



# *Array & Strings*



# Need of Array Variable

- ▶ Suppose we need to store `rollno` of the student in the integer variable.

Declaration

```
int rollno;
```

- ▶ Now we need to store `rollno` of 100 students.

Declaration

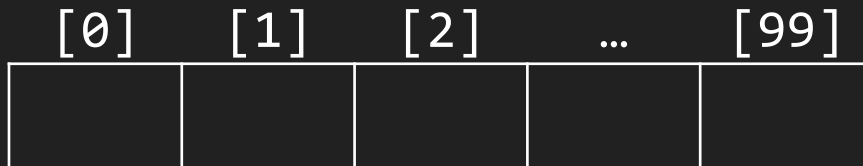
```
int rollno101, rollno102, rollno103, rollno104...;
```

- ▶ This is **not appropriate** to declare these many integer variables.  
e.g. 100 integer variables for `rollno`.
- ▶ Solution to declare and store multiple variables of similar type is an **array**.
- ▶ An **array** is a variable that can store multiple values.

# Definition: Array

- ▶ An array is a fixed size sequential collection of elements of same data type grouped under single variable name.

```
int rollno[100];
```



## Fixed Size

Here, the size of an array is 100 (fixed) to store rollno

## Sequential

It is indexed to 0 to 99 in sequence

## Same Data type

All the elements (0-99) will be integer variables

## Single Name

All the elements (0-99) will be referred as a common name rollno

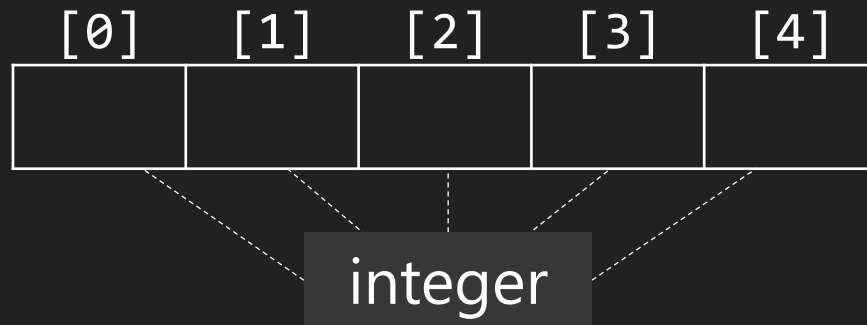
# Declaring an array

## Syntax

```
data-type variable-name[size];
```

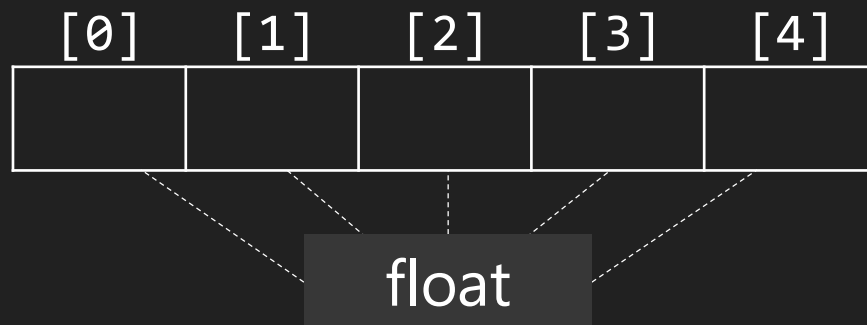
## Integer Array

```
int mark[5];
```



## Float Array

```
float avg[5];
```



- ▶ By default array index starts with 0.
- ▶ If we declare an array of size 5 then its index ranges from 0 to 4.
- ▶ First element will be stored at `mark[0]` and last element will be stored at `mark[4]` not `mark[5]`.
- ▶ Like integer and float array we can declare array of type `char`.

# Initializing and Accessing an Array

Declaring, initializing and accessing single integer variable

```
int mark=90;      //variable mark is initialized with value 90
printf("%d",mark); //mark value printed
```

Declaring, initializing and accessing integer array variable

```
int mark[5]={85,75,76,55,45}; //mark is initialized with 5 values
printf("%d",mark[0]); //prints 85
printf("%d",mark[1]); //prints 75
printf("%d",mark[2]); //prints 65
printf("%d",mark[3]); //prints 55
printf("%d",mark[4]); //prints 45
```

	[0]	[1]	[2]	[3]	[4]
mark[5]	85	75	65	55	45

# Read(Scan) Array Elements

## Reading array without loop

```
1 void main()
2 {
3     int mark[5];
4     printf("Enter array element=");
5     scanf("%d",&mark[0]);
6     printf("Enter array element=");
7     scanf("%d",&mark[1]);
8     printf("Enter array element=");
9     scanf("%d",&mark[2]);
10    printf("Enter array element=");
11    scanf("%d",&mark[3]);
12    printf("Enter array element=");
13    scanf("%d",&mark[4]);
14
15    printf("%d",mark[0]);
16    printf("%d",mark[1]);
17    printf("%d",mark[2]);
18    printf("%d",mark[3]);
19    printf("%d",mark[4]);
20 }
```

## Reading array using loop

```
1 void main()
2 {
3     int mark[5],i;
4     for(i=0;i<5;i++)
5     {
6         printf("Enter array element=");
7         scanf("%d",&mark[i]);
8     }
9     for(i=0;i<5;i++)
10    {
11        printf("%d",mark[i]);
12    }
13 }
```



# Develop a program to count number of positive or negative number from an array of 10 numbers.

## Program

```
1 void main(){
2     int num[10],i,pos,neg;
3     pos = 0;
4     neg = 0;
5     for(i=0;i<10;i++)
6     {
7         printf("Enter array element=");
8         scanf("%d",&num[i]);
9     }
10    for(i=0;i<10;i++)
11    {
12        if(num[i]>0)
13            pos=pos+1;
14        else
15            neg=neg+1;
16    }
17    printf("Positive=%d,Negative=%d",pos,neg);
18 }
```

## Output

```
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
Enter array element=-1
Enter array element=-2
Enter array element=3
Enter array element=4
Enter array element=5
Positive=8,Negative=2
```

# Develop a program to read n numbers in an array and print them in reverse order.

## Program

```
1 void main()
2 {
3     int num[100],n,i;
4     printf("Enter number of array elements=");
5     scanf("%d",&n);
6     //loop will scan n elements only
7     for(i=0;i<n;i++)
8     {
9         printf("Enter array element=");
10        scanf("%d",&num[i]);
11    }
12    //negative loop to print array in reverse order
13    for(i=n-1;i>=0;i--)
14    {
15        printf("%d\n",num[i]);
16    }
17 }
```

## Output

```
Enter number of array
elements=5
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
5
4
3
2
1
```



# Practice Programs

- 1) Develop a program to calculate sum of n array elements in C.
- 2) Develop a program to calculate average of n array elements in C.
- 3) Develop a program to find largest array element in C.
- 4) Develop a program to print sum of second and second last element of an array.
- 5) Develop a program to copy array elements to another array.
- 6) Develop a program to count odd and even elements of an array.



# *Multi Dimensional Array*



# Declaring 2 Dimensional Array

## Syntax

```
data-type variable-name[x][y];
```

## Declaration

```
int data[3][3]; //This array can hold 9 elements
```

```
int data[3][3];
```

	Column-0	Column-1	Column-2
Row-0	data[0][0]	data[0][1]	data[0][2]
Row-1	data[1][0]	data[1][1]	data[1][2]
Row-2	data[2][0]	data[2][1]	data[2][2]

- ▶ A two dimensional array can be seen as a table with '**x**' rows and '**y**' columns.
- ▶ The row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1).

# Initializing and Accessing a 2D Array: Example-1

## Program

```
1 int data[3][3] = {
2   {1,2,3}, //row 0 with 3 elements
3   {4,5,6}, //row 1 with 3 elements
4   {7,8,9}  //row 2 with 3 elements
5 };
6 printf("%d",data[0][0]); //1
7 printf("%d",data[0][1]); //2
8 printf("%d\n",data[0][2]); //3
9
10 printf("%d",data[1][0]); //4
11 printf("%d",data[1][1]); //5
12 printf("%d\n",data[1][2]); //6
13
14 printf("%d",data[2][0]); //7
15 printf("%d",data[2][1]); //8
16 printf("%d",data[2][2]); //9

1 // data[3][3] can be initialized like this also
2 int data[3][3]={ {1,2,3},{4,5,6},{7,8,9}};
```

	Column-0	Column-1	Column-2
Row-0	1	2	3
Row-1	4	5	6
Row-2	7	8	9

# Initializing and Accessing a 2D Array: Example-2

## Program

```
1 int data[2][4] = {
2   {1,2,3,4}, //row 0 with 4 elements
3   {5,6,7,8}, //row 1 with 4 elements
4   };
5 printf("%d",data[0][0]); //1
6 printf("%d",data[0][1]); //2
7 printf("%d",data[0][2]); //3
8 printf("%d\n",data[0][3]); //4
9
10 printf("%d",data[1][0]); //5
11 printf("%d",data[1][1]); //6
12 printf("%d",data[1][2]); //7
13 printf("%d",data[1][3]); //8
```

```
1 // data[2][4] can be initialized like this also
2 int data[2][4]={ {1,2,3,4},{5,6,7,8}};
```

	Col-0	Col-1	Col-2	Col-3
Row-0	1	2	3	4
Row-1	5	6	7	8

# Read(Scan) 2D Array Elements

## Program

```
1 void main(){
2     int data[3][3],i,j;
3     for(i=0;i<3;i++)
4     {
5         for(j=0;j<3;j++)
6         {
7             printf("Enter array element=");
8             scanf("%d",&data[i][j]);
9         }
10    }
11    for(i=0;i<3;i++)
12    {
13        for(j=0;j<3;j++)
14        {
15            printf("%d",data[i][j]);
16        }
17        printf("\n");
18    }
19 }
```

	Column-0	Column-1	Column-2
Row-0	1	2	3
Row-1	4	5	6
Row-2	7	8	9

## Output

```
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
Enter array element=6
Enter array element=7
Enter array element=8
Enter array element=9
123
456
789
```

# Develop a program to count number of positive, negative and zero elements from 3 X 3 matrix

## Program

```
1 void main(){
2     int data[3][3],i,j,pos=0,neg=0,zero=0;
3     for(i=0;i<3;i++)
4     {
5         for(j=0;j<3;j++)
6         {
7             printf("Enter array element=");
8             scanf("%d",&data[i][j]);
9             if(data[i][j]>0)
10                 pos=pos+1;
11             else if(data[i][j]<0)
12                 neg=neg+1;
13             else
14                 zero=zero+1;
15         }
16     }
17     printf("positive=%d,negative=%d,zero=%d",pos,neg,zero);
18 }
```

## Output

```
Enter array element=9
Enter array element=5
Enter array element=6
Enter array element=-3
Enter array element=-7
Enter array element=0
Enter array element=11
Enter array element=13
Enter array element=8
positive=6,negative=2,zero=1
```

# Practice Programs

1. Develop a program to perform addition of two matrix.
2. Develop a program to perform multiplication of two matrix.





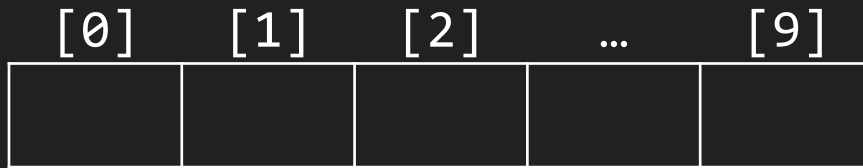
# *String* *(Character Array)*



# Definition: String

- ▶ A String is a one-dimensional array of characters terminated by a `null('\0')`.

```
char name[10];
```



- ▶ Each character in the array occupies one byte of memory, and the last character must always be `null('\0')`.
- ▶ The termination character (`'\0'`) is important in a string to identify where the string ends.



# Declaring & Initializing String

Declaration

```
char name[10];
```

Initialization method 1:

```
char name[10]={ 'D', 'A', 'R', 'S', 'H', 'A', 'N', '\0' };
```

Initialization method 2:

```
char name[10]="DARSHAN";  
//'\0' will be automatically inserted at the end in this type of declaration.
```

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
name[10]	D	A	R	S	H	A	N	\0		

# Read String: scanf()

## Program

```
1 void main()
2 {
3     char name[10];
4     printf("Enter name:");
5     scanf("%s",name);
6     printf("Name=%s",name);
7 }
```

## Output

```
Enter name: Darshan
Name=Darshan
```

## Output

```
Enter name: CE Darshan
Name=CE
```

- ▶ There is no need to use address of (&) operator in **scanf** to store a string.
- ▶ As string **name** is an array of characters and the name of the array, i.e., **name** indicates the base address of the string (character array).
- ▶ **scanf()** terminates its input on the first whitespace(space, tab, newline etc.) encountered.

# Read String: gets()

## Program

```
1 #include<stdio.h>
2 void main()
3 {
4     char name[10];
5     printf("Enter name:");
6     gets(name); //read string including white spaces
7     printf("Name=%s",name);
8 }
```

## Output

```
Enter name:Parul Institute
Name=Parul Institute
```

- ▶ **gets()**: Reads characters from the standard input and stores them as a string.
- ▶ **puts()**: Prints characters from the standard.
- ▶ **scanf()**: Reads input until it encounters whitespace, newline or End Of File (EOF) whereas **gets()** reads input until it encounters newline or End Of File (EOF).
- ▶ **gets()**: Does not stop reading input when it encounters whitespace instead it takes whitespace as a string.

# String Handling Functions : strlen()

- ▶ C has several inbuilt functions to operate on string. These functions are known as string handling functions.
- ▶ **strlen(s1)**: returns length of a string in integer

## Program

```
1 #include <stdio.h>
2 #include <string.h> //header file for string functions
3 void main()
4 {
5     char s1[10];
6     printf("Enter string:");
7     gets(s1);
8     printf("%d",strlen(s1)); // returns length of s1 in integer
9 }
```

## Output

```
Enter string: CE Darshan
10
```

# String Handling Functions: strcmp()

- ▶ `strcmp(s1,s2)`: Returns 0 if s1 and s2 are the same.
- ▶ Returns less than 0 if  $s1 < s2$ .
- ▶ Returns greater than 0 if  $s1 > s2$ .

## Program

```
1 void main()
2 {
3     char s1[10],s2[10];
4     printf("Enter string-1:");
5     gets(s1);
6     printf("Enter string-2:");
7     gets(s2);
8     if(strcmp(s1,s2)==0)
9         printf("Strings are same");
10    else
11        printf("Strings are not same");
12 }
```

## Output

```
Enter string-1:Computer
Enter string-2:Computer
Strings are same
```

## Output

```
Enter string-1:Computer
Enter string-2:Computer
Strings are same
```

# String Handling Functions

For examples consider: `char s1[]="Their", s2[]="There";`

Syntax	Description
<code>strcpy(s1,s2)</code>	Copies 2 <sup>nd</sup> string to 1 <sup>st</sup> string. <code>strcpy(s1,s2)</code> copies the string <code>s2</code> in to string <code>s1</code> so <code>s1</code> is now "There". <code>s2</code> remains unchanged.
<code>strcat(s1,s2)</code>	Appends 2 <sup>nd</sup> string at the end of 1 <sup>st</sup> string. <code>strcat(s1,s2);</code> a copy of string <code>s2</code> is appended at the end of string <code>s1</code> . Now <code>s1</code> becomes "TheirThere"
<code>strchr(s1,c)</code>	Returns a pointer to the first occurrence of a given character in the string <code>s1</code> . <code>printf("%s",strchr(s1,'i'));</code> Output : ir
<code>strstr(s1,s2)</code>	Returns a pointer to the first occurrence of a given string <code>s2</code> in string <code>s1</code> . <code>printf("%s",strstr(s1,"he"));</code> Output : heir



# String Handling Functions (Cont...)

For examples consider: `char s1[]="Their", s2[]="There";`

Syntax	Description
<code>strrev(s1)</code>	Reverses given string. <code>strrev(s1);</code> makes string s1 to "riehT"
<code>strlwr(s1)</code>	Converts string s1 to lower case. <code>printf("%s", strlwr(s1));</code> Output : their
<code>strupr(s1)</code>	Converts string s1 to upper case. <code>printf("%s", strupr(s1));</code> Output : THEIR
<code>strncpy(s1, s2, n)</code>	Copies first n character of string s2 to string s1 <code>s1=""</code> ; <code>s2="There"</code> ; <code>strncpy(s1, s2, 2);</code> <code>printf("%s", s1);</code> Output : Th
<code>strncat(s1, s2, n)</code>	Appends first n character of string s2 at the end of string s1. <code>strncat(s1, s2, 2);</code> <code>printf("%s", s1);</code> Output : TheirTh

# String Handling Functions (Cont...)

For examples consider: `char s1[]="Their", s2[]="There";`

Syntax	Description
<code>strncmp(s1,s2,n)</code>	Compares first n character of string s1 and s2 and returns similar result as strcmp() function. <code>printf("%d",strcmp(s1,s2,3));</code> Output : 0
<code>strrchr(s1,c)</code>	Returns the last occurrence of a given character in a string s1. <code>printf("%s",strrchr(s2,'e'));</code> Output : ere



*Thank you*

