|  |  |
| --- | --- |
| **CS118 Programming Fundamentals** | **LAB 08** 2DArray,String,Structure |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES** | |

# **ARRAY**

Arrays are of two types:

1. One-dimensional arrays
2. [Multidimensional arrays](https://www.programiz.com/c-programming/c-multi-dimensional-arrays)

[**Multidimensional arrays**](https://www.programiz.com/c-programming/c-multi-dimensional-arrays)

C programming language allows multidimensional arrays. Here is the general form of a multidimensional array declaration

type name[size1][size2]...[sizeN];

The simplest form of multidimensional array is the two-dimensional array. To declare a two-dimensional integer array of size [x][y], you would write something as follows .

type arrayName [ x ][ y ];

For example, the following declaration creates a two dimensional integer array .

int a[5][10];

A two-dimensional array can be considered as a table which will have x number of rows and y number of columns. A two-dimensional array a, which contains three rows and four columns can be shown as follow.



Thus, every element in the array a is identified by an element name of the form a[ i ][ j ], where 'a' is the name of the array, and 'i' and 'j' are the subscripts that uniquely identify each element in 'a'.

# **Initializing Two-Dimensional Arrays**

Arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row has 4 columns.

int a[3][4] = {

{0, 1, 2, 3} , /\* initializers for row indexed by 0 \*/

{4, 5, 6, 7} , /\* initializers for row indexed by 1 \*/

{8, 9, 10, 11} /\* initializers for row indexed by 2 \*/

};

The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to the previous example −

int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};

# **Accessing Two-Dimensional Array Elements**

An element in a two-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example −

int value = a[1][2];

The above statement will take the 3rd element from the 2nd row of the array. You can verify it in the above figure. Let us check the following program where we have used a nested loop to handle a two-dimensional array.

|  |
| --- |
| #include <stdio.h>  int main( )  {  int stud[3][2] ;  int i, j ;  for ( i = 0 ; i < 3 ; i++ )  {  for ( j= 0 ; j < 2 ; j++ )  {  printf ( "\n Enter roll no. and marks" ) ;  scanf ( "%d", &stud[i][j]) ;  }  }  for ( i = 0 ; i < 3 ; i++ )  {  for ( j= 0 ; j < 2 ; j++ )  {  printf ( "%d ", stud[i][j] ) ;  }  printf ( "\n");}  return 0;  } |

**Passing an entire array to function**

We will explore it by two ways :

1. One in which we passed individual elements of an array to a function
2. Another in which we pass an entire array to a function.(pointer)

**Example**

|  |
| --- |
| **#include <stdio.h>**  **int calculateSum(int age[],int n);**  **int main() {**  **int result, n,age[10];**    **printf("Enter a number of array ");**  **scanf("%d",&n);**    **for (int i = 0; i < n; i++) {**  **printf("enter a element of array");**  **scanf("%d",&age[i]);**  **}**  **// age array is passed to calculateSum()**  **result = calculateSum(age,n);**  **printf("Result = %d", result);**  **return 0;**  **}**  **int calculateSum(int age[],int n) {**  **int sum = 0;**  **for (int i = 0; i < n; ++i) {**  **sum += age[i];**  **}**    **return sum;**  **}** |

**String**

In C programming, a string is a sequence of characters terminated with a null character \0. For example:

char c[] = "c string";

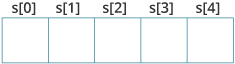
When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.

Memory diagram of strings in C programming

**How to declare a string?**

Here's how you can declare strings:

char s[5];



Here, we have declared a string of 5 characters.

**How to initialize strings?**

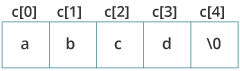
You can initialize strings in a number of ways.

char c[] = "abcd";

char c[50] = "abcd";

char c[] = {'a', 'b', 'c', 'd', '\0'};

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



Let's take another example:

char c[5] = "abcde";

Here, we are trying to assign 6 characters (the last character is '\0') to a char array having 5 characters. This is bad and you should never do this.

Read String from the user

You can use the scanf() function to read a string.

The scanf() function reads the sequence of characters until it encounters [whitespace](https://stackoverflow.com/questions/30033582/what-is-the-symbol-for-whitespace-in-c) (space, newline, tab etc.).

**Example 1: scanf() to read a string**

**#include <stdio.h>**

**int main()**

**{**

**int i;**

**char name[20];**

**printf("Enter name: ");**

**for(i=0;i<9;i++)**

**{**

**scanf("%c",&name[i]);**

**}**

**for(i=0;i<9;i++)**

**{**

**printf("%c",name[i]);**

**}**

**return 0;**

**}**

**Example 2**

#include <stdio.h>

int main()

{

char name[20];

printf("Enter name: ");

scanf("%s", name);

printf("Your name is %s.", name);

return 0;

}

**Output**

Enter name: Dennis Ritchie

Your name is Dennis.

Even though Dennis Ritchie was entered in the above program, only "Ritchie" was stored in the name string. It's because there was a space after Dennis.

|  |  |
| --- | --- |
| **Sr.No.** | **Function & Purpose** |
| 1 | **strcpy(s1, s2);**  Copies string s2 into string s1. |
| 2 | **strcat(s1, s2);**  Concatenates string s2 onto the end of string s1. |
| 3 | **strlen(s1);**  Returns the length of string s1. |
| 4 | **strcmp(s1, s2);**  Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2. |
| 6 | **Strlwr()**  Converts a string to lowercase. |
| 7 | **Strupr()**  Converts a string to uppercase. |
| 8 | **Strdup()**  Duplicates a string. |

**Example**

The following example uses some of the above-mentioned functions −

#include <stdio.h>

#include <string.h>

int main () {

int i, j, k ;

char str1[12] = "Hello";

char str2[12] = "World";

char source[ ] = "Folks!" ;

char target[30] = "Hello" ;

char string1[ ] = "Jerry" ;

char string2[ ] = "Ferry" ;

char str3[12];

int len ;

/\* copy str1 into str3 \*/

strcpy(str3, str1);

printf("strcpy( str3, str1) : %s\n", str3 );

/\* concatenates str1 and str2 \*/

strcat( str1, str2);

printf("strcat( str1, str2): %s\n", str1 );

/\* total lenghth of str1 after concatenation \*/

len = strlen(str1);

printf("strlen(str1) : %d\n", len );

i = strcmp ( string1, "Jerry" ) ;

k = strcmp ( string1, "Jerryboy" ) ;

printf ( "\n%d %d %d", i, k ) ;

return 0;}

When the above code is compiled and executed, it produces the following result −

strcpy( str3, str1) : Hello

strcat( str1, str2): HelloWorld

strlen(str1) : 10

# **STRUCTURE IN C**

Arrays allow to define type of variables that can hold several data items of the same kind. Similarly structure is another user defined data type available in C that allows to combine data items of different kinds.

Suppose you want to keep track of your books in a library. You might want to track the following attributes about each book −

* Title
* Author
* Subject
* Book ID

# **Defining a Structure**

To define a structure, you must use the **struct** statement. The struct statement defines a new data type, with more than one data member. The format of the struct statement is as follows .

struct [structure tag] {

member definition;

member definition;

...

member definition;

} [one or more structure variables];

The **structure tag** is optional and each member definition is a normal variable definition, such as int i; or float f; or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional. Here is the way you would declare the Book structure .

struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id;

} book;

# **Accessing Structure Members**

To access any member of a structure, we use the **member access operator (.)**. The member access operator is coded as a period between the structure variable name and the structure member that we wish to access. You would use the keyword **struct** to define variables of structure type. The following example shows how to use a structure in a program.

**Example**

#include <stdio.h>

#include <string.h>

struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id;

};

int main( ) {

struct Books Book1;

struct Books Book2; /\* Declare Book2 of type Book \*/

/\*accessing book 1 information \*/

printf("Enter BOOK TITLE");

scanf("%s",& Book1.title);

printf("Enter BOOK Author");

scanf("%s",& Book1.author);

printf("Enter BOOK Subject");

scanf("%s",&Book1.subject);

printf("Enter BOOK ID");

Book1.book\_id = 6495700;

/\* print Book1 info \*/

printf( "Book 1 title : %s\n",Book1.title);

printf( "Book 1 author : %s\n",Book1.author);

printf( "Book 1 subject : %s\n",Book1.subject);

printf( "Book 1 book\_id : %d\n",Book1.book\_id);

return 0;

}

# **Structures as Function Arguments**

You can pass a structure as a function argument in the same way as you pass any other variable or pointer.

#include <stdio.h>

#include <string.h>

struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id;

};

/\* function declaration \*/

void printBook( struct Books book );

int main( ) {

struct Books Book1; /\* Declare Book1 of type Book \*/

struct Books Book2; /\* Declare Book2 of type Book \*/

/\* book 1 specification \*/

strcpy( Book1.title, "C Programming");

strcpy( Book1.author, "Nuha Ali");

strcpy( Book1.subject, "C Programming Tutorial");

Book1.book\_id = 6495407;

/\* print Book1 info \*/

printBook( Book1 );

return 0;

}

void printBook( struct Books book ) {

printf( "Book title : %s\n", book.title);

printf( "Book author : %s\n", book.author);

printf( "Book subject : %s\n", book.subject);

printf( "Book book\_id : %d\n", book.book\_id);

}

**Lab Task**

**Question # 01:**

Write a program to obtain transpose of a 4 x 4 matrix. The transpose of a matrix is obtained by exchanging the elements of each row with the elements of the corresponding column.

**Question # 02:**

Write a c program in C for addition of two matrices of same size.

**Question # 03:**

Write a c program in c in which you take a 2 dimensional array and calculate the sum of each row and each column.

**Question # 04:**

Write a program to delete all vowels from a sentence. Assume that the sentence is not more than 60 characters long.

**Question # 05:**

Write a program to count the length of string without built in function.

**Question # 06:**

Write a program that extracts part of the given string from the specified position. For example if the string is “Working with string is fun”, then it from position 4, 4 character are to be extracted then program should return string as a “king”.

**Question # 07:**

Define a struct type with the name Length that represents a length in yards, feet, and inches. Define an add() function that will add two Length arguments and return the sum as type Length. Define a second function, show(), that will display the value of its Length argument. Write a program that will use the Length type and the add() and show() functions to sum an arbitrary number of lengths in yards, feet, and