

Plan lekcji



#nic_szczególnego

Proponujemy kolejne łatwe zadanie z konkursu PIWO 2018/19 – *Plan lekcji*. Tekst zadania zamieszczony jest [tutaj](#). Oryginalny pomysł zadania pochodzi z serwisu Codeforces (zadanie 586A).

Nasza bohaterka Halinka jest studentką i dostała rozpiszkę zajęć w ciągu dnia. Każda zajęta godzina jest oznaczona jedyneką, a każda wolna – zerem. Podczas wolnych godzin dziewczyna oczywiście nie musi przebywać na uczelni, ale jeśli jest to pojedyncza godzina, taka że przed nią są zajęcia i po niej też, to nie opłaca się jej wracać do domu i zostaje na uczelni. Na przykład w poniższym przykładzie podkreślone godziny Halinka spędza na uczelni (mimo, że nie ma wtedy zajęć):

1 0 1 1 1 0 0 1 0 1 0

Pytanie brzmi: ile łącznie godzin spędzi Halinka na uczelni?

Dane wejściowe stanowią: n – ilość godzin opisanych w planie zajęć (nie większa od 100), oraz lista zer i jedynek a_1, a_2, \dots, a_n , określających, czy Halinka ma na danej godzinie zajęcia.

Rozwiązanie zadania rozpoczynamy od wczytania danych:

```
int n;
cin >> n;
int a[101];
for(int i = 1; i <= n; i++)
    cin >> a[i];
```

Proszę zwrócić uwagę na rozmiar tablicy a [:]: deklarujemy rozmiar na maksymalny, określony w zadaniu. Nie ma żadnej premii za oszczędzanie pamięci (zwłaszcza przy tak mikrych strukturach danych) – oczywiście dotyczy to zadań programistycznych na serwisach edukacyjnych oraz na zawodach algorytmicznych. Wystarczy tylko zmieścić się w podanym limicie pamięci, który z reguły wynosi grube megabajty.

Dlaczego jednak 101, a nie 100, jak w zadaniu? Elementy tablicy numerowane są od zera, zatem, jeśli chcemy mieć komórkę o numerze 100 (dla maksymalnych danych), wtedy powinniśmy zadeklarować tablicę o jedną komórkę większą, a komórki o numerze 0 nie będziemy używać. Tym samym staramy się przyzwyczaić Czytelnika, że jeśli dane w zadaniu są numerowane od 1, to przechowująca je struktura powinna być o jeden element większa, niż wskazuje na to rozmiar danych (wspominaliśmy już o tym w podrozdziale *Drużyna, w szeregu zbiórka!*)*.

Teraz dokonamy pewnej modyfikacji danych w tablicy: przespacerujemy się po niej i jeśli napotkamy samotne zero otoczone jedynekami, wtedy zamieniamy go na jedynekę. Należy tylko

*Ktoś mógłby powiedzieć, że można po prostu zapisać dane z indeksami o jeden mniejszymi, ale jest to prośenie się o kłopoty, bo łatwo zapomnieć, że oryginalny indeks uległ zmianie.

uważać, aby sprawdzać tylko te pozycje w tablicy, które mają sąsiadów z obydwu stron, zatem dotyczy to elementów o indeksach od 2 do $n - 1$:

```
for(int i = 2; i <= n - 1; i++)
    if(a[i - 1] == 1 && a[i + 1] == 1)
        a[i] = 1;
```

Zaraz, zaraz, przecież tu brakuje sprawdzenia, czy $a[i]$ jest zerem! Ale czy to jest potrzebne? Nie, bo jeśli $a[i]$ było akurat jedyneką, to pozostanie jedyneką i krzywdą mu się nie stanie.

I co teraz? Teraz wystarczy zsumować zawartość tablicy $a[]$ i to będzie wynik zadania:

```
int sum = 0;
for(int i = 1; i <= n; i++)
    sum += a[i];
```

Operator `+=` oznacza powiększenie lewej strony o wartość wyrażenia z prawej strony. Oczywiście tym razem sumujemy od pierwszego elementu do ostatniego.

Oto kompletny program (na końcu dodaliśmy instrukcję wypisującą wynik na ekranie):

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int a[101];
    for(int i = 1; i <= n; i++)
        cin >> a[i];
    for(int i = 2; i <= n - 1; i++)
        if(a[i - 1] == 1 && a[i + 1] == 1)
            a[i] = 1;
    int sum = 0;
    for(int i = 1; i <= n; i++)
        sum += a[i];
    cout << sum << endl;
}
```

Niech Czytelnik przetestuje ten program na danych z treści zadania, a potem niech go wyśle na konkurs PIWO 2018/19, aby tam dokonać ostatecznego sprawdzenia jego poprawności.