Quick Sort

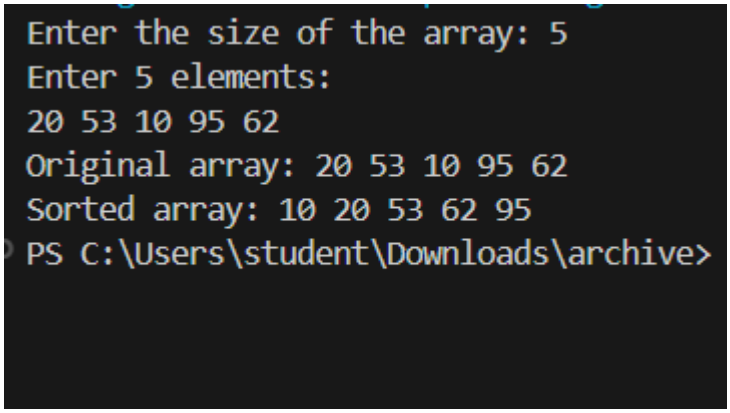
Code:

```
#include <stdio.h>
int partition(int arr[], int low, int high) {
   int pivot = arr[high];
   int i = low - 1;
   for (int j = low; j < high; j++) {
      if (arr[j] <= pivot) {
         i++;
         int temp = arr[i];
         arr[i] = arr[j];
         arr[j] = temp;
      }
   }
   int temp = arr[i + 1];
   arr[i + 1] = arr[high];
   arr[high] = temp;
   return i + 1;
}
void quickSort(int arr[], int low, int high) {
   if (low < high) {

      int pivotIndex = partition(arr, low, high);
      quickSort(arr, low, pivotIndex - 1);
      quickSort(arr, pivotIndex + 1, high);
   }
```

```
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
void inputArray(int arr[], int size) {
    printf("Enter %d elements:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
}
int main() {
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    inputArray(arr, size);
    printf("Original array: ");
    printArray(arr, size);
    quickSort(arr, 0, size - 1);
    printf("Sorted array: ");
    printArray(arr, size);
    return 0;
}
```

Output:



Anaiysis Of Quick Short:

In this program implements the QuickSort algorithm to sort an array of integers:

- partition: Chooses a pivot and rearranges the array so that elements smaller than the pivot are on the left and larger ones are on the right. It returns the pivot's index.
- quickSort: Recursively sorts the subarrays formed by partitioning.
- printArray: Prints the array elements.
- inputArray: Accepts user input to populate the array.

```
/* low --> starting index, high --> ending index */
quick sort (arr[], low, high)
{
  if (low < high)
  {
  /* Pi is partitioning index, arr[Pi] is now at
                                    right place */
  Pi = partision (arr, low, high);

  quicksort (arr, low, Pi-1); // Before Pi
  quicksort (arr, pi+1, high); // After Pi
}
}

Partition (arr[], low, high)
{
// pivot (Element to be placed at right
          position
   Pivot = arr (high);
    i = (low - 1)
for (J = low; j < high - 1; j++)
 {
  if (arr[j] < pivot)
  {
   i++;
  Swap arr[i] & arr[j]
  }
} Swap arr[i+1] and arr[high]
    return (i+1)
  }
```