Zadanie IGR: rozwiązanie

(C++)

Olimpiada
Informatyczna
Juniorów

Zacznijmy od typowej struktury programu w C++:

```
#include<iostream>   ← tego potrzebujemy do czytania/pisania
using namespace std; ← żeby nie musieć pisać std::cin, std:cout, itd.

int main()
{

     ← tu będzie nasz program

}
```

W tym zadaniu podana jest najpierw liczba zawodników n, a potem ich kolejne wyniki. Wiemy, że zawodników nie będzie niż 1000. Zadeklarujmy więc zmienną typu `int` na liczbę zawodników, a także tablicę rozmiaru 1000, do której wpiszemy wyniki:

```
int n;
int wyniki[1000];
```

Teraz wczytamy dane z wejścia – najpierw jest liczba zawodników, z którą jest prosto:

```
cin >> n;
```

Aby wczytać wyniki, trzeba użyć pętli `for`, która zapisze kolejne wyniki do komórek tablicy `wyniki[0]`, `wyniki[1]`, `wyniki[2]`, ..., `wyniki[n-1]`.

```
for(int j=0; j<n; j++)
    cin >> wyniki[j];
```

Zauważmy, że zawodników może być dużo mniej niż 1000 – część tablicy pozostanie po prostu niewykorzystana, czym nie musimy się martwić.
Ale czy musimy się „zabezpieczać" na wypadek $n > 1000$? Nie musimy. System testujący nasz program zawsze podaje takie dane, jakie są obiecane w zadaniu. Nie ma potrzeby go sprawdzać (pisząc, na przykład, `if (n>1000) { ... }`).

Inaczej byłoby, gdybyśmy pisali aplikację biznesową dla klientów – wtedy trzeba się przygotowywać na to, że użytkownicy wpisują nieprawidłowe dane. Ale to opowieść na inną okazję – na Olimpiadzie po prostu wierzymy, że dane są takie, jakie miały być.

Mając już wczytane wyniki, zacznijmy od znalezienia mistrza – zawodnika, który ma najwięcej punktów. Pierwszego zawodnika w tablicy (czyli z komórki `0`) robimy „kandydatem", a następnie przechodzimy przez wszystkich pozostałych zawodników. Jeśli któryś ma więcej punktów niż aktualny kandydat, sam staje się kandydatem.

```
int kandydat = 0;
for(int j = 1; j < n; j++)
    if (wyniki[j] > wyniki[kandydat])
        kandydat = j;
```

Po zakończeniu tej pętli, w zmiennej `kandydat` znajdzie się numer zawodnika, który ma najwięcej punktów.

Jak teraz znaleźć drugiego w kolejności zawodnika? Jest kilka sposobów, opiszemy jeden z najprostszych. Zacznijmy od przestawienia znalezionego mistrza na pozycję 0:

```
swap(wyniki[kandydat], wyniki[0]);
```

Funkcja swap służy właśnie do tego, czego nam trzeba – zamienia miejscami dwie zmienne. Aby móc jej używać, trzeba na początku programu dopisać jeszcze:

```
#include<algorithm>
```

Teraz będziemy szukać wicemistrza, dokładnie w ten sam sposób, jak przed chwilą. Skoro już jednak najlepszy wynik jest w komórce `wyniki[0]`, to teraz będziemy ją ignorować żeby nie znaleźć najlepszego wyniku, tylko następny w kolejności. Innymi słowy, zaczniemy od kandydata na pozycji `wyniki[1]` i będziemy szukać w reszcie tablicy:

```
kandydat = 1;
for(int j = 2; j < n; j++)
    if (wyniki[j] > wyniki[kandydat])
        kandydat = j;
```

Teraz przestawiamy wicemistrza na pozycję 1 i ten sam algorytm wykonujemy po raz trzeci:

```
swap(wyniki[kandydat], wyniki[1]);

kandydat = 2;
for(int j = 3; j < n; j++)
    if (wyniki[j] > wyniki[kandydat])
        kandydat = j;
```

Na koniec dla porządku przestawimy brązowego medalistę na pozycję 2:

```
swap(wyniki[kandydat], wyniki[2]);
```

Teraz najlepsi zawodnicy są w komórkach `wyniki[0]`, `wyniki[1]` i `wyniki[2]`. Mamy więc już działający algorytm, ale nie jest zbyt ładny – trzy razy powtarzamy prawie taki sam kod, co nie jest dobrą praktyką. W ogólności dobrze jest unikać *copy-paste*, bo jest to proszenie się o kłopoty.

Na szczęście tutaj łatwo pozbędziemy się kopiowania: nasze poszukiwanie kandydata startuje raz od 0, raz od 1 i raz od 2. Zróbmy więc po prostu pętlę, która wykona się trzy razy:

```
for(int s = 0; s <= 2; s++)
{
    int kandydat = s;
    for(int j = s+1; j < n; j++)
        if (wyniki[j] > wyniki[kandydat])
            kandydat = j;
    swap(wyniki[kandydat], wyniki[s]);
}
```

Gdybyśmy powyższą pętlę rozpisali na kolejne wykonania (z $s = 0$, $s = 1$ i $s = 2$), otrzymamy poprzednią wersję programu. Ta jest jednak znacznie krótsza i łatwiejsza w czytaniu.

Pozostaje wypisać wyniki i program jest gotowy:

```
#include <iostream>
#include <algorithm>

using namespace std;

int main()
{
    int n;
    int wyniki[1000];
    cin >> n;

    for(int i = 0; i < n; i++)
        cin >> wyniki[i];

    for(int s = 0; s < 3; s++)
    {
        int kandydat = s;
        for(int i = s+1; i < n; i++)
            if (wyniki[i] > wyniki[kandydat])
                kandydat = i;

        swap(wyniki[kandydat], wyniki[s]);
        cout << wyniki[s] << '\n';
    }
}
```

Nie jest to, oczywiście, jedyny sposób na rozwiązanie zadania – można napisać, na przykład, bardzo krótki program używając funkcji `sort`, która porządkuje całą tablicę w kolejności rosnącej. Warto o niej doczytać, bo może być przydatna w dalszych etapach Olimpiady, ale w tym zadaniu obeszliśmy się bez niej.