```
#include <iostream>

int partition(int arr[], int low, int high) {
    int pivot = arr[high];  // 选择最后一个元素作为基准
    int i = low - 1;

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            // 交换元素
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    // 将基准元素放到正确位置
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
}
```

这是快速排序算法中的分区函数。它选择数组中的最后一个元素作为基准（pivot），然后通过将小于基准的元素移到基准的左边、大于基准的元素移到基准的右边来对数组进行分区。它使用两个指针i和j，其中i表示小于基准的子数组的末尾位置。在遍历过程中，如果当前元素小于基准，则将其交换到i的位置，然后i右移。最后，将基准元素放置在i+1的位置，以确保左侧的元素小于等于基准，右侧的元素大于等于基准。

```
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        // 递归地对两个子数组进行快速排序
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

这是快速排序算法的主函数。它采用分治的思想，首先判断数组是否需要排序（即low < high），如果是，则通过调用分区函数将数组分成两个子数组，并分别递归地对这两个子数组进行快速排序。

```
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int size = sizeof(arr) / sizeof(arr[0]);

    quickSort(arr, 0, size - 1);
```

```
    std::cout << "排序结果: ";
    for (int i = 0; i < size; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

这是主函数，用于测试快速排序算法。它创建一个包含待排序元素的数组，并计算数组的大小。然后，它调用quickSort函数对数组进行排序。最后，它输出排序结果。

希望这些注释和解释能帮助您理解快速排序算法的思路和实现。