

Scenariusz zajęć nr 4

Temat: Taneczny palindrom

Cele:

W trakcie zajęć uczeń zapozna się z następującymi pojęciami:

- Podstawy użytkowania środowiska programistycznego (IDE) dla języka C++ (Code::Blocks lub Dev-Cpp).
- Wczytywanie danych liczbowych (`cin >>`),
- Wypisywanie danych liczbowych (`cout << ... << endl`),
- Deklarowanie zmiennych o typie całkowitym (`int`), • Deklarowanie i używanie tablicy o wartościach całkowitych,
- Używanie instrukcji pętli `for`,
- Używanie instrukcji warunkowej `if ...`,
- Używanie warunków logicznych z operatorami porównania `<=`, `==` oraz `!=`,
- Używanie operatorów logicznych koniunkcji `&&` i alternatywy `||`,
- Używanie operatora przypisania w formie `+=`,
- Używanie funkcji bibliotecznej `min()`,
- Używanie instrukcji `return 0` w celu natychmiastowego zakończenia programu,
- Używanie instrukcji `continue` w pętli,
- Sprawdzanie parzystości zmiennej,
- Sprawdzanie, czy dana struktura wykazuje cechy palindromu.

Wstęp:

Uruchomienie środowiska programistycznego, zapoznanie się z edytorem kodu źródłowego i sposobem kompilacji programu (kontynuacja).

Dla nauczyciela:

W razie braku zainstalowanego środowiska można skorzystać z dowolnego środowiska dostępnego w sieci Internet, na przykład `cpp.sh`.

Przebieg zajęć:

Zapoznanie się z treścią zadania

Treść zadania jest dołączona do scenariusza zajęć w formie pliku `cpp_04-taneczny.pdf`.

Dla nauczyciela:

Tematem zajęć jest zadanie *Taneczny palindrom* dostępne na serwisie szkopul.edu.pl, wzorowane na zadaniu *Palindrome Dance* (1040A) z serwisu codeforces.com. Zadanie można rozwiązać w dowolnym języku programowania, a jego stopień trudności należy ocenić jako właściwy dla uczestników początkowego kursu programowania, którzy potrafią tworzyć

bardziej złożone konstrukcje logiczne.

Wskazówki do zadania

Wczytujemy informacje o kostiumach do tablicy k na pozycje $i = 1, 2, \dots, n$. Cenę do zapłacenia za zakupienie nowych kostiumów oznaczmy przez $price$ (na początku równa 0). Będziemy przeglądać komórki tej tablicy parami: $k[i]$ oraz $k[j]$, gdzie:

$$i = 1, 2, 3, \dots, j = n, n-1, n-2, \dots$$

Mamy tutaj kilka możliwości:

- $k[i] = 0, k[j] \neq 0$ – musimy dokupić taki kostium, jaki ma tancerz o numerze j , czyli $price$ powiększamy o jego cenę;
- $k[i] \neq 0, k[j] = 0$ – musimy dokupić taki kostium, jaki ma tancerz o numerze i , czyli $price$ powiększamy o jego cenę;
- $k[i] = k[j] \neq 0$ – nie trzeba nic kupować;
- $k[i] = k[j] = 0$ – dokupujemy dwie sztuki tańszego kostiumu;
- $k[i] \neq k[j]$, przy czym $k[i], k[j] \neq 0$ – kostiumów nie da się dobrać (piszemy „NIE” i kończymy program).

Sprawdzanie parami kontynuujemy dla $i < j$. Jeśli pętla obejdzie spokojnie do końca, wtedy dobranie kostiumów dla całego zespołu jest możliwe. Trzeba tylko uwzględnić sytuację, kiedy n jest nieparzyste i mamy dodatkową osobę na środku, czyli na pozycji $m = (n + 1)/2$. Jeśli $k[m] \neq 0$, nie trzeba nic dokupywać, w przeciwnym razie dokupujemy tańszy kostium.

Wynikiem programu jest wartość zmiennej $price$.

Kod przykładowego programu w C++

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

int main()
{
    int k[1001], n, a, b;
    cin >> n >> a >> b;
    for(int i = 1; i <= n; i++)
        cin >> k[i];
    int price = 0;
    if(n % 2 && k[(n + 1) / 2] == 0)
        price = min(a, b);
    int i = 1, j = n;
    while(i < j)
    {
        if(k[i] && k[j])
        {
```



Fundusze Europejskie
Polska Cyfrowa



Rzeczpospolita
Polska

Unia Europejska
Europejski Fundusz
Rozwoju Regionalnego



```
    if(k[i] != k[j])
    { cout << "NIE" << endl; return 0;}
    else continue;
}
if(k[i] == 0 && k[j] == 0)
    price += 2 * min(a, b);
else if(k[i] == 1 || k[j] == 1)
    price += a;
else
    price += b;
i++; j--;
}
cout << price << endl;
return 0;
}
```

Podsumowanie i dodatkowe uwagi:

Rozmiar tablicy $k[]$ wynosi o 1 więcej niż maksymalny rozmiar danych n . Dokładna dyskusja tego problemu znajduje się w scenariuszu do zajęć nr 2 z zadaniem *Plan lekcji*.

W kodzie programu wykorzystujemy fakt, że liczba 0 występująca jako wartość logiczna jest traktowana tak samo jak `false`, zaś liczba różna od zera jest traktowana jak `true`.

Nie jest konieczne używanie dwóch indeksów i oraz j w celu sprawdzenia, czy układ jest (może stać się) palindromem, gdyż dla danego i wartość j wynosi zawsze $n - i + 1$. Jednak w ten sposób łatwiej i przejrzystiej skonstruować kod wewnątrz pętli, no i unika się ewentualnego błędu w wyliczeniu wartości drugiego indeksu (łatwo zapomnieć o dodaniu 1).

Użycie instrukcji `return 0` pozwala na szybkie zakończenie działania programu (funkcji `main`), gdy już wiadomo, że nie uda się uzyskać palindromu.

Użycie instrukcji `continue` umożliwia zapisanie kodu wewnątrz pętli bez dodatkowego poziomu nawiasów klamrowych.