Selection Sort

Successively find the $n$th smallest value and swap it with the current $n$th value.

Analysis

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>array to be sorted</td>
<td>int[]</td>
<td>varying</td>
<td>received &amp; passed back</td>
<td>arr</td>
</tr>
<tr>
<td># of elements in array</td>
<td>int</td>
<td>varying</td>
<td>received</td>
<td>n</td>
</tr>
<tr>
<td>index of smallest element</td>
<td>int</td>
<td>varying</td>
<td>local</td>
<td>smallIndex</td>
</tr>
<tr>
<td>temporary for swapping</td>
<td>int</td>
<td>varying</td>
<td>local</td>
<td>temp</td>
</tr>
<tr>
<td>index counters</td>
<td>int, int</td>
<td>varying</td>
<td>local</td>
<td>pass, j</td>
</tr>
</tbody>
</table>

Algorithm

1. For pass from 0 to n-2 by 1 do (scan the sublist starting at index pass)
   1.1 Set smallIndex to pass (current index of smallest)
   1.2 For j from pass+1 to n-1 by 1 do
      1.2.1 If arr[j] < arr[smallIndex] then (found smaller)
      1.2.1.1 Set smallIndex to j
   1.3 If smallIndex not equal pass (exchange elements if needed)
      1.3.1 Set temp to arr[pass]
      1.3.2 Set arr[pass] to arr[smallIndex]
      1.3.3 Set arr[smallIndex] to temp
void SelectionSort (int arr[], int n)
{
    for (int pass = 0; pass < n-1; pass++)
    {
        // index of the smallest value in sublist
        int smallIndex = pass;

        // scan the sublist starting at pass+1
        for (int j = pass+1; j < n; j++)
            if (arr[j] < arr[smallIndex]) // found smaller
                smallIndex = j;

        // swap values, if needed
        if (smallIndex != pass)
        {
            int temp = arr[pass];
            arr[pass] = arr[smallIndex];
            arr[smallIndex] = temp;
        } // end swap
    } // end for
} // end SelectionSort