Załącznik Nr 1

do zarządzenia Nr 182/2023

z dnia 21 lipca 2023 r.

**SYLABUS PRZEDMIOTU W SZKOLE DOKTORSKIEJ**

|  |  |  |
| --- | --- | --- |
| **Lp.** | **Elementy składowe sylabusa** | **Opis** |
| **1** | Nazwa przedmiotu w języku polskim oraz angielskim | Kurs języka Rust |
| **2** | Dyscyplina/ dyscypliny naukowe (jeżeli dotyczy) | Informatyka |
| **3** | Nazwa jednostki organizującej kształcenie  | Instytut Matematyczny UWr, Kolegium Doktorskie Matematyki UWr. |
| **4** | Jednostka prowadząca przedmiot/moduł | Instytut Informatyki UWr, Wydział Matematyki i Informatyki |
| **5** | Kod przedmiotu/ modułu  | --- |
| **6** | Rodzaj przedmiotu/ modułu | fakultatywny |
| **7** | Rok kształcenia | wszystkie roczniki |
| **8** | Semestr | zimowy |
| **9** | Formy\* , metody\*\* i tryb \*\*\* prowadzenia przedmiotu | wykład i laboratoria |
| **10** | Treści programowe |  1. Podstawowe typy danych\* literały i operatory, krotki, tablice i tablice dynamiczne2. Typy własne\* struktury, typy wyliczeniowe, stałe3. Typowe pojęcia z zakresu programowania\* zmienne i mutowalność, typy skalarne i złożone, funkcje, komentarze, sterowanie przepływem4. Pojęcie posiadania\* reguły posiadania, zakres zamiennej, odwołania i pożyczki, odwołania mutowalne, wiszące odwołania, reguły odwołań, typ wycinek5. Używanie struktur\* definiowanie struktur i tworzenie ich instancji\* składnia metody6. Typ wyliczeniowy i dopasowanie wzorca\* definiowanie wyliczenia\* operator sterowania przepływem match\* sterowanie przepływem za pomocą if let7. Wykorzystanie modułów do porządkowania kodu i jego ponownego wykorzystania\* mod i Filesystem\* sterowanie widocznością za pomocą pub \* odwoływanie się do nazw w różnych modułach8. Typowe kolekcje\* przechowywanie list wartości za pomocą wektorów\* przechowywanie w łańcuchach tekstu zakodowanego za pomocą UTF-8\* przechowywanie kluczy z powiązanymi wartościami w mapach skrótów9. Obsługa błędów\* błędy nienaprawialne z makrem panic!\* błędy do naprawienia za pomocą Result\* panikować czy nie panikować10. Typy generyczne, cechy i czasy życia\* usuwanie duplikacji przez wyodrębnienie funkcji\* generyczne typy danych\* cechy – definiowanie wspólnego zachowania\* sprawdzanie odwołań za pomocą czasów życia\* parametry generycznego typu, granice cech i czas życia w połączeniu11. Pisanie automatycznych testów\* jak pisać testy\* sterowanie sposobem uruchamiania testów\* organizacja testów12. Projekt we/wy – budowa programu wiersza poleceń\* akceptowanie argumentów wiersza poleceń\* czytanie pliku\* refaktoryzacja w celu poprawienia modułowości i obsługi błędów \* tworzenie funkcjonalności biblioteki przy użyciu TDD\* praca ze zmiennymi środowiskowymi\* pisanie komunikatów o błędach do standardowego błędu zamiast standardowego wyjścia13. Funkcje języka funkcyjnego: iteratory i domknięcia\* domknięcia – anonimowe funkcje, które mogą przechwycić swoje środowisko\* przetwarzanie ciągów elementów za pomocą iteratorów\* ulepszanie projektu we/wy\* porównywanie wydajności – pętle a iteratory14. Więcej informacji o cargo i crates.io\* wersje niestandardowe z profilami wydania\* publikacja skrzynki w Crates.io\* przestrzenie robocze Cargo\* instalowanie wersji binarnych z Crates.io za pomocą cargo install \* Rozszerzanie Cargo za pomocą niestandardowych poleceń15. Inteligentne wskaźniki\* używanie Box do wskazywania danych na kopcu\* traktowanie inteligentnych wskaźników jak zwykłych odwołań z cechą Deref\* uruchamianie kodu czyszczącego z cechą Drop \* rc – inteligentny wskaźnik ze zliczaniem odwołań\* RefCell oraz wzorzec wewnętrznej mutowalności\* odwołania cykliczne mogą prowadzić do wycieku pamięci16. Współbieżność bez obaw\* użycie wątków do równoległego uruchamiania kodu\* używanie przekazywania komunikatów do transferu danych między wątkami\* współbieżność ze współdzieleniem zasobów\* elastyczna współbieżność z cechami Sync i Send17. Własności programowania obiektowego w języku Rust\* charakterystyka języków obiektowych\* wykorzystywanie obiektów cech, które dopuszczają wartości różnych typów \* implementowanie wzorca projektu obiektowego18. Wzorce i dopasowanie\* wszystkie miejsca, w których można korzystać ze wzorców\* składnia wzorca19. Funkcje zaawansowane\* niebezpieczny Rust, zaawansowane czasy życia, zaawansowane cechy, typy zaawansowane, zaawansowane funkcje i domknięcia |
| **11** | Język wykładowy | polski |
| **12** | Zakładane efekty uczenia się w zakresie:Wiedza:…………………………………………………………..Umiejętności:…………………………………………………………..Kompetencje społeczne:………………………………………………………….. | Celem przedmiotu jest zaznajomienie studentów z językiem Rust, w tym cech które sprawiają, że wyróżnia się na tle innych języków programowania. Chodzi głównie o mechanizmy zapobiegające naruszeniom pamięci, oraz pozwalające na bezpieczne operowanie wątkami.## Wiedza\* rozumie, czym charakteryzuje się programowanie: obiektowe, imperatywne, funkcyjne i strukturalne [K\_W06]\* rozumie zalety i pułapki programowania współbieżnego\* zna podstawowe konstrukcje programistyczne oraz standardowe typy danych w języku Rust (przypisanie, instrukcje sterujące, wywoływanie funkcji i przekazywanie parametrów, typy całkowite i zmiennopozycyjne)\* Rozumie pojęcie składni języka programowania.Zna podstawowe struktury danych i wykonywane na nich operacje (tablice, napisy, rekordy, pliki, krotki, listy, stosy, kolejki i drzewa) [SD\_W01, SD\_W02] ## Umiejętności\* potrafi pisać, debugować, kompilować i uruchamiać programy w języku Rust, z użyciem mechanizmów które sprawiają, że wyróżnia się na tle innych języków programowania (bezpieczeństwo pamięci i wielowątkowości). [SD\_U07] |
| **13** | Metody weryfikacji zakładanych efektów uczenia się | Około 35 programów komputerowych sprawdzonych automatycznie na platformie codewars.com, przedstawione osobiście podczas ćwiczeń, oraz 3 projekty programistyczne w tym ostatni o dowolnej tematyce. |
| **14** | Obciążenie pracą doktoranta |  |
|  | Formy aktywności doktoranta | Średnia liczba godzin na zrealizowanie aktywności |
| Godziny zajęć (wg planu kształcenia) z nauczycielem:- wykład: 30- ćwiczenia: - laboratorium: 30- seminarium:- inne: | Łącznie 60 godzin zajęć |
| Praca własna doktoranta, np.:- czytanie wskazanej literatury;- przygotowanie zaliczeniowej pracy pisemnej;- przygotowanie wystąpienia ustnego;- realizacja projektu grupowego;- przygotowanie do egzaminu;- inne | Studiowanie literatury - 10 godzin.Przygotowanie do zajęć - 30 godzin. |
| Suma godzin | 100 godzin |
| Liczba punktów ECTS (jeżeli jest wymagana) |  |
| **15** | Warunki zaliczenia przedmiotu: metody potwierdzania uzyskania efektów uczenia się i kryteria oceny | Zdobycie 50% możliwych do zdobycia punktów.Punkty można zdobywać poprzez:- 7 list zadaniowych (po 5 programów każda)- projekt programistyczny- projekt webassembly- projekt finalnyDodatkowe punkty można zdobyć poprzez:- Ilość zrobionych zadań w konkursie AdventOfCode- Zajęcie jednego z pierwszych 20 miejsc w klanie codewars.com |
| **16** | Podstawowa literatura przedmiotu | Steve Klabnik, Carol Nichols: Programowanie w języku Rust (Wydanie: 1, 2019, Wydawnictwo Naukowe PWN)The Rust Programming Languageby Steve Klabnik and Carol Nichols, with contributions from the Rust Community (free online version) |

\* wykład, seminarium, ćwiczenia, warsztaty, lektoraty, laboratoria

\*\* prezentacja, projekt, analiza przypadku, dyskusja, metoda problemowa

\*\*\* stacjonarnie/zdalnie