C Arrays

An array is a variable that can store multiple values. For example, if you want to store 100 integers, you can create an array for it.

```
int data[100];
```

How to declare an array?

```
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.

It's important to note that, the size and type of an array cannot be changed once it is declared.

Access Array Elements

You can access elements of an array by indices.

Suppose you declared an array ***mark*** as above. The first element is ***mark[0]***, the second element is ***mark[1]*** and so on.

Few keynotes:

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

How to initialize an array?

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

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

You can also initialize an array like this.

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

Here, we haven't specified the size. However, the compiler knows its size is 5 as we are initializing it with 5 elements.
Here,

```
mark[0] is equal to 19
mark[1] is equal to 10
mark[2] is equal to 8
mark[3] is equal to 17
mark[4] is equal to 9
```

Change Value of Array elements

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

// make the value of the third element to -1
mark[2] = -1;

// make the value of the fifth element to 0
mark[4] = 0;
```

Input and Output Array Elements

Here's how you 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 you 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;
}
```

Output

```
Enter 5 integers: 1
-3
34
0
3
Displaying integers: 1
-3
34
0
3
```

Here, we have used a `for` loop to take 5 inputs from the user and store them in an array. Then, using another `for` loop, these elements are displayed on the screen.

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, 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];
   }

   average = sum/n;
   printf("Average = %d", average);

   return 0;
}
```

Output

```
Enter n: 5
Enter number1: 45
Enter number2: 35
Enter number3: 38
Enter number4: 31
Enter number5: 49
Average = 39
```

Here, we have computed the average of *n* numbers entered by the user.

Access elements out of its bound!

Suppose you declared an array of 10 elements. Let's say,

```
int testArray[10];
```

You can access the array elements from `testArray[0]` to `testArray[9]`.

Now let's say if you try to access `testArray[12]`. The element is not available. This may cause unexpected output (undefined behavior). Sometimes you might get an error and some other time your program may run correctly.

Hence, you should never access elements of an array outside of its bound.

Multidimensional arrays

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

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

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

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

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

Here, the array *y* can hold 24 elements.

Initializing a multidimensional array

Here is how you can initialize two-dimensional and three-dimensional arrays:

Initialization of a 2d array

```
// Different ways to initialize two-dimensional array

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

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

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

Initialization of a 3d array

You can initialize a three-dimensional array in a similar way like 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 1: Two-dimensional array to store and print values

```
// C program to store temperature of two cities of a week and display it.
#include <stdio.h>
const int CITY = 2;
const int WEEK = 7;
int main()
{
  int temperature[CITY][WEEK];

  // Using nested loop to store values in a 2d array
  for (int i = 0; i < CITY; ++i)
  {
    for (int j = 0; j < WEEK; ++j)
    {
      printf("City %d, Day %d: ", i + 1, j + 1);
      scanf("%d", &temperature[i][j]);
    }
  }
  printf("\nDisplaying values: \n\n");

  // Using nested loop to display vlues of a 2d array
  for (int i = 0; i < CITY; ++i)
  {
    for (int j = 0; j < WEEK; ++j)
    {
      printf("City %d, Day %d = %d\n", i + 1, j + 1, temperature[i][j]);
    }
  }
  return 0;
}
```

Output

```
City 1, Day 1: 33
City 1, Day 2: 34
City 1, Day 3: 35
City 1, Day 4: 33
City 1, Day 5: 32
City 1, Day 6: 31
City 1, Day 7: 30
City 2, Day 1: 23
City 2, Day 2: 22
City 2, Day 3: 21
City 2, Day 4: 24
City 2, Day 5: 22
City 2, Day 6: 25
City 2, Day 7: 26

Displaying values:

City 1, Day 1 = 33
City 1, Day 2 = 34
City 1, Day 3 = 35
City 1, Day 4 = 33
City 1, Day 5 = 32
City 1, Day 6 = 31
City 1, Day 7 = 30
City 2, Day 1 = 23
City 2, Day 2 = 22
City 2, Day 3 = 21
City 2, Day 4 = 24
City 2, Day 5 = 22
City 2, Day 6 = 25
City 2, Day 7 = 26
```

Example 2: Sum of two matrices

```c
// 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;
}
```

Output

```
Enter elements of 1st matrix
Enter a11: 2;
Enter a12: 0.5;
Enter a21: -1.1;
Enter a22: 2;
Enter elements of 2nd matrix
Enter b11: 0.2;
Enter b12: 0;
Enter b21: 0.23;
Enter b22: 23;

Sum Of Matrix:
2.2     0.5
-0.9    25.0
```

Example 3: Three-dimensional array

```c
// C Program to store and print 12 values entered by the user

#include <stdio.h>
int main()
{
  int test[2][3][2];

  printf("Enter 12 values: \n");

  for (int i = 0; i < 2; ++i)
  {
    for (int j = 0; j < 3; ++j)
    {
      for (int k = 0; k < 2; ++k)
      {
        scanf("%d", &test[i][j][k]);
      }
    }
  }

  // Printing values with proper index.

  printf("\nDisplaying values:\n");
  for (int i = 0; i < 2; ++i)
  {
    for (int j = 0; j < 3; ++j)
    {
      for (int k = 0; k < 2; ++k)
      {
        printf("test[%d][%d][%d] = %d\n", i, j, k, test[i][j][k]);
      }
    }
  }

  return 0;
}
```

Output

```
Enter 12 values:
1
2
3
4
5
```

```
6
7
8
9
10
11
12

Displaying Values:
test[0][0][0] = 1
test[0][0][1] = 2
test[0][1][0] = 3
test[0][1][1] = 4
test[0][2][0] = 5
test[0][2][1] = 6
test[1][0][0] = 7
test[1][0][1] = 8
test[1][1][0] = 9
test[1][1][1] = 10
test[1][2][0] = 11
test[1][2][1] = 12
```

Pass arrays to a function in C

In C programming, you can pass en entire array to functions. Before we learn that, let's see how you can pass individual elements of an array to functions.

Passing individual array elements

Passing array elements to a function is similar to passing variables to a function.

Example 1: Passing an array

```
#include <stdio.h>
void display(int age1, age2)
{
    printf("%d\n", age1);
    printf("%d\n", age2);
}

int main()
{
    int ageArray[] = {2, 8, 4, 12};

    // Passing second and third elements to display()
    display(ageArray[1], ageArray[2]);
    return 0;
}
```

Output

Example 2: Passing arrays to functions

```
// Program to calculate the sum of array elements by passing to a function

#include <stdio.h>
float calculateSum(float age[]);

int main() {
    float result, age[] = {23.4, 55, 22.6, 3, 40.5, 18};

    // age array is passed to calculateSum()
    result = calculateSum(age);
    printf("Result = %.2f", result);
    return 0;
}

float calculateSum(float age[]) {

  float sum = 0.0;

  for (int i = 0; i < 6; ++i) {
          sum += age[i];
  }

  return sum;
}
```

Output

```
Result = 162.50
```

To pass an entire array to a function, only the name of the array is passed as an argument.

```
result =  calculateSum(age);
```

However, notice the use of `[]` in the function definition.

```
float calculateSum(float age[]) {
... ..
}
```

This informs the compiler that you are passing a one-dimensional array to the function.

Passing Multidimensional Arrays to a Function

To pass multidimensional arrays to a function, only the name of the array is passed to the function(similar to one-dimensional arrays).

Example 3: Passing two-dimensional arrays

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

```c
void displayNumbers(int num[2][2]);
int main()
{
    int num[2][2];
    printf("Enter 4 numbers:\n");
    for (int i = 0; i < 2; ++i)
        for (int j = 0; j < 2; ++j)
            scanf("%d", &num[i][j]);

    // passing multi-dimensional array to a function
    displayNumbers(num);
    return 0;
}

void displayNumbers(int num[2][2])
{
    printf("Displaying:\n");
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
           printf("%d\n", num[i][j]);
        }
    }
}
```

Output

```
Enter 4 numbers:
2
3
4
5
Displaying:
2
3
4
5
```

Note: In C programming, you can pass arrays to functions, however, you cannot return arrays from functions.

Code to Calculate Average Using Arrays

```c
#include <stdio.h>
int main()
{
    int n, i;
    float num[100], sum = 0.0, average;
    printf("Enter the numbers of elements: ");
    scanf("%d", &n);
    while (n > 100 || n <= 0)
    {
        printf("Error! number should in range of (1 to 100).\n");
        printf("Enter the number again: ");
        scanf("%d", &n);
    }
    for(i = 0; i < n; ++i)
    {
        printf("%d. Enter number: ", i+1);
        scanf("%f", &num[i]);
        sum += num[i];
    }
    average = sum / n;
    printf("Average = %.2f", average);
    return 0;
}
```

Output

```
Enter the numbers of elements: 6
1. Enter number: 45.3
2. Enter number: 67.5
3. Enter number: -45.6
4. Enter number: 20.34
5. Enter number: 33
6. Enter number: 45.6
Average = 27.69
```

Example: Display Largest Element of an array

```
#include <stdio.h>

int main()
{
    int i, n;
    float arr[100];

    printf("Enter total number of elements(1 to 100): ");
    scanf("%d", &n);
    printf("\n");

    // Stores number entered by the user
    for(i = 0; i < n; ++i)
    {
       printf("Enter Number %d: ", i+1);
       scanf("%f", &arr[i]);
    }

    // Loop to store largest number to arr[0]
    for(i = 1; i < n; ++i)
    {
       // Change < to > if you want to find the smallest element
       if(arr[0] < arr[i])
           arr[0] = arr[i];
    }
    printf("Largest element = %.2f", arr[0]);

    return 0;
}
```

Output

```
Enter total number of elements(1 to 100): 8

Enter Number 1: 23.4
Enter Number 2: -34.5
Enter Number 3: 50
Enter Number 4: 33.5
Enter Number 5: 55.5
Enter Number 6: 43.7
Enter Number 7: 5.7
Enter Number 8: -66.5
```

Example: Program to Add Two Matrices

```c
#include <stdio.h>
int main(){
    int r, c, a[100][100], b[100][100], sum[100][100], i, j;

    printf("Enter number of rows (between 1 and 100): ");
    scanf("%d", &r);
    printf("Enter 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");
    for(i=0; i<r; ++i)
        for(j=0; j<c; ++j)
        {
            printf("Enter element a%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];
        }

    // Displaying the result
    printf("\nSum of two matrix is: \n\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 number of rows (between 1 and 100): 2
Enter 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 a11: -4
Enter element a12: 5
Enter element a13: 3
Enter element a21: 5
Enter element a22: 6
Enter element a23: 3

Sum of two matrix is:

-2   8   7

10   8   6
```

Example: Find the product of two square matrices

```
#include<stdio.h>

int main() {

    int a[10][10], b[10][10], c[10][10], n, i, j, k;


    printf("Enter the value of N (N <= 10): ");

    scanf("%d", & n);

    printf("Enter the elements of Matrix-A: \n");


    for (i = 0; i < n; i++) {

        for (j = 0; j < n; j++) {

            scanf("%d", & a[i][j]);

        }

    }


    printf("Enter the elements of Matrix-B: \n");


    for (i = 0; i < n; i++) {

        for (j = 0; j < n; j++) {

            scanf("%d", & b[i][j]);

        }

    }


    for (i = 0; i < n; i++) {

        for (j = 0; j < n; j++) {

            c[i][j] = 0;

            for (k = 0; k < n; k++) {

                c[i][j] += a[i][k] * b[k][j];

            }

        }

    }
    printf("The product of the two matrices is: \n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d\t", c[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Example: to find the product of two rectangular matrices

```
#include<stdio.h>
int main() {
    int m, n, p, q, i, j, k;
    int a[10][10], b[10][10], res[10][10];

    printf("Enter the order of first matrix\n");
    scanf("%d%d", & m, & n);
    printf("Enter the order of second matrix\n");
    scanf("%d%d", & p, & q);

    if (n != p) {
        printf("Matrix is incompatible for multiplication\n");
    } else {
        printf("Enter the elements of Matrix-A:\n");
        for (i = 0; i < m; i++) {
            for (j = 0; j < n; j++) {
                scanf("%d", & a[i][j]);
            }
        }
        printf("Enter the elements of Matrix-B:\n");
        for (i = 0; i < p; i++) {
            for (j = 0; j < q; j++) {
                scanf("%d", & b[i][j]);
            }
        }

        for (i = 0; i < m; i++) {
            for (j = 0; j < q; j++) {
                res[i][j] = 0;
                for (k = 0; k < p; k++) {
                    res[i][j] += a[i][k] * b[k][j];
                }
            }
        }

        printf("The product of the two matrices is:-\n");

        for (i = 0; i < m; i++) {
            for (j = 0; j < q; j++) {
                printf("%d\t", res[i][j]);
            }
            printf("\n");
        }
    }

    return 0;
}
```

Multiply Matrices by Passing it to a Function

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

```
// function to get matrix elements entered by the user
void getMatrixElements(int matrix[][10], int row, int column) {

  printf("\nEnter elements: \n");

  for (int i = 0; i < row; ++i) {
    for (int j = 0; j < column; ++j) {
      printf("Enter a%d%d: ", i + 1, j + 1);
      scanf("%d", &matrix[i][j]);
    }
  }
}

// function to multiply two matrices
void multiplyMatrices(int first[][10],
               int second[][10],
               int result[][10],
               int r1, int c1, int r2, int c2) {

  // Initializing elements of matrix mult to 0.
  for (int i = 0; i < r1; ++i) {
    for (int j = 0; j < c2; ++j) {
      result[i][j] = 0;
    }
  }

  // Multiplying first and second matrices and storing it in result
  for (int i = 0; i < r1; ++i) {
    for (int j = 0; j < c2; ++j) {
      for (int k = 0; k < c1; ++k) {
        result[i][j] += first[i][k] * second[k][j];
```

```
            }
         }
      }
   }

// function to display the matrix
void display(int result[][10], int row, int column) {

   printf("\nOutput Matrix:\n");
   for (int i = 0; i < row; ++i) {
      for (int j = 0; j < column; ++j) {
         printf("%d  ", result[i][j]);
         if (j == column - 1)
            printf("\n");
      }
   }
}

int main() {
   int first[10][10], second[10][10], result[10][10], r1, c1, r2, c2;
   printf("Enter rows and column for the first matrix: ");
   scanf("%d %d", &r1, &c1);
   printf("Enter rows and column for the second matrix: ");
   scanf("%d %d", &r2, &c2);

   // Taking input until
   // 1st matrix columns is not equal to 2nd matrix row
   while (c1 != r2) {
      printf("Error! Enter rows and columns again.\n");
      printf("Enter rows and columns for the first matrix: ");
      scanf("%d%d", &r1, &c1);
      printf("Enter rows and columns for the second matrix: ");
```

```
    scanf("%d%d", &r2, &c2);
  }

  // get elements of the first matrix
  getMatrixElements(first, r1, c1);

  // get elements of the second matrix
  getMatrixElements(second, r2, c2);

  // multiply two matrices.
  multiplyMatrices(first, second, result, r1, c1, r2, c2);

  // display the result
  display(result, r1, c2);

  return 0;
}
```

Program to Add Two Matrices

```
#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");
  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;
}
```

Program to Find the Transpose of a Matrix

```
#include <stdio.h>
int main() {
  int a[10][10], transpose[10][10], r, c;
  printf("Enter rows and columns: ");
```

```
scanf("%d %d", &r, &c);

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

// printing the matrix a[][]
printf("\nEntered matrix: \n");
for (int i = 0; i < r; ++i)
for (int j = 0; j < c; ++j) {
  printf("%d  ", a[i][j]);
  if (j == c - 1)
  printf("\n");
}

// computing the transpose
for (int i = 0; i < r; ++i)
for (int j = 0; j < c; ++j) {
  transpose[j][i] = a[i][j];
}

// printing the transpose
printf("\nTranspose of the matrix:\n");
for (int i = 0; i < c; ++i)
for (int j = 0; j < r; ++j) {
  printf("%d  ", transpose[i][j]);
  if (j == r - 1)
  printf("\n");
```

```
 }
 return 0;
}
```

Store Numbers and Calculate Average Using Arrays

```
#include <stdio.h>
int main() {
  int n, i;
  float num[100], sum = 0.0, avg;

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

  while (n > 100 || n < 1) {
    printf("Error! number should in range of (1 to 100).\n");
    printf("Enter the number again: ");
    scanf("%d", &n);
  }

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

  avg = sum / n;
  printf("Average = %.2f", avg);
  return 0;
}
```

C program code will insert an element into an array

```
#include <stdio.h>

int main()
{
   int array[50], position, c, n, value;

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

   printf("Enter %d elements\n", n);

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

   printf("Please enter the location where you want to insert an new element\n");
   scanf("%d", &position);

   printf("Please enter the value\n");
   scanf("%d", &value);

   for (c = n - 1; c >= position - 1; c–)
      array[c+1] = array[c];

   array[position-1] = value;

   printf("Resultant array is\n");

   for (c = 0; c <= n; c++)
      printf("%d\n", array[c]);
   return 0;
}
```

C program to merge two arrays into one array

```
#include <stdio.h>
void merge(int [], int, int [], int, int []);
int main()
{
   int a[100], b[100], m, n, c, sorted[200];
   printf("Input number of elements in first array\n");
   scanf("%d", &m);
   printf("Input %d integers\n", m);
   for (c = 0; c < m; c++) {
      scanf("%d", &a[c]);
   }

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

   printf("Input %d integers\n", n);
   for (c = 0; c < n; c++) {
      scanf("%d", &b[c]);
   }

   merge(a, m, b, n, sorted);
   printf("Sorted array:\n");
```

```
  for (c = 0; c < m + n; c++) {
    printf("%d\n", sorted[c]);
  }
  return 0;
}

void merge(int a[], int m, int b[], int n, int sorted[]) {
  int i, j, k;
  j = k = 0;

  for (i = 0; i < m + n;) {
    if (j < m && k < n) {
      if (a[j] < b[k]) {
        sorted[i] = a[j];
        j++;
      }
      else {
        sorted[i] = b[k];
        k++;
      }
      i++;
    }

    else if (j == m) {
      for (; i < m + n;) {
        sorted[i] = b[k];
        k++;
        i++;
      }
    }
```

```
        else {
            for (; i < m + n;) {
                sorted[i] = a[j];
                j++;
                i++;
            }
        }
    }
}
```

program find maximum or largest element present in an array

```
#include <stdio.h>

int main()
{
    int array[100], maximum, size, c, location = 1;

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

    printf("Enter %d integers\n", size);

    for (c = 0; c < size; c++)
        scanf("%d", &array[c]);

    maximum = array[0];

    for (c = 1; c < size; c++)
    {
        if (array[c] > maximum)
        {
            maximum  = array[c];
            location = c+1;
        }
    }

    printf("Maximum element is present at location %d and it's value is %d.\n",
location, maximum);
    return 0;
}




#include <stdio.h>

int main()
{
    long array[100], *maximum, size, c, location = 1;

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

    printf("Enter %ld integers\n", size);

    for ( c = 0 ; c < size ; c++ )
        scanf("%ld", &array[c]);

    maximum  = array;
    *maximum = *array;

    for (c = 1; c < size; c++)
    {
        if (*(array+c) > *maximum)
        {
           *maximum = *(array+c);
           location = c+1;
        }
    }

    printf("Maximum element found at location %ld and it's value is %ld.\n", location,
*maximum);
    return 0;
}
```