```
FindMinInd(A[1..n] : array; ℓ : index in A)
@1   m ← ℓ
@2   for j = ℓ + 1 to n do
@3      if A[j] < A[m] then
@4         m ← j
@5   done
@6   return m
```

Твърдение 0.1. *При всеки вход от масив A от n числа и индекс $\ell \leq n$, процедурата $FindMinInd$ завършва след като извършва не повече от $3(n-\ell)+3$ операции и поне $n-\ell+1$ операции.*

Доказателство. Ред @1 се изпълнява веднъж. След това, контролният ред за цикъла на ред @2 се изпълнява $(n-\ell)+1$ пъти. При първите $(n-\ell)$ от тях се изпълнява и тялото на цикъла, тоест още ред @3 и евентуално ред @4. Поради това цикълът завършва, като се изпълняват не повече от $3(n-\ell)+1$ операции за това и поне $(n-\ell)+1$. Накрая има още една операция от ред @6. □

Следствие 0.1. *Нека $k = n-\ell+1$ е броят на елементите в подмасива $A[\ell..n]$ при изпълнението на процедурата $FindMaxInd$ с аргументи $A[1..n]$ и ℓ. $Time(n,\ell) = Time(k) \in \Theta(k)$ и $Time(n,\ell) \in O(n)$.*

Доказателство. Видяхме, че $Time(k) = Time(n,\ell) \leq 3(n-\ell)+3 = 3k$. Следователно $\frac{Time(k)}{k} \leq 3$. От друга страна $Time(k) \geq (n-\ell)+1 = k+1 \geq 1.k$. Следователно $1.k \leq Time(k) \leq 3.k$ за всяко k, което означава, че $Time(k) \in \Theta(k)$.

Тъй като $k \leq n$, то $Time(n,\ell) \leq 3n+3 \leq 4n$ за $n > 3$. Следователно $Time(n,\ell) \in O(n)$. □

Твърдение 0.2. *При всеки вход на масив A от n числа и индекс $\ell \leq n$, процедурата $FindMinInd$ връща индекс m със следните свойства:*

1. *$A[m] = \min A[\ell..n]$,*
2. *ако $m' \in \{\ell,\ldots,n\}$ и $A[m'] = \min A[\ell..n]$, то $m \leq m$.*

Доказателство. Нека m_j е стойността на параметъра m непосредствено преди проверката за край на цикъл с брояч $j+1$ на ред @2. Тогава $m_\ell = \ell$ и резултатът от процедурата е $m = m_n$. Ще покажем следния *инвариант*:

$$I(j): m_j \in \{\ell,\ldots,j\} \text{ и } A[m_j] = \min A[\ell..j] \text{ и ако } m' \in \{\ell,\ldots,m_j-1\}, \text{ то } A[m'] > A[m_j].$$

Доказателството извършваме с индукция по j.

- $j = \ell$. Тогава $m_\ell = \ell$ и това е единственият елемент в множеството $\{j,\ldots,\ell\}$. Поради това първата част е ясна. Втората също е очевидна, защото

$$\{\ell,\ldots,m_j-1\} = \{\ell,\ldots,\ell-1\} = \emptyset.$$

- Нека $I(j)$ е истина и да допуснем, че $j+1 \leq n$. Тогава се изпълнява тялото на for-цикъла с аргумент брояч $j+1$. При това се променя параметърът m. По определение, непосредствено преди изпълнението на тялото на цикъла, $m = m_j$, а непосредствено след това $m = m_{j+1}$. Така че, разликата между m_j и m_{j+1} се дължи единствено на изпълнението на тялото на цикъла върху m_j, като:

$$m_{j+1} = \begin{cases} j+1, \text{ ако } A[j+1] < A[m_j] \\ m_j, \text{ иначе} \end{cases}.$$

  Сега, ако $m_{j+1} = j+1 \in \{\ell,\ldots,j+1\}$, то $A[m_{j+1}] < A[m_j] \overset{I(j)}{=} \min A[\ell..j]$ и следователно, ако $m' \in \{\ell,\ldots,m_{j+1}-1\} = \{\ell,\ldots,j\}$, то $A[m'] > A[m_{j+1}]$. От друга страна $A[m_{j+1}] = A[j+1]$ и от предишното разсъждение следва, че $A[m_{j+1}] = \min A[\ell..j+1]$.

  Ако пък $m_{j+1} = m_j$, то $A[j+1] \geq A[m_j]$ и следователно

$$A[m_j] = \min(A[m_j], A[j+1]) = \min(\min A[\ell..j], A[j+1]) = \min A[\ell..j+1].$$

□

Теорема 0.1. *Нека P е произволна (детерминирана) процедура, която получава на вход масив $A[1..n]$ от числа и връща индекс m, за който $A[m] = \min A[1..n]$. Ако единствените операции, в които участват елементите на масива $A[1..n]$ са сравнения $\{<,\leq,=,>,\geq\}$, то броят на тези сравнения е поне $n-1$.*

Доказателство. Тъй като единствените операции, които се извършват от P върху масива $A[1..n]$ са сравнения, то P зависи единствено от релацията $R = \{(i,j) \mid A[i] < A[j]\}$, Да разгледаме граф $G_k = (V, E_k)$, чиито върхове са $V = \{1, \ldots, n\}$, а E_k са множествата $\{i, j\}$, за които едно от първите k сравнения, направени от P, е било сравнение $A[i] \sim A[j]$ или $A[j] \sim A[i]$ за някое $\sim \in \{<, \leq, =, >, \geq\}$. Да допуснем, че общият брой сравнения, направени от P, е $N < n - 1$. Тогава графът $G_N = (V, E_N)$ не е свързан. Следователно, ако резултатът от P върху $A[1..n]$ е m, то има елемент $m' \in \{1, \ldots, n\}$, който е недостижим от m в графа G. Нека $A'[1..n]$ е масивът, за който:

$$A'[i] = \begin{cases} A[i] \text{ ако } m \to^*_{G_N} i \\ A[i] - A[m'] + A[m] - 1, \text{ иначе.} \end{cases}$$

Тогава резултатите от сравненията, които прави P върху A и A' ще са едни и същи. Наистина, тъй като сравненията не зависят от транслация, единствената разлика може да бъде ако се сравняват $A'[i]$ и $A'[j]$, за които $m \to^*_{G_N} i$ и $m \not\to^*_{G_N} j$. Това, обаче не се случва при вход A и следователно (по индукция) няма да се случи и при вход A'. Следователно P ще върне резултат m и при вход A'. Но $A'[m'] = A[m'] - A[m'] + A[m] - 1 = A[m] - 1 < A[m]$. Това е противоречие. Следователно, върху всеки вход от n елемента, P извършва поне $n - 1$ операции за сравнение. □

```
SelectionSort(A[1..n]: array)
@1   for ℓ = 1 to n do
@2       m ← FindMinInd(A, ℓ)
@4       swap(A, m, ℓ)
@5   done
```

Твърдение 0.3. *При вход $A[1..n]$ от n числа, процедурата SelectionSort променя A до негова пермутация A', която е сортирана във възходящ ред: Нещо повече, времевата сложност на SelectionSort е $\Theta(n^2)$.*

Доказателство. От предишната лема знаем, че ред @2 отнема $\Theta(n - \ell)$ време, докато редовете @3 и @4 отнемат време $\Theta(1)$. Следователно, ℓ-тата итерация на for-цикъла отнема:

$$T(n, \ell) \in \Theta(n - \ell) \text{ време, по-точно } (n - \ell) + 1 \leq T(n, \ell) \leq 3(n - \ell) + 6.$$

Следователно, времето необходимо за изпълнението на цялата процедура е:

$$2 + \sum_{\ell=1}^{n} T(n, \ell) \in \Theta\left(2 + \sum_{\ell=1}^{n} (n - \ell)\right) = \Theta\left(2 + \sum_{\ell=0}^{n-1} \ell\right) = \Theta\left(2 + \binom{n-1}{2}\right) = \Theta\left(n^2\right).$$

С това показахме, че в частност процедурата завършва. Сега ще покажем коректността. Нека A'_ℓ и π_ℓ са масивите A и $\pi[1..\ell]$ непосредствено преди $(\ell + 1)$-та итерация на for-цикъла. Тъй като единствените операции, които се извършват на масива A са *swap*, то A'_ℓ е пермутация на елементите на A. Нещо на ред @2 се намира най-малкият елемент $A[m]$ на масива $A'_\ell[\ell..n]$. Следователно, след размяната, $A'_{\ell+1}[\ell] \leq \min A'_{\ell+1}[\ell + 1..n]$. Оттук, по индукция, получаваме, че елементите $A'_\ell[1..\ell]$ са сортирани във възходящ ред. При $\ell = n$, получаваме, че $A' = A'_n$ е сортиран във възходящ ред и тъй като A' представлява пермутация на елементите на A, то SelectionSort наистина сортира всеки свой вход A. □