KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Wprowadzenie do informatyki [S1Inf1>WdIang]

Przedmiot

Kierunek studiów
Informatyka

Rok/Semestr
1/1

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
angielski

Forma studiów
stacjonarne

Wymagalność
obieralny

Liczba godzin

Wykład
30

Laboratorium
16

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

4,00

Koordynatorzy

prof. dr hab. inż. Jerzy Nawrocki
jerzy.nawrocki@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z matematyki zgodnie z podstawą programową kształcenia ogólnego dla szkół II stopnia oraz umiejętność pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji/mieć gotowość do podjęcia współpracy w ramach zespołu. Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu

Zapoznanie studentów z podstawowymi obszarami informatyki, a także wsparcie w nabyciu podstawowych umiejętności programistycznych. W ramach przedmiotu prezentowane są paradygmaty programowania oraz podstawowe modele, pojęcia i techniki wykorzystywane w informatyce. Opanowanie prezentowanego materiału zapewnia przyszłym informatykom podstawy niezbędne w dalszych studiach oraz w przyszłej pracy zawodowej.

Przedmiotowe efekty uczenia się

Wiedza:

1. ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną w zakresie kluczowych zagadnień informatyki, oraz wiedzę szczegółową w zakresie wybranych zgadnień tej dyscypliny nauki
2. ma wiedzę o istotnych kierunkach rozwoju i najważniejszych osiągnięciach informatyki oraz innych pokrewnych dyscyplin naukowych, w szczególności elektroniki, telekomunikacji oraz automatyki i robotyki
3. ma podstawową wiedzę o cyklu życia systemów informatycznych, zarówno sprzętowych jak i programowych, a w szczególności o zachodzących w nich kluczowych procesach
4. ma wiedzę nt. kodeksów etycznych dotyczących informatyki, jest świadomy zagrożeń związanych z przestępczością elektroniczną, oraz rozumie specyfikę systemów krytycznych ze względów bezpieczeństwa (ang. mission-critical systems)
5. ma podstawową wiedzę nt. patentów, ustawy prawo autorskie i prawa pokrewne oraz ustawy o ochronie danych osobowych oraz transferu technologii w szczególności w odniesieniu do rozwiązań informatycznych

Umiejętności:
1. potrafi odpowiednio posługiwać się technikami informacyjno-komunikacyjnymi, znajdującymi zastosowanie na różnych etapach realizacji przedsięwzięć informatycznych
2. potrafi dostrzec w procesie formułowania i rozwiazywania zadań informatycznych również aspekty pozainformatyczne, w szczególności kwestie społeczne, prawne i ekonomiczne
3. potrafi organizować, współdziałać i pracować w grupie, przyjmując w niej różne role oraz potrafi odpowiednio określić priorytety służące realizacji określonego przez siebie lub innych zadania

Kompetencje społeczne:
1. rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe
2. ma świadomość znaczenia wiedzy w rozwiązywaniu problemów inżynierskich oraz zna przykłady i rozumie przyczyny wadliwie działających systemów informatycznych, które doprowadziły do poważnych strat finansowych, społecznych lub też do poważnej utraty zdrowia, a nawet życia

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca:
a) w zakresie wykładów:
- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach
b) w zakresie laboratoriów:
- na podstawie oceny bieżącego postępu realizacji zadań
Ocena podsumowująca:
Sprawdzanie założonych efektów kształcenia realizowane jest przez:
- ocenę przygotowania studenta do poszczególnych sesji zajęć laboratoryjnych oraz ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych,
- ocenianie ciągłe, na każdych zajęciach (odpowiedzi ustne) - premiowanie przyrostu umiejętności posługiwania się poznanymi zasadami i metodami,
- ocenę wiedzy i umiejętności związanych z realizacją zadań laboratoryjnych poprzez kolokwium/a w semestrze,
- ocenę wiedzy i umiejętności wykazanych na egzaminie pisemnym po zakończeniu semestru.
Uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:
- omówienia dodatkowych aspektów zagadnienia,
- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,
- uwagi związane z udoskonaleniem materiałów dydaktycznych,
- wskazywanie trudności percepcyjnych studentów umożliwiające bieżące doskonalenia procesu dydaktycznego.

Treści programowe

Przedmiot obejmuje następujące treści:
Programowanie imperatywne i język C i Python (proste programy, deklarowanie i przetwarzanie zmiennych prostych, wczytywanie i drukowanie zmiennych, instrukcja warunkowa, instrukcje pętli, tablice, rekordy, koncepcja modularyzacji, funkcje).
Układy cyfrowe (algebra Boole'a, bramki logiczne, sumator, przerzutnik, rejestr, realizacja operacji logicznych w języku C). Architektura von Neumanna i język asemblera (elementy architektury procesora,

podstawowe rozkazy, proste programy, system szesnastkowy, liczby całkowite i reprezentacja liczb ujemnych, model von Neumanna, rozkazy skoku, wywoływanie podprogramu z wykorzystaniem instrukcji call/ret).
Przetwarzanie tekstu w języku C i Python (reprezentacja łańcuchów znaków, wczytywanie, przetwarzanie i drukowanie łańcuchów tekstu, elementy standardowej biblioteki wejścia/wyjścia) oraz w języku AWK (koncepcja języka AWK, proste programy, wzorce, wyrażenia regularne, zmienne i funkcje standardowe).
Podstawy programowania obiektowego w języku C++ i Python.
Programowanie współbieżne i równoległe (ewolucja systemów operacyjnych, procesy, implementacja wątków w językach C i Python, podział czasu procesora, interferencja obliczeń, biblioteka pthreads, zarządzanie wątkami, synchronizacja dostępu, semafory, problem producent-konsument, problem czytelników i pisarzy, problem zakleszczenia).
Bazy danych (ewolucja, relacyjny model danych, język SQL, zapytania, projekcja, selekcja, zarządzanie danymi, wzorce, zagnieżdżanie zapytań).
Sztuczna inteligencja i język naturalny (test Turinga, program ELIZA, części mowy i niejednoznaczności, analiza leksykalna i generator lex, gramatyki formalne, translacja, gramatyki bezkontekstowe).
Sieci komputerowe (architektura sieci komputerowej, komunikacja w sieci, stos protokołów internetowych, pakiety, zagnieżdżanie danych, gniazda TCP, serwisy webowe).
Złożoność obliczeniowa i metody numeryczne (dynamiczny przydział pamięci, dynamiczne struktury danych, grafy, listy, stosy, kolejki, sortowanie stogowe, złożoność obliczeniowa, reprezentacja liczb rzeczywistych, stabilność numeryczna, algorytmy numeryczne).
Mikrokontrolery i komputerowe systemy sterowania (specyfika systemów czasu rzeczywistego, architektura systemu sterowania, manipulacja bitami, komunikacja z urządzeniami zewnętrznymi, architektura Arduino, podstawy programowania Arduino w języku C).
Inżynieria oprogramowania (analiza problemu, wymagania funkcjonalne i przypadki użycia, wybrane diagramy języka UML, wstępny projekt interfejsu użytkownika, wymagania pozafunkcjonalne, architektura, implementacja i testowanie).
Profesjonalizm w informatyce (zasady skutecznego działania, obopólna wygrana, synergia).

Tematyka zajęć

Wykłady i powiązane laboratoria obejmują następujące tematy:
* Porównanie C i Pythona: proste programy, deklarowanie i przetwarzanie prostych zmiennych, odczytywanie i
wyświetlanie zmiennych, instrukcje warunkowe, instrukcje pętli, tablice, rekordy, koncepcje modularizacji i funkcje;
* Podstawowe układy cyfrowe: algebra Boole'a, bramki logiczne, sumatory, dekodery i multipleksery, przerzutniki, rejestry,
cyfrowe układy biochemiczne;
* Budowa prostego komputera: architektura von Neumanna, mikroprocesory kontra mikrokontrolery, maszyna Turinga;
* Język asemblera NASM: instrukcje całkowite, proste programy, system szesnastkowy, liczby całkowite, reprezentacja liczb ujemnych, instrukcje skoku i ich implementacja;
* Podprogramy w C i Pythonie: przekazywanie parametrów, funkcja jako parametr, implementacja podprogramów w języku asemblera;
* Przetwarzanie tekstu w C i Pythonie: reprezentacja ciągu, przetwarzanie tekstu, standardowe elementy biblioteki wejścia/wyjścia, wyrażenia regularne, AWK;
* Podstawy programowania obiektowego w C++ i Pythonie: rekordy kontra klasy, dziedziczenie, dynamiczna alokacja pamięci, listy;

Metody numeryczne: reprezentacja liczb rzeczywistych - standard IEEE 754-1985, stabilność numeryczna, metoda Newtona i pierwiastek kwadratowy, szereg Maclaurina i funkcje trygonometryczne;

Złożoność obliczeniowa: notacja dużego O, rodzaje złożoności obliczeniowej, złożoność sortowania na stosie, brutalna siła i jej złożoność, problemy optymalizacji i decyzji, transformacja wielomianowa problemów, problem zatrzymania;

Bazy danych: przetwarzanie plików, relacyjny model danych, język SQL, projekcja, selekcja, operacja łączenia,
utrzymywanie danych, wzorce, zagnieżdżanie zapytań, grupowanie, manipulacja bazą danych z Pythona;

Uczenie maszynowe: główne podejścia do uczenia maszynowego, koncepcja uczenia nadzorowanego, entropia informacji, metoda ID3;
* Programowanie równoległe: ewolucja systemów operacyjnych, procesy kontra wątki, współdzielenie czasu, interferencja obliczeniowa, standard POSIX, biblioteka pthreads, zarządzanie wątkami, synchronizacja dostępu, semafory, problem producenta-konsumenta, problem czytelników-pisarzy, impas;
Sieci komputerowe: warstwowa architektura sieci komputerowych, stos protokołów internetowych, pakiety, protokół TCP/IP i gniazda TCP, protokół HTTP, język HTML i JavaScript, tworzenie usług sieciowych;
* Inżynieria oprogramowania: analiza problemów, wymagania funkcjonalne i przypadki użycia, wybrane diagramy UML,
projektowanie interfejsu użytkownika, wymagania niefunkcjonalne, architektura oprogramowania, wdrażanie i testowanie;
Profesjonalizm IT: zasady efektywności, zasada win-win, synergia, licencje open source i oprogramowania, patenty.

Metody dydaktyczne

1. wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań, demonstracja narzędzi programistycznych,
2. ćwiczenia laboratoryjne: rozwiązywanie zadań, dyskusja.

Literatura

Podstawowa:
1. Język ANSI C, B. W. Kernighan, D. M. Ritchie, Wydawnictwa Naukowo-Techniczne, dowolne wydanie
2. Układy cyfrowe, B. Wilkinson, WKiŁ, Warszawa, 2000
3. Wprowadzenie do przetwarzania tekstów w języku AWK, J. Nawrocki, W. Complak, ProDialog 2, 23-46, Poznań, 1994.
4. Sieci komputerowe, J.F. Kurose, K.W. Ross, Helion, 2006

Uzupełniająca:
1. 7 nawyków skutecznego działania, S. Covey, Rebis, 2003
2. Język C. Szkoła programowania, Wydanie VI, Prata S., Helion, 2016

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	100	4,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	48	2,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	52	2,00