KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Programowanie obiektowe [S2ETI1>ProgrObiekt]

Przedmiot

Kierunek studiów
Edukacja techniczno-informatyczna

Rok/Semestr
1/1

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
15

Laboratorium
30

Inne (np. online)
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

4,00

Koordynatorzy

dr hab. inż. Agnieszka Rybarczyk
agnieszka.rybarczyk@put.poznan.pl

Wykładowcy

dr hab. inż. Agnieszka Rybarczyk
agnieszka.rybarczyk@put.poznan.pl

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z zakresu podstaw programowania i znać podstawowe metody stosowane przy rozwiązywaniu prostych zadań informatycznych. Student powinien posiadać umiejętność rozwiązywania podstawowych problemów, którymi zajmuje się informatyka, samodzielnego pisania, modyfikowania i testowania programów komputerowych oraz umiejętność pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji. W zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu

1. Zapoznanie się metodyką programowania obiektowego. Nabycie praktycznej umiejętności projektowania i implementacji, uruchomienia i testowania programów obiektowych. 2. Przekazanie studentom podstawowej wiedzy z zakresu programowania obiektowego na przykładzie języka C++ oraz zapoznanie z teoretycznymi i praktycznymi problemami związanymi z metodologią programowania w tym języku. 3. Rozwijanie u studentów umiejętności rozwiązywania prostych problemów i konstruowania prostych algorytmów.

Przedmiotowe efekty uczenia się

Wiedza:
1. ma rozszerzoną i pogłębioną wiedzę z matematyki, fizyki, chemii potrzebną w obszarze technicznym, przydatną do formułowania i rozwiązywania złożonych zadań z zakresu edukacji techniczno-informatycznej [k2_w01].
2. ma wiedzę z zakresu komputerowego wspomagania edukacji technicznej [k2_w09].
3. ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie algorytmów, architektury systemów komputerowych, systemów operacyjnych, technologii sieciowych, języków programowania, grafiki, sztucznej inteligencji, baz danych, wspomagania decyzji, systemów uczących się i inżynierii oprogramowania [k2_w10].

Umiejętności:
1. potrafi wykorzystać nabytą wiedzę matematyczną do opisu procesów, tworzenia modeli oraz zapisu algorytmów [k2_u01].
2. ma umiejętność samokształcenia i potrafi określić kierunki dalszego uczenia się [k2_u03].
3. potrafi pozyskiwać informacje z literatury, baz danych oraz innych źródeł (w języku ojczystym i angielskim), integrować je, dokonywać ich interpretacji i krytycznej oceny, wyciągać wnioski oraz formułować i wyczerpująco uzasadniać opinie [k2_u04].

Kompetencje społeczne:
1. rozumie potrzebę uczenia się przez całe życie. potrafi inspirować i organizować proces uczenia się innych osób [k2_k01].
2. potrafi współdziałać i pracować w grupie, przyjmując w niej różne role [k2_k03].

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekt Forma oceny Kryteria oceny
kształcenia
K2_U01, K2_U03, K2_U04:
• Projekt i sprawozdanie z projektu przygotowywane częściowo w trakcie zajęć, a częściowo po ich zakończeniu; „obrona” przez studenta sprawozdania z realizacji projektu oraz ocena zgodności z przedstawionymi wymaganiami
• Ocena wiedzy i umiejętności związanych z realizacją zadań projektowych / laboratoryjnych poprzez kolokwium; zaliczenie ma charakter pisemny
• Ocena wiedzy i umiejętności wykazanych na zaliczeniu pisemnym o charakterze problemowym

Treści programowe

Program wykładów z przedmiotu obejmuje następujące zagadnienia.
Wprowadzenie do języków i paradygmatów programowania; definicja paradygmatu, omówienie i przedstawienie paradygmatu obiektowego .Przesłanki programowania obiektowego wynikające z analizy źródeł kryzysu oprogramowania. Idea nowego paradygmatu programowania, który wspiera tworzenie programów o wysokiej jakości. Poszukiwanie optymalnego języka programowania i metodyk właściwych dla budowy uniwersalnych modułów programowych do wielokrotnego użytku. Omówienie podejścia obiektowego. Krótkie przedstawienie historii języków obiektowych. Definicje podstawowych pojęć obiektowych: obiekt, atrybuty (zmienne) obiektu, metody obiektu, przesyłanie komunikatów wyzwalających wywołania metod obiektów, interfejsy klas, obiekty jako wystąpienia klas.
Wprowadzenie do języka C++. Różnice między C i C++. Porównanie rozwiązań prostych problemów w sposób funkcjonalny i obiektowy. Definiowanie klas: składowe klasy, statyczne składowe klasy, modyfikatory dostępu. Przykłady definiowania klas obejmujące: definicje konstruktorów i destruktorów klas, operatorów przeciążonych, zmiennych i metod klasowych. Hermetyczność implementacji klas jako mechanizm ograniczania związków między modułami programowymi. Relacja przyjaźni między klasami. Poznanie typów operatorów kopiowania obiektów złożonych. Przeciążanie operatorów, strumieniowe operatory wprowadzania i wyprowadzania danych. Dziedziczenie klas i relacja podtypu między klasami. Definicja nowych cech klas pochodnych, przesłanianie metod i zmiennych. Dziedziczenie: klasy bazowe i klasy pochodne, dziedziczenie wielobazowe, dziedziczenie wirtualne. Funkcje wirtualne, definiowanie, wywoływanie, klasy abstrakcyjne. Obsługa wyjątków. Wzorce funkcji. Wzorce klas. Klasy pojemnikowe.
W ramach zajęć laboratoryjnych studenci zapoznają się dogłębnie z językiem programowania C++. Ćwiczenia polegają na samodzielnym tworzeniu programów zawierających podstawowe konstrukcje

języków obiektowych przedstawianych na wykładach. Dodatkowo, przerabiane są biblioteki systemowe w zakresie: kolekcji, strumieni. Zapoznawanie się z językiem C++ kończy się samodzielną realizacją projektów obejmujących implementację programów.

Metody dydaktyczne

1. Wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy.
2. Ćwiczenia laboratoryjne: ćwiczenia praktyczne, wykonywanie eksperymentów, dyskusja, praca w zespole.

Literatura

Podstawowa

1. Programowanie zorientowane obiektowo, Bertrand Mayer, Helion, Warszawa, 2005
2. Metody obiektowe w teorii i praktyce, Ian Graham, WNT, Warszawa, 2004
3. Język C++, Bjarne Stroustrup, WNT, Warszawa, 1994
4. Thinking in C++, B. Eckel, Helion 2003.
5. Programowanie obiektowe, Peter Coad, Edward Yourdon, Read Me, 1994
6. Analiza obiektowa, Peter Coad, Edward Yourdon, Read Me, 1994
7. Nowoczesne projektowanie w C++, Andrei Alexandrescu, WNT, 2005
8. Symfonia C++, J. Gręboszl, Oficyna Kallimach, Kraków, 2001.

Uzupełniająca

1. Język C++, J. Kisilewicz, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2005.
2. Wprowadzenie do programowania w języku C++, J. Kniat, WPP, Poznań, 1995
3. Pasja C++, J. Gręboszl, Oficyna Kallimach, Kraków, 2001
4. Programowanie w języku C++, J. Kniat , Nakom, Poznań, 2002.
5. Szkoła Programowania Język C++, S. Prata, Robomatic, 2002

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	47	4,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	32	0,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	15	0,00