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*
Koordynator	*dr Krzysztof Balicki*
Imię i nazwisko osoby prowadzącej / osób prowadzących	*dr Krzysztof Balicki, dr inż. Wojciech Kozioł*

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

2. WYMAGANIA WSTĘPNE

Programowanie obiektowe, algorytmy i złożoność

3. 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_03	Student umie ułożyć i analizować (w tym śledzić) algorytm zgodny ze specyfikacją i zapisać go w wybranym języku programowania.	K_U11
EK_04	Student potrafi pisać proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego i programowania w logice.	K_U12

3.3 Treści programowe

A. 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.

B. 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
Ćwiczenia laboratoryjne: praca w grupach (rozwiązywanie zadań, dyskusja).

4. METODY I KRYTERIA OCENY

4.1 Sposoby weryfikacji efektów uczenia się

Symbol efektu	Metody oceny efektów uczenia się (np.: kolokwium, egzamin ustny, egzamin pisemny, projekt, sprawozdanie, obserwacja w trakcie zajęć)	Forma zajęć dydaktycznych (w, ćw, ...)
EK_01	obserwacja w trakcie zajęć	wykład, lab
EK_02	kolokwium	wykład, lab
EK_03	kolokwium	lab
EK_04	kolokwium	lab

4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

Efekt	Ocena	Kryteria oceny
EK_01	zal	Student zna podstawowe paradygmaty programowania. Zna języki programowania reprezentujące poznane paradygmaty w stopniu umożliwiającym mu pisanie prostych programów.
EK_02	dst	Student zna podstawowe paradygmaty programowania. Zna języki programowania reprezentujące poznane paradygmaty w stopniu umożliwiającym mu pisanie prostych programów, przy czym co najmniej jeden spośród tych języków programowania zna w stopniu umożliwiającym mu swobodną realizację nawet skomplikowanych algorytmów.
	db	Student zna podstawowe paradygmaty programowania. Zna języki programowania reprezentujące poznane paradygmaty w stopniu umożliwiającym mu pisanie prostych programów, przy czym co najmniej jeden spośród tych języków programowania zna w stopniu umożliwiającym mu swobodną realizację nawet skomplikowanych algorytmów. Student potrafi wybrać język programowania, który jego zdaniem jest najodpowiedniejszy do rozwiązania danego problemu i na podstawie wiedzy zdobytej na wykładach umie uzasadnić swój wybór.
	bdb	Student zna podstawowe paradygmaty programowania. Zna języki programowania reprezentujące poznane paradygmaty w stopniu umożliwiającym mu pisanie prostych programów, przy czym zna co najmniej dwa spośród języków programowania reprezentujących różne

		paradygmaty programowania w stopniu umożliwiającym mu swobodną realizację nawet skomplikowanych algorytmów. Student trafnie określa wybór języka, który najlepiej nadaje się do rozwiązania danego problemu. Na podstawie wiedzy zdobytej na zajęciach i z literatury potrafi szczegółowo uzasadnić swój wybór.
EK_03	dst	Student potrafi analizować proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego i programowania w logice.
	db	Student potrafi analizować proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego, imperatywnego, obiektowego i programowania w logice. Student umie dokonać analizy algorytmu, wychwycić i poprawić błędy.
	bdb	Student potrafi analizować proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego, imperatywnego, obiektowego i programowania w logice. Student umie dokonać analizy algorytmu, wychwycić i poprawić błędy i jeśli to możliwe zaproponować bardziej optymalny sposób rozwiązania problemu.
EK_04	dst	Student potrafi pisać proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego i programowania w logice.
	db	Student potrafi pisać proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego, programowania w logice, imperatywnego i obiektowego, przy czym zna dobrze co najmniej jeden spośród tych języku i potrafi w nim tworzyć złożone aplikacje.
	bdb	Student potrafi pisać proste programy użytkowe w co najmniej jednym języku programowania funkcyjnego, programowania w logice, imperatywnego i obiektowego, przy czym zna dobrze dwa lub więcej spośród tych języku i potrafi w nich tworzyć złożone aplikacje.

Zasady uzyskania oceny końcowej:

Wykład:
Zaliczenie wykładów odbywa się poprzez pisemne sprawdzenie wiadomości.

Laboratoria:
Ocena końcowa z laboratoriów wystawiana jest na podstawie ocen zdobytych z kolokwiów z języków Haskell i PROLOG. Wpływ na ocenę końcową wywierać będę również ewentualne kartkówki i ocena aktywności studenta podczas zajęć. Warunkiem koniecznym do uzyskania zaliczenia z laboratorium jest zaliczenie obu kolokwiów na ocenę pozytywną oraz obecność na zajęciach.

5. 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

6. PRAKTYKI ZAWODOWE W RAMACH PRZEDMIOTU

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

7. LITERATURA

Literatura podstawowa:

1. Bryan O'Sullivan, Don Stewart, John Goerzen: *Real World Haskell*. O'Reilly Media 2008
2. Kees Doets, Jan van Eijck: *The Haskell Road to Logic, Math and Programming*. King's College Publications 2004
3. Graham Hutton: *Programming in Haskell.* Cambridge University Press 2007
4. Ivan Bratko: *Prolog Programming for Artificial Intelligence.* Addison-Wesley 2000
5. 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