United International University

Dept. of Electrical and Electronic Engineering (EEE)

Course No. : **EEE 122**
Course Title: **Structured Programming Laboratory**

Lab Sheet 6
Arrays in C

Outcomes

After finishing this lab students should be able to ...

1. define an array, initialize an array and refer to individual elements of an array
2. pass arrays to functions.
3. use arrays to store, sort and search lists and tables of values.
4. use arrays to store, sort and search lists and tables of values.

Contents

1 C - Arrays

C **Array** is a collection of data that holds fixed number of values of same type. For example: if you want to store marks of 100 students, you can create an array for it.

```
float marks[100];// floating array
int a[10];       // integer array
char b[10];      // character array   i.e. string
```

The size and type of arrays cannot be changed after its declaration.

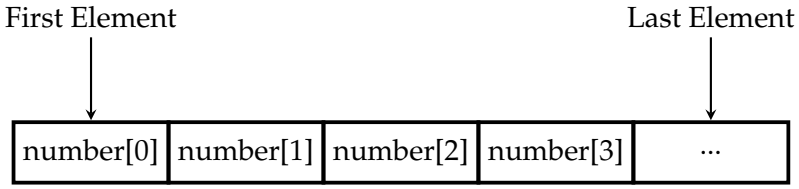
Few key notes:

- Array might be belonging to any of the data types.
- Array size must be a constant value.
- Always, Contiguous (adjacent) memory locations are used to store array elements in memory.

– It is a best practice to initialize an array to zero or null while declaring, if we dont assign any values to array.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as `int numbers[100]` and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



1.1 Declaring Arrays

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 —

```
data_type array_name[array_size];
```

This is called a **single-dimensional** array. The **array_size** must be an integer constant greater than zero and **data_type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement —

```
double balance[10];
```

Here balance is a variable array which is sufficient to hold up to 10 double numbers.

1.2 Initializing Arrays

You can initialize an array in C either one by one or using a single statement as follows —

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

The number of values between braces cannot be larger than the number of elements that we declare for the array between square brackets [ ].

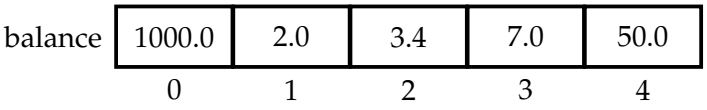
If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write —

```
double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

You will create exactly the same array as you did in the previous example.
Following is an example to assign a single element of the array—

```
balance[4] = 50.0;
```

The above statement assigns the 5^{th} element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. Shown below is the pictorial representation of the array we discussed above —



Few key notes:

- Arrays have 0 as the first index not 1. In this example, **`balance[0]`**
- If the size of an array is n, to access the last element, (n-1) index is used. In this example, **`balance[4]`**
- Suppose the starting address of **`balance[0]`** is 2120d. Then, the next address, **`balance[1]`** will be 2124d, address of **`balance[2]`** will be 2128d and so on. It's because the size of a **`int`** is 4 bytes.

1.3 Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example —

```
double salary = balance[3];
```

The above statement will take the 4^{th} element from the array and assign the value to salary variable. The following example Shows how to use all the three above mentioned concepts viz. declaration, assignment, and accessing arrays —

```
#include<stdio.h>

int main(void){

  int n[10]; //n is an array of 10 integers
  int i,j;

  //initialize elements of array n to 0
  for ( i = 0; i < 10; i++ ) {
    n[i] = i + 100; //set element at location i to i + 100
  }

  //output each array element's value/
  for (j = 0; j < 10; j++ ) {
    printf("Element[%d] = %d\n", j, n[j] );
  }
  return 0;
}
```

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

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

Example:

Write a C program to find the average of n numbers using arrays

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

#include <stdio.h>
int main(void){
  int i, n, sum = 0, average;
  printf("Enter n: ");
  scanf("%d", &n);
  int marks[n];
  for(i=0; i<n; ++i){
    printf("Enter number%d: ",i+1);
    scanf("%d", &marks[i]);
    sum += marks[i];
  }
  average = sum/n;

  printf("Average = %d", average);

  return 0;
}
```

In C programming, you can create an array of arrays known as multidimensional array. For example–

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

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

	Col-1	Col-2	Col-3	Col-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, you can declare a three-dimensional (3d) array. For example—

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

Here,The array `y` can hold 24 elements.
You can think this example as: Each 2 elements have 4 elements, which makes 8 elements and each 8 elements can have 3 elements. Hence, the total number of elements is 24.

2 Programming Examples

Example: 1

Description: Write a C program to input values into an array and display them

Source Code

```
#include<stdio.h>
int main(void){
  int arr[50],i;
  for(i=0;i<5;i++){
    printf ("Enter element for arr[%d]:",i);
    scanf("%d",&arr[i]) ;
  }
  printf ("The array elements are:\n");
  for(i=0;i<5;i++)
  printf("%d\t",arr[i]);
  printf ("\n");
  return 0;
}
```

Output

```
Enter element for arr[0]:8
Enter element for arr[1]:11
Enter element for arr[2]:26
Enter element for arr[3]:17
Enter element for arr[4]:36
The array elements are:
8 11  26  17  36
```

Example: 2

Description: Write a program in C using array which finds largest and smallest elements in a array.

Source Code

```
#include <stdio.h>
int main(void){
  int i,n;
  float a[50],large,small;
  printf("\nHow many elements:");
  scanf("%d",&n);
  printf("\nEnter elements:\n");
  for(i=0;i<n;i++)
    scanf("%f",&a[i]);
  large=a[0];
  small=a[0];
  for(i=1;i<n;i++){
    if(a[i]>large)
    large=a[i];
    else if(a[i]<small)
    small=a[i];
  }
  printf("\nLargest element is:%.2f",large);
  printf("\nSmallest element is:%.2f",small);
  return 0;
}
```

Output

```
How many elements:5

Enter elements:
23 12 87 29 34

Largest element is:87.00
Smallest element is:12.00
```

Example: 3

Description: Write a C program to search for an item in the array.

Source Code

```
#include <stdio.h>
#define SIZE 10
int main(void){
  int i,arr[SIZE]={23,12,56,98,76,14,65,11,19,45};
  int item;
  printf ("Enter the item to be searched : ") ;
  scanf("%d",&item);
  for(i=0;i<SIZE;i++){
    if(item==arr[i]){
      printf ("%d found at position %d\n",
                          item, i+1);
      break;
    }
  }
  if(i==SIZE)
  printf("Item %d not found in array\n" , item) ;
  return 0;
}
```

Output

```
Enter the item to be
searched : 14
14 found at position 6
```

Example: 3

Description: Write a C program to search for an item in the array.

Source Code

```
  #include <stdio.h>
  #define SIZE 10
  int main(void){
  int i,arr[SIZE]={23,12,56,98,76,14,65,11,19,45};
  int item;
  printf ("Enter the item to be searched : ") ;
  scanf("%d",&item);
  for(i=0;i<SIZE;i++){
  if(item==arr[i]){
  printf ("%d found at position %d\n",
  item, i+1);
  break;
  }
  }
  if(i==SIZE)
  printf("Item %d not found in array\n" , item) ;
  return 0;
  }
```

Output

```
   Enter the item to be
   searched : 14
   14 found at position 6
```

Example: 4

Description:Write a C program to input and display a matrix

Source Code

```
#include<stdio.h>
int fact(int n);
int main(void){
  int n,result;
  printf("\n Enter the number:");
  scanf("%d",&n);
  result=fact(n);
  printf("\n The factorial is=%d",result);
  return 0;
}
int fact(int m){
  if(m == 0) return 1 ;
  return m*fact(m-1);
}
```

Output

```
Enter the elements of matrix
 (3x4) row wise ;
12 23 45 21
87 56 22 37
20 10 30 40
The matrix that you have
entered is ;
12   23   45   21
87   56   22   37
20   10   30   40
```

Example: 5

Description: Write a C program to sort some numbers using bubble sort algorithm

Source Code

```
#include <stdio.h>
#define SIZE 10
int main(void){
  int i,j;
  int arr[SIZE]={23,12,56,98,76,14,65,11,19,45};

  printf("\n Before sorting our array is :");
  for(i = 0; i < SIZE; i++){
    printf( " %d",arr[i]);
  }

  for(i = 0; i < SIZE - 1; i++){
    for(j = 0; j < SIZE - i -1; j++){
      if(arr[j]>arr[j+1]){
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
      }
    }
  }

  printf("\n After sorting our array is  :");
  for(i = 0; i < SIZE; i++){
    printf( " %d",arr[i]);
  }
return 0;
}
```

Output

```
Before sorting our array is :
23 12 56 98 76 14 65 11 19 45

After sorting our array is :
11 12 14 19 23 45 56 65 76 98
```

3 Practice session

Sl	Source Code

Practice 1	`#include <stdio.h>`
	`int main(void){`
	`  int arr[4]={2,4,8,16},i=4,j;`
	`  while(i){`
	`  j=arr[i]+i;`
	`  i--;`
	`  }`
	`  printf("j=%d\n",j);`
	`  return 0;`
	`}`

Practice 2	`#include <stdio.h>`
	`int main(void){`
	`  int i=0,sum=0,arr[6]={4,2,6,0,5,10};`
	`  while(arr[i]){`
	`  sum=sum+arr[i] ;`
	`  i++;`
	`  }`
	`  printf("sum=%d\n",sum);`
	`  return 0;`
	`}`

Practice 3	`#include <stdio.h>`
	`int main(void){`
	`  int x [10] , y[3][4] , z[2][3][5] ;`
	`  printf("%u \t %u \t %u\n",sizeof(x),sizeof(y),sizeof(z));`
	`  return 0;`
	`}`

Practice 4	`#include <stdio.h>`
	`int main(void){`
	`  int i,j,arr[3][4]={{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};`
	`  for(i=0;i<4;i++){`
	`  for(j=0;j<3;j++)`
	`  printf("%3d" , arr [j ] [i] ) ;`
	`  printf("\n");`
	`  }`
	`  return 0;`
	`}`

4 Lab Assignments

1. Write a program to find out the determinant of a matrix.
2. Write a program to print the Pascal triangle using a 2-D array.
   We'll calculate and store all the elements of Pascal triangle in a 2-D array. Form the figure we can observe that-
   (a) All the elements of Column 0 are 1.
   (b) All the elements for which Row and Column are same, are 1.
   (c) Any other element can be obtained by adding two elements of previous row as
   $$a[i][j] = a[i-1][j-1] + a[i-1][j];$$
   where i and j represent row and column number.
3. Write a program to print the elements of a matrix spirally. For example, if the matrix is

```
2 5 8
1 3 7
6 2 9
```

The output should be-

```
2 5 8 7 9 2 6 1 3
```

Acknowledgment

First OBE version, prepared by:
B.K.M. Mizanur Rahman,
Assistant Professor,
Department of EEE, UIU

Second Update , prepared by:
Nazmul Alam,
Part Time Faculty,
Department of EEE, UIU