is greater then second	a>b returns (1) True
>= (Greater then equal to)	check whether first number is greater then equal to second number or not	a>=b returns true
< (less than)	check first number is smaller then second	a<b returns false
<= (less than equal to)	check whether first number is smaller then equal to second.	a<=b false
== (equal to)	checks whether the first number is equal to second number.	a==b return false
!= (Not equal to)	checks whether the fist number is not equal to second.	a!=b true a<>b return true

// Program

```
#include <stdio.h>
#include <conio.h>
void main()
{ int a, b;
  Printf("\n Enter two numbers:");
  Scanf("%d %d", &a, &b);
  Printf("\n a>b is = %d", a>b);
  Printf("\n a<b is = %d", a<b);
  getch(); }
```

Enter two numbers: 4 2

a > b is = 1 => return true.
a < b is = 0 => return false

3) Assignment Operator

is used to assign a value to a variable.

operator	Description	
=	Assigns values from right to left	C = a + b assigns value of a + b into C.
+= Add & Assign	It adds the right operand to left operand and Assigns the value to left operand.	C += a is equal to C = C + a
-= Subtract & Assign	It subtracts the right operand from left operand and assigns the value to left operand.	C -= a is equal to C = C - a
*= Multiply & Assign	It multiplies the right operand with left operand and assigns the value to left operand.	C *= a is equal to C = C * a
/= Divide & Assign	It divides left operand with the right operand and assign the value to left operand.	C /= a is equal to C = C / a
% = Modulus & Assign	It divides left operand with the right operand and assign the remainder value to left operand	C % = a is equal to C = C % a

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int a, b, c;
a=4; // assignment operator
b=2; // assignment operator
a+=b; // value of a is now 6
a-=4 // value of a is now 2
a*=3; // value of a is now 6
b+=a; // value of b is now 6
getch()
}
```

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int a, b;
  a=4; // assignment operator
  b=2; // " "
  a+=b;
  Printf("\n value of a = %d", a);
  a-=2;
  Printf("\n value of a = %d", a);
  getch();
}
```

Value of a = 6
Value of a = 4

4) LOGICAL OPERATORS =>

Performs logical operations based on given condition. C has following three logical operators and the operators are:

Operator	Description
and (Logical AND)	It checks all the conditions associated with this and returns true if and only if all the condition are "true", Any one false condition makes it "false".
or (Logical OR)	It checks all the conditions associated with this and returns true if "any one" among them are true. It will return false if not even one condition is true.
not (Logical Not)	This operator is used to reverse the output of logical condition. This means if some condition is true it will convert it to false and if some condition is false it will make it true.

operator Name	symbol
AND	&&
OR	||
NOT	!

5) Increment / Decrement OPERATOR

Increment operator (++) / Decrement operator (--) are further classified into:

1) Prefix Increment / Decrement operator
2) Postfix Increment / Decrement operators.

1) Prefix Increment / Decrement

When ++ or -- operator are placed before the operand (variable) then this is called Prefix increment decrement operator.

For Example: ++a; --b;

Note: If there is Prefix increment/decrement then we need to first evaluate the value and then process.

2) Postfix Increment / Decrement

When ++ or -- operator are placed after the operand (variable) then this is called postfix Increment / decrement operator.

For Example: a++; b--;

Note: if there is postfix increment/decrement then we need to first process the things and then value will be either incremented or decremented.

Increment / Decrement OPERATOR

For Example:

```
a = 5 ; //
++a ; //
--a ; //
+ +a ; //
- -a ; //
a ; //
- -a ; //
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=5, b=10;
Printf("\n Value of a = %d and Value of b = %d", a, b);
Printf("\n Value of a = %d", a++); Value of a = 5
Printf("\n Value of a = %d", ++a); Value of a = 7
Printf("\n Value of b = %d", ++b); Value of b = 11
Printf("\n Value of b = %d", b++); Value of b = 11
Printf("\n Last Value = %d", (a++)+(++b)); Last Value = 20
getch();
}
```

a = 7, b = 13

7 + 13 = 20 => Last value

6) SIZEOF OPERATOR

This operator is used to find the size of a variable / datatype in C programming language. This return the size of a variable / datatype in terms of bytes.

For example:

```
int a;
sizeof(int); // will return 2
sizeof(float); // will return 4
sizeof(a); // will return 2 as a is of datatype int
```

SIZEOF OPERATOR

Program to understand the concept of size of operator:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=5;
float b=6.6;
Printf("\n Size of integer = %d bytes", sizeof(int));
Printf("\n Size of float = %d bytes", sizeof(float));
Printf("\n Size of variable a = %d bytes", sizeof(a));
Printf("\n Size of variable b = %d bytes", sizeof(b));
getch();
}
```

output

Size of integer = 2 bytes
Size of float = 4 "
a " a = 2 "
" " b = 4 "

Ternary operator

Conditional Operator

- Conditional operators are also referred to as "Ternary Operator"
- It has ? and : as operators. The syntax is:

Syntax

(test expression) ? true section : false section

Example:

```
(a > b) ? a : b;
```

Here the test expression is checked and if the condition is found true then the true section is executed otherwise false section is executed.

Let's take an example to understand the concept of conditional operator.

Program

```
#include<stdio.h>
#include<conio.h>
int main()
{
  int a, b, max;
  Printf("\n Enter Two numbers:");
  Scanf("%d %d", &a, &b);
  max = (a>b) ? a : b;
  Printf("\n Max Number = %d", max);
  getch();
  return;
}
```

Output

Enter two numbers: 5 10

Max number = 10

Ques: write a Program to find max number between three numbers.

```
#include<stdio.h>
#include<conio.h>

int main()
{
  int a, b, c, max;

Printf("\n Enter the three numbers:");
Scanf("%d %d %d", &a &b);
Max = (a>b && a>c) ? a : (b>a && b>c) ? b : c;

Printf("\n Max Number = %d", max);
getch();
return 0;
}
```

Output

Enter two numbers: 10 20 30

Max number = 30

Basic structure of C Program

```
// Program of addition of two numbers.
#include <stdio.h>
#include <conio.h>
int main()
{
   int a, b, add;
Printf("\n Enter two numbers:");
Scanf("%d %d", &a, &b);
Add = a+b;
Printf("\n Addition = %d", Add);
getch();
return 0;
}
```

output
Enter two numbers: 4 8
Addition = 12

// Program of addition, multiplication, subtra, divis of two number

```
# include <stdio.h>
# include <conio.h>
  int main()
{
 int a, b, add, sub, mult, div;
Printf("\n Enter two numbers:");
Scanf("%d %d", &a, &b);
 Add = a+b;
 Sub = a-b;
 mult = a*b;
 div = a/b;

Printf("\n Addition = %d", Add);
Printf("\n Subtraction = %d", Sub);
Printf("\n multiplication = %d", mult);
Printf("\n division = %d", div);

getch();
return 0;
}
```

output
Enter two numbers 4 8
Addition = 12
Subtraction = -4
multiplication = 32
division = 1/2

Program to Convert temperature from Fahrenheit to Celsius -

```
#include <stdio.h>
int main()
{
 float c, f;
Printf("\n Enter Temperature in celcius:");
Scanf("%f", &c);
f = (c*9/5)+32;
Printf("\n Temp in Fah = %f", f);
return 0;
}
```

Out

Enter tem in celcius: 37

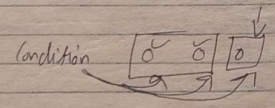
Temp in Fah = 98.599998

CONTROL STATEMENTS

Control statements are used to control the flow of execution of a program. We know that the C program gets executed step by step in the same order the statement appears on the screen. But there may be some situations where we may like to execute or not to execute statements based on a condition. This flow of execution may be controlled by using control statements.

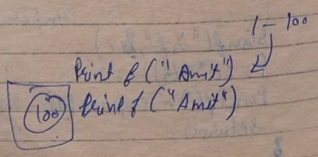
1) Selection statement

a) if ... else
b) Switch
c) goto — jump statement



2) Looping statement:

a) while
b) do ... while
c) for



a) if .... else Statement

This is a bi-directional conditional control statement. This checks the condition and if the condition is true, it executes the true section and if the condition is false it executes the false section.

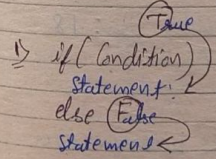
The syntax is:

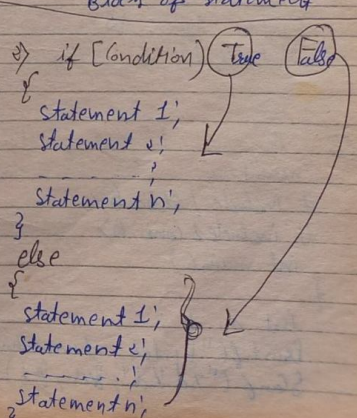
1) if (condition)
statement;

↑ single statement

2) if (condition)
{
Statement 1;
Statement 2;
.....;
Statement n;
}

↑ Block of statement





IF... ELSE STATEMENT

Ques: Program to check whether an age is eligible to vote or not?

```
#include<stdio.h>
int main()
{
int age;
Printf("\n Enter Age:");
Scanf("%d", &age);
if(age >=18)
Printf("\n Eligible for voting");
else
Printf("\n Not Eligible for voting");
return 0;
}
```

Output

Enter Age : 15
Not eligible for voting

Output

Enter Age : 18
Eligible for voting

Ques: Program to find greater number between two numbers.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int a, b;
Printf("Enter two numbers:");
Scanf("%d %d", &a, &b);
if(a>b)
Printf("\n Max Number = %d", a);
else
Printf("\n Max Number = %d", b);
return 0;
}
```

Output

Enter two numbers 5 8
Max number = 8

Output

Enter two numbers 8 5
Max number = 8

Ques → Program to find the greatest of three numbers in C

```
#include<stdio.h>
int main()
{
int a, b, c;
Printf("Enter three numbers:");
Scanf("%d %d %d", &a, &b, &c);
if(a>b && a>c)
Printf("\n Max No = %d", a);
else if(b>a && b>c)
Printf("\n Max No = %d", b);
else
Printf("\n Max No = %d", c);
return 0;
}
```

Output

Enter three number : 5 6 4
Max No = 6

SWITCH STATEMENT IN C

Switch statement is an alternate to the if statement when using switch statement the choice of user is passed and based on the choice there are multiple case statements which performs the activity as per the choice supplied.

Syntax

```
Switch (choice)
{
  case value:
       statement(s);
       break;
  case value:
       statement(s);
       break;
  ---------
  ---------
  default:
       statement(s);
}
```

Ques: Program to print the name of the day as per user's choice between 1-7.

```
#include<stdio.h>
int main()
{
int ch;
Printf("\n Enter any number from 1-7:");
scanf("%d",&ch);
Switch (ch)
{
case 1:
Printf("\n Sunday");
break;
case 2:
Printf("\n Monday")
break;
case 3:
Printf("\n tuesday");
break;
case 4:
Printf("\n Wednesday")
break;
case 5:
Printf("\ Thursday");
break;
case 6:
Printf("\n Friday");
case 7:
Printf("\n Saturday");
break;
default:
Printf("\n wrong video");
break;
}
return 0;
}
```

Output Enter any number from 1-7:- 3
Tuesday

output

Enter any number from 1-7 : 10

Wrong choice

Ques: Program to accept two numbers and perform calculation as per user's choice.

```
#include<stdio.h>
int main()
{
int a, b, ch;
Printf("\n Enter Two Numbers:");
Scanf("%d%d", &a, &b);
Printf("\n1->Add \n2->Sub \n3->Mul \n4->Div
        \nEnter your choice:");
Scanf("%d", &ch);
Switch(ch)
{
Case 1:
Printf("\n Addition=%d", a+b);
break;
Case 2:
Printf("\n Subtraction=%d", a-b);
break;
Case 3:
Printf("\n Multiplication=%d", a*b);
break;
Case 4:
Printf("\n Division = %d", a/b);
break;
default:
Printf("\n Wrong choice");
break;
}
return 0;
}
```

Output: enter two numbers: 5 4

```
1->add
2->sub
3->mul
enter your choice: 3
multiplication=20
```

```
1->add
2->sub
3->mul
enter your choice: 5
Wrong choice.
```

Fall Through

When a case has no break, it will keep executing the next case statements till the end or till it finds the break statement. This is called fall through.

```
#include<stdio.h>
int main()
{
int ch;
Printf("\n Enter any number from 1-7:");
Scanf("%d", &ch);
switch(ch)
{
Case 1:
Printf("\n Sunday");
break;
```

```
case 2:
Printf("\n Monday")
case 3:
Printf("\n Tuesday");
case 4:
Printf("\n Wednesday")
break;
case 5:
Printf("\n Thursday");
case 6:
Printf("\n Friday");
break;
case 7:
Printf("\n Saturday");
break;
default:
Printf("\n wrong choice");
break;
}
return 0;
}
```

Output:

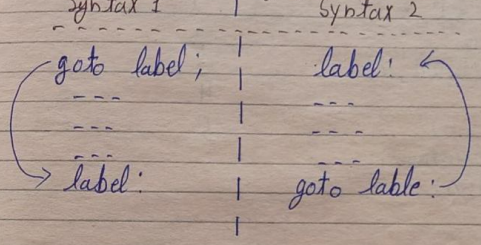
enter any number: 2

Monday
Tuesday
wednesday

GOTO STATEMENT IN C

The goto statement is a jump statement which is also referred to as unconditional jump statement. The goto statement can be used to jump from anywhere to anywhere within a function.

The syntax for using goto statement is:



Ques: write a program to check a number is odd or even using goto statement.

```
#include<stdio.h>
int main()
{
int a;
Printf("\n Enter Number:");
Scanf("%d", &a);
if(a%2==0)
goto even;
else
goto odd;
even:
Printf("\n Number is even");
return 0;
odd:
Printf("\n Num is odd");
return 0;
}
```

a = 5
a = 4

Output: Enter a number: 5
number is odd

Ques: write a program to print first "N" natural numbers.

```
#include<stdio.h>
int main()
{
int n, i=1;
Printf("\n Enter Number:");
Scanf("%d", &n);
Printnum:
Printf("\n%d", i);
i++;
if(i<=n)
goto printnum;
return0;
}
```

	n=5
1	i=1
2	2
3	3
4	4
5	5
	6

Output

Enter number: 5	Enter number: 1065
1	1
2	2
3	⋮
4	1064
5	1065

2) Looping statement:

1) Initialization
2) Condition
3) Increment / Decrement

i = 1 2 3 10

1) Initialization = i = 1
2) Condition = (i<=10)
3) Increment = i = i+1

1) while loop

Syntax

```
initialization
while (condition)
-> {
    _____;
    _____;
increment / decrement
}
```

2) while loop

Question	Initialization	Condition	Increment/Decre
1) 1-10	i = 1	i<=10	i = i+1
2) 10-1	i = 10	i>=1	i = i-1
3) 2-20	i = 2	i<=20	i = i+2

i: 2 4 6 8 20

1) C Program to print first N Natural numbers

```
# include<stdio.h>
int main()
{
  int n, i;
  Printf("\n Enter number:");
  Scanf("%d", &n);
   i=1; // initialization
  while(i<=n)
    {
      Printf("\n%d", i);
       i=i+1;
    }
    return 0;
  }
```

Dry Run
i = 1 2 3 4 5 6

output Enter number : 5
1
2
3
4
5

2) Print even numbers from 1 to n

```
# include<stdio.h>
int main()
{
  int n, i;
  Printf("\n Enter value of n:");
  Scanf("%d", &n);
   i=1; // initialization
  while(i<=n) // Condition
  ->{
     if (i%2==0)
        Printf("\n%d", i)
     i=i+1; // increament
  }
  -> return 0;
}
```

n [10]
i = 1 2 3 4 5 6 7 8 9 10 11

output
2
4
6
8
10

ii) Print odd numbers from 1 to n

```
# include<stdio.h>
int main()
{
  int n, i;
  Printf("\n Enter value of n:");
  Scanf("%d", &n);
   i=1; // initialization
  while(i<=n) // Condition
  {
     if (i%2!=0)
        Printf("\n%d", i)
     i=i+1; // increament
  }
   return 0;
}
```

n [10]
i = 1 2 3 4 5 6 7 8 9 10 11

output
1
3
5
7
9

Ques: Program to find sum of first (n) natural numbers

```
#include<stdio.h>
int main()
{
    int n, i, sum;
    printf("\n Enter Value of N");
    scanf("%d", &n);
    i=1; // initialization
    sum=0;
    while(i<=n) // Condition
    {
        sum = sum+i;
        i = i+1;
    }
    printf("\n sum = %d", sum);
    return 0;
}
```

n[5]

i = 1 x 3

sum = ~~0 x 1 2 6 10~~ 15

sum = sum + i
= 0+1
= 1

sum = 1+2 = 3
sum = 3+3 = 6
sum = 6+4 = 10
sum = 10+5 = 15

output

Enter value of N = 5
15 = sum

Ques write a Program to find sum of digits of a given number

```
#include<stdio.h>
int main()
{
    int i, sum=0;
    printf("\n Enter No:");
    scanf("%d", &i);
    while(i>0)
    {
        sum = sum+i%10;
        i = i/10;
    }
    printf("\n Sum of digits = %d", sum);
    return(0);
}
```

dry run

1 0 2 3 8

sum = 0

sum = sum + i%10
= 0+8
= 8

sum = sum + i%10
= 8+3
= 11

sum = sum + i%10
= 11+2
= 13

output

Enter No: 2 38

sum of digits = 13

C Program to find factorial of a given Number.

```
#include <stdio.h>
int main()
{
  int i, fac = 1;
  Printf("\n Enter Number to find fac: ");
  Scanf("%d", &i);
  while (i >= 1)
  {
    fac = fac * i;
    i--;
  }
  Printf("\n factorial = %d", fac);
  return (0);
```

Dry Run

i	fac = fac x i
5	= 120 x 1
fac (120)	= 120

output

factorial = 120

Ques write a program to find Reverse of a number

```
#include <stdio.h>
int main()
{
  int i, rev = 0;
  Printf("\n Enter No to find reverse: ");
  Scanf("%d", &i);
  while (i > 0)
  {
    rev = (rev * 10) + i % 10;
    i = i / 10;
  }
  Printf("\n Reverse = %d", rev);
  return 0;
}
```

Dry Run

i = 0 . ~~8~~ . ~~2~~ . ~~3~~

rev = 0

rev = (rev * 10) + i % 10
= (0 * 10) + 8
= 0 + 8
= 8

rev = (rev * 10) + i % 10
= (8 * 10) + 3
= 80 + 3
= 83

rev = (rev * 10) + i % 10
= (83 * 10) + 2
= 830 + 2
= 832

Ques Palindrome Number Program in C

Ans

```
#include<stdio.h>
int main()
{
int i, n, rev=0;
Printf("\n Enter No to check:");
Scanf("%d", &i);
n=i;
while(i>0)
{
rev=(rev*10)+i%10
i=i%10;
}
if (rev==n)
printf("Palindrom");
else
Printf("Not Palindrom");
return 0;
}
```

Dry Run

i = 0 . 5 . 2 . 5

n = 525

rev = 0 5 52

rev = (rev*10) + i%10
= (0*10) + 5
= 0 + 5
= 5

rev = (rev*10) + i%10
= (5*10) + 2 = 52

rev = (rev*10) + i%10
= (52*10) + 5
= 520 + 5
= 525

b) Do - While Loop

Syntax:

```
do
{
  Statement
  Statement;
} while(Condition);
```

a) while

```
while [Condition]
{
  ——
  ——
}
```

Do - while loop

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

output

1 2 3 4 5 6 7 8 9 10

Exit Control loop

while loop

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

1 2 3 4 5 6 7 8 9 10

Entry Control loop

c) For Loop

Syntax of for loop

```
for(initialization; condition; increment/dec)
{
  statement 1;
  statement 2;
  statement 3;
}
```

C Program to Print from 1 to n.

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

1 — n

1> i=1

2> i<=n

3> i++/i=i+1

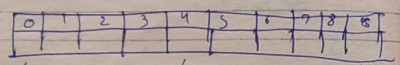
output Enter number: 5

1
2
3
4
5

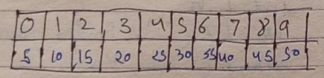
Chapter → ARRAY

- An Array is a Collection of similar types of data. It is a Contiguous memory which is indexed from 0 to n-1.
- So In simpler words we may say that if we need to store same type of values in large quantity then we use the Concept of Array.

```
int a[10];
```



Ques: Program to accept and display an array.




```
#include<stdio.h>
int main()
{
int a[10], i;
for(i=0; i<10; i++)
{
printf("\n Enter Number:");
Scanf("%d", &a[i]);
}
clrscr();
printf("\n Array elements are:");
for(i=0; i<10; i++)
printf("\n%d", a[i]);
return 0;
}
```

Input

```
Enter Number = 5
Enter number = 10
. . . . . -15
```

Output

```
Array Elements are
```

Comparative Study between while/do-while/for loop:
Program Snippet to Print from 1-10.

```
i=1;
while(i<=10)
{
printf("\n%d", i);
i++;
}
```

```
i=1;
do
{
printf("\n%d", i);
i++;
} while(i<=10);
```

```
for(i=1; i<=10; i++)
{
printf("\n%d", i);
}
```

Program Snippet to find factorial

```
#include<stdio.h>
int main()
{
int n, fac=1;
printf("\n Enter Num:");
Scanf("%d", &n);
while(n>0)
{
fac=fac*n;
n--;
}
printf("\n Factorial=%d", fac);
return 0;
}
```

```
#include<stdio.h>
int main()
{
int n, fac=1;
printf("\n Enter Num:");
Scanf("%d", &n);
do
{
fac=fac*n
n--;
} while(n>0);
printf("\n factorial =%d", fac);
return 0;
}
```

```
//include<stdio.h>
int main()
{
int n, fac=1;
Printf("\n Enter Num:");
Scanf("%d", &n);
for(; n>0; n--)
{
fac=fac*n;
}
Printf("\n Factorial =%d", fac);
return 0;
}
```

Ques: Program to find sum of array elements.

```
#include<stdio.h>
int main
{
int a[10], i, sum=0;
for(i=0; i<10; i++)
{
Printf("\n Enter Number:");
Scanf("%d", &a[i]);
}
for (i=0; i<10; i++)
{
Sum = sum+a[i];
}
Printf("\n Sum of Array Numbers = %d", sum);
return 0;
}
```

a[i]	0	1	2	3	4	5	6	7	8	9
	5	2	3	1	4	2	3	4	1	1

Sum = 0 i = 0 1 2 3

sum = sum + a[i]
= 10 + a[3]
= 10 + 1 = 11

Ques: Program to find sum of even numbers and product of odd numbers.

```
#include<stdio.h>
int main()
{
int a[10], i, sum=0, pro=1;
for (i=0; i<10; i++)
{
Printf("\n Enter number:");
Scanf("%d", &a);
}
```

0	1	2	3	4	5	6	7	8	9
2	3	1	4	2	1	3	2	2	1

```
for(i=0; i<10; i++)
{
if (a[i]%2==0)
sum = sum+a[i];
else
pro = pro*a[i];
}
Printf("\n Sum of Even = %d and Product of odd = %d", sum, pro);
return 0;
}
```

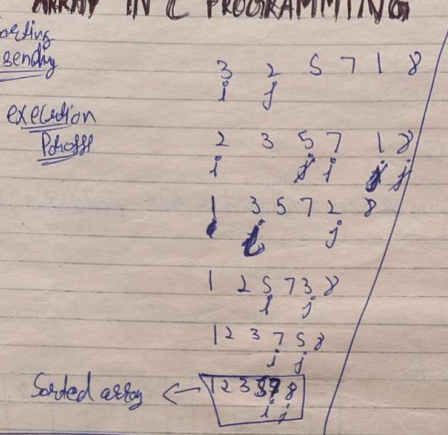
Ques: Program to print alternate elements of the array.

```
#include<stdio.h>
int main()
{
int a[10], i;
for(i=0; i<10; i++)
{
Printf("\n Enter Number:");
Scanf("%d", &a[i]);
}
for (i=0; i<10; i=i+2)
Printf("\n %d", a[i])
}
return 0;
}
```

0	1	2	3	4	5	6	7	8	9
5	10	15	20	25	30	35	40	45	50

ARRAY IN C PROGRAMMING

Sorting
Ascending



execution
Process

Ques: Program to sort element in an array.

```
#include<stdio.h>
int main()
{
int a[10],i,j,t;
for(i=0;i<10;i++)
{
Printf("\n Enter Number:");
Scanf("%d",&a[i]);
}
for(i=0;i<9;i++)
for(j=i+1;j<10;j++)
{
if(a[i]>a[j])
{
t=a[i];
a[i]=a[j];
a[j]=t;
}
}
Printf("\n Array after Sorting is:");
for(i=0;i<10;i++)
Printf("\n%d",a[i]);
return 0;
}
```

0	1	2	3	4	5	6	7	8	9
3	5	2	7	8	1	3	9	5	6

output

Enter Number
3
5
2
7
8
1
3
9
5
6

Array after Sorting
1
2
3,3
5
6
7
8
9

Two dimensional array

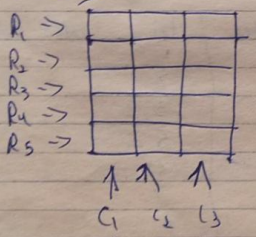
Array is basically of two types:

- One Dimensional Array
- Multi Dimensional Array
  - ⇒ 2 Dimensional Array

a [10] [10] → Column
↳ Rows

int a[5] ⇒ a [5][4][3][2][1] → x-axis

int a[5][3] ⇒



Ques: Accept & Display a 3*3 Matrix

```
#include <stdio.h>
#include<conio.h>
int main()
{
int a[3][3],i,j;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
Printf("\n Enter value for 2D Array:");
Scanf("%d",&a[i][j]);
}
Printf("\n2D Array Elements are:");
for(i=0;i<3;i++)
{
Printf("\n");
for(j=0;j<3;j++)
Printf("%d\t",a[i][j]);
}
return(0);
}
```

Ques Program to print Sum of elements of a 3*3 matrix.

```
#include <stdio.h>
int main()
{
int a[3][3],i,j,sum=0;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
{
Printf("\n Enter value for 2D Array:");
scanf("%d",&a[i][j]);
}
Printf("\n2D Array Elements are:");
for(i=0;i<3;i++)
{
Printf("\n");
for(j=0;j<3;j++)
Printf("%d\t",a[i][j]);
}
```

```
for (i=0; i<3; i++)
for(i=0; j<3; j++)
  Sum = Sum+a[i][j]
  Printf("\n Sum of 2D array is =%d", Sum);
  return 0;
}
```

	0	1	2
0	2	3	1
1	4	1	3
2	2	5	7

Ques: Program to search a given number in a 2D array.

```
#include<stdio.h>
int main()
{
 int a[3][3], i, j, Key;
 for(i=0; i<3; i++)
  for(j=0; j<3; j++)
  {
   Print("\n Enter Value for 2D Array:");
   Scanf("%d",&a[i][j]);
  }
   Printf("\n 2D Array Elements are:");
   for(i=0; i<3; i++)
   {
    Printf("\n");
    for(j=0; j<3; j++)
     Printf("%d\t",a[i][j]);
   }
```

Ques: Addition of two matrix.

```
#include<stdio.h>
int main()
{
 int a[3][3], b[3][3],i, j;
 Printf("\n Enter Values for first matrix:");
 for(i=0; i<3; i++)
  for(j=0; j<3; j++)
 {
  Printf("\n Enter Value for 2D Array:");
  Scanf("%d", &a[i][j]);
 }
  Printf("\n Enter Values for Second matrix:");
  for(i=0; i<3; i++)
  for(j=0; j<3; j++)
  {
   Printf("\n Enter Value for 2D Array:");
   Scanf("%d", &b[i][j]);
  }
 Printf("\n First matrix is:");
 for(i=0; i<3; i++)
 {
  Printf("\n");
  for(j=0; j<3; j++)
   Printf("%d\t", a[i][j]);
 }
  Printf("\n Second Matrix is:");
  for(i=0; j<3; i++)
  {
   Printf("\n");
   for(j=0; j<3; j++)
  } Printf("%d\t", b[i][j]);
```

```
for(i=0; i<3; i++)
 for(j=0; j<3; j++)
  C[i][j] = a[i][j] + b[i][j];

 Printf("\n Addition of Matrix is:");
 for(i=0; i<3; i++)
{
 Printf("\n");
  for(j=0; j<3; j++)
  Printf("%d\t", C[i][j]);
}
 return 0;
}
```

Ques: Multiplication of two matrix.

```
#include <stdio.h>
int main()
{
int a[3][3], b[3][3] i, j, k;
Printf("\n Enter values for first matrix:");
 for(i=0; i<3; i++)
 for(j=0; j<3; j++)
{
 Printf("\n Enter Value for 2D Array:");
  Scanf("%d", &a[i][j]);
}
 Printf("\n Enter Values for second matrix:");
 for(i=0; i<3; i++)
 for(j=0; j<3; j++)
{
 Printf("\n Enter Value for 2D Array");
  Scanf("%d", &b[i][j]);
}

Printf("\n First Matrix is:");
 for(i=0; i<3; i++)
{
 Printf("\n");
 for(j=0; j<3; j++)
 Printf("%d\t", a[i][j]);
}
  Printf("\n Second Matrix is:");
  for(i=0; i<3; i++)
{
```

```
printf("\n");
for(j=0;j<3;j++)
printf("%d\t", b[i][j]);
}
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
c[i][j]=0;
for(k=0;k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}
}
}
printf("\n Multiplication of Matrix is:");
for(i=0;i<3;i++)
{
printf("\n");
for(j=0;j<3;j++)
printf("%d\t",c[i][j]);
}
```

Ques> Program to insert a given value at a given index in an array.

```
#include<stdio.h>
int main()
{
int a[10], i, j, pos, val;
for(i=0;i<9;i++)
{
printf("\n Enter Number:");
scanf("%d",&a[i]);
}
printf("\n Enter index to Insert:");
scanf("%d",&pos);
printf("\n Enter value to Insert:");
scanf("%d",&val);
for(i=8;i>=pos;i--)
a[i+1]=a[i];
a[pos]=val;
printf("\n Array after Insertion:");
for(i=0;i<10;i++)
printf("\n %d",a[i]);
return 0;
}
```

Ques Program to remove a given number from the array.

```
#include <stdio.h>
int main()
{
int a[10], i, j, t, index, key;
for(i=0; i<10; i++)
{
Printf("\n Enter Number:");
Scanf("%d", &a[i]);
}
Printf("\n Enter Element to Remove from the list:");
Scanf("%d", &Key);
index = 9;
for(i=0; i<=index; i++)
{
if(a[i]==Key)
{
for(j=i+1; j<index; j++)
a[j-1] = a[j];
i--;
index--;
}
}
Printf("\n Array after removing %d is:", Key);
for(i=0; i<=index; i++)
Printf("\n %d", a[i]);
return 0;
}
```

STRING

~~String~~ Character array is called string. Basically string is a collection of many characters. In simple words we can say that the words / sentences are referred as string in C programming.

Way to create a String

Syntax:

```
char Var_name[size];
```

Example:

```
char name[50];
```

This creates an array of 50 character string.

```
#include<stdio.h>
int main()
{
char a[50];
Printf("\n Enter Your Name:");
Scanf("%s", &a);
Printf("\n Your Name is %s", a);
return 0;
}
```

Ques: Accept a string and print it character by character.

```
#include<stdio.h>
int main()
{
char a[50];
```

```
int i;
Printf("\n Enter Your Name!");
gets(a);
i=0;
while(a[i]!='\0')
{
 Printf("%c", a[i]);
 i++;
}
return 0;
}
```

↙ Imp. Concept

'\0' = Null

Ques: Accept a string and Count total vowels & Consonants.

Eg

```
#include<stdio.h>
int main()
{
 char a[50];
 int i, Vol=0, Cons=0;
 Printf("\n Enter Your Name:");
 gets(a);
 i=0;
while(a[i]!='\0') {if(a[i]<=65&&a[i]>=90||a[i]>=97
~~while(a[i]!='\0')~~ && a[i]<=122)
{
 if(a[i]=='a'||a[i]=='e'||a[i]=='i'||a[i]=='o'
||a[i]=='U'||a[i]=='A'||a[i]=='E'||a[i]=='I'
||a[i]=='O'||a[i]=='U')
Vol++;
else
Cons++;
```

ASCII	
A = 65	a = 97
B = 66	b = 98
C = 67	c = 99
⋮	⋮
Z = 90	z = 122

```
i++;
}
Printf("\n Total vowels = %d and total Consonent
 = %d", Vol, Cons);
```

Ques: Concatenated two strings into one.

```
#include<stdio.h>
int main()
{
 char a[50], b[50], c[50]; int i, j;
 Printf("\n Enter First string!");
 gets(a);
 Printf("\n Enter Second String!");
 gets(b);
 i=j=0
 while(a[i]!='\0')
{
 c[i]=a[i];
 i++;
}
 while(b[j]!='\0')
{
 c[i]=b[j];
 i++; j++;
}
 c[i]='\0';
 Printf("\n Concatenated string = %s", c); return 0;
}
```

Character Functions in C Programming

Use <ctype.h> as header file.

1. isalpha() => checks whether character is alphabetic
2. isdigit() => checks whether character is digit
3. isalnum() => checks whether character is alphanumeric
4. isspace() => checks whether character is space
5. islower() => checks whether character is lower case
6. isupper() => checks whether character is upper case
7. tolower() => checks whether character is alphabetic & converts to lower case
8. toupper() => checks whether character is alphabetic & converts to upper case

```
#include <stdio.h>
int main()
{
char ch;
Printf("\n Enter an alphabet:");
Scanf("%c",&ch);
if(isalpha(ch))
 Printf("\n the character is an alphabet:");
 else
Printf("\n the character is not an alphabet:");
 if(isdigit(ch))
Printf("\n The character is a Digit:");
else
Printf("\n the character is not a Digit:");
if(isalnum(ch))
Printf("\n the character is either an alphabet or
        a Digit:");
else
Printf("\n The character is neither an alphabet
        not a Digit:");

if(isspace(ch))
 Printf("\n The character is a space:");
else
 Printf("\n The character is not a space:");
 if(islower(ch))
Printf("\n The character is a lower case alphabet:");
else
Printf("\n The character is not a lower case alphabet:");
if(isupper(ch))
 Printf("\n The character is a upper case alphabet:");
else
Printf("\n The Character is not a Upper case alphabet");
Printf("\n the character is lower case=%c",tolower(ch));
Printf("\n The character in Upper case=%c",toupper(ch));
return 0;
}
```

Functions

Functions are sub-program i.e. the part of a program which performs a specific type of operation. Remember that it is not a complete program whereas it is a part of the program.

Function are broadly divided into two parts

1) Library functions
2) User-defined functions

Functions

Benefits / Advantages of Function:

- Avoid repetition of codes
- Increases program readability.
- Divide a complex problem into simpler ones.
- Reduces chances of error.
- Modifying a program becomes easier by using function

1) Library functions / Built-in functions

Library functions / Built-in functions are those which are already defined means whose functionality is already defined and we need to just call them in order to use them. They will perform the functionality only when it is called

Functions present in math.h header file:

1) sqrt() - return the square root of the given number
2) pow() - returns x to the power y.
3) round() - rounds up the number
4) floor() - rounds the number to next nearest integer.
5) sin() - returns sin value.
6) cos() - returns cos value.
7) tan() - returns tan value.

2) User-defined functions

The function defined by user has a certain functionality is a called a user defined function.

User-defined functioned consists of three parts.

1) Function Prototype
2) Function Call
3) Function Definition.

1) Function Prototype

Function prototype is just an information to the compiler about the function structure / definition.

Function Prototype contains three parts.

return_type Function_name(arguments);

2) Function Call
In order to invoke / execute a function we need to call them by the function name.

3) Function Definition
The body / functionality that a function will perform is written in its definition block.

A function may be written using four ways.

1) No ARGUMENT NO RETURN -> Void add(Void)
2) With ARGUMENT NO RETURN Void add(int, int)
3) NO ARGUMENT WITH RETURN int add(Void);
4) WITH ARGUMENT WITH RETURN int add(int, int);

Argument => The value which we Pass to function
return => The value that a function return

Add two numbers using function.

1) No Argument No Return

```
#include <stdio.h>
Void add(Void);
int main()
{
Add();
return 0;
}
Void add(Void)
{
int a, b, Sum;
Printf("\n Enter Two Numbers");
Scanf("%d%d", &a, &b);
Sum = a+b;
Printf("\n Addition = %d", Sum);
}
```

2) With Argument No Return

```
#include <stdio.h>
Void add(int, int);
int main()
{
Int a, b;
Printf("\n Enter Two Numbers");
Scanf("%d%d", &a, &b);
Add(a, b);
return 0;
}
Void add(int a, int b)
{
int Sum;
Sum = a+b;
Printf("\n Addition = %d", Sum);
}
```

3) No Argument with Return

```
#include<stdio.h>
int add(void);
int main()
{
int sum;
sum=add();
printf("\n Addition = %d", sum);
return 0;
}
int add( )
{
int a, b, c;
printf("\n Enter Two Numbers");
scanf("%d %d", &a, &b);
c = a+b;
return (c);
}
```

4) No Argument with Return

```
#include<stdio.h>
int add(int, int);
int main()
{
int a, b, sum;
printf("\n Enter Two Numbers");
scanf("%d %d", &a, &b);
sum = add(a, b);
printf("\n Addition = %d", sum);
return 0;
}
int add(int a, int b)
{
int c;
c = a+b;
return(c);
}
```

WANR

Ques: write a function to check whether a given number is prime or Not

```
#include<stdio.h>
void prime(int);
int main()
{
int i;
printf("\n Enter No to check");
scanf("%d", &i);
prime(i);
return 0;
}
```

Call by Value

```
void prime(int i)
{
int x, count=0;
for(x=1; x<=i; x++)
{
if(i%x==0)
count++;
}
if(count==2)
printf("\n The No is prime");
else
printf("\n Not Prime");
}
```

WANR

Ques Write a function to find sum of array elements.

```
#include<stdio.h>
Void arrayadd(int[], int);
int main()
{
  int a[100], size, i;
  Printf("\n Enter size of Array:");
  Scanf("%d", &size);

  for(i=0; i<size; i++)
  {
    Printf("\n Enter Number:");
    Scanf("%d", &a[i]);
  }
  arrayadd(a, size);
  return 0;
}
Void arrayadd(int a[], int size)
{
  int Sum=0, i;
  for(i=0; i<size; i++)
      Sum = Sum+a[i];
  printf("\n Sum of Array = %d", Sum);
}
```

WANR

Ques ⇒ Program to search a number using function.

```
#include<stdio.h>
⇒ Void Sort(int[], int, int);
int main()
{
  int a[100], size, i, num
  printf("\n Enter size of Array:");
  Scanf("%d", &size);
  for(i=0; i<size; i++)
  {
    printf("\n Enter Number:");
    Scanf("%d", &a[i]);
  }
    printf("\n Enter No. to search:");
    Scanf("%d", &num);
    Sort(a, size, num);
    return 0;
}
  Void Sort(int a[], int size, int num)
{
  int i, flag=0, pos;
  for(i=0; i<size; i++)
  {
    if(a[i]==num)
    {
      flag=1; pos=i+1;
      break;
    }
  }
  if(flag==1)
    Printf("No. found at %d", pos);
```

```
else
    printf("\n No Not found");
}
```

Ques :- write a C program to add/ Subtract two matrix using function.

```
#include<stdio.h>
void addmat(int [][10], int[][10], int, int);
int main()
{
  int a[10][10], b[10][10], row, col, i, j;
  printf("\n Enter No of rows & Column: ");
  scanf("%d %d", &row, &col);
  for(i=0; i<row; i++)
    for(j=0; j<col; j++)
    {
      printf("\n Enter Number: ");
      scanf("%d", &a[i][j]);
    }
  for(i=0; i<row; i++)
  for(j=0; j<col; j++)
  {
    printf("Enter No: ");
    scanf("%d", &b[i][j]);
  }
  addmat(a, b, row, col);
  return(0);
}

void addmat(int a[][10], int b[][10],
            int row, int col)
{
  int c[10][10], i, j;
  for(i=0; i<row; i++)
  for(j=0; j<col; j++)
    c[i][j] = a[i][j]+b[i][j];
  for(i=0; i<row; i++)
  {
    printf("\n");
    for(j=0; j<col; j++)
      printf("%d\t", c[i][j]);
  }
}
```

add

POINTER

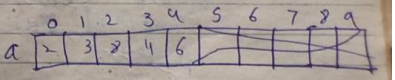
1. Pointer are used to store address of variable
2. Pointer are used to create Run-time memory.

↳ Data structure

```
int a;
a = 5;
```

a [5]
→ [100] [hexa-decimal]

1. int a[10]



POINTER

add

```
#include<stdio.h>
int main()
{
int a, b, *p, *q, c;
Printf("\n Enter two Numbers:");
Scanf("%d %d", &a, &b);
-> p = &a;
=> q = &b;
=> c = *p + *q;

Printf("\n addition = %d", c);
return 0;
}
```

p = &a;
② => *p

a ⑤
1 → 100

b 12
q → 2001

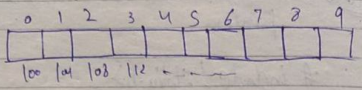
Ques :-> Program to check a number is odd/even using pointer

```
#include<stdio.h>
int main()
{
=> int a, *p;
printf("\n Enter No to check for odd/even");
Scanf("%d", &a);
p = &a;
if (*p%2 == 0)
    printf("Even");
else
    print("odd");
return 0;
}
```

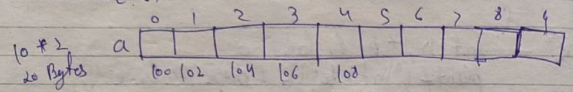
Ques :-> Program to check a number is prime or not using pointer

```
#include<stdio.h>
int main()
{
int n, *p, i, count = 0;
printf("\n Enter No to check for prime:");
Scanf("%d", &n)
p = &n;
for(i=1; i<=*p; i++)
{
    if (*p % i == 0)
        count++;
}
if (count == 2)
    printf("Prime No:");
else
    Printf("Not Prime");
return 0;
}
```

Pointer with Array



int a[10];



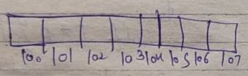
char a[10]

a[0] = 10

a[5] = 15

1 Byte

float a[10];



Ques Pointer with array to find sum of array element.

```
#include<stdio.h>
int main()
{
 int a[10], i, *p, sum=0;          a[i] = *(p+i)
  for(i=0; i<10; i++)
   {
    printf("\n Enter Number:");
    scanf("%d", &a[i]);
   }
    p = &a[0];
    for(i=0; i<10; i++)
          sum = sum + *(p+i);
    printf("\n Sum of array elements = %d", sum);
    return 0;
}
```

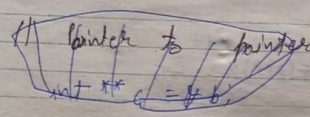
Ques Pointer with Array to found total even/odd numbers

```
#include<stdio.h>
int main()
{
 int a[10], i, *p, even=0, odd=0;
   for(i=0; i<10; i++)
  {
   printf("\n Enter Number:");
   scanf("%d", &a[i]);
  }
    p = &a[0];
    for(i=0; i<10; i++)
     { if (*(p+i)%2==0)
            even++;
       else
            odd++;
     }
    printf("\n Total even = %d total odd = %d", even, odd);
    return 0;
}
```

Pointer with function

```
#include<stdio.h>
=> void add(int*, int*);
   int main()
{
   int a, b, *p, *q;
  printf("\n Enter Two Number:");
   scanf("%d %d", &a, &b);
     p = &a;
     q = &b;
     add(p, q);
```

```
return 0;
}
Void add (int *p, int *q)
{
    int c;
    c = *p + *q;
    Printf("\n Addition = %d", C);
}
```




```
int a=3;
int *b;
 b = &a;

Cout<<" The address of a is "<<&a<<endl;
Cout<<" The address of a is "<<b<<endl;

Cout<<" The value at address b is "<<*b<<endl;

int **c = &b;

Cout<<" The address of b is "<<&b<<endl;
Cout<<" The address of b is "<<c<<endl;
Cout<<" The value at address c is "<<*c<<endl;
Cout<<" The value at address value_at(value_at(c)) is"
                                    <<**c<<endl;
```

output

```
The address of a is 0x61ff08
The address of a is 0x61ff08

The value at address b is 3

The address of b is 0x61ff04
The address of b is 0x61ff04

The value at address c is 0x61ff08
The value at address value_at
                (value_at(c)) is 3
```