

A] Insertion Sort Program in C

```
#include <stdio.h>

// Function to print an array
void printArray(int array[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
}

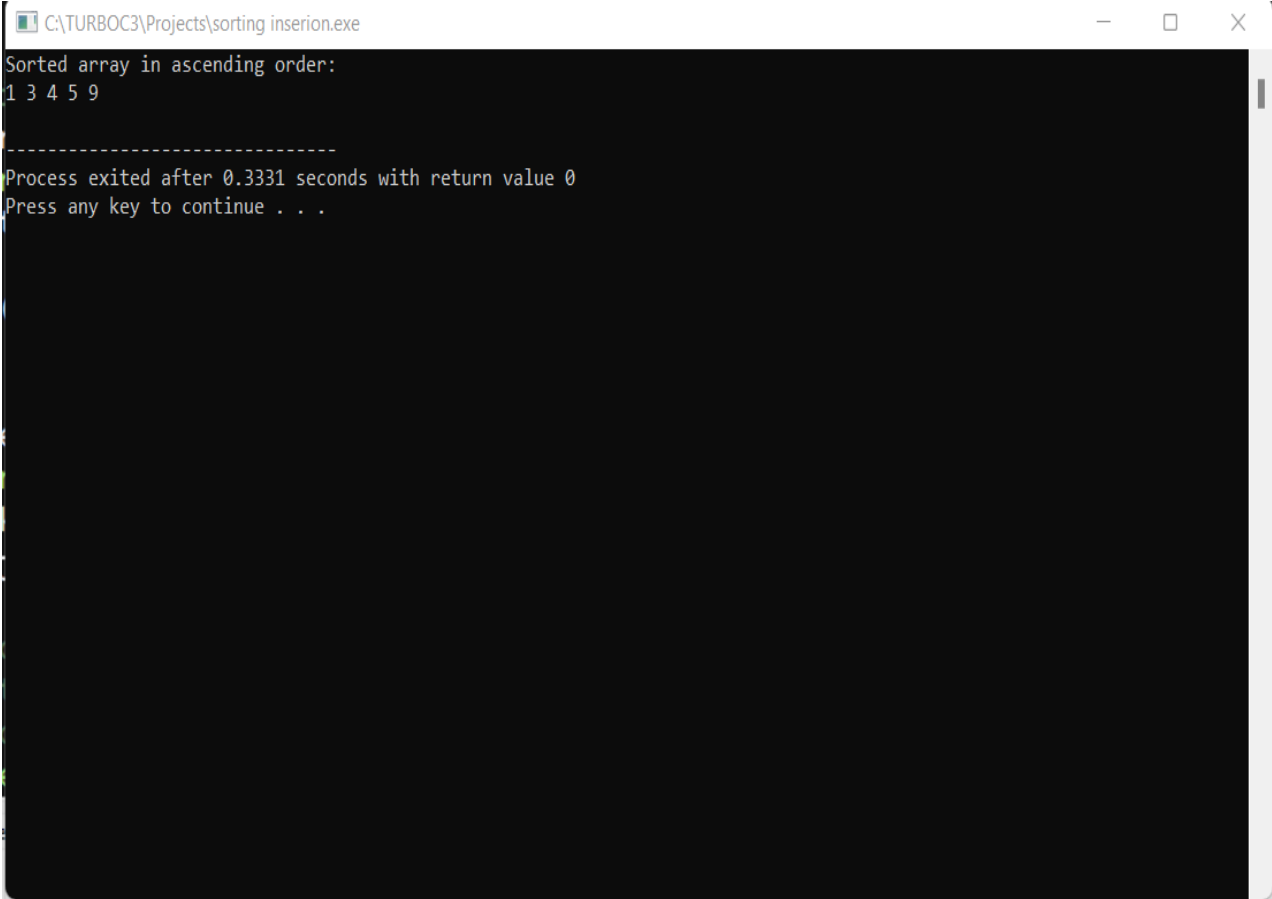
void insertionSort(int array[], int size) {
    for (int step = 1; step < size; step++) {
        int key = array[step];
        int j = step - 1;

        // Compare key with each element on the left of it until an element smaller
        // than
        // it is found.
        // For descending order, change key<array[j] to key>array[j].
        while (key < array[j] && j >= 0) {
            array[j + 1] = array[j];
            --j;
        }
        array[j + 1] = key;
    }
}
```

```
// Driver code

int main() {
    int data[] = {9, 5, 1, 4, 3};
    int size = sizeof(data) / sizeof(data[0]);
    insertionSort(data, size);
    printf("Sorted array in ascending order:\n");
    printArray(data, size);
}
```

Output:



The screenshot shows a TurboC3 console window titled "C:\TURBOC3\Projects\sorting inserion.exe". The output of the program is displayed on a black background with white text. It shows the sorted array in ascending order: 1 3 4 5 9. Below this, a separator line is shown, followed by the message "Process exited after 0.3331 seconds with return value 0" and "Press any key to continue . . .".

```
C:\TURBOC3\Projects\sorting inserion.exe
Sorted array in ascending order:
1 3 4 5 9

-----
Process exited after 0.3331 seconds with return value 0
Press any key to continue . . .
```

B] Merge sort in C

```
#include <stdio.h>

// Merge two subarrays L and M into arr
void merge(int arr[], int p, int q, int r) {

    // Create  $L \leftarrow A[p..q]$  and  $M \leftarrow A[q+1..r]$ 
    int n1 = q - p + 1;
    int n2 = r - q;

    int L[n1], M[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[p + i];
    for (int j = 0; j < n2; j++)
        M[j] = arr[q + 1 + j];

    // Maintain current index of sub-arrays and main array
    int i, j, k;

    i = 0;
    j = 0;
    k = p;

    // Until we reach either end of either L or M, pick larger among
    // elements L and M and place them in the correct position at A[p..r]
```

```
while (i < n1 && j < n2) {  
    if (L[i] <= M[j]) {  
        arr[k] = L[i];  
        i++;  
    } else {  
        arr[k] = M[j];  
        j++;  
    }  
    k++;  
}
```

```
// When we run out of elements in either L or M,  
// pick up the remaining elements and put in A[p..r]
```

```
while (i < n1) {  
    arr[k] = L[i];  
    i++;  
    k++;  
}
```

```
while (j < n2) {  
    arr[k] = M[j];  
    j++;  
    k++;  
}  
}
```

```
// Divide the array into two subarrays, sort them and merge them
```

```

void mergeSort(int arr[], int l, int r) {
    if (l < r) {

        // m is the point where the array is divided into two subarrays
        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        // Merge the sorted subarrays
        merge(arr, l, m, r);
    }
}

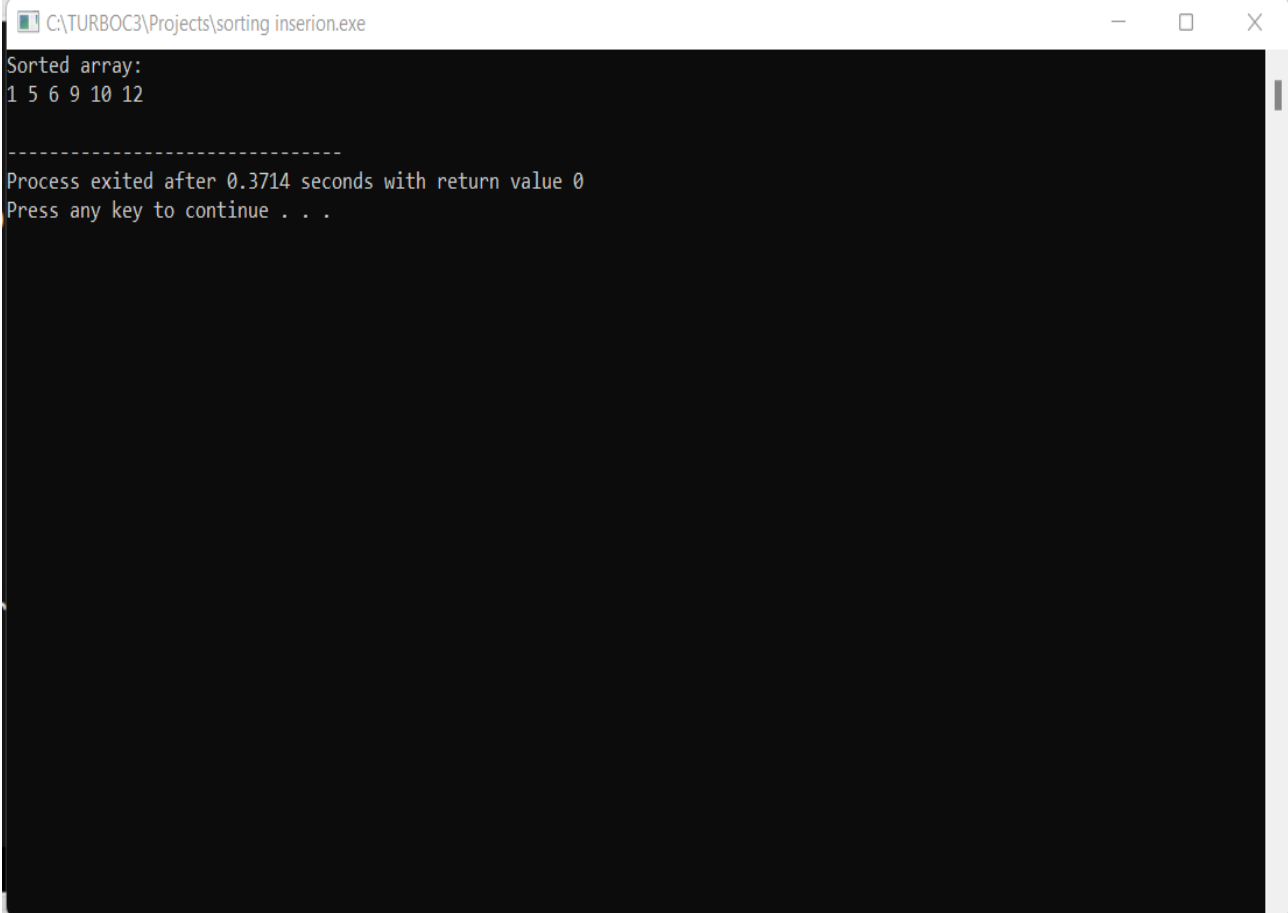
// Print the array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program
int main() {
    int arr[] = {6, 5, 12, 10, 9, 1};
    int size = sizeof(arr) / sizeof(arr[0]);
    mergeSort(arr, 0, size - 1);
    printf("Sorted array: \n");
}

```

```
    printArray(arr, size);  
}
```

Output:



The screenshot shows a TurboC++ console window titled "C:\TURBOC3\Projects\sorting inserion.exe". The output text is as follows:

```
Sorted array:  
1 5 6 9 10 12  
  
-----  
Process exited after 0.3714 seconds with return value 0  
Press any key to continue . . .
```

C] Quick Sort Program in C

```
#include <stdio.h>

// function to swap elements
void swap(int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}

// function to find the partition position
int partition(int array[], int low, int high) {

    // select the rightmost element as pivot
    int pivot = array[high];

    // pointer for greater element
    int i = (low - 1);

    // traverse each element of the array
    // compare them with the pivot
    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
```

```
// if element smaller than pivot is found
// swap it with the greater element pointed by i
i++;

// swap element at i with element at j
swap(&array[i], &array[j]);
}
}

// swap the pivot element with the greater element at i
swap(&array[i + 1], &array[high]);

// return the partition point
return (i + 1);
}
```

```
void quickSort(int array[], int low, int high) {
    if (low < high) {

        // find the pivot element such that
        // elements smaller than pivot are on left of pivot
        // elements greater than pivot are on right of pivot
        int pi = partition(array, low, high);
```



```
// recursive call on the left of pivot
quickSort(array, low, pi - 1);

// recursive call on the right of pivot
quickSort(array, pi + 1, high);
}
}
```

```
// function to print array elements
void printArray(int array[], int size) {
    for (int i = 0; i < size; ++i) {
        printf("%d ", array[i]);
    }
    printf("\n");
}
```

```
// main function
int main() {
    int data[] = {8, 7, 2, 1, 0, 9, 6};

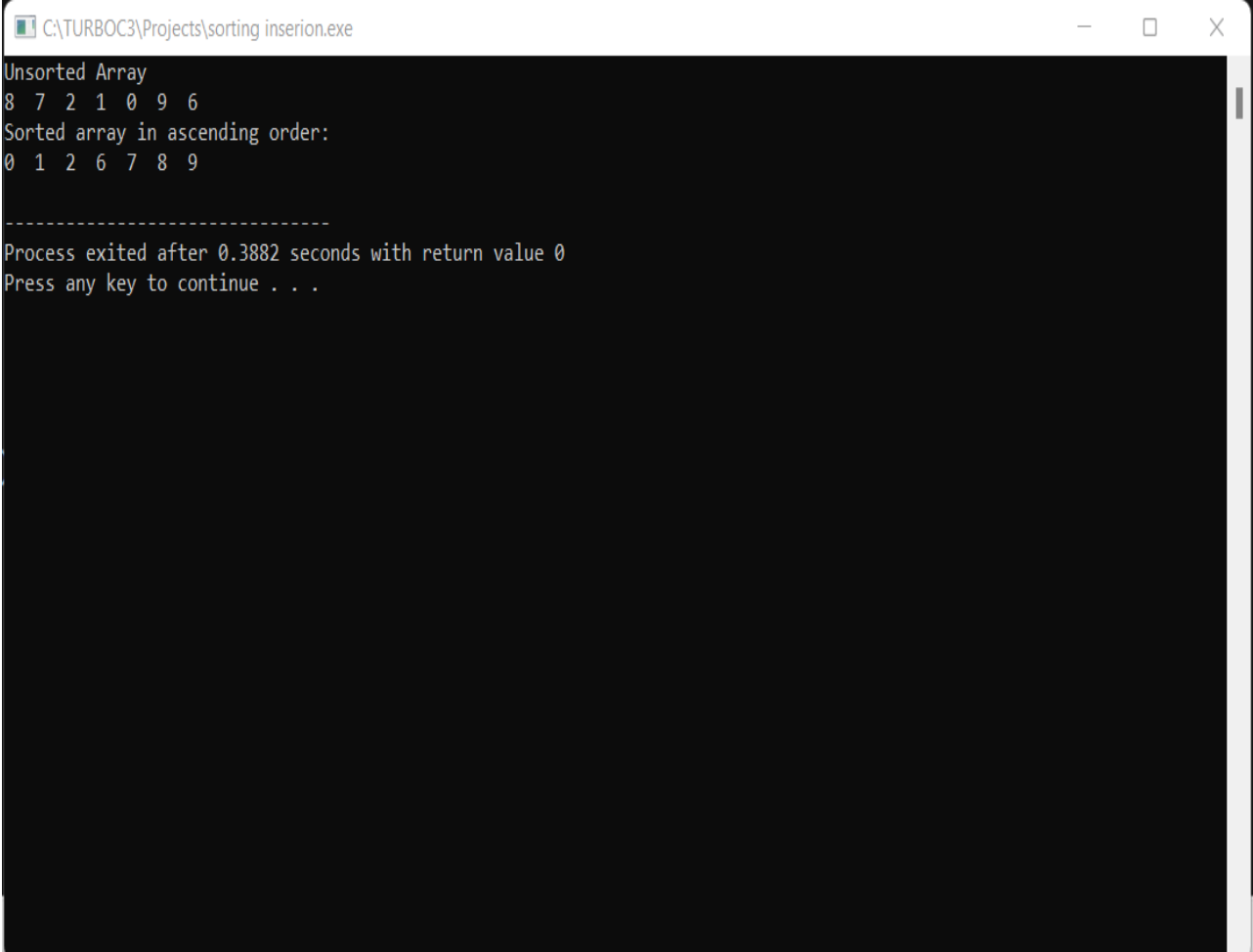
    int n = sizeof(data) / sizeof(data[0]);

    printf("Unsorted Array\n");
    printArray(data, n);
}
```

```
// perform quicksort on data
quickSort(data, 0, n - 1);

printf("Sorted array in ascending order: \n");
printArray(data, n);
}
```

Output:



A screenshot of a TurboC3 console window titled "C:\TURBOC3\Projects\sorting inserion.exe". The window has a black background with white text. The output shows the unsorted array [8, 7, 2, 1, 0, 9, 6], the sorted array in ascending order [0, 1, 2, 6, 7, 8, 9], and a message indicating the process exited after 0.3882 seconds with a return value of 0. The prompt "Press any key to continue . . ." is visible at the bottom.

```
C:\TURBOC3\Projects\sorting inserion.exe
Unsorted Array
8 7 2 1 0 9 6
Sorted array in ascending order:
0 1 2 6 7 8 9

-----
Process exited after 0.3882 seconds with return value 0
Press any key to continue . . .
```

D] Heap Sort in C

```
#include <stdio.h>

// Function to swap the the position of two elements
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(int arr[], int n, int i) {
    // Find largest among root, left child and right child
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] > arr[largest])
        largest = left;

    if (right < n && arr[right] > arr[largest])
        largest = right;

    // Swap and continue heapifying if root is not largest
    if (largest != i) {
        swap(&arr[i], &arr[largest]);
    }
}
```

```

        heapify(arr, n, largest);
    }
}

// Main function to do heap sort
void heapSort(int arr[], int n) {
    // Build max heap
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // Heap sort
    for (int i = n - 1; i >= 0; i--) {
        swap(&arr[0], &arr[i]);

        // Heapify root element to get highest element at root again
        heapify(arr, i, 0);
    }
}

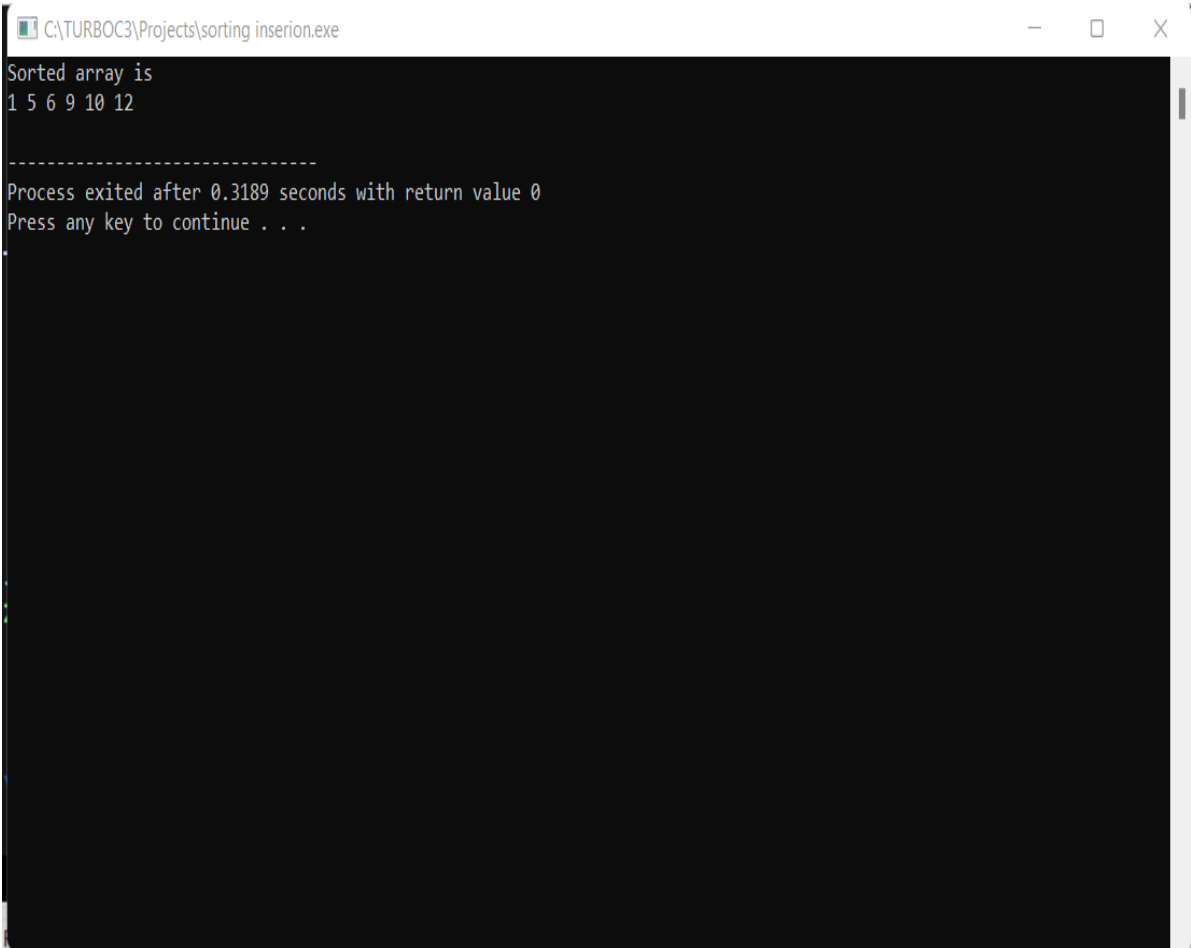
// Print an array
void printArray(int arr[], int n) {
    for (int i = 0; i < n; ++i)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver code
int main() {

```

```
int arr[] = {1, 12, 9, 5, 6, 10};  
  
int n = sizeof(arr) / sizeof(arr[0]);  
  
heapSort(arr, n);  
  
printf("Sorted array is \n");  
printArray(arr, n);  
}
```

Output:



The screenshot shows a TurboC++ console window titled "C:\TURBOC3\Projects\sorting inserion.exe". The output of the program is displayed on a black background with white text. It shows the sorted array and a message indicating the process has exited.

```
C:\TURBOC3\Projects\sorting inserion.exe  
Sorted array is  
1 5 6 9 10 12  
  
-----  
Process exited after 0.3189 seconds with return value 0  
Press any key to continue . . .
```