KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Języki i paradygmaty programowania	**1010331521010334960**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**(brak)**	**1 / 2**
Ścieżka obieralności/specjalność	Przedmiot oferowany w języku:	Kurs (obligatoryjny/obieralny)
-	**polski**	**obligatoryjny**

Stopień studiów:	**Forma studiów** (stacjonarna/niestacjonarna)
I stopień	**stacjonarna**

Godziny	Liczba punktów
Wykłady: **30** Ćwiczenia: **-** Laboratoria: **30** Projekty/seminaria: **-**	**6**

Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny)	(ogólnouczelniany, z innego kierunku)
(brak)	**(brak)**

Obszar(y) kształcenia i dziedzina(y) nauki i sztuki	**Podział ECTS (liczba i %)**
nauki techniczne	**6 100%**
nauki techniczne	**6 100%**

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Beata Jankowska
email: beata.jankowska@put.poznan.pl
tel. +48 61 665 37 24
Wydział Elektryczny
ul. Piotrowo 3A 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	Student ma podstawową wiedzę w zakresie matematyki, obejmującą algebrę, analizę, logikę, probabilistykę oraz elementy matematyki dyskretnej i stosowanej.
2	**Umiejętności:**	Student potrafi: posłużyć się środowiskami i platformami programistycznymi do pisania, wykonywania i testowania programów kodowanych w językach programowania imperatywnego; przygotować i przedstawić krótką prezentację poświęconą wynikom realizacji zadania. inżynierskiego.
3	**Kompetencje społeczne**	Student ma świadomość: odpowiedzialności za pracę własną; konieczności podporządkowania się zasadom pracy w zespole; ponoszenia odpowiedzialności za wspólnie realizowane zadania.

Cel przedmiotu:

Zapoznanie studentów ze stylem programowania obiektowego. Opanowanie przez nich umiejętności posługiwania się konstrukcjami języków obiektowych, w tym - projektowania i implementowania obszernych algorytmów w językach obiektowych C++ i Java.

Opanowanie zasad doboru stylu i języka programowania do charakteru zadania.

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

1. Student ma uporządkowaną i podbudowaną teoretycznie wiedzę w zakresie podstawowych algorytmów i ich analizy, technik projektowania algorytmów, abstrakcyjnych struktur danych i ich implementacji, problemów obliczeniowo trudnych. - [K_W04]

2. Student ma uporządkowaną i podbudowaną teoretycznie wiedzę w zakresie podst. konstrukcji programistycznych, implementacji algorytmów, paradygmatów i stylów programowania, metod weryfikacji poprawności programów, języków formalnych, kompilatorów, platform programistycznych. - [K_W05]

Umiejętności:

1. Student potrafi konstruować algorytmy z wykorzystaniem podstawowych technik algorytmicznych i dokonać analizy ich złożoności. - [K_U09]

2. Student potrafi posłużyć się środowiskami i platformami programistycznymi do pisania, wykonywania i testowania prostych programów kodowanych w językach programowania imperatywnego, obiektowego i deklaratywnego. - [K_U10]

3. Student potrafi opracować dokumentację dotyczącą realizacji zadania inżynierskiego i przygotować tekst zawierający omówienie wyników realizacji tego zadania. - [K_U03]

Kompetencje społeczne:

1. Student ma świadomość ważności dokładnego wykonania projektu, zachowania standardów notacyjnych, przestrzegania poprawności językowej i terminowego oddania prac. - [K_K07]
2. Student ma świadomość ważności i rozumie pozatechniczne aspekty i skutki działalności inżyniera-informatyka i związaną z tym odpowiedzialność za podejmowane decyzje. - [K_K02]

Sposoby sprawdzenia efektów kształcenia

Wykład: egzamin pisemny.

Laboratorium: zaliczenie na podstawie wejściówek, sprawdzianów i aktywności programistycznej na zajęciach, oraz - opcjonalnie - rozwiązania zadania projektowego (implementacja w C++, dokumentacja pisemna).

Kryterium egzaminacyjne i zaliczeniowe: od 50,1%.

Treści programowe

Wykład.

Klasyfikacja stylów programowania. Cele i zasady modelowania obiektowego. Język UML -najczęściej używane diagramy struktur i zachowań. Podstawowe paradygmaty programowania zorientowanego obiektowo (hermetyzacja, dziedziczenie, polimorfizm) i ich implementacja w języku C++. Biblioteki wejścia-wyjścia w języku C++. Obsługa błędów i obsługa wyjątków w języku obiektowym. Przeciążanie nazw funkcji i operatorów. Dynamiczne zarządzanie pamięcią w językach obiektowych. Programowanie uogólnione, biblioteka STL. Programowanie wielowątkowe. Wyrażenia regularne i klasa boost::regex.

Elementy programowania w języku Java: kod bajtowy, implementacja klas i obiektów, implementacja wejścia-wyjścia, pakiety, interfejsy, programowanie wielowątkowe, historyczne aplety.

SOLID - zasady efektywnego programowania w języku obiektowym.

Laboratorium.

Projektowanie algorytmów i ich implementacja w językach C++ i Java.

Zastosowane metody kształcenia: wykłady - wykład prowadzony w sposób interaktywny z formułowaniem pytań do grupy studentów lub do wskazywanych konkretnych studentów; wykład uzupełniony materiałami do samodzielnego studiowania w systemie Moodle;

Laboratoria - zajęcia na uczelni uzupełnione materiałami do samodzielnego wykonywania zadań w systemie Moodle; praca w zespołach.

Aktualizacja 2017: Cele i zasady modelowania obiektowego. Język UML - najczęściej używane diagramy struktur i zachowań. Wyrażenia regularne i klasa boost::regex. SOLID - zasady efektywnego programowania w języku obiektowym.

Literatura podstawowa:

1. Stroustrup B., Język C++. Kompendium wiedzy. Wydanie IV, Helion, 2014.
2. Grębosz J., Symfonia C++ standard. Programowanie w języku C++ orientowane obiektowo. Tom I i II, Wydanie 3B, Helion, 2010.
3. Prata S., Język C++. Szkoła programowania. Wydanie VI, Helion, 2012.
4. Schildt H., Java. Przewodnik dla początkujących. Wydanie VI, Helion, 2015.
5. Darwin I.F., Java. Receptury. Wydanie III, Helion, 2015.

Literatura uzupełniająca:

1. Prata S., Język C. Szkoła programowania. Wydanie V, Helion, 2006.
2. Eckel B., Thinking in C++. Edycja polska, Helion, 2012.
3. Eckel B., Thinking in Java. Edycja polska. Wydanie IV, Helion, 2006.

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. Wykłady	30
2. Ćwiczenia laboratoryjne	30
3. Udział w konsultacjach i i egzaminie	15
4. Bieżące przygotowanie do ćwiczeń laboratoryjnych	30
5. Przygotowanie do sprawdzianów	30
6. Przygotowanie do egzaminu	15

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	150	6
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	75	3
Zajęcia o charakterze praktycznym	75	3