

Unit 5

Array and String

- Introduction to Array, Declaration, Initialization
- Types of Arrays (1-D Array, Multi-dimensional Array)
- String, Array of String
- String Handling Function (strlen(), strrev(),strupr(), strlwr(), strcpy(), strcat(), strcmp())

Array:

- An array is the collection of similar type of data items
- It is a variable that can store multiple values.
- For example, if we want to store 100 integers, we can create an array for it.

```
int data[100]
```



ARRAY IN C

The diagram shows three array declarations labeled a, b, and c. Each declaration is shown in a row of boxes representing the array elements. To the right of each row is a circular icon indicating whether the declaration is valid.

- a: [5 | 6 | 10 | 13 | 56 | 76 | 1 | 2 | 4 | 8] ✓ (Valid)
- b: ['a' | 'b' | 'c' | 'd' | 'e'] ✓ (Valid)
- c: ['a' | 'b' | 1 | 5.6 | 'e' | 34 | 2 | 3] ✗ (Invalid)

Array declaration:

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 –

Syntax: `dataType arrayName[arraySize];`

For example: `float mark[5];`

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

Array Initialization:

It is possible to initialize an array during declaration. For example,

```
int mark[5] = {19, 10, 8, 17, 9};
```

We can also initialize an array like this.

```
int mark[] = {19, 10, 8, 17, 9};
```

Accessing array elements

- We can access elements of an array by indices.
- Suppose we declared an array mark as: `int mark[5] = {19, 10, 8, 17, 9};`
- The first element is `mark[0]`, the second element is `mark[1]` and so on.

Input and Output Array Elements

Here's how we can take input from the user and store it in an array element.

```
// take input and store it in the 3rd element
scanf("%d", &mark[2]);

// take input and store it in the ith element
scanf("%d", &mark[i-1]);
```

Here's how we can print an individual element of an array.

```
// print the first element of the array
printf("%d", mark[0]);

// print the third element of the array
printf("%d", mark[2]);
```

```
// print ith element of the array
printf("%d", mark[i-1]);
```

Example 1: Array Input/Output

```
// Program to take 5 values from the user and store them in an array
// Print the elements stored in the array

#include <stdio.h>

int main() {

    int values[5];

    printf("Enter 5 integers: ");

    // taking input and storing it in an array
    for(int i = 0; i < 5; ++i) {
        scanf("%d", &values[i]);
    }

    printf("Displaying integers: ");

    // printing elements of an array
    for(int i = 0; i < 5; ++i) {
        printf("%d\n", values[i]);
    }
    return 0;
}
```

Example 2: Calculate Average

```
// Program to find the average of n numbers using arrays
```

```

#include <stdio.h>

int main() {

    int marks[10], i, n, sum = 0;
    double average;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    for(i=0; i < n; ++i) {
        printf("Enter number%d: ",i+1);
        scanf("%d", &marks[i]);

        // adding integers entered by the user to the sum variable
        sum += marks[i];
    }

    // explicitly convert sum to double
    // then calculate average
    average = (double) sum / n;

    printf("Average = %.2lf", average);

    return 0;
}

```

Calculate sum of all array elements

```

#include<stdio.h>
void main(){

    int arr[5];
    printf("Enter array elements:");
    for(int i = 0; i < 5; i++)
        scanf("%d", &arr[i]);
}

```

```
printf("Array elements are:");
for(int i = 0; i < 5; i++)
    printf("%d ", arr[i]);
int sum = 0;
for(int i = 0; i < 5; i++)
    sum += arr[i];
printf("Sum =%d", sum);
}
```

Two-dimensional array (2D array)

In C programming, we can create an array of arrays. These arrays are known as multidimensional arrays.

```
float x[3][4];
```

Here, x is a two-dimensional (2d) array. The array can hold 12 elements. We can think the array as a table with 3 rows and each row has 4 columns.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Similarly, we can declare a three-dimensional (3d) array. For example,

```
float y[2][4][3];
```

Initialization of a 2d array

2D array can be initialize in following way:

```
int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[][3] = {{1, 3, 0}, {-1, 5, 9}};
```

Initialization of a 3d array

We can initialize a three-dimensional array in a similar way to a two-dimensional array. Here's an example,

```
int test[2][3][4] = {
    {{3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2}},
    {{13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9}}};
```

Example 2: Sum of two matrices

```
// C program to find the sum of two matrices of order 2*2

#include <stdio.h>
int main()
{
    float a[2][2], b[2][2], result[2][2];

    // Taking input using nested for loop
    printf("Enter elements of 1st matrix\n");
    for (int i = 0; i < 2; ++i)
        for (int j = 0; j < 2; ++j)
        {
            printf("Enter a%d%d: ", i + 1, j + 1);
            scanf("%f", &a[i][j]);
        }

    // Taking input using nested for loop
    printf("Enter elements of 2nd matrix\n");
    for (int i = 0; i < 2; ++i)
```

```

for (int j = 0; j < 2; ++j)
{
    printf("Enter b%d%d: ", i + 1, j + 1);
    scanf("%f", &b[i][j]);
}

// adding corresponding elements of two arrays
for (int i = 0; i < 2; ++i)
    for (int j = 0; j < 2; ++j)
    {
        result[i][j] = a[i][j] + b[i][j];
    }

// Displaying the sum
printf("\nSum Of Matrix:");

for (int i = 0; i < 2; ++i)
    for (int j = 0; j < 2; ++j)
    {
        printf("%.1f\t", result[i][j]);

        if (j == 1)
            printf("\n");
    }
return 0;
}

```

String

- In C programming, a string is a sequence of characters terminated with a null character \0.
- Strings are used for storing text/characters.
- For example:

```
char c[] = "c string";
```

String declaration

Here's how we can declare strings:


```
char s[5];
```

String Initialization

We can initialize strings in a number of ways.

```
char c[] = "abcd";  
  
char c[50] = "abcd";  
  
char c[] = {'a', 'b', 'c', 'd', '\0'};  
  
char c[5] = {'a', 'b', 'c', 'd', '\0'};
```

Accessing strings

Since strings are actually arrays in C, we can access a string by referring to its index number inside square brackets [].

```
Example: char greetings[] = "Hello World!";  
         printf("%c", greetings[0]);
```

In the above example H is printed because index 0 refers to first item.

Input and output of string

- We can use the we can use printf() and scanf() function to read and display a string.
- Alternatively, we can use gets() and puts() functions to read and display strings.
- The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab, etc.).
- To read all the characters we can use gets() function.

Example :

```
#include <stdio.h>  
int main()  
{  
    char name1[20];  
    printf("Enter name: ");  
    scanf("%s", name);
```

```
printf("Wer name is %s.", name);  
return 0;  
}
```

String functions

- We need to often manipulate strings according to the problem.
- C supports a large number of string handling functions in the standard library "string.h".
- Few commonly used string handling functions are discussed below:

Commonly Used String Functions

1. strlen() - calculates the length of a string
2. strcpy() - copies a string to another
3. strcmp() - compares two strings
4. strcat() – concatenates(joins) two strings
5. strrev() – reverses the given string
6. strlwr() – converts the string to lowercase
- 7.strupr() – converts the string to uppercase

Strings handling functions are defined under "string.h" header file.

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

1. strlen()

1. The strlen() function calculates the length of a given string.
2. The strlen() function takes a string as an argument and returns its length.

```
#include <stdio.h>  
#include <string.h>  
int main()  
{  
    char a[20]="Program";  
    char b[20]={'P','r','o','g','r','a','m','\0'};  
  
    printf("Length of string a = %d \n",strlen(a));  
    printf("Length of string b = %d \n",strlen(b));  
  
    return 0;
```

```
}
```

Output

```
Length of string a = 7  
Length of string b = 7
```

2. Strcpy()

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

```
#include <stdio.h>  
#include <string.h>  
  
int main() {  
    char str1[20] = "C programming";  
    char str2[20];  
  
    // copying str1 to str2  
    strcpy(str2, str1);  
  
    puts(str2); // C programming  
  
    return 0;  
}
```

Output

```
C programming
```

3. strcmp()

The strcmp() compares two strings character by character. If the strings are equal, the function returns 0.

```

#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) = 1
strcmp(str1, str3) = 0

```

4. strcat()

- In C programming, the strcat() function concatenates (joins) two strings.
- Example: C strcat() function

```

#include <stdio.h>
#include <string.h>
int main() {
    char str1[100] = "This is ", str2[] = "program.com";

    // concatenates str1 and str2
    // the resultant string is stored in str1.
    strcat(str1, str2);

    puts(str1);
    puts(str2);
}

```

```
    return 0;
}
```

Output

```
This is program
program.com
```

5. strrev()

The `strrev(string)` function returns reverse of the given string. Let's see a simple example of `strrev()` function.

```
#include<stdio.h>
#include <string.h>
int main(){
    char str[20];
    printf("Enter string: ");
    gets(str);//reads string from console
    printf("String is: %s",str);
    printf("\nReverse String is: %s",strrev(str));
    return 0;
}
```

Output:

```
Enter string: helloeveryone
String is: helloeveryone
Reverse String is: enoyreveolleh
```

6. strlwr()

The `strlwr(string)` function returns string characters in lowercase.

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

```

#include <string.h>
int main(){
    char str[20];
    printf("Enter string: ");
    gets(str);//reads string from console
    printf("String is: %s",str);
    printf("\nLower String is: %s",strlwr(str));
    return 0;
}

```

Output:

```

Enter string: HELLOworld
String is: HELLOworld
Lower String is: helloworld

```

7.strupr()

Thestrupr(string) function returns string characters in uppercase.

```

#include<stdio.h>
#include <string.h>
int main(){
    char str[20];
    printf("Enter string: ");
    gets(str);//reads string from console
    printf("String is: %s",str);
    printf("\nUpper String is: %s",strupr(str));
    return 0;
}

```

Output:

```

Enter string: helloworld
String is: helloworld
Upper String is: HELLOWORLD

```