Arrays:

An array is defined as finite ordered collection of homogenous data, stored in contiguous memory locations.

Here the words,

- ✓ Finite means data range must be defined.
- ✓ Ordered means data must be stored in continuous memory addresses.
- ✓ Homogenous means data must be of similar data type.

Example where arrays are used,

- ✓ to store list of Employee or Student names,
- ✓ to store marks of a students,
- ✓ or to store list of numbers or characters etc.

` Declaring an Array

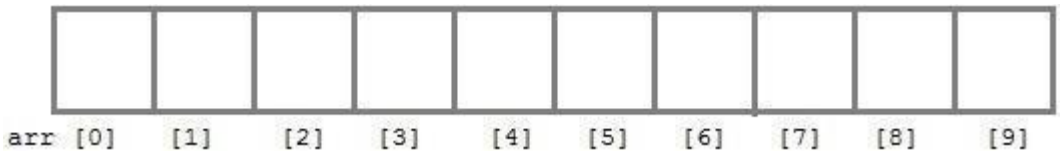
Like any other variable, arrays must be declared before they are used. General form of array declaration is,

data-type variable-name[size];

For example:

int arr[10];

In the above example the array name arr is declared of type integer and of size 10 .Each arr holds 2 bytes as the size of integer is 2 bytes .



Here int is the data type, arr is the name of the array and 10 is the size of array. It means array arr can only contain 10 elements of int type. Index of an array starts from 0 to size-1 i.e first element of arr array will be stored at arr[0] address and last element will occupy arr[9].

TYPES OF C ARRAYS:

- ✓ One – Dimensional Arrays
- ✓ Two – Dimensional Arrays
- ✓ Multi-Dimensional Arrays

One – Dimensional Arrays

Sequential collections of data of same type with one subscript is called One-Dimensional Arrays.

type arrayName [arraySize];

For example,

float mark[5];// it contains one subscript which is used to store the size of array.

Here, we declared an array, mark, of floating-point type and size 5. Meaning, it can hold 5 floating-point values.

Initialization of an Array (Imp)

After an array is declared it must be initialized. Otherwise, it will contain garbage value (any random value).

An array can be initialized at either compile time or at runtime

Compile time Array initialization of One- Dimensional Array

Compile time initialization of array elements is same as ordinary variable initialization.
The values for the array is initialized in the code directly.

The general form of initialization of array is,

type array-name[size] = { list of values };

int marks[4]={ 67, 87, 56, 77 }; //integer array initialization
float area[5]={ 23.4, 6.8, 5.5 }; //float array initialization

Program Example :

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i;
 int arr[]={2,3,4};  //Compile time array initialization
 for(i=0 ; i<3 ; i++) {
   printf("%d\t",arr[i]);
 }
 getch();
}
```

Output

2 3 4

Runtime Array initialization of One- Dimensional Array

An array can also be initialized at runtime using scanf() function. This approach is usually used for initializing large array, or to initialize array with user specified values.

Example,

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int arr[4];
 int i, j;
 printf("Enter array element");
 for(i=0;i<4;i++)
 {
  scanf("%d",&arr[i]);    //Run time array initialization
 }
 for(j=0;j<4;j++)
 {
  printf("%d\n",arr[j]);
 }
 getch();
}
```

Two – Dimensional Arrays

The arrays with contains rows and columns (2 Subscript) is called as Two-Dimensional Arrays.

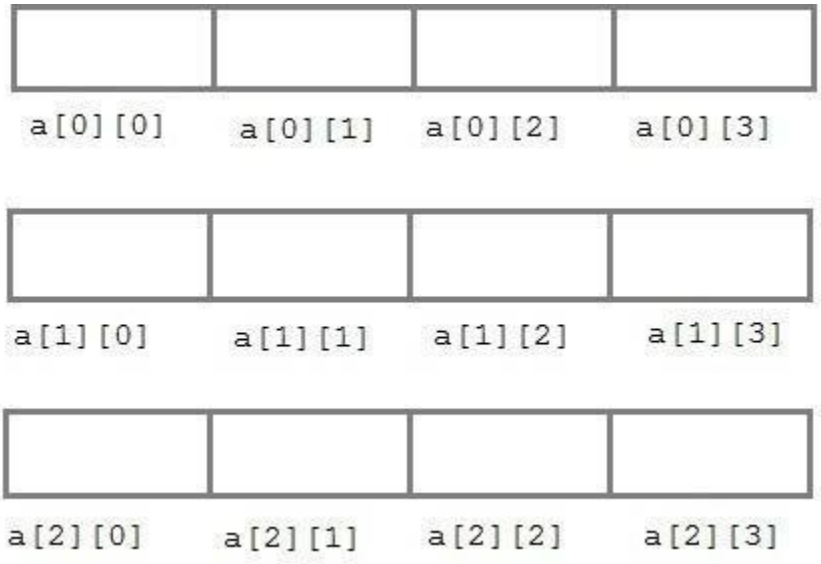
Two Dimensional Array stores the values in the form of matrix.

Syntax :

type array-name[row-size][column-size] ;

Example :

int a[3][4];



In the above example the arrays consists of 3 rows and 4 columns.

Compile time Array initialization of Two- Dimensional Array

Compile time initialization of array elements is same as ordinary variable initialization.

The values for the array is initialized in the code directly.

Syntax:

type array-name[row_size][col-size] = { list of values };

int arr[2][2] = {10,20,30,40};

EXAMPLE PROGRAM FOR TWO DIMENSIONAL ARRAY IN C:

```
#include<stdio.h>

int main()
{
  int i,j;
  // declaring and Initializing array
  int arr[2][2] = {10,20,30,40};
  for (i=0;i<2;i++)
  {
    for (j=0;j<2;j++)
    {
      Printf(“%d”,a[i][j]);
    }
Printf(“\n”);
  }
return 0;
}
```

Output:

```
10 20
30 40
```

Runtime Array initialization of Two- Dimensional Array

An array can also be initialized at runtime using scanf() function. This approach is usually used for initializing large array, or to initialize array with user specified values.

EXAMPLE PROGRAM FOR TWO DIMENSIONAL ARRAY IN C:

```
#include<stdio.h>

int main()
{
  int i,j;
  // declaring 2darray
  int arr[2][2] ;
  for (i=0;i<2;i++) // run time code
  {
    for (j=0;j<2;j++)
```

```
    {
      scanf(“%d”,&a[i][j]);
    }
  }

  for (i=0;i<2;i++)
  {
    for (j=0;j<2;j++)
    {
      Printf(“%d”,a[i][j]);
    }
Printf(“\n”);
  }
return 0;
}
```

Output:

10 20 30 40

10 20

30 40

How to pass arrays to a function in C Programming?

In C programming, a single array element or an entire array can be passed to a function.

This can be done for both one-dimensional array and 2-dimensional array.

Passing One-dimensional Array In Function

Single element of an array can bepassed in similar manneras passing variable to a function.

Passing an entire one-dimensional array to a function

While passing arrays as arguments to the function, only the name of the array is passed (,i.e, starting address of memory area is passed as argument).

```
#include<stdio.h>
Void main()
{
Int arr[5]={10,20,30,40,50};
Sum=add(arr);
Printf(“%d”,sum);    }
```

```
Int add(int b[])
{
int i,s=0;
for (i=0; i<5;i++)
{
        s=s+b[i];
}
return s;
}
```

Output: 150

String

- String is nothing but the collection of the individual array elements or characters stored at contiguous memory locations
- String is enclosed within Double quotes.
- String always Terminated with NULL Character (‘/0’).

Syntax:
Char Stringname[sizeofcharacters];

Example : char name[10];

Initializing String [Character Array] :

Whenever we declare a String then it will contain garbage values inside it. We have to initialize String or Character array before using it. Process of assigning some legal default data to String is Called **Initialization of String.**

char name [] = {'P','R','I','T','E','S','H','\0'};

In the above example name is assigned with a string PRITESH and Ends with Null Character.

Program example for basic string concept

```
Void main()
{
Char name[10];
Printf(“Enter yourname\n”);
Scanf(“%s”,name);
Printf(“The entered name is %s”, name);
}
```

Output:
Enter your name Pradhan
The entered name is Pradhan

Inbuilt String input and Output function.

- **gets()**

  Reads characters from the standard input (stdin) and stores them as a C string
  Syntax:
  gets(<variable-name>)

- **puts()**

  puts() can be used to display message.

  puts(string_Variable_name) ;

```
#include<stdio.h>
void main()
{
char name[20];
printf("\nEnter the Name : ");
gets(name);
puts(name);
}
```

In above example gets(name) is used to get the input from user and puts(name) is to print the name readed from user.

- **getchar()**

  getchar() function is also one of the function which is used to accept the single character from the user.

  Syntax for Accepting String and Working :

```
/* getchar accepts character & stores in ch */
char ch =getchar();
```

- **putchar()**

  putchar() displays character stored in variable.

Syntax

putchar(Variable);

```
#include<stdio.h>
void main()
{
char c;
printf("\nEnteracharacter");
c=getchar();
putchar(c);
}
```

Inbuilt String Functions in C

❖ strlen() : Finding Length of String

```
void main()
{
 Char name=”Hello”;
 Int len=strlen(name);
 Printf(“The length of the String is %d”,len);

}
```

Output: 5

❖ Strcat()

It is used to combine two strings together.

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

int main()
{
char str1[] = "This is ", str2[] = "crossmove";
//concatenates str1andstr2andresultantstringisstoredinstr1.
strcat(str1,str2);

puts(str1);
puts(str2);
```

```
return 0;
}
```

Output

```
This is crossmove
        Crossmove
```

❖ strcpy()

The strcpy() function copies the string pointed by source (including the null character) to the character array destination.

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

int main()
{
   char str1[10]="awesome";
   char str2[10];
   char str3[10];

   strcpy(str2, str1);
   strcpy(str3, "well");
   puts(str2);
   puts(str3);

   return 0;
}
```

Output : awesome
Well

❖ strcmp()

The strcmp() compares two strings character by character. If the first character of two strings are equal, next character of two strings are compared. This continues until the corresponding characters of two strings are different or a null character '\0' is reached.

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

int main()
{
  char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
```

```
        int result;
    //comparing strings str1 and str2
    result = strcmp(str1, str2);
    printf("strcmp(str1, str2) = %d\n", result);
    //comparing strings str1 and str3
    result = strcmp(str1, str3);
    printf("strcmp(str1, str3) = %d\n", result);

    return 0;
        }
```

Output

strcmp(str1, str2) = 32
strcmp(str1, str3) = 0

The first unmatched character between string str1 and str2 is third character. The ASCII value of 'c' is 99 and the ASCII value of 'C' is 67. Hence, when strings str1 and str2 are compared, the return value is 32.

When strings str1 and str3 are compared, the result is 0 because both strings are identical.

❖ **strlwr( )**

strlwr( ) function converts a given string into lowercase.

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

OUTPUT:
modify

❖ **strrev()**

strrev( ) function reverses a given string in C language.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char name[30] = "Hello";
```

```
    printf("String before strrev( ) : %s\n",name);
    printf("String after strrev( ) :
    %s",strrev(name)); return 0;
  }
   OUTPUT:
  Stringbefore strrev():
  Hello String after
  strrev()

  : olleH
```

❖ **strupr()**

strupr( ) function converts a given string into uppercase.

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

int main()
{
  char str[]="Modify This String To
  Upper"; printf("%s\n",strupr(str));
  return 0;
}
```

Output : MODIFY THIS STRING TO UPPER