Praca domowa III—Biblioteka z API

16 stycznia 2026

1 Instrukcja

Celem projektu jest implementacja słownika z użyciem drzewiastej struktury danych. Struktura powinna mieć dwa interfejsy: interfejs Rustowy i interfejs C. Kluczem słownika jest typ `u64`, a jego wartościami są ciągi znakowe (własna implementacja prostej alternatywy dla `String`). Do implementacji możesz wybrać dowolną spośród następujących struktur:

1. Drzewo AVL,
2. Drzewo Czerwono-Czarne,
3. B-Drzewo.

Termin projektu: 30.01.2026. Każdy tydzień opóźnienia skutkuje ograniczeniem maksymalnej liczby punktów o 5.

W sekcji 3 znajdziesz wymaganą do zaimplementowania funkcjonalność. Za jej pełną implementację otrzymać możesz 15pkt. W sekcji 4 znajdziesz rzeczy, których nie możesz robić w projekcie.

Jeśli w poleceniu jest napisane, że należy zaimplementować strukturę/cechę/funkcję o nazwie `X`, która nie jest jednak definiowana szczegółowo przez polecenie, to jej implementacja jest dowolna.

Niedozwolone jest używanie dowolnych zewnętrznych bibliotek. **Niedozwolone** jest używanie struktur biblioteki standardowej, które alokują pamięć (`String`, `Box`, `Vec`, `BTreeMap`, itd.).

Ocena za projekt liczona jest ze wzoru

$$k = \max(r + p),$$

gdzie k—ocena końcowa, r—punkty z sekcji 3, p—punkty z sekcji 4.

2 Biblioteki

W implementacji przyda się biblioteka `libc` (`https://crates.io/crates/libc`). Jest to jedyna dozwolona biblioteka zewnętrzna.

3 Wymagana funkcjonalność

1. (2pkt) Dodawanie elementów z kluczem do słownika.
2. (2pkt) Wyszukiwanie elementów po kluczu ze słownika.
3. (1pkt) Sprawdzenie istnienia klucza w słowniku.
4. (2pkt) Usuwanie elementów po kluczu ze słownika.
5. (3pkt) Makro pozwalające utworzyć słownik z podanymi wartościami na podanych kluczach (analogicznie do `vec!`).
6. (3pkt) Udostępniony interfejs operowania na tej strukturze w C.

7. (1pkt) Przykładowe użycie utworzonej struktury w Rust (używające wszystkich metod struktury).

8. (1pkt) Przykładowe użycie utworzonej struktury w C (używające wszystkich funkcji struktury).

4 Praktyki zakazane

Pojawienie się któregokolwiek z poniższych elementów w projekcie skutkuje odjęciem tylu punktów, ile sprecyzowano.

1. Użycie metod `unwrap`, `expect` lub makra `panic`. −1pkt za 1 lub 2 użycia, −2pkt za więcej użyć. Dozwolone jednak używanie w testach.

2. Używanie struktury, gdzie właściwy byłby `enum`, to jest trzymanie w strukturze pól, których wartość czasami ma, a czasami nie ma znaczenia, np.

   ```
   struct IP {
   v4: String,
   v6: String,
   is_v4: bool
   }
   ```

   -1pkt za jedno wystąpienie, -2pkt za więcej.

3. Ignorowanie błędów, kiedy powinny być obsłużone, np. wołanie funkcji, która zwraca `Result<T, E>` i sprawdzanie przez `if let` tylko przypadku `Ok`, kiedy błąd ma znaczenie (jeśli nie ma, to oczywiście dozwolone). -1pkt za jedno lub 2 wystąpienia, -2pkt za więcej.

4. Funkcje dłuższe niż 30 linijek. -1pkt za 1 lub 2, -2pkt za więcej. Funkcje takie są dozwolone w testach.

5. Funkcje, które niepotrzebnie przejmują własność obiektów, kiedy mogłyby je tylko pożyczać. -1pkt za 1-3 takich funkcji. -2pkt za więcej.

6. Używanie `clone`, kiedy jest niepotrzebny (tzn. kod kompiluje się po jego usunięciu, a nie wpływa to na logikę). -1pkt za jedno lub więcej użyć.

7. Wycieki pamięci, -2pkt.