
POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

# PJC Zadania 7

Rozwiązania należy przesłać w postaci pojedynczego pliku o rozszerzeniu `.cpp`

## Zadanie 1

Napisz funkcję `show_ordered_character_occurrences()`, która pobierze przez argument `std::string`a i wyświetli wszystkie jego znaki wraz z ich krotnościami, w kolejności malejącej względem tych krotności (czyli najpierw znaki występujące najczęściej, a potem najrzadziej). Każdy ze znaków winien być wyświetlony tylko raz.

## Zadanie 2

W funkcji `main()` stwórz mapę kojarzącą każdy otwierający nawias (okrągły, kwadratowy, klamrowy i trójkątny) z jego zamykającą wersją. Stwórz potem drugą mapę, która będzie kojarzyła zamykające nawiasy z otwierającymi.

Następnie napisz funkcję `has_balanced_brackets()`, która przyjmie `std::string`a zawierającego różne znaki (w tym nawiasy wspomniane wyżej) oraz mapy kojarzące pary nawiasów – również wspomniane wyżej. Funkcja ta winna zwracać prawdę lub fałsz zależnie od tego, czy nawiasy w przekazanym tekście są *zbalansowane*. W tym celu użyj adaptera `std::stack` według poniższego pseudokodu:

- Idąc po kolei przez wszystkie znaki:
    - Jeżeli napotkany znak to nawias otwierający, to dodaj go na stos;
    - Jeżeli napotkany znak to nawias zamykający, to sprawdź, czy na górze stosu jest odpowiadający mu nawias otwierający (wyjmując przy okazji ten element).

Jeżeli tak, to przejdź do następnej iteracji. Jeżeli nie, to nawiasy nie są zbalansowane;

- Jeżeli po przejściu przez wszystkie znaki stos nie jest pusty, to nawiasy nie są zbalansowane.

## Zadanie 3

Napisz funkcję `chunk()`, która przyjmie `std::vector` przechowujący typ wyliczeniowy o nazwie `action` z możliwymi wartościami: `move_left`, `move_right` oraz `idle` (wpierw stwórz ten typ) i zwróci `std::vector` przechowujący trójelementowe tablice tego samego typu, co elementy wektora wejściowego. Jeżeli liczba elementów wektora wejściowego jest niepodzielna przez 3, to ostatnią tablicę dopełnij wartościami `action::idle`. Następnie napisz funkcję `effectively_idle()`, która będzie mogła przyjąć obiekt zwracany przez funkcję `chunk()` oraz sprawdzi, czy wszystkie przekazane partycje (wspomniane wcześniej trójelementowe tablice) efektywnie reprezentują brak ruchu (brak ruchu jest reprezentowany przez dwa, wykluczające się ruchy, czyli lewo-prawo lub prawo-lewo i dodatkowy ruch `idle` lub po prostu przez trzy ruchy `idle`).