Wprowadzenie i podstawy języka

3 października 2025

Instrukcja

1. Skonfiguruj IDE do pracy z Rustem.
2. Utwórz nowy projekt.
3. Dodaj do projektu bibliotekę `rand`.
4. Zaimplementuj program w kolejnej sekcji.
5. Napraw wszystkie ostrzeżenia kompilatora.
6. Znajdź powszechne problemy z kodem używając `cargo clippy` i je napraw.

Program

Zaimplementuj program z następującą funkcjonalnością.

1. Rozpocznij pętlę `loop`.
2. Wyświetl wiadomość „Podaj liczbę”.
3. Pobierz od użytkownika z `STDIN` string.
4. Przekonwertuj string na liczbę x typu `u64`. W wypadku błędu przerwij pętlę z wartością `true`. Użyj wyrażenia `match`.
5. Jeśli podana liczba to 0, to przerwij pętlę z wartością `false` (`break false;`).
6. Dodaj do x losową liczbę z przedziału $[0, 5]$ i wyświetl nową wartość x.
7. Zapisz do zmiennej tablicę 10-elementową, zawierającą kolejne potęgi x (zaimplementuj to w osobnej funkcji, która zwraca tablicę).
8. Dla każdej liczby w tablicy sprawdź, czy spełniona jest dla niej hipoteza Collatza (limit 100 iteracji) (użyj zagnieżdżonych pętli `for`) (zaimplementuj w osobnej funkcji lub funkcjach) i zapisz tę informację do osobnej, 10-elementowej tablicy.

9. Tablicę z poprzedniego punktu zapisz w dowolnym formacie do pliku o nazwie ”xyz.txt”(użyj przykładu z `https://doc.rust-lang.org/stable/std/fs/struct.File.html`, ale zamiast operatora `?`, którego jeszcze nie było, użyj metody `expect`). W wypadku błędu przerwij pętlę z `true`.

10. Wróć do 1.

11. Jeśli pętla się zakończy, to wartość z niej zwróconą zapisz do zmiennej i wyświetl komunikat, czy zakończyła się z woli użytkownika (`false`) czy z powodu błędu (`true`).

12. Na sam koniec programu dodaj wymyśloną przez siebie funkcjonalność, która używa krotek i przerywania zewnętrznej pętli z etykietą z zagnieżdżonej pętli (`break 'label;`). Musi wystąpić funkcja zwracająca krotkę składającą się z różnych typów (nie może to być więc np. `(i32, i32)`) i przypisanie wyniku tej funkcji do zmiennych.

Wymagania

- Kod kompiluje się przy użyciu stabilnego kompilatora Rust.
- Brak ostrzeżeń z kompilatora i ‘clippy‘.
- Pełna implementacja zadanej funkcjonalności

Ocena (3 pkt)

- 2 pkt: Pełna funkcjonalność (pełna implementacja zgodna z wymaganiami).
- 1 pkt: Prezentacja rozwiązania i odpowiedź na pytania prowadzącego.

W tym tygodniu osoby nieobecne na laboratorium mogą otrzymać 3 pkt za zadanie wykonane w domu.