№3

```
 Array size = 10000
Minimal element = 3 has index = 8130
Maximal element = 24995 has index = 6149
Sorting time: 2991 microseconds
Minimal element = 3 has index = 0
Maximal element = 24995 has index = 9999
```

Увеличиваем длину массива:

```
 Array size = 100000
Minimal element = 1 has index = 8842
Maximal element = 25000 has index = 1292
Sorting time: 13116 microseconds
Minimal element = 1 has index = 0
Maximal element = 25000 has index = 99996
```

```
 Array size = 500000
Minimal element = 1 has index = 59001
Maximal element = 25000 has index = 88616
Sorting time: 65910 microseconds
Minimal element = 1 has index = 0
Maximal element = 25000 has index = 499986
```

```
 Array size = 1000000
Minimal element = 1 has index = 16429
Maximal element = 25000 has index = 1417
Sorting time: 155220 microseconds
Minimal element = 1 has index = 0
Maximal element = 25000 has index = 999959
```

Следует отметить, что если длина массива значительно больше длины диапазона значений его элементов, то элементы будут повторяться и индекс первого вхождения наибольшего элемента не обязательно будет последним.

№4

```
 Comparing for array size = 1000000
FOR duration: 838103 microseconds
CILK_FOR duration: 617843 microseconds

Comparing for array size = 100000
FOR duration: 87045 microseconds
CILK_FOR duration: 59175 microseconds

Comparing for array size = 10000
FOR duration: 8910 microseconds
CILK_FOR duration: 5924 microseconds

Comparing for array size = 1000
FOR duration: 885 microseconds
CILK_FOR duration: 665 microseconds

Comparing for array size = 500
FOR duration: 442 microseconds
CILK_FOR duration: 446 microseconds

Comparing for array size = 100
FOR duration: 113 microseconds
CILK_FOR duration: 171 microseconds

Comparing for array size = 50
FOR duration: 105 microseconds
CILK_FOR duration: 112 microseconds

Comparing for array size = 10
FOR duration: 40 microseconds
CILK_FOR duration: 134 microseconds
```

№5

При малых длинах массива эффективность `cilk_for` относительно обычного `for` снижается, так как для запуска параллельной программы cilk взаимодействует с ОС для создания необходимых потоков/процессов. `cilk_for` целесообразно использовать для ресурсоемких задач на аппаратном обеспечении, позволяющем выполнять несколько потоков одновременно.

`cilk_for` более эффективен, когда необходимо параллелизовать большое количество операций. `cilk_spawn` и `cilk_sync` дают разработчику больше контроля над синхронизацией потоков и не требуют оформления операций в виде цикла, что более удобно для единичных ветвлений.