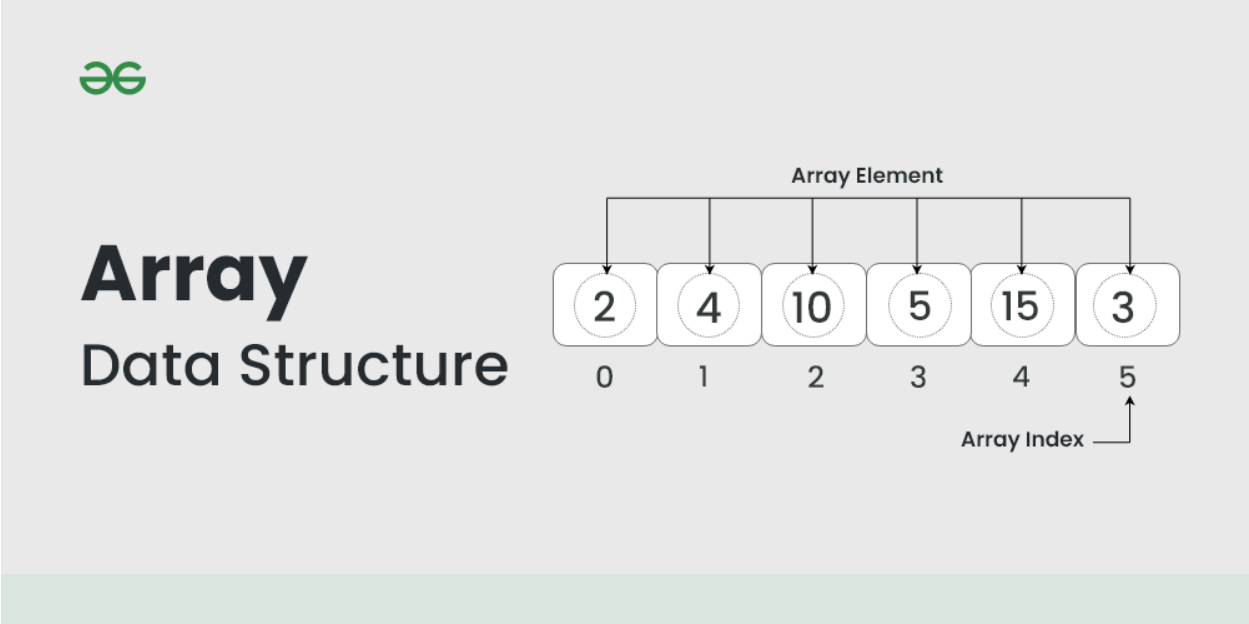
Cyber Security Engineering Department Lectures
Prepared by Assist. Prof. Imad Matti

هياكل البيانات

Array Data Structure

Chapter 1



Prepared by Assist . Prof .
Imad Matti
AL-mamoon University College / Cyber Security Engineering Department

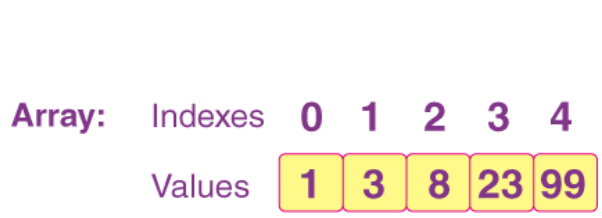
2024

What is Array?

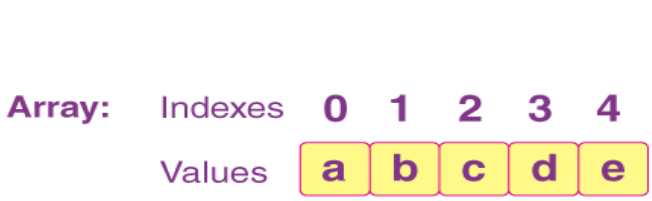
Array is a data structure that is used to store variables that are of similar data types at contiguous locations.

An array is a group of similar elements or data items of the same type collected at contiguous memory locations.

Array for Integral value:

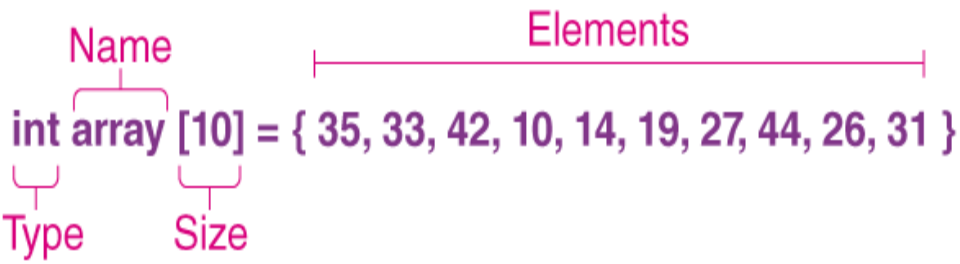


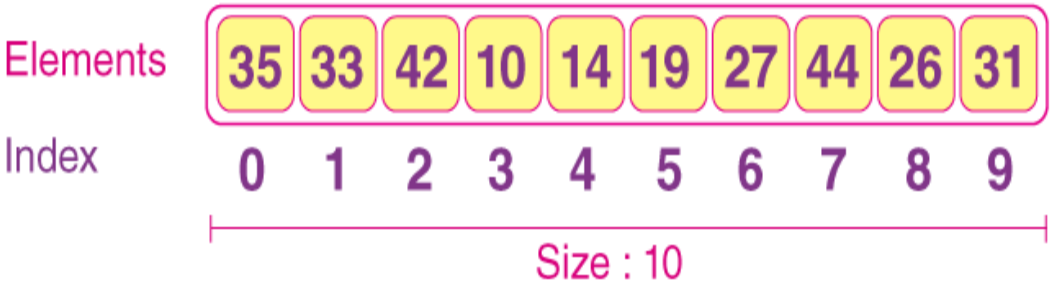
Array for Character value:



Representation of an Array:

Arrays can be represented in several ways, depending on the different languages. To make you understand, we can take one example of the C language. The picture below shows the representation of the array.

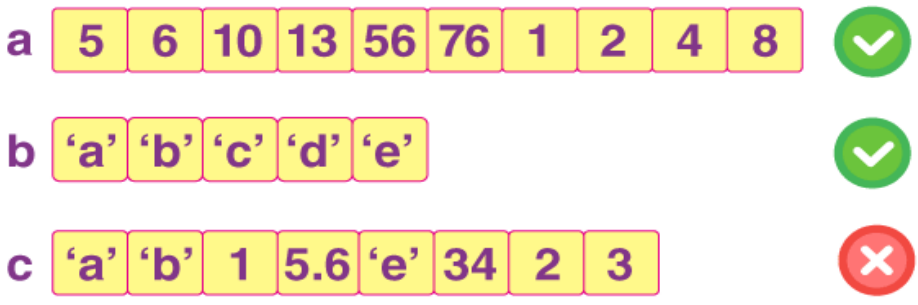




Arrays always store the same type of values. In the above example:

- int is a type of data value.
- Data items stored in an array are known as elements.

Important: Array can store only the same type of data items.



Declaration Syntax of Array:

Example 1: For integral value

int A[10];

Here 10 means, this array A can have 10 integer elements.

2	5	8	44	21	11	7	9	3	1

Example 2: For character value

char B[10];

This array B can have 10 character elements.

f	d	a	b	n	j	l	s	e	Y

Declaration of Array in python

Arrays can be declared in various ways in different languages.

```
# In Python, all types of lists are created same way
      arr = []
```

Initialization of Array in python

Arrays can be initialized in different ways in different languages.

```
# This list will store integer type elements
arr = [1, 2, 3, 4, 5]
```

# This list will store character type elements (strings in Python)
arr = ['a', 'b', 'c', 'd', 'e']

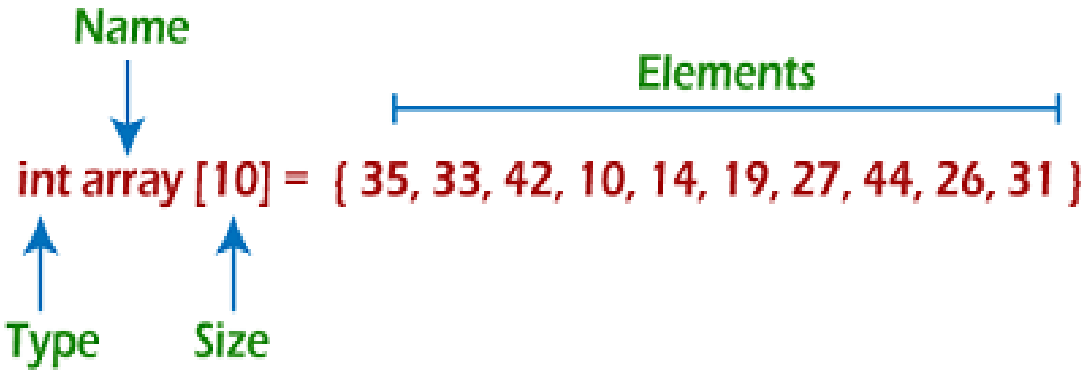
# This list will store float type elements
arr = [1.4, 2.0, 24.0, 5.0, 0.0] # All float values

Python:

```
my_array = [7, 12, 9, 4, 11]

print( my_array[0] )
```

Representation of an array in C language

We can represent an array in various ways in different programming languages. As an illustration, let's see the declaration of array in C language -

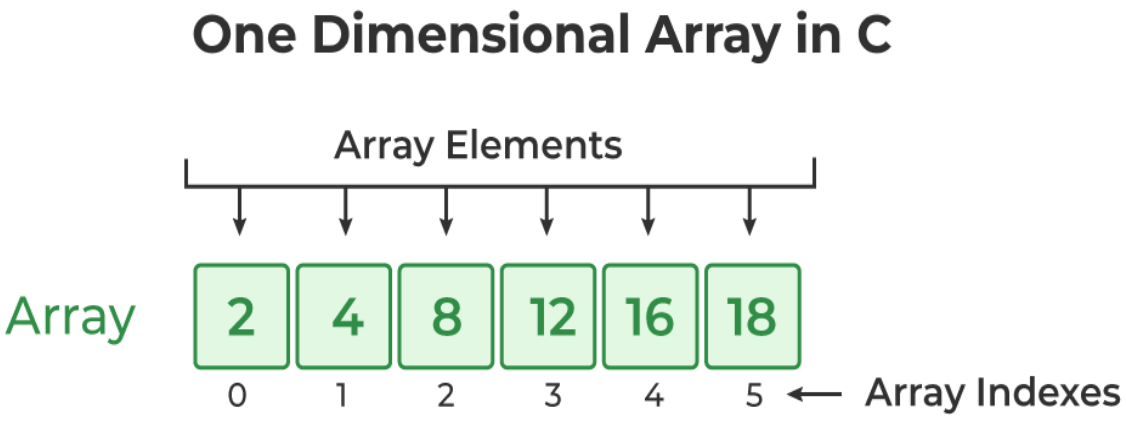


There are mainly three types of the array:

- One Dimensional (1D) Array
-
- Two Dimension (2D) Array
-
- Multidimensional Array

One-Dimensional Arrays in C

A one-dimensional array can be viewed as a linear sequence of elements. We can only increase or decrease its size in a single direction.



How Do You Initialize an Array?

You can initialize an array in four different ways:

- Method 1:

int a[6] = {2, 3, 5, 7, 11, 13};

- Method 2:

```
int arr[]= {2, 3, 5, 7, 11};
```

- Method 3:

```
int n;

scanf(“%d”,&n);

int arr[n];

for(int i=0;i<5;i++)

{

scanf(“%d”,&arr[i]);

}
```

- Method 4:

```
int arr[5];

arr[0]=1;

arr[1]=2;

arr[2]=3;

arr[3]=4;

arr[4]=5;
```

How Can You Access Elements of Arrays in Data Structures?

You can access elements with the help of the index at which you stored them. Let's discuss it with a code:

```
#include<stdio.h>

int main(){

int a[5] = {2, 3, 5, 7, 11};

printf(“%d\n”,a[0]); // we are accessing

printf(“%d\n”,a[1]);

printf(“%d\n”,a[2]);

printf(“%d\n”,a[3]);

printf(“%d”,a[4]);

return 0;

}
```

Array Examples in Python

In Python, arrays are represented using **lists**, which are dynamic arrays that can change in size. However, we can still treat lists similarly to arrays for basic operations. If you need fixed-size arrays, you can use the array module, but lists are more commonly used in Python.

Example 1: List (Array) in Python

```
# Declare a list of integers
arr = [10, 20, 30, 40, 50]

# Access elements using indices
print("First element:", arr[0])
print("Second element:", arr[1])
print("Third element:", arr[2])

# Modify an element
arr[4] = 100
print("Modified fifth element:", arr[4])
```

Explanation:

- We declare a list arr with initial values [10, 20, 30, 40, 50].
- Access elements using their indices in the same way as in C.
- Modify the fifth element by assigning a new value (arr[4] = 100).

Output:

```
First element: 10
Second element: 20
Third element: 30
Modified fifth element: 100
```

Traversal operation

This operation is performed to traverse through the array elements. It prints all array elements one after another. We can understand it with the below program -

```
1. #include <stdio.h>
2. void main() {
3.    int Arr[5] = {18, 30, 15, 70, 12};
```

```
4. int i;
5.    printf("Elements of the array are:\n");
6.    for(i = 0; i<5; i++) {
7.       printf("Arr[%d] = %d,  ", i, Arr[i]);
8.    }
9. }
```

Output

```
Elements of the array are:
Arr[0] = 18,  Arr[1] = 30,  Arr[2] = 15,  Arr[3] = 70,  Arr[4] = 12,
```

Example of One Dimensional Array in C

```c
#include <stdio.h>

int main(){
    int arr[5] = { 1, 2, 4, 8, 16 };

    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    arr[3] = 9721;

    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Output

```
1 2 4 8 16

1 2 4 9721 16
```

Array of Characters (Strings)

The strings are essentially an array of character that is terminated by a NULL character ('\0').

Syntax

```
char name[ ] = "this is an example string"
```

Here, we don't need to define the size of the array. It will automatically deduce it from the assigned string.

Example

```
// C++ Program to illlustrate the use of string
#include <stdio.h>

int main()
{

    // defining array
    char str[] = "This is cyber security engineering department";

    printf("%s", str);

    return 0;
}
```

Output

This is cyber security engineering department

Two Dimensional Array:

- It is a list of lists of the variable of the same data type.
- Its dimension can be increased from 2 to 3 and 4 so on.
- They all are referred to as a multi-dimension array.
- The most common multidimensional array is a 2D array.

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

Applications of Arrays:

- 2D Arrays are used to implement **matrices**.
- Arrays can be used to implement various data structures like a heap, stack, queue,

The first dimension represents the number of rows **[2]**, while the second dimension represents the number of columns **[3]**. The values are placed in row-order, and can be visualized like this:

	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

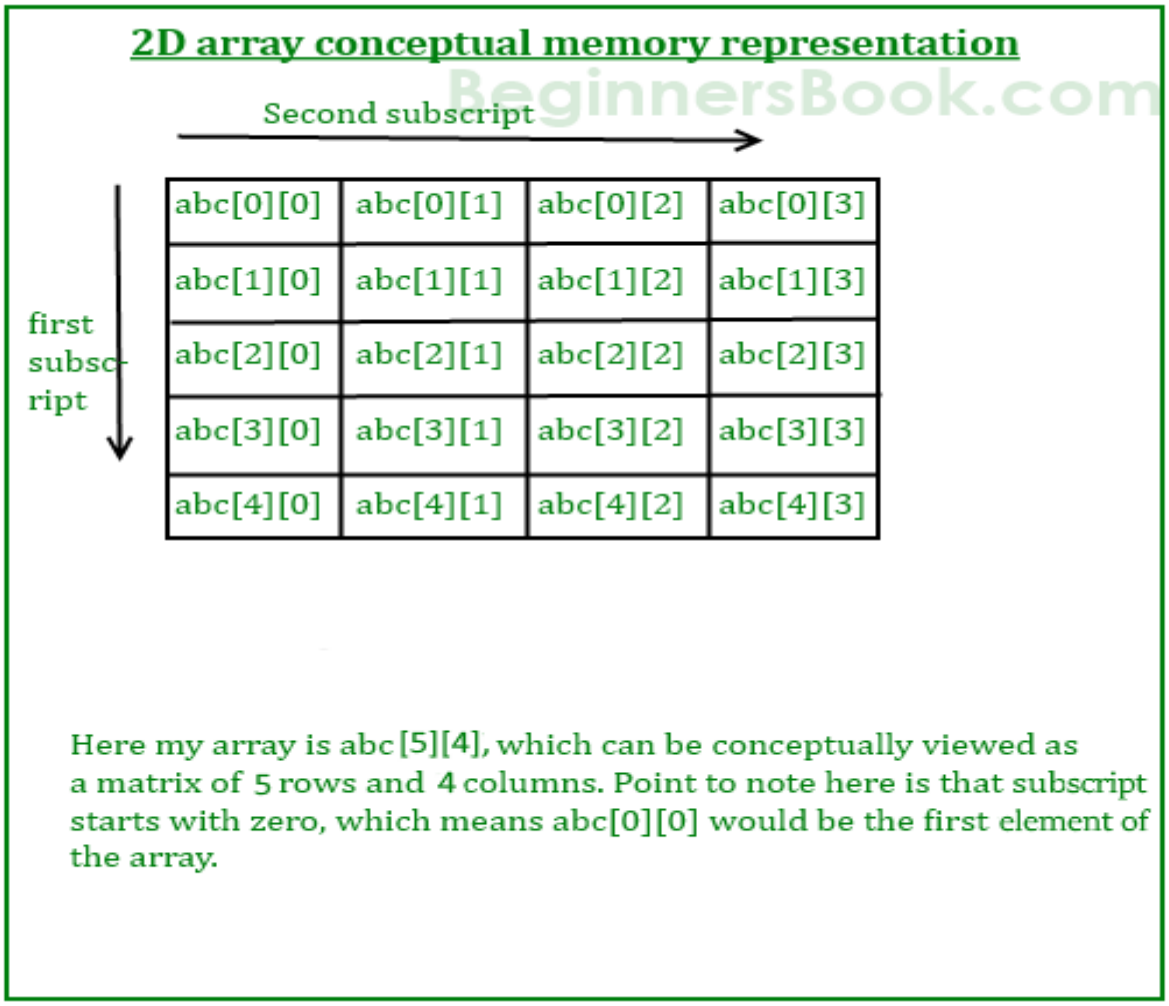
Example

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

int i, j;
for (i = 0; i < 2; i++) {
  for (j = 0; j < 3; j++) {
    printf("%d\n", matrix[i][j]);
  }
}
```

The following program demonstrates how to store the elements entered by user in a 2d array

```
#include<stdio.h>
int main(){
  /* 2D array declaration*/
  int abc[5][4];
  /*Counter variables for the loop*/
  int i, j;
  for(i=0; i<5; i++) {
    for(j=0;j<4;j++) {
      printf("Enter value for abc[%d][%d]:", i, j);
      scanf("%d", &abc[i][j]);
```

```
    }
  }
  return 0;
}
```



The following program demonstrates how to store the elements entered by user in a 2d array and how to display the elements of a two dimensional array.

```
#include<stdio.h>
int main(){
   /* 2D array declaration*/
   int disp[2][3];
   /*Counter variables for the loop*/
   int i, j;
   for(i=0; i<2; i++) {
      for(j=0;j<3;j++) {
         printf("Enter value for disp[%d][%d]:", i, j);
         scanf("%d", &disp[i][j]);
      }
   }
   //Displaying array elements
   printf("Two Dimensional array elements:\n");
   for(i=0; i<2; i++) {
      for(j=0;j<3;j++) {
         printf("%d ", disp[i][j]);
         if(j==2){
            printf("\n");
         }
      }
   }
   return 0;
}
```

Output:

```
Enter value for disp[0][0]:1
Enter value for disp[0][1]:2
Enter value for disp[0][2]:3
Enter value for disp[1][0]:4
Enter value for disp[1][1]:5
Enter value for disp[1][2]:6
Two Dimensional array elements:
1 2 3
4 5 6
```

Example of 2D Array:

The below example demonstrates the row-by-row traversal of a 2D array.

```
// C Program to illustrate the 2D array
#include <stdio.h>

int main() {

    // Create and initialize an array with 3 rows
      // and 2 columns
    int arr[3][2] = { { 0, 1 }, { 2, 3 }, { 4, 5 } };

    // Print each array element's value
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
            printf("arr[%d][%d]: %d     ", i, j, arr[i][j]);
        }
          printf("\n");
    }
    return 0;
}
```

Output

```
arr[0][0]: 0    arr[0][1]: 1
arr[1][0]: 2    arr[1][1]: 3
arr[2][0]: 4    arr[2][1]: 5
```

C Program to read and print a RxC Matrix, R and C must be input by User

```c
#include <stdio.h>

#define MAXROW      10
#define MAXCOL      10

int main()
{
    int matrix[MAXROW][MAXCOL];
    int i,j,r,c;

    printf("Enter number of Rows :");
    scanf("%d",&r);
    printf("Enter number of Cols :");
    scanf("%d",&c);

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

    printf("\nMatrix is :\n");
    for(i=0;i< r;i++)
    {
        for(j=0;j< c;j++)
        {
            printf("%d\t",matrix[i][j]);
        }
        printf("\n");   /*new line after row elements*/
```

```
    }
    return 0;
}
```

Output

```
Enter number of Rows :3
Enter number of Cols :3

Enter matrix elements :
Enter element [1,1] : 1
Enter element [1,2] : 1
Enter element [1,3] : 1
Enter element [2,1] : 2
Enter element [2,2] : 2
Enter element [2,3] : 2
Enter element [3,1] : 3
Enter element [3,2] : 3
Enter element [3,3] : 3

Matrix is :
1       1       1
2       2       2
3       3       3
```

C program to find sum of all elements of each row of a matrix

```c
#include <stdio.h>

#define MAXROW 10
#define MAXCOL 10

int main() {
  int matrix[MAXROW][MAXCOL];
  int i, j, r, c;
  int sum, product;

  printf("Enter number of Rows :");
  scanf("%d", &r);
  printf("Enter number of Cols :");
  scanf("%d", &c);

  printf("\nEnter matrix elements :\n");
  for (i = 0; i < r; i++) {
    for (j = 0; j < c; j++) {
      printf("Enter element [%d,%d] : ", i + 1, j + 1);
      scanf("%d", &matrix[i][j]);
    }
  }
  printf("\n");
  /*sum of all rows*/
  for (i = 0; i < r; i++) {
    sum = 0; /*initializing sum*/
    for (j = 0; j < c; j++) {
      printf("%d\t", matrix[i][j]); /*print elements*/
      sum += matrix[i][j];
    }
    printf("\tSUM : %d", sum);
    printf("\n"); /*after each row print new line*/
  }
}
```

Output

```
Enter number of Rows :3
Enter number of Cols :3

Enter matrix elements :
Enter element [1,1] : 1
Enter element [1,2] : 2
Enter element [1,3] : 3
Enter element [2,1] : 4
Enter element [2,2] : 5
Enter element [2,3] : 6
Enter element [3,1] : 7
Enter element [3,2] : 8
Enter element [3,3] : 9

1       2       3               SUM : 6
4       5       6               SUM : 15
7       8       9               SUM : 24
```

Chapter 1 Lab. works

واجبات مختبرية

Q1: what is the output of the following piece of code?

```
int myNumbers[] = {10, 25, 50, 75, 100};
int length = sizeof(myNumbers) / sizeof(myNumbers[0]);

printf("%d", length);
```

Q2: what is the output of the following piece of code?

```
// An array storing different ages
int ages[] = {20, 22, 18, 35, 48, 26, 87, 70};

float avg, sum = 0;
int i;

// Get the length of the array
int length = sizeof(ages) / sizeof(ages[0]);

// Loop through the elements of the array
for (i = 0; i < length; i++) {
  sum += ages[i];
}

// Calculate the average by dividing the sum by the length
avg = sum / length;

// Print the average
printf("The average age is: %.2f", avg);
```

Q3: what is the output of the following piece of code?

```
int myNumbers[] = {25, 50, 75, 100};
printf("%d", myNumbers[0]);

int myNumbers[] = {25, 50, 75, 100};
myNumbers[0] = 33;
printf("%d", myNumbers[0]);
```

Q4: what is the output of the following piece of code?

```
int myNumbers[] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", myNumbers[i]);
}
```

Q5: what is the output of the following piece of code?

```
int myNumbers[] = {25, 50, 75, 100};
int length = sizeof(myNumbers) / sizeof(myNumbers[0]);
int i;

for (i = 0; i < length; i++) {
  printf("%d\n", myNumbers[i]);
}
```

Answer the following MCQ questions:

1. What is a data structure?

a) Programming language
b) collection of algorithms
c) way to store and organize data
d) type of computer hardware

2. Which of these best describes an array?

a) A data structure that shows a hierarchical behavior
b) Container of objects of similar types
c) Arrays are immutable once initialized
d) Array is not a data structure

3. How do you initialize an array in C?
a) int arr[3] = (1,2,3);
b) int arr(3) = {1,2,3};
c) int arr[3] = {1,2,3};
d) int arr(3) = (1,2,3);

4. Assuming int is of 4bytes, what is the size of int arr[15];?
a) 15
b) 19
c) 11
d) 60

5. In general, the index of the first element in an array is _____
a) 0
b) -1
c) 2
d) 1

6. What is the order of a matrix?
a) number of rows X number of columns
b) number of columns X number of rows
c) number of rows X number of rows
d) number of columns X number of columns

7. What are the advantages of arrays?
a) Objects of mixed data types can be stored
b) Elements in an array cannot be sorted
c) Index of first element of an array is 1
d) Easier to store elements of same data type

8. What are the disadvantages of arrays?
a) Data structure like queue or stack cannot be implemented
b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
c) Index value of an array can be negative
d) Elements are sequentially accessed

9. Elements in an array are accessed _____
a) randomly
b) sequentially
c) exponentially
d) logarithmically

10) Which one of the following is the size of int arr[9] assuming that int is of 4 bytes?

1. 9
2. 36
3. 35
4. None of the above

11) What is the output of the below code?

```
1. #include <stdio.h>
2. int main()
3. {
4.    int arr[5]={10,20,30,40,50};
5.    printf("%d", arr[5]);
6.
7.     return 0;
8. }
9.
```

1. Garbage value
2. 10
3. 50
4. None of the above

12. What is the output of the below code?

```
1. #include <stdio.h>
2. int main()
3. {
4.    int arr[5]={10,20,30,40,50};
5.    printf("%d", arr[4]);
6.
7.     return 0;
8. }
```

1. Garbage value
2. 10
3. 50
4. None of the above

13. **The following piece of cod is used for**

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        a[i ][j] += b[i][j];
```

a. One dimensional array.
b. Two dimensional arrays.
c. Three dimensional arrays.
d. All of the above.

Selected applications of programs on arrays

Program to find Largest Element of an Array in C using Loops
Example:

Given **Array: [1, 2, 4, 1, 6, 4]**

Algorithm:

Initially, Largest Element: 1

Now we start traversing the array.

Index: 1
Index element > Largest Element, so updating the largest element with 2.
Index: 2
Index element > Largest Element, so updating the largest element with 4.
Index: 3
Index element < Largest Element, so Largest element remains the same.
Index: 4
Index element > Largest Element, so updating the largest element with 6.
Index: 5
Index element < Largest Element, so Largest element remains the same.

So **Largest Element** is 6.

Program/Source Code

```
1./*
2. * C Program to find the largest number in an array using loops
3. */
4.
5.#include <stdio.h>
6.
7.int main()
8.{
9.    int size, i, largest;
10.
11.            printf("\n Enter the size of the array: ");
12.            scanf("%d", &size);
```

```
13.             int array[size];  //Declaring array
14.
15.             //Input array elements
16.
17.             printf("\n Enter %d elements of the array: \n", size);
18.
19.             for (i = 0; i < size; i++)
20.             {
21.                 scanf(" %d", &array[i]);
22.             }
23.
24.              //Declaring Largest element as the first element
25.             largest = array[0];
26.
27.             for (i = 1; i < size; i++)
28.             {
29.                 if (largest < array[i])
30.                 largest = array[i];
31.             }
32.
33.             printf("\n largest element present in the given array is :
    %d", largest);
34.
35.             return 0;
36.         }
```

Program Explanation

1. Take the size of the array as input from the user.
2. Then, initialize an array of size given by the user.
3. Using **for** loop, take array element as input from users and insert them into the array.
4. After inserting all the elements of the array, consider the very first element of array to be the **largest**.
5. Run a for loop, from 1 to **arraySize-1**, extracting array element one by one and comparing it to the **largest** element.
6. If the **largest** element is smaller than the element being compared, then

the largest element is updated with the value of the current element of the array.
7. In the end, the **largest** element will hold the actual largest value present in the array.

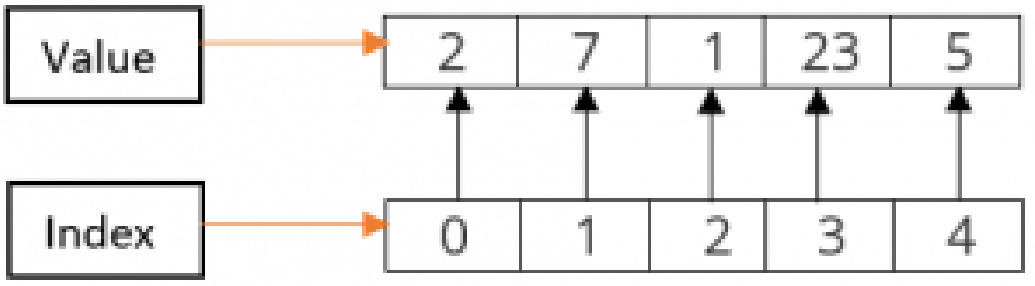
Runtime Test Cases

Here is the runtime output of the C program where the user is reading array of **6** elements with values as **1, 2, 4, 1, 6, and 4**. Then it finds out the largest element and displays its value.

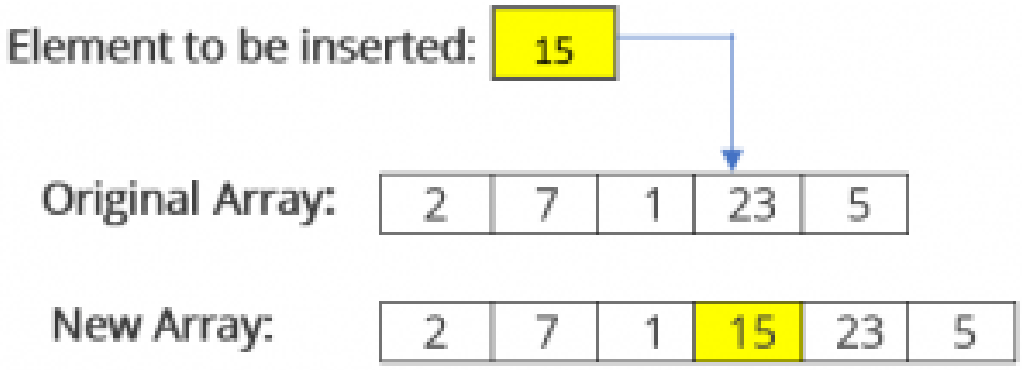
```
Enter the size of the array: 6

Enter 5 elements of the array:
1
2
4
1
6
4

Largest element present in the given array is: 6
```

Write a C Program to insert an element in an Array.

Example: arr[5] = {2,7,1,23,5}



The new element we are entering here is 15 and the position where the element must be inserted is “4”.



Algorithm:

Follow the below steps to solve the problem:

- First get the element to be inserted, say x
- Then get the position at which this element is to be inserted, say pos
- Then shift the array elements from this position to one position forward(towards right), and do this for all the other elements next to pos.
- Insert the element x now at the position pos, as this is now empty.

```
1./*
2. * C program to insert an element in a specified position in a given
   array
3. */
4.
```

```
#include <stdio.h>
void main()
{
    int array[100];
    int i, n, x, pos;

            printf("Enter the number of elements in the array \n");
            scanf("%d", &n);
            printf("Enter the elements \n");

            for (i = 0; i < n; i++)
            {
                scanf("%d", &array[i]);
            }

            printf("Input array elements are: \n");
            for (i = 0; i < n; i++)
            {
                printf("%d ", array[i]);
            }
            printf("\nEnter the new element to be inserted: ");
            scanf("%d", &x);
            printf("Enter the position where element is to be inserted:
    ");
            scanf("%d", &pos);

            //shift all elements 1 position forward from the place
            //where element needs to be inserted
            n=n+1;
            for(i = n-1; i >= pos; i--)
                array[i]=array[i-1];

            array[pos-1]=x; //Insert the element x on the specified
    position

            //print the new array
            for (i = 0; i < n; i++)
```

```
40.             {
41.                 printf("%d ", array[i]);
42.             }
43.         }
```

Program Explanation

1. Initialize an **array** of size 100.
2. Ask the user for number of elements they want to enter into the array. Store it in variable **n**.
3. Take **n** number of elements as input from the user and store it in the **array**.
4. Ask the user for new element, **x** to be inserted and the position **pos** where element needs to be inserted.
5. Increment the value of **n** by 1.
5. Shift all the elements that are next to **pos** to one index forward.
6. Now, **array[pos]** is empty, insert the element **x** into the array.
7. Print all the elements.

Example:

- Consider an example with **n=5** and array elements as **2, 7, 1, 23, 5**.
- Now ask the user for **x** and **pos**. Suppose the value entered by user are **x=15** and **pos=4**.
- Increment the value of **n** by 1 i.e., **n=6**. Run a for loop from **i=n-1 = 5** to **i=pos** = 4 and in each iteration insert the element at previous index i.e., **i-1** to index **i**. Therefore, **array[5] = array[4]** which is equal to **5**.
- Decrement the value of **i** by 1 and repeat the same process **array[4] = array[3]** i.e., array[4] = 23.
- Now **i=3** and **pos=4** so condition in for loop becomes false, come out of the loop.
- Now, insert the element **x** in index **pos-1** i.e., **array[3] = 15**.
- The element is inserted now, run a for loop and print the array with element being inserted. i.e. **2 7 1 15 23 5**

Runtime Test Cases

In this case, the new element is 15 and the position where the element to be inserted is “4”.

```
Enter the number of elements in the array
5
Enter the elements
2 7 1 23 5
Input array elements are:
2 7 1 23 5
Enter the new element to be inserted: 15
Enter the position where element is to be inserted: 4
2 7 1 15 23 5
```

Write C Program to Reverse an Array

Examples:

Input array: [1,2,3,4]

Reversed array: [4,3,2,1]

Algorithm to Reverse an Array:

1. Input size of the array and array elements.
2. Declare two variables pointing to the start and end of array
3. Swap the elements at their places and increment variables pointing to the first element and decrement the variable pointing to the last element.
4. Do it until the first variable is less than the second one.
5. Print the Reversed Array.

Example:

Input array: [7,2,4,1,6,4]

Reversed array: [4,6,1,4,2,7]

```
Start: 0      End: 5        Array After Swapping:
[4,2,4,1,6,7]
Incrementing Start and Decrementing End.

Start: 1      End: 4        Array After Swapping:
[4,6,4,1,2,7]
Incrementing Start and Decrementing End.

Start: 2      End: 3        Array After Swapping:
[4,6,1,4,2,7]

Now after Incrementing Start and Decrementing End,
start is greater than end,
so stopping here and hence, the reversed array is:
Reversed Array: [4,6,1,4,2,7]
```

```c
/*
 *  C program to reverse an array using loops
 */
#include <stdio.h>

int main()
{
   int size;
   printf("Enter size of the array: ");
   scanf("%d",&size);
   printf("Enter Array Elements: ");
   int arr[size];
```

```
    //Input array elements
    for(int i=0;i<size;i++)
    scanf("%d",&arr[i]);
    printf("Entered Array is: ");
    for(int i=0;i<size;i++)
    printf("%d ",arr[i]);

    //Start points at the first element and end points at the last element
    int start=0,end=size-1;
    while(start<end)
    {
        //Swapping elements
        int temp=arr[start];
        arr[start]=arr[end];
        arr[end]=temp;

        //Incrementing start  and decrementing end
        start++;
        end--;
    }

    //Printing reversed array
    printf("\nReversed array is: ");
    for(int i=0;i<size;i++)
    printf("%d ",arr[i]);
    return 0;
}
```

Program Explanation

1. First, input the size of the array.
2. Declare an array and input its elements, then initialize two variables pointing to the first and last index of the array.
3. Swap elements at both indices and increment the first variable and decrement the second variable.

4. Do it until the first variable is less than the second one.
5. Print the reversed array.

Write a program to Add Two Matrix in C

Matrix addition in C is used to add two matrices. i.e. calculate and print the sum of them.
Example:
Given two matrices of same size, this program will add the corresponding elements of each matrix and print the result.

Input:

First Matrix:

2	5
32	**65**
23	**76**

Second Matrix:

23	65
35	2
4	0

Output Matrix:

25	70
67	**67**
27	**76**

Problem Description
Write a C program which calculates the addition of two matrices of the same size.

Problem Solution
In order to add two matrices, we need to add the corresponding elements of each matrix. Suppose we have two matrices of size m x n and p x q.

Algorithm:

1. Take the number of rows and columns as input.
2. Store the number in a variable.
3. Initialize the first matrix.
4. Initialize the second matrix.
5. Initialize the result matrix.
6. Inside a loop, add the corresponding elements of the two matrices and store them in the result matrix.
7. Print the result matrix.

```c
/* C Program to add two matrices of same size using for loop */

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int r, c;
    printf("Enter the number of rows: ");
            scanf("%d", &r);
            printf("Enter the number of columns: ");
            scanf("%d", &c);
            int i, j;
            int **a = (int **)malloc((unsigned) r * sizeof(int *));
            int **b = (int **)malloc((unsigned) r * sizeof(int *));
            int **res = (int **)malloc((unsigned) r * sizeof(int *));
            for (i = 0; i < r; i++)
            {
                a[i] = (int *)malloc((unsigned) c * sizeof(int));
                b[i] = (int *)malloc((unsigned) c * sizeof(int));
                res[i] = (int *)malloc((unsigned) c * sizeof(int));
            }
            printf("Enter the elements of first matrix:\n");
            for (i = 0; i < r; i++)
            {
                for (j = 0; j < c; j++)
                {
```

```
28.                 printf("[%d][%d]: ", i, j);
29.                 scanf("%d", &a[i][j]);
30.             }
31.         }
32.         printf("Enter the elements of second matrix:\n");
33.         for (i = 0; i < r; i++)
34.         {
35.             for (j = 0; j < c; j++)
36.             {
37.                 printf("[%d][%d]: ", i, j);
38.                 scanf("%d", &b[i][j]);
39.             }
40.         }
41.         for (i = 0; i < r; i++)
42.         {
43.             for (j = 0; j < c; j++)
44.             {
45.                 res[i][j] = a[i][j] + b[i][j];
46.             }
47.         }
48.         printf("The result matrix is:\n");
49.         for (i = 0; i < r; i++)
50.         {
51.             for (j = 0; j < c; j++)
52.             {
53.                 printf("%6.d", res[i][j]);
54.             }
55.             printf("\n");
56.         }
57.     }
58.
```

Program Explanation

1. The program will ask the user to enter the number of **rows** and **columns** and will return the result matrix.
2. The program will loop through the number of **rows** and **columns** and add the corresponding elements of the two matrices and store them in the **result** matrix.

3. Here, the memory allocation is done a bit differently. We'll use **malloc** to allocate the memory for the two matrices and the **result matrix**.
4. First we allocate the rows of the three matrices and then we allocate the columns.

Runtime Test Cases

In this case, we enter "3" for the number of rows and "2" for the number of columns as input for the addition of two matrices.

```
Enter the number of rows: 3
Enter the number of columns: 2
Enter the elements of first matrix:
[0][0]: 2
[0][1]: 5
[1][0]: 32
[1][1]: 65
[2][0]: 23
[2][1]: 76
Enter the elements of second matrix:
[0][0]: 23
[0][1]: 65
[1][0]: 35
[1][1]: 2
[2][0]: 4
[2][1]: 0
The result matrix is:
    25    70
    67    67
    27    76
```