



## Zadanie 1

**Otwarto:** środa, 2 października 2024, 00:00

**Wymagane do:** niedziela, 20 października 2024, 23:59

# Klasyfikacja medalowa

## Wstęp

Biblioteka standardowa języka C++ udostępnia implementacje wielu struktur danych, takich jak `pair`, `tuple`, `array`, `vector`, `unordered_set`, `set`, `string`, `unordered_map`, `map`, `queue` itp., a także implementacje podstawowych algorytmów, np. `sort`, `lower_bound`, `upper_bound`, `max_element` itd. Celem pierwszego zadania zaliczeniowego jest przeciwiczenie korzystania z tej biblioteki. Studenci powinni:

- poznać podstawy korzystania z STL-a,
- uświadomić sobie konieczność weryfikacji poprawności danych wejściowych,
- nauczyć się podejmowania decyzji programistycznych,
- ugruntować w sobie konieczność testowania programu.

## Polecenie

Napisać program wyznaczający klasyfikację medalową krajów uczestniczących w igrzyskach olimpijskich. Program czyta dane ze standardowego wejścia. Poprawna linia z danymi wejściowymi ma jeden z następujących trzech formatów:

- informacja o zdobyciu przez kraj medalu,
- informacja o odebraniu krajowi medalu,
- prośba o wypisanie aktualnej klasyfikacji medalowej.

Informacja o zdobyciu przez kraj medalu składa się z nazwy kraju i rodzaju medalu oddzielonych pojedynczą spacją. Nazwa kraju zawiera tylko litery języka angielskiego, składa się z co najmniej dwóch liter, pierwsza litera jest wielka, wewnątrz nazwy mogą występować spacje. Rodzaj medalu jest to cyfra 1 (złoto), 2 (srebro) lub 3 (brąz). Jako rodzaj medalu może być też podana cyfra 0, co oznacza tylko dodanie kraju do rankingu, ale bez medalu. Kraj może być dodawany do rankingu wielokrotnie.

Informacja o odebraniu krajowi medalu to napis składający się ze znaku `-`, po którym następuje napis w formacie analogicznym jak dla informacji o zdobyciu medalu, przy czym rodzaj odbieranego medalu jest cyfrą dodatnią.

Prośba o wypisanie aktualnej klasyfikacji to napis składający się ze znaku `=`, po którym występują trzy liczby całkowite z przedziału od 1 do 999999. Te liczby oznaczają wagi, z jakimi brane są do porównania liczby medali odpowiednio złotych, srebrnych i brązowych. Liczby oddzielone są pojedynczą spacją. Liczby nie mają zer wiodących. Jeśli nie wprowadzono jeszcze informacji o żadnym medalu, to niczego nie wypisuje.

Linie wejściowe są rozdzielone znakiem nowej linii. Po ostatniej linii znak nowej linii jest opcjonalny. Poprawne dane wejściowe nie zawierają innych białych znaków niż wymienione powyżej.

Program wypisuje klasyfikację medalową na standardowe wyjście. Nazwę każdego kraju wypisuje się w osobnej linii, poprzedzając ją miejscem w klasyfikacji. W przypadku remisu nazwy krajów wypisuje się w kolejności leksykograficznej, podając to samo miejsce w klasyfikacji. Każda linia kończy się znakiem nowej linii.

## Obsługa błędów

Program na bieżąco sprawdza, czy dane wejściowe nie zawierają błędów. Za błędną uznaje się również informację o odebraniu medalu, którego nie ma. Dla każdej błędnej linii program wypisuje na standardowe wyjście diagnostyczne komunikat

ERROR L

?

gdzie `L` oznacza numer linii. Linie są numerowane od 1. Komunikat o błędzie kończy się znakiem nowej linii. Program ignoruje zawartość błędnych linii.

## Wymagania formalne

Program powinien kończyć się kodem 0.

Oczekiwane rozwiązanie nie powinno zawierać definicji własnych struktur i klas, a przynajmniej takich, które zawierają dane. Zamiast tego należy intensywnie korzystać z kontenerów i algorytmów dostarczanych przez standardową bibliotekę języka C++. Obsługę wejścia i wyjścia należy zrealizować za pomocą strumieni.

Rozwiązanie należy umieścić w pliku `medals.cpp`, który należy wstawić do Moodle. Rozwiązanie będzie kompilowane na maszynie students poleceniem

```
g++ -Wall -Wextra -O2 -std=c++20 medals.cpp -o medals
```

## Ocenianie rozwiązania

### Ocena automatyczna

Za testy automatyczne zostanie przyznana ocena z przedziału od 0 do 6 punktów. Za błędną nazwę pliku zostanie odjęty 1 punkt. Za ostrzeżenia wypisywane przez kompilator zostanie odjęty 1 punkt. Nie ma punktów ułamkowych.

### Ocena jakości kodu

Ocena jakości kodu jest z przedziału od 0 do 4 punktów. Nie ma punktów ułamkowych. Odejmujemy punkty za:

- brzydki [styl](#) (niepoprawne wstawianie spacji, wcięć, odstępów, brak komentarzy, magiczne stałe itd.);
- dłubanie własnych klas, struktur lub algorytmów zamiast użycia STL-owych;
- brak nazw typów, niewiele mówiące nazwy typów;
- rozwlekłą lub nieelegancką strukturę programu, rozpatrywanie zbyt wielu warunków brzegowych, powtarzanie kodu, nieefektywne korzystanie z klasy string, np. `if (str != "")` zamiast `if (!str.empty())`, trzymanie liczb jako napisów i używanie w każdym porównaniu komparatora konwertującego napis na liczbę itp.;
- korzystanie z wejścia-wyjścia dostarczanego przez bibliotekę C zamiast ze strumieni lub dłubanie własnego kodu zamiast użycia np. funkcji `getline`;
- potencjalne wycieki pamięci albo korzystanie z `new` i `delete`;
- zły dobór typów całkowitoliczbowych, np. używanie `int` lub `unsigned` zamiast `size_t` jako typu wartości zwracanej przez metody `size`, `length`, używanie `int` lub `long` zamiast `int32_t`, gdy potrzebujemy typu 32-bitowego.
- używanie typu zmiennoprzecinkowego;
- zły wybór kontenera, np. `map`, gdy wystarczyłoby `unordered_map`;
- wprowadzanie sztucznych ograniczeń na rozmiar danych;
- nieusuwanie lub nieefektywne usuwanie niepotrzebnych już danych;
- inne znalezione i niewymienione w powyższych kryteriach błędy.

Dodajemy jeden punkt za skorzystanie z `regex` do walidowania poprawności danych wejściowych. Przy czym ocena nie może być wyższa niż 4 pkt.

Ponadto:

- piętnujemy przekazywanie funkcjom dużych argumentów (np. typu `string`) przez wartość, takie obiekty przekazuje się przez stałą referencję;
- wytykamy nieukrywanie globalnych zmiennych, struktur, funkcji przed światem zewnętrznym za pomocą anonimowej przestrzeni nazw, choć studenci powinni wiedzieć, że można to też osiągnąć podobnie jak w języku C, czyli deklarując je jako `static`;
- sugerujemy stosowanie `using` zamiast `typedef`.

Na razie tylko wskazujemy te błędy i nie odejmujemy za nie punktów, bo są to zagadnienia pojawiające się w drugim zadaniu, w którym już będziemy za to karać.

## Przykład użycia

Przykład użycia jest częścią specyfikacji. Formatowanie wypisywanych rankingów powinno być dokładnie zgodne z formatowaniem w podanym przykładzie użycia. Przykład użycia znajduje się w załączonych w plikach:

 [test\\_1.err](#)


28 września 2024, 21:27

?

-  [test\\_1.in](#)
-  [test\\_1.out](#)
-  [zadanie1\\_testy.zip](#)

29 września 2024, 15:37  
28 września 2024, 21:27  
21 października 2024, 18:40

## Status przesłanego zadania

Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Ocenione
Pozostały czas	Zadanie zostało przesłane 4 dni 1 godzina przed terminem
Ostatnio modyfikowane	środa, 16 października 2024, 22:00
Przesyłane pliki	<div> <a href="#">medals.cpp</a> 16 października 2024, 22:00</div>
Komentarz do przesłanego zadania	<a href="#">▶ Komentarze (0)</a>


## Informacja zwrotna

Ocena	4,00 / 4,00
Ocenione dnia	poniedziałek, 28 października 2024, 13:11
Ocenione przez	JP    Jakub Pawlewicz

Skontaktuj się z nami



Obserwuj nas

 Skontaktuj się z pomocą techniczną

Jesteś zalogowany(a) jako [Krzysztof Hałubek](#) (Wyloguj)

[Podsumowanie zasad przechowywania danych](#)

[Pobierz aplikację mobilną](#)

[Pobierz aplikację mobilną](#)

?

Motyw został opracowany przez

conecti.me

Moodle, 4.1.16 (Build: 20250210) | [moodle@mimuw.edu.pl](mailto:moodle@mimuw.edu.pl)