```
1   void Swap (int *array, size_t first, size_t second) {
2       int dummy = array[first];
3       array[first] = array[second];
4       array[second] = dummy;
5   }
6
7   void InsertionSort (int *array, size_t len) {
8       for (size_t i = 1; i < len; ++i) {
9           size_t j = i;
10          while (j > 0 && array[j] < array[j-1]) {
11              Swap(array, j, j - 1);
12              --j;
13          }
14      }
15  }
```

**Листинг 1 – Алгоритм сортировки вставками**

```
1   void SelectionSort (int *array, size_t len) {
2       for (size_t i = 0; i < len - 1; ++i) {
3           size_t min = i;
4           for (size_t j = i + 1; j < len; ++j) {
5               if (array[j] < array[min])
6                   min = j;
7           }
8           Swap(array, i, min);
9       }
10  }
```

**Листинг 2 – Алгоритм сортировки выбором**

```
1   void BubbleSort (int *array, size_t len) {
2       for (size_t i = 0; i < len - 1; ++i) {
3           bool swapped = false;
4           for (size_t j = len - 1; j > i; --j) {
5               if (array[j] < array[j-1]) {
6                   Swap(array, j, j - 1);
7                   swapped = true;
8               }
9           }
10          if (!swapped) break;
11      }
12  }
```

**Листинг 3 – Алгоритм сортировки пузырьком**

```
1   void MergeSort_recursion (int *array, size_t len, int * tmp) {
2       if (len < 2) return;
3
4       size_t middle = len / 2;
5       MergeSort_recursion(array, middle, tmp);
6       MergeSort_recursion(array + middle, len - middle, tmp);
7       memcpy(tmp, array, middle * sizeof(array[0]));
8
9       size_t i = 0, j = middle, k = 0;
10      while (i < middle && j < len)
11          array[k++] = (array[j] < tmp[i])?array[j++]:tmp[i++];
12      while (i < middle)
13          array[k++] = tmp [i++];
14  }
15
16  void MergeSort(int *array, size_t len) {
17      const size_t half_len = (len + 1) / 2;
18      int *tmp = new int[half_len];
19      MergeSort_recursion(array, len, tmp);
20      delete[] tmp;
21  }
```

**Листинг 4 – Алгоритм сортировки слиянием**

```
1   size_t RandRange(size_t start, size_t end) {
2       return rand() % (end - start) + start;
3   }
4
5   void QuickSort(int *array, size_t len) {
6       Swap(array, 0, RandRange(1, len));
7
8       size_t pivot = 0;
9       for (size_t i = 1; i < len; ++i)
10          if (array[i] < array[0]) Swap(array, ++pivot, i);
11      Swap(array, 0, pivot);
12
13      if (pivot > 1)
14          QuickSort(array, pivot);
15      if ((pivot + 2) < len)
16          QuickSort(array + pivot + 1, len - pivot - 1);
17  }
```

**Листинг 5 – Алгоритм быстрой сортировки**

| 38 | 27 | 43 | 3 | 9 | 82 | 10 |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| 38 | 27 | 43 | 3 | 9 | 82 | 10 |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| 38 | 27 | 43 | 3 | 9 | 82 | 10 |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| 38 | 27 | 43 | 3 | 9 | 82 | 10 |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |