Takie Jasia porządki

#selection_sort
#min_element #swap

Jak postąpił mały Jaś, gdy mama poleciła mu ustawienie zabawek od najmniejszej do największej? Najpierw Jaś ustawił swoje zabawki w przypadkowej kolejności na półce, a potem wziął się za ich porządkowanie. Spośród wszystkich zabawek Jaś wybrał najmniejszą z nich (w jego przypadku był to gwizdek – pamiątka po zeszłorocznym Mundialu):

– Mame! To będzie na początku! – oznajmił tryumfalnie Jaś.

– Ale na początku półki stoi magiczna butelka po soku z gumijagód! – odparła mama.

– To weź i zamień miejscami gwizdek i butelkę! – polecił Jaś.

– No dobrze, Jasiu – odparła mama. – A która zabawka będzie następna?

Jaś dumał i dumał nad pozostałymi zabawkami (z pominięciem gwizdka), aż wskazał z nich najmniejszą (piłkę tenisową z wimbledońskiego turnieju) i zaprezentował mamie:

– Ta może być? – spytał Jaś. – Zrób podobny myk z zamianą miejsc.

– Może, może, synku – rzekła mama kładąc piłkę obok gwizdka, a stojącą tam encyklopedię przełożyła na pozycję do tej pory zajmowaną przez piłkę. – A teraz?

– Teraz będzie kolejna najmniejsza! – Jaś zlustrował rząd pozostałych zabawek i wybrał najmniejszą z nich (pluszowego jeżyka).

– Zuch chłopak! – pochwaliła mama kładąc pluszaka jako trzecią w kolejności, a leżący tam kij bejsbolowy (pamiątka z burzliwych czasów przedszkolnych) powędrował na dotychczasowe miejsce jeżyka. – Tak trzymaj dalej!

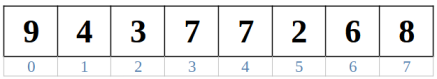
Jaś kontynuował porządki w analogiczny sposób, aż w końcu udało mu się ustawić wszystkie zabawki w szeregu.*

Taki – jakże intuicyjny – sposób porządkowania, nosi nazwę *sortowania przez wybór* (ang. **selection sort**, wymawiaj: *selekszyn sort*). Jak widzieliśmy, na każdym kroku tej metody wybieraliśmy najmniejszy element z nieposortowanej części zbioru i ustawialiśmy go na kolejnej pozycji.

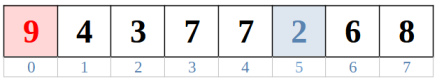
Zauważmy, że Jaś i jego mama całe sortowanie przeprowadzili na półce, bez wykorzystania dodatkowego miejsca na przykład na stole czy podłodze, a zatem wszystko przebiegło według zasady *in situ* (o tym było w poprzednim podrozdziale).

*Jako ostatni dumnie prezentował się drewniany M1 Abrams – replika w skali 1:2.

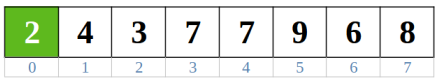
Rolę zbioru zabawek będzie u nas spełniać niewielka tablica liczb całkowitych (oznaczona później przez $A[\,]$, a jej rozmiar – przez N):



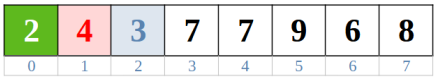
W pierwszym kroku algorytmu szukamy elementu najmniejszego w całej tablicy. Tak wyszło, że jest to element o wartości 2 znajdujacy się na pozycji 5 (oznaczymy go kolorem niebieskim). Ten element (gwizdek!) powinien znaleźć się na początku szeregu (pozycja 0), a tymczasem miejsce to zajmuje element o wartości 9 (oznaczmy ten element kolorem czerwonym):



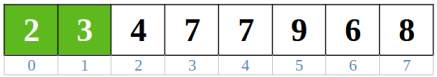
Nie ma rady, robimy *machniom!* i zamieniamy pokolorowane elementy miejscami. Element o wartości 2 (gwizdek) stanowi zalążek posortowanej części tablicy, którą będziemy kolorować na zielono:



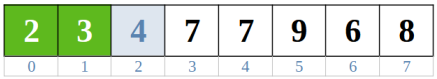
Kolejnej zabawki poszukujemy w odcinku tablicy od pozycji 1 do końca tablicy i znajdujemy ją na pozycji 2 – element o wartości 3, który wypadnie zamienić z zabawką na początku (o wartości 4):



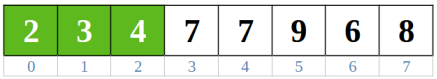
Mamy zatem już dwa elementy ustawione na fest:



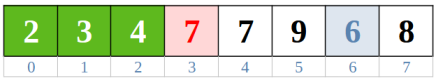
Teraz szukamy następnej najmniejszej zabawki i okazuje się nią element o wartości 4:



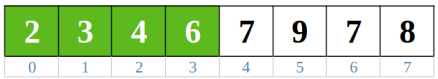
Ta zabawka stoi od razu na dobrym miejscu, więc tylko zatwierdzamy jej wybór:



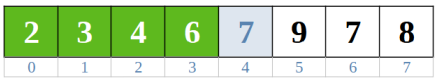
Z pozostałych pięciu zabawek najmniejsza ma wartość 6 i stoi również na pozycji 6:



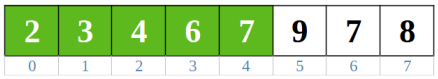
Po przestawieniu z siódemką ląduje ona na pozycji 3:



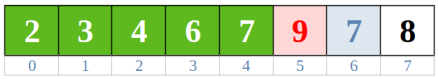
Przeszukujemy pozostałe cztery zabawki i znajdujemy najmniejszą siódemkę...



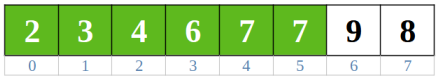
...która stoi od razu na dobrym miejscu:



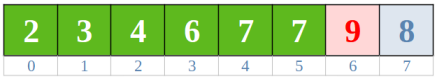
Z pozostałych trzech zabawek najmniejsza jest również siódemka:



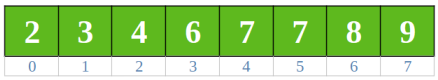
Przestawiamy ją z dziewiątką i zatwierdzamy wybór:



Zostały jeszcze dwie zabawki, a z nich mniejsza jest ósemka:



Dokonujemy ostatniego przestawienia i zatwierdzamy ostateczną kolejność na półce:



Zabawki ułożone na półce, Jaś i jego mama zadowoleni, a my możemy zająć się implementacją algorytmu w postaci programu. Jak w poprzednim przykładzie napiszemy funkcję, której argumentami będzie tablica $A[\,]$ i jej rozmiar N:

```
void selection_sort(int A[], int N)
{
  . . .
}
```

Główna pętla przebiegnie po indeksie elementu (i), od którego zaczyna się nieposortowana część tablicy, a więc od 0 do $N-2$ (ostatni odcinek obejmuje dwa elementy). Na każdym kroku wyszukujemy położenie minimalnego elementu w nieposortowanym odcinku (i_{min}) i zamieniamy go z elementem na pozycji i (jeśli jest taka potrzeba):

```
void selection_sort(int A[], int N)
{
  for(int i{ }; i <= N - 2; i++)
  {
    int i_min = min_element(A + i, A + N) - A;
    if(i_min != i)
      swap(A[i_min], A[i]);
  }
}
```

W powyższym kodzie pojawiła się funkcja `min_element()`, która wyszukuje wskaźnik do najmniejszego elementu tablicy w podanym zakresie: począwszy od $A[i]$ (reprezentowanego przez wskaźnik $A+i$) aż do końca tablicy. Rezultatem tej funkcji jest wskaźnik do najmniejszego elementu – jeśli odejmiemy od niego wskaźnik do początku tablicy, czyli A, wtedy otrzymamy indeks tego elementu (liczbę naturalną).[†]

W celu zamiany położenia dwóch elementów tablicy posłużyliśmy się standardową funkcją `swap()`, która zamienia wartości swoich dwóch argumentów.[‡] Zamiast tego można by oczywiście napisać sekwencję instrukcji w rodzaju:[§]

```
int x = A[i_min];
A[i_min] = A[i];
A[i] = x;
```

Program sprawdzający działanie procedury sortującej jest analogiczny do programu przytoczonego w poprzednim podrozdziale.

Dodajmy na koniec jedną uwagę dotyczącą stabilności sortowania przez wybór. Proszę prześledzić kolejność elementów o wartości 7: w trakcie sortowania zamieniły swoje wzajemne pozycje, a zatem ten schemat sortowania nie ma cechy stabilności (i ma kwadratową złożoność obliczeniową – dlaczego?), co oczywiście nie przekreśla jego użyteczności w pewnych sytuacjach.

[†]Odsyłamy Czytelnika do podrozdziału o wskaźnikach i iteratorach, gdzie można znaleźć pełniejszy opis arytmetyki wskaźników. Mamy nadzieję, że sformułowanie powyższego kodu dla kontenerów z użyciem iteratorów (`begin()`, `end()`, te sprawy) nie będzie trudne dla Czytelnika.

[‡]Funkcja ta działa dla dowolnego (byle zgodnego) typu argumentów, nawet dla złożonych obiektów czy wskaźników. Zauważmy, że w wywołaniu funkcji podajemy konkretne elementy tablicy, a nie same indeksy.

[§]Istnieją też inne, bardziej trickowe sposoby zamiany wartości zmiennych – niektóre nawet nie wymagają pomocniczej zmiennej.