KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Informatyka [S1EiT1>INF2]

Przedmiot

Kierunek studiów
Elektronika i telekomunikacja

Rok/Semestr
2/3

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
30

Laboratorium
30

Inne (np. online)
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

6,00

Koordynatorzy

dr inż. Michał Sybis
michal.sybis@put.poznan.pl

Wykładowcy

Wymagania wstępne

Podstawowe wiadomości z logiki matematycznej i kombinatoryki. Umiejętność formułowania prostych algorytmów. Potrafi pozyskiwać informacje z literatury oraz innych źródeł w języku polskim lub angielskim; potrafi integrować uzyskane informacje, dokonywać ich interpretacji i wyciągać wnioski. Zna ograniczenia własnej wiedzy i umiejętności, rozumie konieczność dalszego dokształcania się.

Cel przedmiotu

Zapoznanie z podstawami inżynierii oprogramowania. Przedmiot wprowadza kolejne zagadnienia zarówno praktyki programowania obiektowego komputerów w języku C++ jak i projektowania struktur danych i algorytmów oraz analizy ich złożoności obliczeniowej.

Przedmiotowe efekty uczenia się

Wiedza:
1. Podstawowa wiedza teoretyczna i praktyczna w zakresie programowania w językach C i C++, ze szczególnym uwzględnieniem projektowania programów poprawnie zbudowanych, zasad konstruowania oprogramowania obiektowego, wykorzystania szablonów, projektowania złożonych programów oraz wykorzystywania oprogramowania bibliotecznego.

2. Wiedza o podstawowych algorytmach (sortowanie, przeszukiwanie zbiorów danych, metody zachłanne, metody prób i błędów, wybrane algorytmy numeryczne) i strukturach danych (pojemniki, listy jedno i dwukierunkowe, drzewa poszukiwań binarnych, drzewa zrównoważone, grafy i metody ich przeszukiwania, dendryty) wykorzystywanych w codziennej praktyce programisty.
3. Przeglądowa wiedza na temat metod stosowanych w projektowaniu złożonego instrumentarium informatycznego oraz obiektowym projektowaniu specjalizowanych struktur danych.

Umiejętności:
1. Przy projektowaniu oprogramowania student potrafi przeprowadzić analizę problemu z punktu widzenia postępowania algorytmicznego stosując kryteria złożoności obliczeniowej, szybkości działania programu, skalowalności zastosowanych rozwiązań, oraz adekwatności przyjętych metod.
2. Przy projektowaniu oprogramowania student potrafi dokonać właściwej dekompozycji problemu, dokonać wyboru adekwatnych struktur danych, wyodrębnić hierarchię obiektów, zidentyfikować relacje między obiektami i ich reprezentantami w programie.
3. Student potrafi krytycznie zanalizować dostępne oprogramowanie biblioteczne pod kątem zastosowania w realizowanym projekcie oraz zaproponować zasady współpracy w konfiguracji zbiorowego programisty.

Kompetencje społeczne:
1. Zrozumienie potrzeby szerszej popularyzacji wiedzy z zakresu nowoczesnych technik informatycznych.
2. Świadomość możliwości i ograniczeń współczesnej informatyki przy jednoczesnym otwarciu na możliwość zastosowań w nowych dziedzinach życia codziennego, gospodarki, techniki i nauki.
3. Umiejętność formułowania własnych opinii na temat aktualnie stosowanych i dostępnych technologii i rozwiązań w projektowaniu nowoczesnych systemów informatycznych.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Zakres semestru pierwszego (z dwóch):
Wiedza i umiejętności nabyte w trakcie wykładów są weryfikowane podczas egzaminu. Egzamin ma formę pisemną. Składa się 11-13 pytań otwartych, które nie muszą być równo punktowane. Próg zaliczeniowy dla egzaminu pisemnego to 50% możliwych do zdobycia punktów. Możliwe jest również weryfikowanie wiedzy studentów na podstawie projektu, mającego charakter indywidualny lub zespołowy, i jego obrony.
Umiejętności nabyte podczas realizacji zajęć laboratoryjnych są oceniane na podstawie: krótkiego testu na początku zajęć (tzw. wejściówki), ocenie pracy studentów na zajęciach, oraz kolokwium końcowego odbywającego się na końcowych zajęciach (lub dwóch kolokwiów odpowiednio w połowie i na końcowych zajęciach). Wagi poszczególnych kryteriów: wejściówka maksymalnie 25 %, ocena pracy studenta na zajęciach maksymalnie 50 % i kolokwium (suma wyników z obu kolokwiów) o wadze co najmniej 50 %. Próg zaliczeniowy to 50% możliwych do zdobycia punktów.
Zakres semestru drugiego (z dwóch):
Wiedza i umiejętności nabyte w trakcie wykładów są weryfikowane podczas egzaminu. Egzamin ma formę pisemną. Składa się 11-13 pytań otwartych, które nie muszą być równo punktowane. Próg zaliczeniowy dla egzaminu pisemnego to 50% możliwych do zdobycia punktów. Możliwe jest również weryfikowanie wiedzy studentów na podstawie projektu, mającego charakter indywidualny lub zespołowy, i jego obrony.
Umiejętności nabyte podczas realizacji zajęć laboratoryjnych są oceniane na podstawie: krótkiego testu na początku zajęć (tzw. wejściówki), ocenie pracy studentów na zajęciach, oraz kolokwium końcowego odbywającego się na końcowych zajęciach (lub dwóch kolokwiów odpowiednio w połowie i na końcowych zajęciach). Wagi poszczególnych kryteriów: wejściówka maksymalnie 25 %, ocena pracy studenta na zajęciach maksymalnie 50 % i kolokwium (suma wyników z obu kolokwiów) o wadze co najmniej 50 %. Próg zaliczeniowy to 50% możliwych do zdobycia punktów.

Treści programowe

Zakres wykładu w pierwszym semestrze obejmuje podstawy programowania w C++, w tym struktury programu, typy danych, instrukcje sterujące, tablice, funkcje oraz podstawy teorii obliczeń i algorytmów, takich jak sortowanie i przeszukiwanie binarne. Drugi semestr skupia się na programowaniu obiektowym, wprowadzając klasy, obiekty, dziedziczenie, polimorfizm, oraz zaawansowane struktury danych i algorytmy grafowe. Zajęcia laboratoryjne w pierwszym semestrze koncentrują się na praktycznych aspektach podstaw programowania, takich jak operacje na zmiennych, tablicach i funkcjach, oraz podstawowe algorytmy sortowania i przeszukiwania. W drugim semestrze laboratoria rozwijają umiejętności

programowania obiektowego i operacji na dynamicznych strukturach danych, z wykorzystaniem biblioteki STL.

Tematyka zajęć

Zakres wykładu w semestrze pierwszym (z dwóch) obejmuje: struktura programu w języku C++, podstawowe typy danych, konwersja danych, reprezentacja uzupełnieniowa liczb binarnych, operatory i wyrażenia, operacje bitowe, instrukcje sterujące, tablice, funkcje, przekazywanie argumentów, wzorce funkcji, przeciążanie funkcji, teoria obliczeń, algorytmy rekurencyjne i zachłanne, algorytmy sortowania, szybkie algorytmy sortowania, złożoność obliczeniowa, przeszukiwanie binarne, tablica mieszająca, jako struktura danych
Zakres wykładu w semestrze drugim (z dwóch) obejmuje: klasy i obiekty klas, konstruktor, destruktor, przeładowanie operatorów, wskaźniki i dynamiczne przydzielanie pamięci, wzorce klas, podstawowe struktury danych z bazy STL (wektory, listy, stosy, kolejki, drzewa, grafy), idea iteratorów, dziedziczenie, polimorfizm, wzorce klas, programowanie zorientowane obiektowo, nowoczesna inżynieria oprogramowania, odwrotna notacja polska, drzewa binarne, przeszukiwanie drzew binarnych, drzewa AVL, zagadnienia teorii grafów, przeszukiwanie grafów, cykl Eulera, Hamiltona, Algorytm Prima, Kruskala, algorytmy znajdywania najkrótszych ścieżek (alg. Dijkstry, alg. Bellmana-Forda, alg. Floyda).
Zakres zajęć laboratoryjnych w semestrze pierwszym (z dwóch) obejmuje: zapoznanie z ideą programowania i językiem programowania, tworzenie zmiennych i operacje na zmiennych, tworzenie tablic i operacje na tablicach, tworzenie funkcji, przeładowanie funkcji, funkcja rekurencyjna, przekazywanie argumentów do funkcji, metody sortowania, wyszukiwanie binarne, tablice hashujące.
Zakres zajęć laboratoryjnych w semestrze drugim (z dwóch) obejmuje: Podstawy programowania obiektowego w języku C++: tworzenie klas, metod, obiektów, przeciążenie operatorów, dziedziczenie, polimorfizm. Tworzenie dynamicznych struktur danych: lista jednokierunkowa, lista dwukierunkowa, drzewo przeszukiwań binarnych. Wykonywanie operacji na strukturach danych, sortowanie, przeszukiwanie, dodawanie i usuwanie elementów, itp. Podstawy wykorzystania biblioteki STL.

Metody dydaktyczne

Wykład: prezentacja multimedialna, ilustrowana przykładami podawanymi na tablicy.
Laboratoria: ćwiczenia praktyczne - realizacja zadań podanych przez prowadzącego.

Literatura

Podstawowa
1. Jerzy Grębosz, Symfonia C++ : programowanie w języku C++ orientowane obiektowo. T. 1/2/3, 2000
2. Jerzy Grębosz, Pasja C++ : szablony, pojemniki i obsługa sytuacji wyjątkowych w języku C++. T. 1/2, 2004
3. Jerzy Grębosz, Opus Magnum C++11 : programowanie w języku C++. T. 1/2/3, 2018
4. Bruce Eckel, Thinking in C++. Edycja polska,
Uzupełniająca
1. Stephen Prata, Język C++.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Wprowadzenie do algorytmów, WNT, Warszawa, 2004.

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	300	12,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	150	6,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	150	6,00