UNIWERSYTET IM. ADAMA MICKIEWICZA W POZNANIU

Paradygmaty programowania

Sylabus zajęć

Informacje podstawowe

Kierunek studiów
Informatyka

Specjalność
-

Jednostka organizacyjna
Wydział Matematyki i Informatyki

Poziom studiów
studia inżynierskie pierwszego stopnia

Forma studiów
studia stacjonarne

Profil studiów
profil ogólnoakademicki

Cykl dydaktyczny
2023/24

Kod zajęć
06INFS.32P.02413.23

Języki wykładowe
polski

Obligatoryjność
Obowiązkowy

Blok zajęciowy
Przedmioty podstawowe

Koordynator zajęć	Arkadiusz Hypki
Prowadzący zajęcia	Arkadiusz Hypki

Okres	**Forma zajęć / liczba godzin / forma zaliczenia**	**Liczba punktów ECTS**
Semestr 2	• Wykład: 30, Zaliczenie z oceną • Ćwiczenia w salach komputerowych: 30, Zaliczenie z oceną	4

Cele kształcenia dla zajęć

Kod	Cel
C1	Wyjaśnienie definicji paradygmatu programowania, ich rodzajów oraz celów dla których powstały.
C2	Wyjaśnienie różnic pomiędzy różnymi paradygmatami programowania.
C3	Pokazanie konkretnych przykładów języków programowania charakterystycznych dla danych paradygmatów programowania.
C4	Przekazanie studentom wiedzy teoretycznej i praktycznej, aby móc wybrać odpowiedni język programowanie do osiągnięcia określonego celu.
C5	Nauka programowania imperatywnego (proceduralne i zorientowane obiektowo).
C6	Nauka programowania deklaratywnego (ogólnego i funkcyjnego).
C7	Nauka programowania sterowanego danymi.
C8	Nauka programowania sterowanego zdarzeniami.
C9	Nauka programowanie współbieżnego i w modelu SIMD.
C10	Nauka programowania na urządzenia wbudowane.

Wymagania wstępne

Wiedza i umiejętności z zajęć Wstęp do informatyki oraz Podstawy programowania.

Efekty uczenia się dla zajęć

Kod	Efekty uczenia się dla zajęć w zakresie	Efekty uczenia się dla kierunku	Metody weryfikacji osiągnięcia efektów uczenia się dla zajęć
Wiedzy – Student/ka:			
W1	zna i rozumie definicje paradygmatów programowania, ich rodzaje oraz cel ich powstania	INF_K3_W02, INF_K3_W03, INF_K3_W04_inz	Test, Projekt
W2	zna i rozumie zasady doboru danego paradygmatu programowania do konkretnego zadania programistycznego w celu jego najlepszego rozwiązania	INF_K3_W02, INF_K3_W03, INF_K3_W04_inz	Test, Projekt
Umiejętności – Student/ka:			
U1	potrafi rozróżniać paradygmaty programowania i jest wstanie dobrać odpowiedni język programowania do postawionego zadania	INF_K3_U02, INF_K3_U04_inz, INF_K3_U05_inz, INF_K3_U06_inz	Projekt
U2	potrafi korzystać ze zintegrowanych środowisk programistycznych, które ułatwiają pisanie oprogramowania w językach programowania wymienionych w celach kształcenia	INF_K3_U05_inz, INF_K3_U06_inz	Projekt
U3	potrafi napisać podstawowe programy w językach programowania wymienionych w celach kształcenia	INF_K3_U05_inz, INF_K3_U06_inz	Projekt
U4	potrafi odnaleźć odpowiednie informacje w dokumentacji technicznej wybranego języka programowania w języku angielskim	INF_K3_U02, INF_K3_U08	Projekt

Treści programowe dla zajęć

Lp.	Treści programowe dla zajęć	Efekty uczenia się dla zajęć	Formy zajęć
1.	Czym są paradygmaty programowania – definicja, rodzaje, cele ich powstania. Główne różnice pomiędzy paradygmatami programowania na przykładzie konkretnych języków programowania.	W1, W2	Wykład
2.	Programowanie imperatywne proceduralne (ang. imperative procedural) na przykładzie języków C/C++, Python.	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
3.	Programowanie imperatywne zorientowane obiektowo (ang. imperative object-oriented) na przykładzie języków Java, C# (krótko o SmallTalk).	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
4.	Programowanie deklaratywne ogólnie (ang. declarative) na przykładzie języków SQL, HTML i programowanie w logice.	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
5.	Programowanie deklaratywne funkcyjne (ang. functional) na przykładzie języków Lisp, Haskell, Scheme, Clojure, Elixir, Java.	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
6.	Programowanie sterowane danymi (ang. data-driven) na przykładzie języków AWK, JQ, XSLT.	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
7.	Programowanie sterowane zdarzeniami (ang. event-driven) na przykładzie JavaScript	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
8.	Programowanie współbieżne i w modelu SIMD na przykładzie OpenMP, MPI	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
9.	Języki wbudowane (ang. embedded) na przykładzie języka Lua	U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
10.	Główne języki programowania ich cechy charakterystyczne, różne podziały języków (zarządzenia pamięcią, typowanie, dynamiczność). Języki interpretowane i kompilowane.	W1, W2, U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych
11.	Współczesne trendy w programowaniu: asyncio w Python, automatyczne zarządzanie pamięcią (garbage collector) w Java, transpilacja w TypeScript, lekka wielowątkowość z wykorzystaniem coroutines.	W1, W2, U1, U2, U3, U4	Wykład, Ćwiczenia w salach komputerowych

Informacje dodatkowe

Forma zajęć	Metody i formy prowadzenia zajęć
Wykład	Wykład z prezentacją multimedialną wybranych zagadnień
Ćwiczenia w salach komputerowych	Metoda laboratoryjna, Metoda warsztatowa

Forma zajęć	Warunki zaliczenia zajęć
Wykład	Na końcową ocenę składa się wynik uzyskany z testu. Skala ocen: bardzo dobry (bdb; 5,0): powyżej 90% punktów dobry plus (+db; 4,5): powyżej 80% punktów dobry (db; 4,0): powyżej 70% punktów dostateczny plus (+dst; 3,5): powyżej 60% punktów dostateczny (dst; 3,0): powyżej 50% punktów niedostateczny (ndst; 2,0): 50% punktów lub mniej
Ćwiczenia w salach komputerowych	Na końcową ocenę składa się wynik uzyskany z projektu. Skala ocen: bardzo dobry (bdb; 5,0): powyżej 90% punktów dobry plus (+db; 4,5): powyżej 80% punktów dobry (db; 4,0): powyżej 70% punktów dostateczny plus (+dst; 3,5): powyżej 60% punktów dostateczny (dst; 3,0): powyżej 50% punktów niedostateczny (ndst; 2,0): 50% punktów lub mniej

Literatura

Obowiązkowa

1. Peter Van Roy, Seif Haridi, Programowanie. Koncepcje, techniki i modele Peter Van Roy, Seif Haridi, 2005

Dodatkowa

1. R. Sebesta, Concepts of Programming Languages, Addison Wesley, 2005

Nakład pracy studenta i punkty ECTS

Rodzaje zajęć studenta	Średnia liczba godzin* przeznaczonych na zrealizowane rodzaje zajęć
Wykład	30
Ćwiczenia w salach komputerowych	30
Przygotowanie do zajęć	10
Czytanie wskazanej literatury	5
Przygotowanie projektu	15
Przygotowanie pracy pisemnej	10
Łączny nakład pracy studenta	**Liczba godzin** 100
Liczba punktów ECTS	**ECTS** 4

* godzina (lekcyjna) oznacza 45 minut

Efekty uczenia się dla kierunku

Kod	Treść
INF_K3_U02	Absolwent/ka potrafi pozyskiwać wiarygodne informacje z literatury, baz wiedzy, Internetu oraz innych źródeł, integrować je, interpretować oraz wyciągać wnioski i formułować opinie
INF_K3_U04_inz	Absolwent/ka potrafi opracować, przeanalizować, zaprojektować klasyczne algorytmy i systemy informatyczne
INF_K3_U05_inz	Absolwent/ka potrafi pisać, uruchamiać i testować programy w wybranym środowisku programistycznym
INF_K3_U06_inz	Absolwent/ka potrafi ocenić, na podstawowym poziomie, przydatność metod i narzędzi informatycznych oraz wybrać i zastosować właściwą metodę i narzędzia do typowych zadań informatycznych
INF_K3_U08	Absolwent/ka potrafi posługiwać się językiem angielskim zgodnie z wymaganiami określonymi dla poziomu B2 Europejskiego Systemu Opisu Kształcenia Językowego oraz zna język angielski w stopniu umożliwiającym czytanie ze zrozumieniem dokumentacji oprogramowania, podręczników i artykułów informatycznych
INF_K3_W02	Absolwent/ka zna i rozumie zaawansowane pojęcia i problemy formujące kanon dyscypliny informatyka
INF_K3_W03	Absolwent/ka zna i rozumie narzędzia, technologie i urządzenia informatyczne właściwe dla wybranych obszarów zastosowań oraz podstawy ich działania
INF_K3_W04_inz	Absolwent/ka zna i rozumie zaawansowane pojęcia, konstrukcje i procesy związane z językami programowania i inżynierią programowania