Structured Programming Language Sessional

CSE 1104

Sessional - 06

Array in C

One Dimensional & Multidimensional Arrays

Department of Computer Science and Engineering

Pabna University of Science and Technology

1 Introduction

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type. 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.

C programming language provides the following types of Arrays:

- One-dimensional arrays
- Multidimensional arrays

1.1 One-dimensional arrays

To declare an array in C, it is needed to specify the type of the elements and the number of elements required by an array as follows:

type arrayName [arraySize];

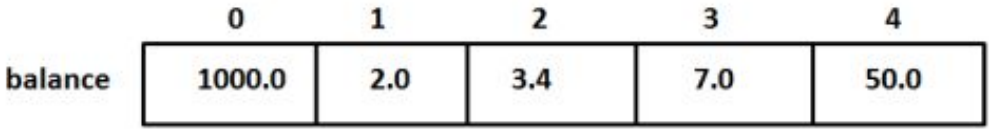
This is called a one-dimensional array. The arraySize must be an integer constant greater than zero and type can be any valid C data type. For example, to declare a 10-element array called balance of type double, the following statement can be used:

double balance[10];

Here balance is a variable array which is sufficient to hold up to 10 double numbers.
We 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 []. The above array can be pictorially represented as follows:



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[9];

The above statement will take the 10^{th} element from the array and assign the value to salary variable.

Example 1: Write a C program to find the largest element of an array.

Algorithm 1 Steps in pseudo code:

```
Step 1: Start
Step 2: Declare array a[n] and variable i, large
Step 3: Read n from User and read all the elements of a[n]
Step 4: Initialize Variable i=1 and large=a[0]
Step 5: Repeat Until i<=n-1
    5.1 if(a[i]>large), set large=a[i]
    increment i=i+1
Step 6: Print "The largest element is": large
step 7: end
```

Code:

```c
#include <stdio.h>
int main()
{
int i, n, a[100],large;
printf("Enter total number of elements(1 to 100): ");
scanf("%d", &n);
// Stores number entered by the user
for(i = 0; i < n; ++i)
{ printf("Enter Number %d: ", i+1);
scanf("%d", &a[i]); }
// Loop to store largest number to arr[0]
large=a[0];
for(i = 1; i < n; ++i)
{
// Change < to > if you want to find the smallest element
if(a[i]>large)
large = a[i];
}
printf("Largest Element = %d", large);
return 0;
}
```

Output:

```
Enter total number of elements(1 to 100): 5
Enter Number 1: 4
Enter Number 2: 3
Enter Number 3: 7
Enter Number 4: 1
Enter Number 5: 2
Largest Element = 7
```

Example 2: Write a C program to find the average of n numbers using arrays.

Code:

```c
#include <stdio.h>
int main()
{
      int a[100], i, n, sum = 0;
      float average;

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

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

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

      average = (float) sum/n;
      printf("Average = %.2f", average);

      return 0;
}
```

Output:

```
Enter number of elements: 5
Enter number1: 4
Enter number2: 1
Enter number3: 3
Enter number4: 2
Enter number5: 5
Average = 3.00
```

1.2 Multidimensional arrays

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

type arrayName [x][y];

Where type can be any valid C data type and arrayName will be a valid C identifier. 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 follows:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

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'.

Multidimensional 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 */
};
```

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 val = a[2][3];

The above statement will take the 4th element from the 3rd row of the array.
Example 1: Write a C program to find the largest element of an array.

Algorithm 2 Steps in pseudo code:

1: Step 1: Start
2: Step 2: Declare array a[100][100],b[100][100],sum[100][100] and variable i,j,r,c
3: Step 3: Read r and c, rows and column respectively
4: Step 4: Read Matrix a and b
5: Step 5: add each corresponding elements of a & b and put it to sum
6: Step 6: Print sum
7: step 7: end

Code:

```
#include <stdio.h>
int main() {
  int r, c, a[100][100], b[100][100], sum[100][100], i, j;
  printf("Enter the number of rows (between 1 and 100): ");
  scanf("%d", &r);
  printf("Enter the number of columns (between 1 and 100): ");
  scanf("%d", &c);

  printf("\nEnter elements of 1st matrix:\n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("Enter element a%d%d: ", i + 1, j + 1);
      scanf("%d", &a[i][j]);
    }

  printf("Enter elements of 2nd matrix:\n");
```

```c
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("Enter element b%d%d: ", i + 1, j + 1);
      scanf("%d", &b[i][j]);
    }

  // adding two matrices
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      sum[i][j] = a[i][j] + b[i][j];
    }

  // printing the result
  printf("\nSum of two matrices: \n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("%d   ", sum[i][j]);
      if (j == c - 1) {
        printf("\n\n");
      }
    }

  return 0;
}
```

Output:

```
Enter the number of rows (between 1 and 100): 2
Enter the number of columns (between 1 and 100): 3

Enter elements of 1st matrix:
Enter element a11: 2
Enter element a12: 3
Enter element a13: 4
Enter element a21: 5
Enter element a22: 2
Enter element a23: 3
Enter elements of 2nd matrix:
Enter element b11: -4
Enter element b12: 5
Enter element b13: 3
Enter element b21: 5
Enter element b22: 6
Enter element b23: 3

Sum of two matrices:
-2   8   7

10   8   6
```

2 Discussion & Conclusion

Based on the focused objective(s) to understand about the array operations, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

3 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a program in C to read n number of values in an array and display it in reverse order.
2. Write a program in C to count a total number of duplicate elements in an array.

3. Write a program in C to find the maximum and minimum element in an array.
4. Write a program in C to separate odd and even integers in separate arrays.

4 Lab Exercise (Submit as a report)

- Write a C Program to Calculate mean, median and Standard Deviation.
- Write a C program to convert Decimal to Binary number system.
- Write a C program to count frequency of each element in an array.
- Write a C Program to Find Transpose of a Matrix.

5 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.