* **Array -**
  Array is a collection of similar data items or elements. Or we can say that array is a collection of homogeneous data ~~elect~~ element stored continuously under a single name.

* Each data item of an array is called an element and each element is unique and located in separated memory location.

* Array of character is known as string.

* Each location of an element in an array has a numerical index no., known as subscript. which is used to identify the element.

* Index is always starts with zero.

* An array can be a single dimensional or multi-dimensional.

* One dimensional array is known as vector and two dimensional arrays are known as matrix.

* Declaration of an array -

  syntax -

  data type array name [size];

  Eg:- int arr[100];
  int sub[100];
  int a[5] = {10, 20, 30, 100, 5}

Date ____
Page ____

* Array subscript (index) always start from zero which is known as lower bound and upper value is known as upper bound.

```
           size  Index 0    1   2   3    4
int   arr[5] = { 20, 60, 90, 100, 120 }
 ↓       ↓          ↓                 ↓
type    name      lower             upper
                  bound             bound
```

→ Input values into an array and display them -

```
# include <stdio.h>
int main()
{
   int arr [5];
   int i ;
   for (i = 0 ; i < 5 ; i++)
   {
      printf ("Enter a value for arr[%d]\n", i);
      scanf ("%d", & arr[i]);
   }
   printf (" The array elements are \n");
   for (i = 0 ; i < 5 ; i++)
   {
      print ("%d\t", arr[i]);
   }
}
```

* Two Dimensional array - (Matrix)

Two dimensional array is popularly known as tables or matrix and can be easily visualized as having rows and columns.

To creat a two dimensional array, specifying both dimensions i.e. rows and columns in square brackets.

syntax -

data type array name [row] [column];

* 2-D array is a collection of 1-D array placed one below the other.

* Total no. of elements in 2-D array is calculated as "row * column".

Eg:- int ar[2][3]

Total no. of elements = row * column

= 2 * 3 = 6

It means the matrix consist of 2 rows and 3 columns.

* Matrix initializes -

int mat[3][3];

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

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

And while initializing, it is necessary to mention the 2nd dimension where 1st dimension is optional.

int mat[][3]; } valid
int mat[2][3];

int mat[][]; } invalid
int mat[2][];

* We can also give the size of the 2-D array by using symbolic constant such as -

```
#define row 2;
#define column 3;
int mat[row][column];
```

* User input in double dimension array -

```
#include <stdio.h>
int main()
{ int ar[50][50], row, col;
  int i,j;
 printf("Enter row\n");
 scanf("%d", &row);
 printf(" Enter column\n");
 scanf("%d", &column);
  for(i=0; i<row; i++);
    {
      for(j=0; j<col; j++);
        { scanf("%d", ar[i][j]);
        }
    }
 printf(" Elements of array is given below\n");
    for(i=0; i<row; i++)
      { for(j=0; j<col; j++)
          { printf("%d", ar[i][j]);
          }
        printf("\n");
      }
}
```

* String (Character array):-
An string is an array of characters. Any group of characters enclosed between double quotes is called a string constant.

* A string is a one-dimensional array of characters terminated by null(\0).

* Character array initialization -

```
char a[10];
char a[10] = {'R', 'A', 'M', '\0'}
char a[10] = {'R', 'A', 'M'}
char a[10] = "RAM";
char a[] = {'R', 'A', 'M'}
```

(If null(\0) is not given at the end of character then the compiler automatically assume it so it is not necessary to giving null at ~~every~~ last character of word.)

* String input -
As we know, 'scanf' is ~~a~~ ~~co~~ used for input the data and this function can also be used to input the ~~data~~ string using format specifier (%s). and while inputting the string, the address operator (&) is not required.

Eg:- scanf("%s", Ram);

The problem with the scanf function is that it reads a single character at a time, so to read the entire line of text, we use gets() function. and it ends when it find a space

scanf("%s", Ram kumar); (reads only Ram not kumar)
gets(Ram kumar); (it reads all the words)

* String output -

To output a string, we can use printf function with format specifier (%s) as well as 'puts()' function.

Eg:- printf("%s", name);
or, puts(name);

Q Program to read a string and find out its length -

```
#include <stdio.h>
void main()
{
    char str[10];
    int i, c = 0;
    printf("\n Enter a string");
    gets(str);
    for (i=0; str[i] != '\0'; i++)
    {
        c++;
    }
    printf("\n length = %d", c);
}
```

Date
Page

* String Library Function -

These are several string library functions used to manipulate string and the prototypes for these functions are in header file "string.h" There are many string function -

1) strlen() -

This function return the length of the string i.e the no. of characters in the string excluding the terminating ('\0') character.

It accepts a single argument which is pointer to the first character of the string.

Eg:- strlen("Pencil");

output - 6

* Program to find length of character using string function.

```
# include <stdio.h>
# include <string.h>
void main()
{   char str[50];
    printf(" Enter a string - ");
      gets(str);
    printf(" Length of the string is %d \n", strlen(str));
}
```

Output -

Enter a string : Hello World

Length of ——— : 11

* ii) strcmp() -

This function is used to compare two strings. If the two string match, strcmp() return a value 0 otherwise it returns a non-zero value.

* It compare the strings characters by characters and the comparison stop when the end of the string is reached.

* Program to string comparison -

```
#include <stdio.h>
#include <string.h>
void main()
{
   char str1[10], str2[10];
   printf("Enter two string-");
    gets(str1);
    gets(str2);
   if (strcmp(str1, str2) == 0)
    {
       printf("String are same\n");
    }
   else
       {
          printf("String are not same\n");
       }
   }
}
```

iii) strcpy() -

This function is used to copying one string to another string.

The function strcpy(str1, str2) copies str2 to str1 including the null character. Here str2 is the source string and str1 is the destination string.

The old content of the destination string str1 are lost. The function returns a pointer to destination string str1.

* Program to use strcpy() function -

```
# include <stdio.h>
# include <string.h>
void main()
{ char str1[10], str2[10];
  printf("Enter a string-");
  scanf("%s", str2);
  strcpy(str1, str2);
  printf("first string = %s
```

*iv) strcat().

This function is used to append a copy of a string at the end of the other string.

* The null character from str1 is moved and str2 is added at the end of str1.

Q
```
#include <stdio.h>
#include <string.h>
void main()
{ char str1[20] st
```