*Załącznik nr 1.5 do Zarządzenia Rektora UR nr 12/2019*

**SYLABUS**

**dotyczy cyklu kształcenia 2022-2026**

Rok akademicki 2024/2025

1. Podstawowe informacje o przedmiocie

|  |  |
| --- | --- |
| Nazwa przedmiotu | języki i paradygmaty programowania |
| Kod przedmiotu |  |
| Nazwa jednostki prowadzącej kierunek | Instytut Informatyki, Kolegium Nauk Przyrodniczych |
| Nazwa jednostki realizującej przedmiot | Instytut Informatyki, Kolegium Nauk Przyrodniczych |
| Kierunek studiów | informatyka |
| Poziom studiów | studia inżynierskie I-go stopnia |
| Profil | ogólnoakademicki |
| Forma studiów | stacjonarne |
| Rok i semestr/y studiów | rok III semestr 6 |
| Rodzaj przedmiotu | inżynierski przedmiot kierunkowy |
| Język wykładowy | język polski, język angielski |
| Koordynator | dr Krzysztof Balicki |
| Imię i nazwisko osoby prowadzącej / osób prowadzących |  |

1.1 Formy zajęć dydaktycznych, wymiar godzin i punktów ECTS

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Semestr  (nr) | Wykł. | Ćw. | Konw. | Lab. | Sem. | ZP | Prakt. | Inne (jakie?) | **Liczba pkt. ECTS** |
| 6 | 30 |  |  | 30 |  |  |  |  | 5 |

1.2 Sposób realizacji zajęć

zajęcia w formie tradycyjnej

1.3 Forma zaliczenia przedmiotu (z toku)

zaliczenie z oceną

1. Wymagania wstępne

|  |
| --- |
| Podstawy programowania w języku C, Algorytmy i struktury danych, Programowanie obiektowe |

1. Cele, efekty uczenia się, treści programowe i stosowane metody dydaktyczne

3.1 Cele przedmiotu

|  |  |
| --- | --- |
| C1 | W ramach przedmiotu omawiane są cztery obecnie najistotniejsze paradygmaty programowania (programowanie imperatywne, obiektowe, funkcyjne i programowanie w logice) oraz związane z nimi języki programowania. |
| C2 | Kurs ten ma dać uczestnikom szerszy pogląd na programowanie. Pokazać cechy wspólne i różnice między typowymi dla tych paradygmatów językami, jak również pokazać metody tworzenia i kompilacji programów utworzonych w tych językach. |
| C3 | Ważną cechą tych zajęć jest duża liczba praktycznych ćwiczeń, które umożliwiają studentom praktyczne zastosowanie poznanych paradygmatów. |

**3.2 Efekty uczenia się dla przedmiotu**

|  |  |  |
| --- | --- | --- |
| EK (efekt uczenia się) | Treść efektu uczenia się zdefiniowanego dla przedmiotu | Odniesienie do efektów kierunkowych |
| EK\_01 | Student zna zasady formułowania i algorytmizacji zadań oraz podstawowe notacje zapisu algorytmów. | K\_W04 |
| EK\_02 | Student zna podstawowe paradygmaty programowania oraz co najmniej jeden język reprezentujący każdy z poznanych paradygmatów w stopniu umożliwiającym mu pisanie prostych programów użytkowych. | K\_W07 |
| EK\_3 | Student umie ułożyć i analizować (w tym śledzić) algorytm zgodny ze specyfikacją i zapisać go w wybranym języku programowania. | K\_U11 |
| EK\_4 | Student potrafi pisać proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego i programowania w logice. | K\_U12 |

**3.3 Treści programowe**

1. Problematyka wykładu

|  |
| --- |
| 1. Przegląd najważniejszych paradygmatów programowania. |
| 2. Programowanie funkcyjne w języku Haskell. |
| 3. Programowanie deklaratywne w języku PROLOG. |
| 4. Programowanie imperatywne – przypomnienie podstawowych zagadnień. |
| 5. Programowanie obiektowe – przypomnienie podstawowy zagadnień. |
| 6. Paradygmat imperatywny – przegląd i przypomnienie. |
| 7. Wykorzystanie różnych paradygmatów programowania do rozwiązywania problemów algorytmicznych. |
| 8. Inne paradygmaty programowania – wykład podsumowujący. |

1. Problematyka ćwiczeń laboratoryjnych

|  |
| --- |
| 1. Środowisko pracy. Kompilatory i interpretery. |
| 2. Programowanie funkcyjne w języku Haskell. |
| 3. Programowanie deklaratywne w języku PROLOG. |
| 4. Programowanie imperatywne w języku C – powtórzenie wybranych zagadnień. |
| 5. Programowanie obiektowe w języku Java – powtórzenie wybranych zagadnień. |
| 6. Wykorzystanie różnych paradygmatów programowania do rozwiązywania problemów algorytmicznych. |

3.4 Metody dydaktyczne

Wykład: wykład problemowy, wykład z prezentacją multimedialną

Laboratoria: rozwiązywanie zadań, dyskusja

1. Metody i kryteria oceny

4.1 Sposoby weryfikacji efektów uczenia się

|  |  |  |
| --- | --- | --- |
| Symbol efektu | Metody oceny efektów uczenia sie  (np.: kolokwium, egzamin ustny, egzamin pisemny, projekt, sprawozdanie, obserwacja w trakcie zajęć) | Forma zajęć dydaktycznych  (w, ćw, …) |
| Ek\_ 01 | obserwacja w trakcie zajęć | lab |
| Ek\_ 02 | kolokwium | lab |
| Ek\_ 03 | kolokwium | lab |
| Ek\_ 04 | kolokwium | lab |

4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

|  |
| --- |
| Wykład  Zaliczenie bez oceny, efekty kierunkowe EK\_01, EK\_02 właściwe dla wykładu weryfikowane są w trakcie zajęć laboratoryjnych przy pomocy kolokwiów i obserwacji na zajęciach.  Laboratorium  Warunkiem zaliczenia laboratorium jest zaliczenie kolokwiów z języka Haskell oraz z języka Prolog. Ocena końcowa jest średnią ocen z tych dwóch kolokwiów. Aby zaliczyć kolokwium należy zdobyć przynajmniej połowę maksymalnej liczby punktów. Oceny z kolokwiów przyznawane są proporcjonalnie do liczby zdobytych punktów. Pod uwagę brana jest również aktywność na zajęciach, która może obniżyć lub podwyższyć ocenę końcową o pół stopnia. |

1. **Całkowity nakład pracy studenta potrzebny do osiągnięcia założonych efektów w godzinach oraz punktach ects**

|  |  |
| --- | --- |
| **Forma aktywności** | **Średnia liczba godzin na zrealizowanie aktywności** |
| Godziny kontaktowe wynikające z harmonogramu studiów | 60 |
| Inne z udziałem nauczyciela akademickiego  (udział w konsultacjach, egzaminie) | 3 |
| Godziny niekontaktowe – praca własna studenta  (przygotowanie do zajęć, egzaminu, napisanie referatu itp.) | 62 |
| SUMA GODZIN | 125 |
| **SUMARYCZNA LICZBA PUNKTÓW ECTS** | 5 |

1. Praktyki zawodowe w ramach przedmiotu

|  |  |
| --- | --- |
| wymiar godzinowy | - |
| zasady i formy odbywania praktyk | - |

1. Literatura

|  |
| --- |
| Literatura podstawowa:   1. P. Hudak, J. Peterson, J. Fasel: *A Gentle Introduction to Haskell*   https://www.haskell.org/tutorial/index.html   1. W.F.Clocksin, C.S.Mellish, *Prolog. Programowanie*, Wydawnictwo Helion, 2003 2. Bryan O'Sullivan, Don Stewart, John Goerzen: *Real World Haskell*. O'Reilly Media 2008 3. Kees Doets, Jan van Eijck: *The Haskell Road to Logic, Math and Programming*. King's College Publications 2004 4. Graham Hutton: *Programming in Haskell.* Cambridge University Press 2007 5. Ivan Bratko: *Prolog Programming for Artificial Intelligence.* Addison-Wesley 2000 6. W.F.Clocksin, C.S.Mellish, *Prolog. Programowanie*, Wydawnictwo Helion, 2003 |
| Literatura uzupełniająca:   1. Joeren Fokker: *Functional Programming*. Department of Computer Science, Utrecht University 1995 (plik pdf dostępny w Internecie) 2. Hal Daume III, et. al.: *Yet Another Haskell Tutorial*. 2004 (plik pdf dostępny w Internecie) 3. J. R. Fischer: *Prolog tutorial*, http://www.csupomona.edu/~jrfisher/www/prolog\_tutorial/contents.html 4. Dave Stuart Robertson: *Quick Prolog*,  http://www.dai.ed.ac.uk/groups/ssp/bookpages/quickprolog/quickprolog.html 5. Patrick Blackburn, Johan Bos and Kristina Striegnitz: *Learn Prolog Now!*,http://www.learnprolognow.org 6. Brian W. Kernighan, Dennis Ritchie: *Język ANSI C.* WNT, Warszawa 2003 7. Bruce Eckel: *Thinking in Java*. Edycja polska, Wydanie 4, Helion, Gliwice, 2006 8. Marcin Lis: Praktyczny kurs Java. Helion, Gliwice, 2007 |

Akceptacja Kierownika Jednostki lub osoby upoważnionej