

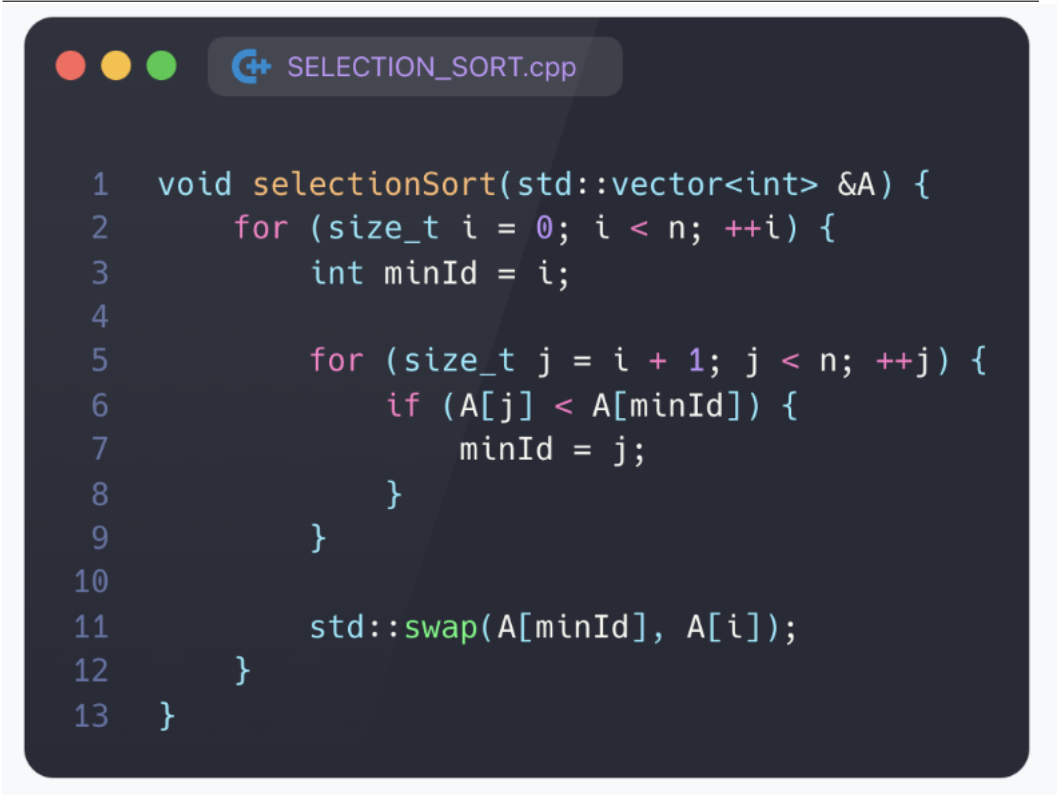
# АиСД | SET-1 | А1

Потякин Арсений

---

## TODO:

1. Сформулируйте условие P1 , которое подходит в качестве инварианта внутреннего цикла алгоритма по j. Представьте краткое обоснование (например, с использованием частичной трассировки выполнения цикла).
2. Сформулируйте условие P2, которое подходит в качестве инварианта внешнего цикла алгоритма по i. Представьте краткое обоснование.
3. Выполните проверку выполнения найденных инвариантов P1 и P2 до входа в каждый из циклов (INIT), во время итерации циклов (MNT), при выходе из цикла (TRM).



```
1 void selectionSort(std::vector<int> &A) {
2     for (size_t i = 0; i < n; ++i) {
3         int minId = i;
4
5         for (size_t j = i + 1; j < n; ++j) {
6             if (A[j] < A[minId]) {
7                 minId = j;
8             }
9         }
10
11         std::swap(A[minId], A[i]);
12     }
13 }
```

### Task 1: (Инвариант внутреннего цикла по j)

$\text{minId}$  - индекс текущего элемента, на место которого мы хотим поставить наименьшее число среди тех, что находятся правее этого индекса. Частичная трассировка: Будем считать, что число с индексом  $i$  минимальнее всех. Цикл идет от  $i + 1$  до  $n - 1$  включительно, на каждой итерации сравнивая элемент с фиксированным индексом  $i$  с элементом с индексом  $j$  правее него. В случае, если элемент с индексом  $j$  меньше, чем элемент с индексом  $i$ , мы меняем значение  $\text{minId}$  на индекс  $j$ , то есть находим минимальное число между  $i$  и  $j$ , где  $j \in [i + 1, i + 2, \dots, n - 1]$ . В таком случае, инвариантом можно считать тот факт, что  $\text{minId}$  хранит минимальное значение среди всех элементов с индексом.  $\in [i, j]$

### Task 2: (Инвариант внешнего цикла по i)

Из пункта 1 нам известно, что внутренний цикл находит минимальное значение среди всех элементов с индексами  $\in [i, j] \Rightarrow [i, n - 1]$ , это значит, что за полный внутренний цикл в переменной  $\text{minId}$  будет храниться индекс минимального числа среди чисел, начиная с индекса  $i$ . А значит мы можем обменять значения элемента с индексом  $i$  и элемента с индексом  $\text{minId} \Rightarrow$  за каждую итерацию внешнего цикла на место элемента с индексом  $i$  будет становиться наименьшее число из тех, чей индекс  $\geq i$ . Это и будем считать за инвариант.

### Task 3.1: (Проверка инварианта P1)

**INIT:**  $\text{minId} = \min(i, j - 1) \iff \min(i, i)$ , то есть индекс некоторого числа, которое является наименьшим на  $[i, i]$ .

**MNT:** Каждая итерация увеличивает диапазон поиска наименьшего числа на 1. То есть через одну итерацию в  $\text{minId}$  будет храниться минимальное число из диапазона  $[i, i + 1]$ , через две итерации будет храниться минимальное число из диапазона  $[i, i + 2]$  и т.д., пока не найдем минимальное число в диапазоне  $[i, n - 1]$ .

**TRM:** После выполнения всех итераций в  $\text{minId}$  будет храниться индекс наименьшего числа из диапазона  $[i, n - 1]$

$\Rightarrow$  P1 выполняется

### Task 3.2: (Проверка инварианта P2)

**INIT:** Так как перед началом цикла мы не передвинули в отсортированную часть ни одного элемента, то можем считать, что множество отсортированных чисел (чей индекс  $< i$ ) пустое, а значит множество отсортировано.

**MNT:** Каждую итерацию в множество отсортированных чисел прибавляется один элемент, при этом множество остается отсортированным.

**TRM:** Размер множества отсортированных чисел равен размеру исходного множества чисел, при этом конечное множество отсортировано.  
 $\implies$  P2 выполняется