INFORMATYKA

Efekty uczenia się i treści programowe zajęć:

Nazwa zajęć**: Algebra liniowa i geometria**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Posiada wiedzę z teorii podstawowych struktur algebraicznych.
- Zna podstawy algebry macierzy.
- Potrafi zbadać rozwiązalność układów liniowych równań algebraicznych, potrafi je rozwiązywać za pomocą operacji elementarnych. Zna algorytm eliminacji Gaussa.
- Ma wiedzę na temat podstawowych własności i metod obliczania wyznaczników. Potrafi ją wykorzystać w rozwiązywaniu układów równań.
- Ma wiedzę na temat podstaw teorii przestrzeni liniowych, potrafi badać własności liniowych kombinacji wektorów. Zna pojęcie i własności przekształcenia liniowego, umie wyznaczyć macierz takiego przekształcenia.
- Potrafi formułować i rozwiązywać zagadnienie własne, zna własności spektralne wybranych klas macierzy.
- Zna podstawy teorii form kwadratowych, potrafi zweryfikować ich określoność i wyznaczyć ich postacie kanoniczne.
- Posiada wiedzę dotyczącą podstaw przestrzeni euklidesowych.

Treści programowe dla zajęć:
- Podstawowe struktury algebraiczne: grupa, pierścień, ciało. Weryfikacja własności grupy. Grupy permutacji, pierścień wielomianów, algorytm Euklidesa, NWD.
- Algebra macierzy, macierz transponowana, macierz hermitowsko-sprzężona. Macierze elementarne, macierze odwracalne, algorytm wyznaczanie macierzy odwrotnej do macierzy trójkątnej.
- Rachunek macierzowy, macierz transponowana, macierz permutacyjna, macierz hermitowsko-sprzężona. Iloczyn skalarny wektorów. Macierze trójkątne, odwracanie macierzy trójkątnych.
- Wyznaczanie postaci zredukowanej i całkowicie zredukowanej, rząd macierzy, Twierdzenie Kroneckera-Capellego. Rozwiązywanie układów liniowych równań algebraicznych za pomocą algorytmu eliminacji Gaussa
- Operacje elementarne, postać zredukowana, rząd macierzy, twierdzenie Kroneckera-Capellego. Odwracanie macierzy za pomocą operacji elementarnych. Układy o macierzach kwadratowych i prostokątnych. Układy równań z macierzami o elementach z ciał skończonych.
- Własności wyznacznika, minory, twierdzenia Laplace'a i Cauchy'ego, obliczanie wyznacznika z wykorzystaniem rozkładu macierzy na iloczyn czynników trójkątnych.
- Rząd macierzy z wykorzystaniem wyznaczników. Macierz dołączona i obliczanie macierzy odwrotnej do macierzy nieosobliwej. Wzory Cramera.
- Obliczanie wyznacznika: z definicji, w oparciu o schemat Sarrusa, z twierdzenia Laplace'a, z rozkładu na czynniki trójkątne (LU). Pojęcie minora i badanie rzędu. Metoda minorów obejmujących. Macierz dołączona i obliczanie macierzy odwrotnej. Wzory Cramera - porównanie z metodą eliminacji Gaussa w kontekście obliczeń na dużych układach równań.
- Przestrzenie liniowe, wektory, liniowa niezależność wektorów, baza i wymiar przestrzeni liniowej, podobieństwo macierzy, macierz przejścia od bazy do bazy. Przekształcenie liniowe, macierz przekształcenia liniowego. Jądro i obraz przekształcenia linowego.
- Przykłady przestrzeni liniowych. Operacje na wektorach. Badanie liniowej niezależności wektorów. Baza przestrzeni liniowej i macierz przejścia od bazy do bazy. Przekształcenie liniowe, macierz przekształcenia liniowego, Wyznaczanie jądra i obrazu przekształcenia liniowego. Podobieństwo macierzy
- Podprzestrzenie niezmiennicze, wektory i wartości własne, wielomian charakterystyczny. Diagonalizacja macierzy.
- Rozwiązywanie zadania własnego. Zależności pomiędzy wyznacznikiem i śladem macierzy, a wartościami własnymi. Zastosowanie diagonalizacji macierzy.
- Formy kwadratowe, metoda Lagrange'a i Jacobiego sprowadzania do postaci kanonicznej, badanie określoności. Przestrzenie liniowe z iloczynem skalarnym, przekształcenia: ortogonalne, unitarne, samosprzężone.

Nazwa zajęć**: Algorytmy kombinatoryczne**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Rozumie specyfikę algorytmów kombinatorycznych.
- Potrafi przeanalizować złożoność obliczeniową algorytmów kombinatorycznych oraz wskazać sposoby jej poprawienia.
- Potrafi strukturom dyskretnym przyporządkować liczby rankingowe w różnych porządkach.
- Potrafi dla danego problemu kombinatorycznego znaleźć różne algorytmy rozwiązujące go.

Treści programowe dla zajęć:

- Kombinatoryczne struktury i algorytmy.
- Generowanie klasycznych struktur kombinatorycznych (zbiory, podzbiory, permutacje, kody Graya).
- Zaawansowane generowanie struktur kombinatorycznych (podziały liczb, podziały zbiorów, liczby Bella i Strirlinga, rodziny Catalana).
- Algorytmy nawrotów (szacowanie wielkości drzewa, funkcje ograniczające).
- Algorytmy heurystyczne (strategie wspinania, schładzania, listy tabu, ewolucyjne).

Nazwa zajęć**: Analiza matematyczna**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Potrafi określić ograniczenia arytmetyki komputerowej w stosunku do pełnej teorii aksjomatycznej liczb rzeczywistych i dobrać odpowiednie metody dla unikania problemów.
- Potrafi wprawnie posługiwać się pojęciem granicy ciągu oraz sprawdzać wykonalność obliczeń na komputerze. Zna podstawowe metody obliczania granic.
- Potrafi operować ciągami rekurencyjnymi, badać ich zbieżność i granice. Potrafi określić podstawowe wady i zalety takich ciągów w zastosowaniach informatycznych.
- Potrafi badać zbieżność szeregów liczbowych i potęgowych. Stosuje podstawowe kryteria zbieżności i zna ograniczenia obliczeń numerycznych.
- Potrafi badać ciągłość funkcji i granice funkcji w punkcie. Stosuje w praktyce informatycznej własności funkcji ciągłych (np. własność Darboux).
- Potrafi rozwijać funkcje w szeregi potęgowe i zna ich zastosowania.
- Potrafi wykorzystać pojęcie metryki w różnych zastosowaniach informatycznych.
- Potrafi stosować pojęcie pochodnej, obliczać pochodne i zna podstawowe zastosowania w informatyce, np. w analizie błędu.
- Potrafi stosować pojęcie całki i całki Riemanna w odpowiednich sytuacjach. Prowadzi obliczenia analityczne i obliczenia numeryczne całek. Zna podstawowe zastosowania takich całek w informatyce.
- Potrafi dostrzec zastosowania całek niewłaściwych w zastosowaniach informatycznych i badać ich zbieżność.

Treści programowe dla zajęć:

- Cele nauczania analizy dla informatyków.
- Szkic teorii aksjomatycznej liczb rzeczywistych, w tym kresy, zapis dziesiętny liczb rzeczywistych. Liczby wymierne. Potęga o wykładniku rzeczywistym. Pierwiastek. Uwagi o arytmetyce komputerowej.
- Arytmetyka komputera, zero (przykłady w różnych programach). Kresy zbiorów liczbowych. (na ćwiczeniach: proste zadania na obliczanie kresów, postacie niedziesiętne liczb rzeczywistych.)
- Ciągi liczbowe: granice właściwe i niewłaściwe. Zbieżność i bezwzględna zbieżność. Ciągi monotoniczne. Podciągi, punkty skupienia i tw. Bolzano-Weierstrassa. Warunek Cauchy'ego i zupełność. Pozostałe informacje o zbieżności ciągów. Liczba e. Ciągi zadane rekurencyjnie w informatyce.
- Granice ciągów, algorytmy obliczania granic
- (problem zbieżności). Interpretacja geometryczna (aplety) (na ćwiczeniach: obliczanie granic, punkty skupienia ciągów).
- Szeregi liczbowe. Suma szeregu. Zbieżność i bezwzględna zbieżność szeregu. Kryteria zbieżności. Podstawy teorii szeregów geometrycznych i potęgowych.
- Wprowadzenie do obliczeń sum szeregów, problem przybliżonego obliczania sumy szeregu. (na ćwiczeniach: obliczanie sum a badanie ich zbieżności, kryterium Leibnitza i reszty szeregu, funkcje = sumy szeregów potęgowych.
- Granica i ciągłość funkcji jednej zmiennej rzeczywistej. Punkt skupienia zbioru. Granica funkcji w punkcie. Ciągłość funkcji (np. spline) i ciągłość jednostajna funkcji.
- Własność Darboux. Twierdzenie Weierstrassa o kresach. Ciąg dalszy informacji o funkcjach zadanych szeregiem potęgowym. Wybrane funkcje elementarne. Funkcje zadane szeregami potęgowymi w informatyce (np.
- funkcje błędu). Metryki i przykłady ich zastosowań w informatyce.

- Wybrane szeregi potęgowe i ich obliczanie. Błąd obliczeniowy (na ćwiczeniach: kilka granic funkcji i badanie ciągłości funkcji zadanych klamrowo, wykorzystanie własności Darboux do obliczania miejsc zerowych równań nieliniowych).
- Rachunek różniczkowy funkcji jednej zmiennej rzeczywistej. Pochodna i jej sens geometryczny. Zastosowania w informatyce (m.in. podstawy interpolacji, funkcje spline). Interpretacja geometryczna pochodnej. Liniowe przybliżanie funkcji (lokalne). Rola wzoru Taylora w szacowaniu błędów. Dla zainteresowanych: funkcje wielu zmiennych i ich zastosowania w informatyce.
- Obliczanie prostych pochodnych, sprawdzanie monotoniczność funkcji i szukanie ekstremu, wzory Taylora dla wybranych funkcji.
- Rachunek całkowy funkcji jednej zmiennej. Funkcja pierwotna i całka nieoznaczona. Podstawowe metody całkowania. Całka Riemanna i jej zastosowania w informatyce (podstawy całkowania numerycznego). Całka niewłaściwa i jej zastosowania w informatyce.
- Obliczenia numeryczne wybranych całek. Przegląd porównawczy metod.

Nazwa zajęć**: Analiza i projektowanie obiektowe**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna rolę analizy i projektowania obiektowego w cyklu życiu oprogramowania.
- Zna podstawowe techniki i metody analizy i projektowania obiektowego.
- Potrafi wykorzystać UML w analizie i projektowaniu obiektowym.
- Potrafi zebrać wymagania w postaci przypadków użycia.
- Potrafi zbudować model wiedzy dziedzinowej w oparciu o wymagania zapisane w postaci przypadków użycia.
- Potrafi wykorzystać diagramy interakcji w procesie projektowania obiektowego.
- Umie przypisać obiektom odpowiedzialności z wykorzystaniem odpowiednich technik i metod.
- Potrafi zbudować projektowy diagram klas.
- Potrafi wykorzystać wzorce projektowe w procesie projektowania obiektowego.

Treści programowe dla zajęć:
- Analiza i projektowanie obiektowe w cyklu życia oprogramowania.
- Programowanie obiektowe, a analiza i projektowanie obiektowe.
- Określanie wymagań przy pomocy przypadków użycia.
- Analiza obiektowa - model wiedzy dziedzinowej.
- Analiza obiektowa - wprowadzanie asocjacji i atrybutów.
- Diagramy interakcji w procesie projektowania obiektowego.
- Projektowanie obiektowe - przypisywanie obiektom odpowiedzialności.
- Projektowy diagram klas.
- Wzorce projektowe.

Nazwa zajęć**: Algorytmy i struktury danych**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna i stosuje podstawowe konstrukcje algorytmiczne, zapisuje je w pseudokodzie i wybranym języku programowania.
- Wykorzystuje procedury i funkcje do formułowania algorytmów, ma wiedzę w zakresie znaczenia i wykorzystania pojęcia rekurencji.
- Zna i stosuje proste i złożone struktury danych, w tym struktury dynamiczne.
- Zna podstawowe techniki projektowania algorytmów i stosuje wiedzę matematyczną do formułowania i rozwiązywania prostych zadań algorytmicznych.
- Konstruuje i implementuje w wybranym języku programowania algorytmy dla średnio zaawansowanego problemu algorytmicznego.
- Ocenia poprawność i złożoność czasową algorytmów, potrafi krytycznie ocenić skonstruowany algorytm.
- Ma świadomość ważności algorytmiki w informatyce i rozumie potrzebę dalszego kształcenia algorytmicznego.

Treści programowe dla zajęć:
- Język algorytmiczny. Pojęcie zmiennej, instrukcja przypisania, instrukcje warunkowe, iteracje, operatory specjalne.
- Pojęcie struktury tablicowej. Przykłady i implementacje prostych problemów algorytmicznych na tablicach 1 i 2-wymiarowych, wyszukiwanie liniowe i binarne.
- Pojęcie procedury. Deklaracja, parametry formalne, wywołanie, przykłady prostych procedur i funkcji.
- Rekurencja. Pojęcie rekurencji, przykłady procedur rekurencyjnych, programowanie dynamiczne.

- Algorytmy sortowania. Sortowanie przez wstawianie, bąbelkowe, przez scalanie, szybkie, przez zliczanie (elementy różne, przypadek ogólny)
- Analiza algorytmów. Notacja asymptotyczna, złożoność optymistyczna, pesymistyczna i średnia, twierdzenie o rekurencji uniwersalnej, klasy złożoności. Poprawność konstrukcji wybranych algorytmów .
- Stosy, kolejki, listy. Pojęcie zbioru dynamicznego. Tablicowa implementacja stosu i operacje na stosie. Tablicowa implementacja kolejki i operacje kolejkowe. Lista dwukierunkowa z dowiązaniami. Operacje na listach. Listy z wartownikiem.
- Pojęcia teorii grafów. Graf prosty, drzewa i ich podstawowe własności, drzewa ukorzenione. Grafy skierowane ważone. Reprezentacja grafów w komputerze.
- Kopce. Podstawowe operacje na kopcach binarnych, sortowanie przez kopcowanie. Implementacja kolejki priorytetowej.
- Drzewa wyszukiwań binarnych. Podstawowe własności, operacje słownikowe, przechodzenie drzewa BST.
- Algorytmy grafowe. Przeszukiwanie grafu wszerz i w głąb, znajdowanie najkrótszych ścieżek.
- Metoda z nawrotami i metoda zachłanna. Problem n królowych, problem plecakowy, problem wyboru zajęć.
- Problemy trudne obliczeniowo. Klasa P i NP, problemy NP-trudne i NP-zupełne.

Nazwa zajęć**: Bazy danych**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Rozumie istotę relacyjnych baz danych; zna podstawowe cechy i zadania systemu zarządzania relacyjną bazą danych; ma świadomość istnienia innych, pozarelacyjnych, modeli danych; potrafi dobrać odpowiednie rozwiązanie do rzeczywistego problemu
- Zna rolę systemu baz danych w systemie informatycznym i jego przykładowe architektury; zna zasady prawidłowego projektowania relacyjnej bazy danych i systemu bazodanowego.
- Rozumie potrzebę normalizacji schematu, nakładania ograniczeń integralnościowych, stosowania odpowiednich poziomów izolacji.
- Wykonuje podstawowe operacje na bazie danych z wykorzystaniem języka SQL; zna metody optymalizacji wykonywania zapytań; potrafi zaprojektować i zaimplementować prosty system bazodanowy.
- Zna fizyczną strukturę zapisu danych w bazie, zna znaczenie typu variable zmiennych, zna różnice między indeksami klastrowanymi a nieklastrowanymi.
- Zna pojęcie i własności transakcji w bazie danych; rozumie trudności wynikające ze współbieżnego wykonywania transakcji i potrafi zastosować odpowiednie rozwiązanie.
- Zna podstawowe pojęcia i własności nierelacyjnych baz danych, rozumie problemy związane z zapisem i odczytem danych w bazach rozproszonych i rozumie model map-reduce agregowania danych.

Treści programowe dla zajęć:

- Modele zapisu bazy danych (Flat File, BDAM, ISAM, modele baz danych (hierarchiczny, sieciowy, relacyjny, obiektowy).
- Relacje dwuargumentowe i wieloargumentowe, algebra relacji, operacje mnogościowe i relacyjne, atrybut, krotka, klucz relacji, implementacja relacji w postaci tabeli dwuwymiarowej.
- Projektowanie relacyjnych baz danych: anomalie,
- funkcyjna zależność atrybutów, postacie normalne, model ER, encja, atrybut, związek, przykłady notacji, diagramy ERD.
- Język SQL: podzbiory DDL, DML i DCL, typy danych,
- złączenia tabel, aliasy, ograniczenia (constraints), type TIMESTAMP, UNIQUEIDENTIFIER, podzapytania; skrypty SQL: zmienne, instrukcje warunkowe, pętle, widoki, procedury, funkcje.
- Fizyczna struktura zapisu danych w bazie relacyjnej MSSQL: strony, zakresy, wielkość strony, podział stron, typy variable zmiennych, unicode, indeksy klastrowane i nieklastrowane.
- Własności ACID transakcji, anomalie, poziomy izolacji, operacje niekonfliktowe i konfliktowe, algorytm blokowania dwufazowego.
- Bazy rozproszone, partycjonowanie, problemy odczytu i zapisu danych, twierdzenie Brewera, bazy szerokokolumnowe, bazy dokumentowe, BASE, model map-reduce.
- Elementy języka DML: tworzenie zapytań pobierających dane z bazy, funkcje kolumnowe i wierszowe, grupowanie danych.
- Elementy języka DML: złączenia tabel (wewnętrzne, zewnętrzne), podzapytania.
- Elementy języka DDL: tworzenie tabel, widoków, procedur i funkcji, tworzenie wyzwalaczy.
- Projektowanie baz danych: usuwanie anomalii.

Nazwa zajęć**: Bezpieczeństwo systemów mobilnych**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Posługuje się podstawowymi pojęciami kryptologii, zna proste metody służące zapewnianiu poufności danych.
- Zna techniki zabezpieczania aplikacji przed atakami, potrafi praktycznie stosować podstawowe mechanizmy zapewniające poufność danych.
- Zdaje sobie sprawę z ograniczeń biometrii, potrafi wykorzystywać czytniki informacji biometrycznej na terminalach abonenckich do zabezpieczania aplikacji.
- Potrafi dostosowywać mechanizmy uwierzytelniania użytkownika do UX (user experience) aplikacji mobilnych.
- Zna architekturę współcześnie używanych platform mobilnych z punktu widzenia bezpieczeństwa.
- Jest świadomy współczesnych zagrożeń dotykających systemy mobilne i potrafi samodzielnie uzupełniać wiedzę w tym zakresie.
- Posiada podstawową wiedzę o zabezpieczeniach sprzętowych używanych w systemach mobilnych.

Treści programowe dla zajęć:
- Podstawowe zasady bezpieczeństwa, model CIA/AIC, poufność danych, terminologia, proste szyfry, tryby szyfrowania. Środowisko programistyczne wybranej platformy mobilnej. Analiza kodu źle napisanych aplikacji.
- Przechowywanie haseł, haszowanie, key stretching, tablice tęczowe, proste ataki. Przykład aplikacji używającej uwierzytelniania hasłem.
- Uwierzytelnianie wieloskładnikowe. S/KEY, HOTP, TOTP, FIDO U2F. Aplikacje wykorzystujące wiele ścieżek uwierzytelniania.
- Biometria – ocena parametrów, krzywe ROC, czytniki odcisków palców, tęczówki, rozpoznawanie twarzy. Podstawowe ataki. Aplikacja wymagająca odcisku palców.
- Struktura GSM i jego cechy bezpieczeństwa. Słabości SS7. Ataki man-in-the-middle. IMSI catching. Szyfr A5/1.
- Podatności na ataki side channel. Budowa pamięci RAM, atak Drammer i pokrewne ataki na struktury pamięci.
- Elementy bezpieczeństwa radiowego. Ataki TEMPEST. TEMPEST optyczny.
- Architektura bezpieczeństwa systemu Android OS i porównanie architektur innych platform mobilnych. Bezpieczeństwo komunikacji międzyprocesowej wybranej platformy mobilnej.
- Bezpieczeństwo sprzętowe współczesnych terminali abonenckich. TEE/REE, bezpieczny rozruch, KeyStore/Keychain, Global Platform, TPM.
- Źródła wiedzy o podatnościach. Samodzielna interpretacja opisów podatności.

Nazwa zajęć**: Człowiek wobec wyzwań globalizacji**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna pojęcie globalizacji w różnych ujęciach dyscyplin naukowych.
- Zna strukturę psychiki człowieka, proces jej kształtowania i konsekwencje oddziaływań w globalnym świecie.
- Zna historyczne uwarunkowania globalizacji.
- Potrafi wskazać kraje rdzenia, półperyferyjne i peryferyjne w kontekście globalizacji.
- Potrafi wskazać szanse i nierówności społeczne wynikające z globalizacji.
- Potrafi opisać postać globalnego człowieka jako konsekwencję życia w zglobalizowanym świecie.
- Wie jakie mają znaczenie państwa narodowe w globalnym świecie.

Treści programowe dla zajęć:
- Analiza pojęcia: globalizacja w ujęciu socjologicznym, ekonomicznym, kulturowym, cyfrowym, informacyjnym.
- Struktura psychiki człowieka, jej kształtowanie i konsekwencje oddziaływań w globalnym świecie.
- Historyczne uwarunkowania globalizacji. Pokój Westfalski (1648 r.) i traktaty pokojowe po II wojnie światowej w drodze do budowania świata globalnego.
- Kraje rdzenia, a kraje półperyferyjne i peryferyjne wobec globalizacji.
- Globalizacja a szanse i nierówności społeczne.
- Postać globalnego człowieka jako konsekwencja życia w zglobalizowanym świecie.
- Znaczenie państw narodowych w globalnym świecie.

Nazwa zajęć**: Dynamika procesów grupowych**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna pojęcia związane z grupą oraz posiada umiejętność pracy z grupą.
- Zna charakterystykę cech składowych dynamiki procesu grupowego.

- Zna cztery fazy procesu grupowego.
- Potrafi radzić sobie z oporem grupowym w pracy zespołowej.

Treści programowe dla zajęć:

- Analiza podstawowych pojęć: grupa, proces grupowy. Interdyscyplinarne ujęcie pojęcia: grupa.
- Charakterystyka cech składowych dynamiki procesu grupowego.
- Prezentacja czterech faz procesu grupowego.
- Opór grupowy i warunki minimalizowania jego skutków w pracy z grupą.

Nazwa zajęć**: E-gospodarka: narzędzia i bezpieczeństwo**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Posługuje się podstawowymi pojęciami kryptologii, zna proste metody służące zapewnieniu poufności danych.
- Zna w zakresie podstawowym aktualny stan prawny w
- zakresie podpisu elektronicznego, elektronicznego obrotu dokumentami.
- Rozumie zasady działania infrastruktury klucza publicznego, posiada umiejętność jej implementacji i posługiwania się nią, zarówno w aspektach prawnych jak i technicznych.
- Zna i potrafi rozpoznawać zabezpieczenia fizyczne oraz metody fizycznej realizacji zabezpieczeń cyfrowych stosowane do środków pieniężnych i dokumentów.
- Zna problematykę znakowania czasem, potrafi cyfrowo znakować czasem dokumenty.
- Zdaje sobie sprawę z wpływu uwarunkowań na sposób współpracy podmiotów i wynikającą z nich potrzebę stosowania zabezpieczeń różnego rodzaju. Potrafi przewidywać i opisywać proste cechy emergentne.
- Rozumie i potrafi stosować proste algorytmy sprawdzania danych księgowych wykrywające oszustwa.
- Zna problematykę kryptowalut, rozumie działanie systemów proof-of-work i łańcucha bloków (blockchain), potrafi się posługiwać klientem protokołu Bitcoin.
- Rozumie funkcje i cele obrotu giełdowego. Zna mechanizmy giełdowe i specyfikę architektury ich systemów informatycznych. Posługuje się podstawową terminologią rynków finansowych. Rozumie treści dokumentów giełdowych, np. Prospektów emisyjnych.

Treści programowe dla zajęć:

- Podstawowe pojęcia kryptologii, poufność danych, terminologia, proste szyfry.
- Algorytmy klucza publicznego, podpis elektroniczny w prawie polskim.
- Infrastruktura klucza publicznego, modele scentralizowany i zdecentralizowany, certyfikaty, szyfrowana poczta GPG, podstawy zarządzania kluczami.
- Zabezpieczenia fizyczne oraz cyfrowe banknotów i druków.
- Pomiar i synchronizacja czasu w systemach teleinformatycznych, znakowanie czasem, notariat cyfrowy. Zegar Lamporta, drzewa Merklego. Czas urzędowy w Polsce.
- Historia zabezpieczeń obrotu gospodarczego i handlu, koncepcje współczesnej gospodarki, polityczne mechanizmy zabezpieczania obrotu gospodarczego.
- Proste gry ekonomiczne - “Chłopska szkoła biznesu”, “dwie trzecie”, etc. Rozwój współpracy wynikający z uwarunkowań początkowych. Próby przewidywania działań systemów emergentnych.
- Algorytmy wykrywania oszustw w sprawozdaniach i danych księgowych – prawo Benforda.
- Protokół i sieć Bitcoin. Protokoły proof-of-work. Poprzednie idee pieniędzy elektronicznych – schemat Chauma.
- Zasady działania giełdy i rodzaje zleceń. Warset, UTP, trading algorytmiczny i o wysokiej częstotliwości. Rynki finansowe. Dokumenty związane z obrotem giełdowym. Infrastruktura giełd.

Nazwa zajęć**: Edukacja informacyjna i źródłowa (przysposobienie biblioteczne)**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna i rozumie wspólne cechy i różnice systemu biblioteczno-informacyjnego uczelni (Biblioteka Uniwersytecka w Poznaniu, biblioteki wydziałowe).
- Zna i rozumie zasady korzystania z czytelni i wypożyczalni, z zasobów elektronicznych oraz otwartych projektów cyfrowych UAM.
- Zna i rozumie typy źródeł informacji w bibliotekach.
- Zna i rozumie wszystkie usługi bibliotek UAM.
- Potrafi korzystać z konta bibliotecznego, wykorzystując pełne jego możliwości.
- Potrafi wyszukiwać i gromadzić materiał do realizacji zajęć, niezbędnych do optymalnego realizowania toku studiów.
- Potrafi korzystać ze źródeł informacji tradycyjnej i elektronicznej, w tym z zasobów dostępnych zdalnie dla studentów UAM oraz w otwartych projektach cyfrowych.

- Potrafi poprawnie sporządzić bibliografię dla tworzonej pracy dyplomowej przy pomocy programów bibliograficznych.
- Potrafi korzystać z usług oferowanych przez biblioteki (np. zamawia lub pobiera kopie do własnego użytku) z poszanowaniem praw autorskich.
- Jest gotów do autonomicznego wyszukiwania informacji i literatury, gromadzenia materiałów, niezbędnych do optymalnego realizowania toku studiów.
- Jest gotów do krytycznej oceny źródeł informacji.
- Jest gotów do sporządzenia bibliografii w pracy dyplomowej.
- Jest gotów do zapobiegania zjawisku plagiatu.

Treści programowe dla zajęć:

- System biblioteczno-informacyjny UAM: charakterystyka cech wspólnych i różniących Bibliotekę Uniwersytecką w Poznaniu i biblioteki wydziałowe, podstawowe zasady korzystania ze wspólnego dla całego Uniwersytetu systemu biblioteczno- informacyjnego, zasady i regulamin korzystania ze zbiorów bibliotecznych, konto czytelnika oraz korzyści wynikające z oferowanych możliwości: zdalny zapis, charakterystyka konta, podstawowe zasady zamówienia, prolongaty, rezerwacji, dostęp zdalny do licencjonowanych zasobów naukowych UAM.
- Wyszukiwanie i zamawianie książek, czasopism. Charakterystyka katalogów bibliotecznych: wyszukiwarka zasobów naukowych UAM, katalog biblioteczny online UAM, najważniejsze katalogi online w Polsce, np.: Biblioteki Narodowej, Katalog KaRo (Katalog Rozproszony Bibliotek Polskich).
- Warsztat naukowy studenta:praktyczne wskazówki dotyczące strategii poszukiwania literatury: wyszukiwanie tematyczne, proste, logiczne, zaawansowane w katalogu online oraz w wyszukiwarce zasobów naukowych UAM z użyciem operatorów boolowskich, wyszukiwanie literatury do zajęć i prac dyplomowych w zdalnych zasobach naukowych UAM (otwartych i licencjonowanych, dziedzinowych bazach danych, e-czasopismach, e-książkach, bibliotekach wirtualnych, repozytoriach).
- Warsztat naukowy studenta: tradycyjne źródła informacji: bibliografie, encyklopedie, słowniki, opracowania, bibliografia: rodzaje, zasady tworzenia przypisów, bibliografia załącznikowa, zautomatyzowane programy do tworzenia bibliografii.
- Plagiat: definicja i konsekwencje, przykłady plagiatu, zapobieganie.

Nazwa zajęć**: Elementy kombinatoryki**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna podstawowe definicje, własności i twierdzenia dotyczące szeregów potęgowych.
- Potrafi zapisać i rozwiązać równanie rekurencyjne jednorodne metodą funkcji tworzących.
- Potrafi zapisać i rozwiązać układ zależności rekurencyjnych jednorodnych metodą funkcji tworzących.
- Potrafi rozwiązać niejednorodne zależności rekurencyjne metodą funkcji tworzących.
- Zna i potrafi zastosować zasadę szufladkową do rozwiązywania problemów kombinatorycznych.
- Potrafi przeliczać struktury nieoznaczone.
- Potrafi przeliczać struktury oznaczone.

Treści programowe dla zajęć:

- Szeregi potęgowe.
- Równania rekurencyjne jednorodne.
- Równania rekurencyjne niejednorodne.
- Układy rekurencji.
- Zasada szufladkowa.
- Struktury nieoznaczone.
- Struktury oznaczone.

Nazwa zajęć**: Elementy przedsiębiorczości**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Rozumie mechanizm prowadzenia własnej firmy oraz potrafi wskazać plusy i minusy prowadzenia firmy w porównaniu z byciem pracownikiem w większej firmie.
- Potrafi wskazać podstawowe formy prawne działania biznesowego i porównać je ze sobą.
- Rozumie czym są koszty pracy oraz podstawowe wskaźniki finansowe firmy.
- Potrafi stworzyć biznesplan oraz rozumie na czym polegają sposoby zarządzania ryzykiem biznesowym.
- Potrafi przeanalizować przykłady nowopowstałych przedsiębiorstw i wyciągnąć wnioski z takiej analizy.

Treści programowe dla zajęć:

- Wyszukiwanie pomysłu na biznes. Rola unikalności pomysłu, zalety i wady. Franczyza i jej przykłady. Firmy oddolne. Czy ty się nadajesz do biznesu.

- Działalność gospodarcza a spółka z o.o. Inne formy prawne prowadzenia biznesu. Osobowość prawna i jej konsekwencje. Procedury zakładania firmy i jej zamykania.
- Koszty pracy. Podatek VAT. Amortyzacja. Płynność finansowa, rentowność. Cechy udanej firmy.
- Szacowanie nakładów i przychodów podczas planowania biznesu. Elementy biznesplanu i ich związek z szansą na sukces. Przykłady biznesplanów.
- Studium przypadku – wybrane przykłady polskich firm oraz analiza czynników wpływających na ich sukces czy porażkę.

Nazwa zajęć**: Frameworki aplikacji webowych Angular i React**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna i umie korzystać z najpopularniejszych frameworków/bibliotek aplikacji webowych, tj. React i Angular, oraz zna związane z nimi dobre praktyki i wzorce projektowe.

Treści programowe dla zajęć:
- Test poziomujący.
- Wprowadzenie do React.
- React - Routing.
- Wprowadzenie do Redux.
- React - testowanie.
- React - wzorce i reużywalne komponenty.
- Kończenie projektu i konsultacje.
- Wprowadzenie do Angular.
- Angular vs React.
- Angular - Change Detection, Zone.js, Chrome DevTools.
- Angular - Directives and pipes.
- Angular - Dependency Injection.
- Wprowadzenie do RxJS, Angular Elements, Angular PWA.
- Projekt Angular/React.
- Test końcowy.

Nazwa zajęć**: Grafika komputerowa**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Rozumie zasady wyrażania transformacji liniowej jako macierz, wektor, iloczyn i pojęcie potoku graficznego. Potrafi zorientować i umieścić obiekty geometryczne w przestrzeni świata za pomocą liniowych transformacji.
- Rozumie ideę stojącą za wykorzystaniem rzutowania perspektywicznego i ortograficznego. Potrafi stworzyć odpowiednie macierze rzutowania potrzebne w potoku graficznym.
- Rozumie optymalizacje potoku graficznego za pomocą algorytmów z-buforu, cullingu i clippingu. Potrafi w OpenGL-u zastosować wiedzę optymalizacji potoku graficznego.
- Rozumie modelowanie światła metodą Phonga. Potrafi stworzyć implementację tego modelu na poziomie karty graficznej.
- Rozumie różnice między teksturowaniem za pomocą mapy tekstury a teksturowaniem proceduralnym, rozumie pojęcie map normalnych. Potrafi oteksturować obiekty.
- Rozumie modelowanie ruchu obiektów wzdłuż krzywych. Potrafi stworzyć animacje obiektów w przestrzeni świata. Rozumie zasady tworzenia krzywych typu Hermite, Bezier i Catmull-Rom.

Treści programowe dla zajęć:
- Potok graficzny, transformacje liniowe, interpretacja w przestrzeni Euklidesa.
- Rzutowanie perspektywiczne i ortograficzne.
- Algorytm z-bufor, culling, clipping.
- Model oświetlenia i cieniowania Phonga.
- Teksturowanie.
- Równanie parametryczne krzywych, trójścian Freneta.

Nazwa zajęć**: Główne problemy współczesnego świata**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna podstawowe zagadnienia etyczne, prawne, społeczne i ekonomiczne uwarunkowań działalności dotyczące informatyki.
- Potrafi pozyskiwać informacje z literatury, baz danych i innych źródeł; potrafi selekcjonować i integrować uzyskane informacje, dokonywać ich interpretacji, a także wyciągać wnioski oraz formułować i uzasadniać opinie.

- Ma umiejętność samokształcenia się, m.in. w celu podnoszenia kompetencji i nabywania nowych umiejętności zawodowych.
- Potrafi wykorzystywać metody analityczne, symulacyjne oraz eksperymentalne do formułowania i rozwiązywania zadań.
- Potrafi stosować nowoczesne narzędzia informatyczne do rozwiązywania sytuacji problemowych z różnych dziedzin.
- Potrafi przy rozwiązywaniu zadań dostrzegać aspekty społeczne, ekonomiczne i prawne.
- Potrafi ocenić potencjalne (nowe) zastosowania narzędzi informatyki i ich konsekwencje dla życia społecznego, gospodarczego oraz wynikające z nich korzyści i zagrożenia.

Treści programowe dla zajęć:

- Podstawowe trendy, wydarzenia i procesy zachodzące we współczesnym świecie.
- Główne problemy ekologiczno-surowcowe występujące w XXI w. – przykłady, przyczyny, konsekwencje, przeciwdziałanie.
- Zagrożenia ekonomiczno-społeczne: geneza, przejawy, skutki, zapobieganie.
- Wyzwania demograficzne współczesnego świata: źródła, przykłady, implikacje, przeciwdziałanie.
- Wojny i konflikty w XXI wieku – analiza wybranych przykładów: przyczyny, charakterystyka, skutki, sposoby zapobiegania.
- Zagrożenia w cyberprzestrzeni: np.: cyberszpiegostwo, cyberprzestępczość, cyberterroryzm, cyberdezinformacja itd. oraz sposoby ich zwalczania.
- Terroryzm i fundamentalizm we współczesnym świecie: przyczyny, przykłady, zapobieganie i zwalczanie.
- Przestępczość zorganizowania i jej główne formy w XXI w. – analiza wybranych przykładów oraz ich implikacji.

Nazwa zajęć**: Historia i filozofia informatyki**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Student zna podstawowe pojęcia i problemy filozofii.
- Student potrafi wymienić i opisać podstawowe zagadnienia filozofii informatyki oraz zagadnienia filozoficzne związane z informatyką.
- Student zna główne fakty historyczne i tendencje w rozwoju informatyki.
- Student zna podstawowe fakty dotyczące historii obliczeń oraz wybrane systemy liczbowe i metody rachunkowe.

Treści programowe dla zajęć:

- Wprowadzenie ogólne do filozofii. Pojęcie filozofii, filozofia a nauki szczegółowe, funkcje filozofii, działy filozofii.
- Ontologia. Podstawowe problemy i stanowiska w ontologii.
- Epistemologia. Podstawowe zagadnienia i koncepcje epistemologiczne.
- Filozofia informatyki. Co to jest filozofia informatyki? Podstawowe problemy filozofii informatyki. Informatyka jako nauka: definicja informatyki, paradygmaty informatyki.
- Zagadnienia filozoficzne związane z informatyką. Sztuczna inteligencja (SI): definicje, rys historyczny, nurty SI, podstawowe zagadnienia SI, przykłady zastosowań w tym sztuczna twórczość i sztuczne życie. Wybrane aspekty filozoficzne sztucznej inteligencji: test Turinga, chiński pokój. Silna i słaba teza SI – za i przeciw. Maszyna a świadomość – przegląd stanowisk. Obliczalność: definicja potoczna a formalizmy, teza Churcha-Turinga i jej konsekwencje filozoficzne. Komputery w matematyce – rodzaje wykorzystania komputerów w matematyce, rodzaje dowodów komputerowych, filozoficzne konsekwencje uznania w matematyce dowodów automatycznych oraz wspomaganych komputerowo. Zagadnienia filozoficzne związane w wirtualną rzeczywistością.
- Historia informatyki. Podstawowe fakty dotyczące historii maszyn liczących oraz historii mechanizacji rozumowań. Omówione zostaną następujące zagadnienia: Prehistoria komputerów (od abakusów i liczydeł do pałeczek Nepera); Pierwsze maszyny liczące; Historia mechanizacji rozumowań; Maszyna analityczna jako pierwowzór współczesnych komputerów; Mechanografia. Wielcy teoretycy XX wieku; Komputery przekaźnikowe; Komputery generacji I-IV oraz historia najnowsza.
- Historia obliczeń.
- Początki liczenia, powstanie bazy, pierwotne metody rachunkowe.
- Sposoby zapisu liczb oraz najciekawsze metody rachunkowe stosowane przez: Egipcjan, Babilończyków, Chińczyków, Greków, Inków, Rzymian.
- Historia powstania oraz rozpowszechnienia się numeracji arabskiej.

Nazwa zajęć**: Historia obliczeń**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna podstawowe pojęcia związane z historią liczenia.
- Zna wybrane systemy liczbowe i metody rachunkowe.
- Zapisuje liczby w wybranych systemach liczbowych.
- Wykonuje obliczenia metodami historycznymi, korzystając m. in. z historycznych pomocy obliczeniowych (abaków, liczydeł, pałeczek Nepera, suwaka logarytmicznego).
- Zna podstawowe fakty z historii maszyn liczących.
- Rozumie znaczenie matematyki (jej historii) dla rozwoju cywilizacyjnego.
- Potrafi w sposób zrozumiały dla laika mówić o zagadnieniach z historii obliczeń.

Treści programowe dla zajęć:

- Liczenie w czasach najdawniejszych (sposoby wyrażania ilości i liczenia u plemion pierwotnych). Rachunki na częściach ciała.
- Sposób zapisu liczb i metody rachunkowe wykorzystywane w starożytnym Egipcie. Ciekawostki dotyczące matematyki Egipskiej.
- Numeracja babilońska, zapisywanie liczb w systemie babilońskim oraz ich odczytywanie. Liczenie na abakach babilońskich i kalkulach.
- Systemy liczbowe oraz sposoby zapisu liczb u Inków, Azteków i Majów (w tym „liczby na sznurkach").
- Systemy liczbowe starożytnej Grecji i Rzymu (różnych epok), elementy matematyki Grackiej, w tym dowody rysunkowe prostych własności liczb i tw. Pitagorasa. Liczenia na abakach greckich i rzymskich. Wybrane fakty z matematyki starożytnej Grecji (w tym paradoksy nieskończoności).
- Zasady obliczeń na liczydłach rosyjskich, chińskich, japońskich (ćwiczenia praktyczne).
- Systemy zapisu liczb oraz metody rachunkowe pochodzące ze starożytnych i średniowiecznych Chin. Rachunki na szachownicach do liczenia.
- Hinduski i arabski system liczenia, ewolucja cyfr arabskich, wybrane metody rachunkowe (rachunki „na piasku"). Wprowadzenie cyfr arabskich do Europy. Liczenie w średniowiecznej Europie, w tym abaki średniowieczne i liczby na palcach.
- „Urządzania" ułatwiające liczenie, w tym kości Nepera i suwaki logarytmiczne. Ćwiczenia praktyczne w ich wykorzystaniu.
- Pierwsze maszyny liczące XVII wieku (maszyna Schicarda, Pascalina, maszyna Leibniza), pokaz i opis metod działania. Pierwsze polskie maszyny liczące.
- Prezentacja projektów studenckich z zakresu historii liczenia i wybranych zagadnień historii matematyki.

Nazwa zajęć**: Inżynieria oprogramowania**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Potrafi stworzyć model obiektowy prostego systemu (np. w języku UML).
- Potrafi oceniać przydatność różnych paradygmatów i związanych z nimi środowisk programistycznych do rozwiązywania różnego typu problemów.
- Potrafi projektować oprogramowanie zgodnie z metodyką obiektową.
- Potrafi ocenić, na podstawowym poziomie, przydatność rutynowych metod i narzędzi informatycznych oraz wybrać i zastosować właściwą metodę i narzędzia do typowych zadań informatycznych.
- Potrafi zgodnie z zadaną specyfikacją zaprojektować oraz zrealizować prosty system informatyczny, używając właściwych metod, technik i narzędzi.
- Potrafi stosować techniki prowadzące do otrzymania oprogramowania wysokiej jakości.
- Potrafi posługiwać się przynajmniej jednym z najbardziej popularnych systemów zarządzania wersjami.
- Potrafi posługiwać się wzorcami projektowymi.

Treści programowe dla zajęć:

- Czym jest oprogramowanie, problemy pojawiające się przy rozwoju oprogramowania.
- Cykl życia oprogramowania z uwzględnieniem różnych modeli. Wyróżnienie składowych procesu rozwoju oprogramowania.
- Analiza przykładowych modeli cyklu życia oprogramowania na przykładzie metodyki hybrydowej (Rational Unified Process) oraz zwinnej (SCRUM).
- Analiza wymagań: wymagania funkcjonalne i niefunkcjonalne, specyfikacja wymagań, proces pozyskiwania wymagań (scenariusze, przypadki użycia), zarządzanie wymaganiami.
- Projektowanie systemu informatycznego. UML w
- projektowaniu. Projektowanie interakcji (diagramy interakcji), modelowanie struktury (model logiczny, diagram klas, diagram wdrożenia), modelowania zachowania (diagram stanów, diagram aktywności).
- Architektura systemu informatycznego, wzorce architektoniczne (klient-serwer, aplikacje wielowarstwowe), wzorce projektowe (MVC, ORM). Wpływ architektury systemu na tworzone oprogramowanie (przykłady).

- Problemy związane z implementacją systemu informatycznego.
- Testowanie oprogramowania. Rodzaje testów. Testy prowadzone przez programistę (testy jednostkowe, testy modułowe, testy integracyjne). Test-driven developement.
- Ciągła integracja. Scenariusze testowe i testy akceptacyjne.
- Testy użyteczności.
- Utrzymanie i pielęgnacja oprogramowania.
- Wykorzystanie narzędzi wspierających rozwój oprogramowania (repozytorium kodu, issue tracker, bug tracker).
- Zbieranie wymagań przy pomocy przypadków użycia oraz user stories.
- Projektowanie interakcji (diagramy interakcji).
- Projektowanie struktury systemu (model logiczny, diagram klas, diagram wdrożenia).
- Modelowanie zachowania (diagram stanów, diagram aktywności).
- Testowanie oprogramowania przez programistę (testy jednostkowe). Scenariusze testowe i testy akceptacyjne. Testy użyteczności.
- Ciągła integracja.
- Realizacja grupowego mini-projektu w metodyce hybrydowej (RUP). Realizacja w dużym zespole z podziałem ról.

Nazwa zajęć**: Język angielski**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- czyta ze zrozumieniem różnego rodzaju teksty w języku angielskim, w tym publikacje naukowe, analizuje ich treść i wybiera niezbędne informacje
- tworzy ustne wypowiedzi na przygotowane tematy, prezentuje i argumentuje swoje stanowisko, komentuje stanowisko innych;
- wykazuje chęć i potrzebę podjęcia dyskusji na tematy ogólno-akademickie
- samodzielnie korzysta z rożnych źródeł informacji w celu rozbudowania swojej wiedzy ogólno-akademickiej
- pisze logiczne i spójne teksty na różne tematy; rozróżnia język formalny od nieformalnego
- rozumie ustne wypowiedzi wyrażane językiem standardowym z uwzględnieniem różnic między angielskim brytyjskim i amerykańskim
- wyraża się z dużą poprawnością gramatyczną i ortograficzną

Treści programowe dla zajęć:
- Czasy gramatyczne potrzebne do wyrażania różnorodnych czynności osadzonych w czasie (Present Simple and Present Continuous, Past Simple and Past Continuous, Present Perfect and Present Perfect Continuous, Past Perfect, formy wyrażania przyszłości
- Inne struktury gramatyczne potrzebne do wyrażania różnorodnych treści i opinii (np. czasowniki modalne, przymiotniki, strona bierna, zdania warunkowe, mowa zależna)
- Słownictwo dotyczące życia codziennego (jedzenie, podróże, zainteresowania, edukacja, zakupy, pieniądze, technologia)
- Słownictwo związane z bezpośrednim środowiskiem studenta (dom, rodzina, studia, praca)
- Strategie efektywnego czytania w celu zrozumienia ogólnego sensu wypowiedzi; domyślanie się znaczenia nieznanych słów
- Strategie efektywnego czytania w celu wychwytywania niezbędnych szczegółów; definiowanie znaczenia nowych słów; tworzenie powiązań z posiadaną wiedzą
- Strategie efektywnego słuchania w celu zrozumienia ogólnego sensu wypowiedzi; domyślanie się znaczenia nieznanych słów
- Strategie efektywnego słuchania w celu wychwytywania niezbędnych szczegółów; definiowanie znaczenia nowych słów; tworzenie powiązań z posiadaną wiedzą
- Strategie komunikacyjne np. negocjowanie znaczenia, prośba o powtórzenie, opisywanie w sytuacji nieznajomości słów, itp.
- Wyrażanie różnorodnych funkcji językowych np. prośby, opisy, wyrażanie opinii, wyrażanie zgody, brak zgody, pytania o pozwolenie, skargi, itp.

Nazwa zajęć**: Języki formalne i złożoność obliczeniowa**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Ma podstawową wiedzę w zakresie języków formalnych i ich własności.
- Zna formalne modele obliczeń (automaty skończenie stanowe, automaty ze stosem, maszyny Turinga) i ich własności.
- Zna podstawowe rodzaje gramatyk generatywnych, ich własności i związki z automatami.

- Potrafi stosować wprowadzone modele obliczeń i rodzaje gramatyk do definiowania/opisywania języków formalnych.
- Potrafi zaimplementować reprezentacje formalizmów automatów skończenie stanowych, wyrażeń regularnych i gramatyk bezkontekstowych w obiektowym języku programowania.
- Potrafi wykorzystać gotowe biblioteki do obsługi automatów skończenie stanowych i wyrażeń regularnych w rozwiązaniu praktycznych problemów.

Treści programowe dla zajęć:

- Pojęcia podstawowe: alfabet, słowo, język. Operacje na językach - mnogościowe i swoiste (konkatenacja, domknięcie Kleene'ego). Rozpoznawanie języka vs.
- generowanie języka.
- Automat skończenie stanowy (wersja deterministyczna i niedeterministyczna), język akceptowany przez automat skończenie stanowy.
- Determinizacja automatu skończenie stanowego i jej koszt. Informacja o automacie minimalnym. Automat z pustym przejściem.
- Lemat o pompowaniu dla języków regularnych i jego zastosowanie.
- Wyrażenia regularne i języki oznaczane przez te wyrażenia. Twierdzenie Kleene'ego. Zamkniętość języków regularnych na operacje regularne, dopełnienie i przecięcie.
- Gramatyki typu 3 (regularne), generowane przez nie języki i ich związek z językami akceptowanymi przez automaty skończenie stanowe. Inne typy gramatyk i hierarchia Chomsky'ego języków.
- Rozstrzygalne problemy dla języków regularnych.
- Gramatyki bezkontekstowe i automaty ze stosem. Postaci normalne dla gramatyk bezkontekstowych.
- Lemat o pompowaniu dla języków bezkontekstowych. Języki poza klasą CF. Rozstrzygalność problemu słowa i algorytm CYK.
- Maszyna Turinga - model podstawowy i modele równoważne. Języki akceptowane i rozstrzygane przez maszynę Turinga. Maszyna Turinga jako model obliczeń numerycznych - informacja o tezie Churcha.
- Złożoność czasowa i pamięciowa maszyny Turinga.
- Problemy jako języki. Klasy złożoności obliczeniowej. Redukowalność i NP-zupełność. Twierdzenie Cooka, hipoteza P-NP
- Reprezentacja automatu skończenie stanowego (deterministycznego bądź niedeterministycznego z pustymi przejściami) w obiektowym języku programowania.
- Zewnętrzne, otwarto-źródłowe biblioteki do obsługi automatów skończenie stanowych i ich praktyczne zastosowania.
- Algorytm tworzenia automatu skończenie stanowego równoważnego wyrażeniu regularnemu POSIX.
- Standard POSIX rozszerzonych wyrażeń regularnych i jego praktyczne zastosowania. Użycie rozszerzonych wyrażeń regularnych w wybranych językach programowania i bibliotekach.

Nazwa zajęć**: Java oraz JSON i XML**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna środowiska programistyczne IntelliJ, podstawy Gita i Mavena.
- Zna podstawowe pojęcia, zasady i techniki stosowane przy testowaniu, debuggowaniu i walidacji oprogramowania.
- Rozumie czym są kolekcje i strumienie oraz wyrażenia Lambda.
- Zna podstawy JSONa i XMLa oraz potrafi wykonać serializację i deserializacją w Javie.
- Potrafi połączyć się z bazą danych i wykonać podstawowe operacje przy użyciu frameworka Hibernate.
- Zna kilka wzorców projektowych.
- Zna podstawy aplikacji webowych w Javie.
- Potrafi wykonać niewielki projekt programistyczny z wykorzystaniem zaawansowanych technik, narzędzi i zasad.
- Zna pojęcie REST API i potrafi go wykorzystać w praktyce.

Treści programowe dla zajęć:

- Przygotowanie środowiska programistycznego (IntelliJ, Git, Maven, Logger).
- Debuggowanie i testowanie (JUnit).
- Strumienie oraz Eclipse Collection, wyrażenia Lambda.
- JSON i XML – serializacja i deserializacja.
- Hibernate – implementacja technologii odwzorowania obiektowo-relacyjnego dla aplikacji Java.
- Apache Tomcat – serwer aplikacji webowych.
- REST API

Nazwa zajęć**: Języki programowania JavaScript**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Opisuje wybrane koncepcje i struktury języka JavaScript/TypeScript.
- Identyfikuje wybrane zastosowania technologii JavaScript/TypeScript.
- Wskazuje metody projektowania, wdrażania i testowania aplikacji JavaScript/TypeScript.
- Tworzy proste aplikacje w technologii JavaScript/TypeScript z wykorzystaniem wybranych bibliotek i frameworków.
- Rozwiązuje problemy i zadania jako uczestnik zespołu.
- Analizuje dostępne materiały i źródła wiedzy w celu rozwiązywania problemów programistycznych.

Treści programowe dla zajęć:

- Wprowadzenie do JavaScript/TypeScript. Podstawowe zastosowania i struktury językowe.
- Architektura aplikacji klient-serwer w kontekście programowania JavaScript/TypeScript.
- Wprowadzenie do tworzenia prostych aplikacji klienckich JavaScript/TypeScript. Przykładowa architektura i podstawowe komponenty aplikacji klienckiej JavaScript/TypeScript.
- Korzystanie z API REST z poziomu aplikacji klienckiej JavaScript/TypeScript.
- Routing w aplikacjach klienckich JavaScript/TypeScript.
- Debugowanie, testowanie i obsługa błędów w aplikacjach klienckich JavaScript/TypeScript.
- Wprowadzenie do tworzenia prostych aplikacji serwerowych JavaScript/TypeScript. Przykładowa architektura i podstawowe komponenty aplikacji serwerowej JavaScript/TypeScript.
- Debugowanie, testowanie i obsługa błędów w aplikacjach serwerowych JavaScript/TypeScript.
- Prezentacja grupowa rozwiązań problemów i zadań projektowych wypracowanych w trakcie semestru.

Nazwa zajęć**: Kryptografia z elementami algebry**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna podstawowe struktury algebraiczne.
- Potrafi określić strukturę algebraiczną związaną z systemem kryptograficznym.
- Potrafi zaimplementować działania na elementach struktur algebraicznych.
- Wykorzystuje twierdzenia matematyczne w analizie systemów kryptograficznych.
- Zna terminologie współczesną terminologię kryptologiczną.
- Analizuje bezpieczeństwo algorytmów kryptologicznych. Potrafi wskazać wady i zalety danego rozwiązania kryptologicznego.
- Potrafi obliczyć złożoność obliczeniową algorytmów wykorzystywanych do konstrukcji systemów kryptologicznych.
- Potrafi efektywnie implementować podstawowe systemy kryptologiczne.
- Potrafi wykorzystać w implementacji istniejące biblioteki kryptograficzne.

Treści programowe dla zajęć:

- Działania wewnętrzne w zbiorze i ich własności. Zasadnicze pojęcia teorii grup – podgrupy, warstwy.
- Operacje elementarne na bitach. Notacja wielkie O. Algorytmy czasu wielomianowego oraz wykładniczego ze względu na liczbę bitów danych. Złożoność obliczeniowa wykonywania działań grupach skończonych.
- Podstawowe protokoły kryptograficzne. Bezpieczeństwo doskonałe. Bezpieczeństwo obliczeniowe. Kandydaci na funkcje jednokierunkowe. Problem logarytmu dyskretnego w grupie. Problem i-tego bitu logarytmu dyskretnego.
- Rząd elementu w grupie. Grupy cykliczne. Homomorfizm grup. Jądro oraz obraz homomorfizmu. Grupa ilorazowa i twierdzenie o izomorfizmie.
- Algorytmy szyfrowania z kluczem publicznym. Algorytm ElGamala i RSA. Obliczanie logarytmu dyskretnego metodą wielkich i małych kroków. Równoważność między problemem faktoryzacji a łamaniem RSA.
- Ciała skończone i ich własności. Reszty i niereszty kwadratowe w grupach skończonych. Obliczanie pierwiastków w ciałach skończonych.
- Algorytm ElGamala na krzywej eliptycznej. Problem logarytmu dyskretnego na krzywej eliptycznej. Protokoły uzgadniania kluczy. Protokół Diffiego-Hellmana na krzywej eliptycznej.
- Pierścienie. Jedności pierścieni. Obraz i jądro homomorfizmu pierścieni. Pierścienie wielomianów. Algorytm Euklidesa dla wielomianów. Ideały i pierścienie ilorazowe. Chińskie twierdzenie o resztach.
- Zaawansowany standard szyfrowania danych AES. Transformacje SubByte i MixColumn w AES.

Nazwa zajęć**: Kompresja danych**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Rozumie podstawowe pojęcia związane z kompresją danych i teorią informacji.
- Potrafi skonstruować kody Huffmana, Golomba, Rice'a, Tunstall'a dla podanych danych.
- Potrafi wygenerować znacznik w kodowaniu arytmetycznym oraz wykazać jego jednoznaczność.

- Rozumie ideę słownikowych algorytmów kompresji i potrafi podać ich przykłady.
- Zna zagadnienie kodowania predykcyjnego i potrafi podać przykłady algorytmów.
- Zna miary kompresji stratnej i potrafi je stosować.
- Rozumie potrzebę stosowania i zna przykładowe algorytmy kwantyzacji skalarnej i wektorowej.
- Rozumie ideę kodowania różnicowego i korzyści z niej wynikające.
- Zna ideę transformat wykorzystywanych w kompresji i potrafi zaprezentować niektóre.
- Potrafi wykorzystywać transformaty do kompresji danych (kodowanie podpasmowe).
- Zna ideę kompresji falkowej oraz przykładowe systemy korzystające z przekształcenia falkowego (JPEG2000, Dirac).
- Zna ideę schematów analiza-synteza.

Treści programowe dla zajęć:

- Wprowadzenie i podstawowe pojęcia.
- Kodowanie Huffmana. Kody Rice'a, Tunstall'a, Golomba.
- Idea kodowania słownikowego na przykładzie wybranych algorytmów, kodowanie arytmetyczne, kodowanie predykcyjne (transformata BWT).
- Podstawy kompresji stratnej - stosowane miary, entropia warunkowa, modele.
- Kwantyzacja skalarna i wektorowa.
- Kodowanie różnicowe.
- Transformaty.
- Kodowanie wykorzystujące transformaty.
- Kompresja falkowa, Dirac, JPEG2000.
- Schematy typu analiza-synteza.

Nazwa zajęć**: Laboratorium systemów mobilnych**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna podstawowe cechy systemów mobilnych. Rozumie różnice wynikające z ograniczeń urządzeń mobilnych.
- Potrafi zgodnie z zadaną specyfikacją zaprojektować oraz stworzyć interfejs użytkownika dla aplikacji mobilnej.
- Potrafi uruchamiać i testować programy dla aplikacji mobilnych na urządzeniu oraz emulatorze.
- Posiada umiejętność wykorzystywania najważniejszych bibliotek programistycznych dla urządzeń mobilnych w przynajmniej jednym środowisku.
- Potrafi stworzyć aplikację mobilną zawierającą dane nieulotne przechowywane lokalnie lub w chmurze.
- Potrafi zaprojektować i zaimplementować wykonywanie długich i złożonych obliczeniowo operacji na urządzeniach mobilnych.
- Posiada wiedzę na temat publikowania i udostępniania aplikacji mobilnych w jednym ze sklepów przeznaczonych do sprzedawania aplikacji mobilnych.
- Potrafi zaprojektować i zaimplementować prostą aplikację mobilną.

Treści programowe dla zajęć:

- Architektura systemu Android. Omówienie podstawowych komponentów (Aktywności, Intencje, struktura projektu). Stworzenie pierwszej aplikacji w systemie Android, cykl życia aktywności).
- Android - tworzenie interfejsu użytkownika (views, layouts, view binding).
- Android Jetpack. Architektura MVVM, LiveData, Data binding.
- Android Navigation Architecture Component,
- Przesyłanie danych pomiędzy fragmentami.
- Zaawansowane elementy interfejsu użytkownika (Animacja, zakładki, RecyclerView, AppBar).
- Intencje jawne, Intencje niejawne, wykonywanie zadań w tle. Serwisy.
- Notyfikacje, GPS, Uprawnienia w systemie Android.
- Dane nieulotne (Shared preferences, SQLite, Android Room).
- Flutter – różnice i podobieństwa podczas tworzenia aplikacji mobilnych. Uruchomienie pierwszej aplikacji i struktura projektu. Dart.
- Tworzenie interfejsu użytkownika we frameworku Flutter. Widżety stanowe i bezstanowe. Nawigacja.
- Wtyczki w języku Flutter pub.dev.
- Wydanie aplikacji w Google Play.
- Tworzenie aplikacji mobilnej.

Nazwa zajęć**: Logika i teoria mnogości**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Potrafi pracować w rachunku zdań.

- Potrafi pracować w rachunku predykatów.
- Posługuje się podstawowymi pojęciami teorii mnogości.

Treści programowe dla zajęć:

- Rachunek zdań: język rachunku zdań, zapisywanie schematów zdań, tautologie, schematy wnioskowania, postaci normalne.
- Funkcje logiczne (KPN, APN, zupełne układy funkcji logicznych, wielomiany Żegałkina).
- Rachunek predykatów: język rachunku predykatów, zapisywanie schematów zdań, tautologie.
- Teoria mnogości: algebra zbiorów, relacje, funkcje.

Nazwa zajęć**: Matematyka dyskretna**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna i rozumie i potrafi wykorzystywać różne metody dowodzenia implikacji. Potrafi stosować zasadę indukcji matematycznej
- Zna i rozumie podstawowe zasady i prawa przeliczania.
- Potrafi stosować podstawowe zasady i prawa przeliczania. Umie wykorzystywać zasadę szufladkową. Umie przeprowadzić dowody prostych tożsamości kombinatorycznych.
- Zna, rozumie i potrafi się posługiwać podstawowymi pojęciami teorii grafów.
- Zna przykłady klasycznych zastosowań teorii grafów. Rozumie i potrafi posługiwać się klasycznymi algorytmami teorii grafów. Rozumie znaczenie praktyczne teorii grafów - umie podać przykłady, w których stosuje się poznane zagadnienia i fakty teorii grafów w praktyce.
- Potrafi układać i identyfikować wybrane zależności rekurencyjne oraz rozwiązywać je różnymi metodami.

Treści programowe dla zajęć:

- Metody dowodzenia implikacji. Zasada szufladkowa. Twierdzenie o indukcji matematycznej.
- Podstawowe zasady i prawa przeliczania - zasada bijekcji, prawa dodawania i mnożenia.
- Schematy wyboru. Zasada włączania i wyłączania.
- Współczynniki wielomianowe. Tożsamości kombinatoryczne.
- Podstawowe pojęcia teorii grafów.
- Klasyczne problemy i algorytmy grafowe – problemy: najkrótszych ścieżek, optymalnego drzewa rozpiętego, chińskiego listonosza, wędrującego komiwojażera, przydziału zadań, kolorowania grafów i map.
- Zależności rekurencyjne. Układanie i rozwiązywanie prostych i liniowych równań rekurencyjnych.

Nazwa zajęć**: Metody numeryczne**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Rozumie różnicę pomiędzy arytmetyką liczb rzeczywistych, a arytmetyką komputerową; potrafi wyjaśnić wpływ arytmetyki zmiennopozycyjnej na działanie algorytmu.
- Potrafi wskazać, który z podanych algorytmów jest dla danego problemu bardziej efektywny.
- Zna metody pozwalające na przybliżone rozwiązanie wybranych zagadnień matematycznych.

Treści programowe dla zajęć:

- Arytmetyka zmiennopozycyjna - reprezentacja zmiennopozycyjna liczb (podstawy teoretyczne i standard IEEE 754). Algorytm sumacyjny Kahana. Błędy w obliczeniach, uwarunkowanie zadania, numeryczna poprawność i stabilność algorytmów.
- Interpolacja wielomianowa - zadanie interpolacji Lagrange'a i Hermite'a, bazy Newtona + zastosowanie uogólnionego algorytmu Hornera, ilorazy różnicowe, interpolacja odwrotna, błąd interpolacji, dobór węzłów interpolacji.
- Interpolacja funkcjami sklejanymi.
- Numeryczne obliczanie całek - kwadratury Newtona -Cotesa i Gaussa.
- Rozwiązywanie równań nieliniowych - metoda bisekcji, siecznych, Newtona i Bairstowa, kryteria stopu, rząd metody.
- Wyznaczanie wartości własnych macierzy, pojęcie promienia spektralnego macierzy.
- Przypomnienie pojęcia normy wektorów, wprowadzenie definicji normy macierzy, w tym indukowanej. Uwarunkowanie macierzy. Rozwiązywanie układów liniowych równań algebraicznych - metody bezpośrednie: eliminacja Gaussa bez modyfikacji (niestabilność), z częściowym i pełnym wyborem elementu głównego, rozkład LU, metoda Doolitle'a, rozkład Cholesky'ego-Banachiewicza (z wprowadzeniem pojęcia macierzy dodatnio określonej); iteracyjne poprawianie rozwiązań.
- Metody iteracyjne rozwiązania układów liniowych równań algebraicznych.
- Rozkład SVD i jego zastosowania (m.in. do rozwiązywania LZNK).

Nazwa zajęć**: Ochrona własności intelektualnej**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Wyjaśnia pojęcie, istotę i funkcje prawa własności intelektualnej.
- Opisuje strukturę prawa własności intelektualnej.
- Wyróżnia organy państwowe działające w sferze prawa własności intelektualnej.
- Wyróżnia wolności, prawa i obowiązki obywatela w sferze prawa własności intelektualnej.
- Korzysta ze źródeł prawa własności intelektualnej i proponuje rozwiązania konkretnych problemów.
- Stosuje odpowiednie przepisy związane z prawem własności intelektualnej.
- W sposób zrozumiały, jasny i precyzyjny dla laików omawia prawo własności intelektualnej.
- Wyjaśnia aspekty etyczne i prawne funkcjonowania prawa własności intelektualnej.
- Uczestniczy w przygotowaniu i realizacji projektów w poszanowaniu praw własności intelektualnej.
- Korzysta z utworów w sposób zgodny z prawem

Treści programowe dla zajęć:
- Zajęcia wprowadzające do prawa własności intelektualnej: źródła prawa, przedmiot prawa własności intelektualnej, podstawowe zasady prawa własności intelektualnej.
- Zajęcia wprowadzające do prawa autorskiego: podmiot, przedmiot prawa autorskiego, utwór, rodzaju utworów.
- Osobiste i majątkowe prawa autorskie; dozwolony użytek, odpowiedzialność za naruszenie praw autorskich, przepisy szczególne dotyczące programów komputerowych.
- Umowy w prawie autorskim ze szczególnym uwzględnieniem umów w branży IT.
- Zajęcia wprowadzające do prawa własności przemysłowej: zakres ochrony.
- Prawo własności przemysłowej: znaki towarowe, patenty inne przedmioty prawa własności przemysłowej.

Nazwa zajęć**: Przetwarzanie języka naturalnego**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna zastosowania przetwarzania języka naturalnego w systemach informatycznych.
- Rozumie pojęcie wyrażeń regularnych. Potrafi stosować wyrażenia regularne w programowaniu.
- Zna i rozumie pojęcia: analiza leksykalna oraz analiza składniowa. Potrafi implementować programy typu: lekser i parser.
- Potrafi przetwarzać, przeszukiwać, edytować teksty z wykorzystaniem poleceń systemu operacyjnego Linux.
- Zna i potrafi implementować algorytmy służące do podziału tekstu na tokeny i zdania.
- Zna i rozumie pojęcia: analiza morfologiczna, lematyzacja, stemming.
- Zna i potrafi wykorzystać w programach informatycznych zestawy narzędzi przetwarzania języka naturalnego.
- Zna naiwny klasyfikator Bayesa i potrafi go stosować w zadaniach przetwarzania języka naturalnego.
- Zna pojęcie regresji liniowej i logistycznej wielu zmiennych. Zna algorytm spadku gradientu. Potrafi zastosować metody regresji w zadaniach przetwarzania języka naturalnego.
- Zna i rozumie pojęcie sztucznej sieci neuronowej. Potrafi implementować algorytmy stosowania sieci neuronowych w przetwarzaniu języka naturalnego.
- Zna i rozumie pojęcie statystycznego modelu języka. Zna i rozumie pojęcie kanału zaszumionego. Zna podstawowe zastosowanie statystycznego modelu języka.
- Zna typy korekty pisowni. Potrafi zastosować statystyczny model języka do korekty ortograficznych błędów pisowni.
- Zna pojęcie parsingu płytkiego (ang. shallow parsing) oraz jego zastosowania w przetwarzaniu języka naturalnego. Potrafi implementować algorytm płytkiego parsingu.
- Zna pojęcie parsingu głębokiego (ang. deep parsing). Zna algorytm CYK. Zna i rozumie pojęcie probabilistycznej gramatyki struktur frazowych. Zna i rozumie pojęcie gramatyki zależności.

Treści programowe dla zajęć:
- Zastosowania przetwarzania języka naturalnego w systemach informatycznych.
- Powtórzenie podstaw języka programowania Python na podstawie przykładów związanych z przetwarzaniem języka naturalnego.
- Wyrażenia regularne – definicja matematyczna i zastosowania w informatyce.
- Zadania z wykorzystaniem wyrażeń regularnych w wyszukiwaniu i zamianie napisów w języku Python.
- Wprowadzenie pojęć: analiza leksykalna, analiza składniowa, lekser i parser. Podanie przykładu implementacji leksera i parsera w języku Python.
- Zastosowanie biblioteki PLY (Python Lex & Yacc) w przykładowym programie do analizy wypowiedzi kontrolowanego języka naturalnego.
- Polecenia systemu operacyjnego Linux, służące do przetwarzanie tekstów.

- Poznanie platformy dydaktycznej Bash Box do nauki pisania skryptów w powłoce Bash. Wykonanie zadań podanych na platformie Bash Box.
- Segmentacja tekstu. Algorytmy MaxMatch, BPE i podobne. Format SRX.
- Implementacja algorytmów segmentacji tekstu.
- Analiza morfologiczna. Lematyzacja. POS-tagging. Stemming. Algorytm Portera.
- Implementacja przykładowych algorytmów analizy morfologicznej.
- Zestawy przetwarzania języka naturalnego: NLTK, Spacy.
- Implementacja programów w języku programowania Python z wykorzystaniem bibliotek NLTK oraz Spacy.
- Wprowadzenie do uczenia maszynowego. Naiwny klasyfikator Bayesa.
- Implementacja programu autorskiego z wykorzystaniem metod uczenia maszynowego w przetwarzaniu języka naturalnego.
- Regresja liniowa. Regresja logistyczna. Metoda spadku gradientu. Zastosowanie regresji logistycznej w analizie wydźwięku.
- Sztuczne sieci neuronowe. Zastosowanie sieci neuronowych w przetwarzaniu języka naturalnego. Wektorowe reprezentacje wyrazów i testów.
- Statystyczny model języka. Kanał zaszumiony. Zastosowania statystycznego modelu języka.
- Implementacja programu autorskiego z wykorzystaniem metod statystycznych w przetwarzaniu języka naturalnego.
- Automatyczna korekta pisowni. Typy błędów pisowni. Zastosowanie metod statystycznych w korekcie ortograficznych błędów pisowni.
- Parsing płytki. Zastosowania parsingu płytkiego w przetwarzaniu języka naturalnego.
- Parsing głęboki. Alogrytm CYK. Gramatyki probabilistyczne. Niejednoznaczność leksykalna i strukturalna. Gramatyki zależności.

Nazwa zajęć**: Wstęp do matematyki**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Potrafi określić ograniczenia arytmetyki komputerowej w stosunku do pełnej teorii aksjomatycznej liczb rzeczywistych i dobrać odpowiednie metody dla unikania problemów.
- Potrafi wprawnie posługiwać się pojęciem granicy ciągu oraz sprawdzać wykonalność obliczeń na komputerze. Zna podstawowe metody obliczania granic.
- Potrafi operować ciągami rekurencyjnymi, badać ich zbieżność i granice. Potrafi określić podstawowe wady i zalety takich ciągów w zastosowaniach informatycznych.
- Potrafi badać i określać własności podstawowych funkcji, które napotyka w informatyce.
- Potrafi wykorzystać rachunek wektorowy w odniesieniu do obiektów geometrycznych. Operuje równaniami w opisach geometrycznych.
- Potrafi rozpoznać i wykorzystać metryki i różne miary odległości w informatyce. Rozpoznaje różnice w ich zastosowaniu.

Treści programowe dla zajęć:
- Cele nauczania matematyki dla informatyków. Szkic teorii aksjomatycznej liczb rzeczywistych, w tym kresy, zapis dziesiętny liczb rzeczywistych. Liczby wymierne. Potęga o wykładniku rzeczywistym. Pierwiastek. Uwagi o arytmetyce komputerowej. Arytmetyka komputera, zero (przykłady w różnych programach). Kresy zbiorów liczbowych. Proste zadania na obliczanie kresów, postacie niedziesiętne liczb rzeczywistych.
- Ciągi liczbowe: granice właściwe i niewłaściwe. Zbieżność i bezwzględna zbieżność. Ciągi monotoniczne. Podciągi, punkty skupienia i tw. Bolzano-Weierstrassa. Warunek Cauchy'ego i zupełność. Pozostałe informacje o zbieżności ciągów. Symbole Landaua i obliczanie granic z nimi związanych. Liczba e.
- Ciągi zadane rekurencyjnie w informatyce. Granice ciągów, algorytmy obliczania granic (problem zbieżności).
- Problem stopu w algorytmach. Interpretacja geometryczna. Obliczanie granic, punkty skupienia ciągów.
- Funkcje. Pojęcie funkcji (intuicyjnie, nie jako relacji), argumenty, wartości, dziedzina, przeciwdziedzina, zbiór wartości. Funkcje monotoniczne, wypukłe, parzyste, nieparzyste, okresowe. Iniekcje, suriekcje, bijekcje. Przegląd podstawowych funkcji elementarnych (afiniczna, kwadratowa, wykładnicza, logarytmiczna, trygonometryczne), ich wykresów i własności.
- Podstawy geometrii analitycznej. Podstawy rachunku wektorów i ich zastosowania. Równania prostych i powierzchni. Równanie prostej - postać kierunkowa i ogólna, warunek równoległości i prostopadłości prostych, odległość dwóch punktów i punktu od prostej, równanie okręgu. Krzywe i powierzchnie zadane parametrycznie.

- Pojęcie metryki. Przegląd metryk Minkowskiego, uwagi o nazwach metryk w materiałach anglojęzycznych (np. Czebyszewa czy Mahattan). Do wyboru omówienie: metryka Hamminga na łańcuchach znaków (kodowanie, kody z wykrywaniem błędów), metryka
- Levensheima (przetwarzanie informacji, analiza plagiatów itp.), odległości cosinusowe i miary podobieństwa w informatyce - np. w rozpoznawaniu wzorców (grafika) czy porównywaniu łańcuchów znaków. Konstrukcja i zastosowania metryk wagowych (też w informatyce) - kule w takich metrykach. Metryki w analizie obrazów.

Nazwa zajęć**: Programowanie mikrokontrolerów**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna środowisko STM32Cube i potrafi z niego korzystać.
- Zna środowisko MBED OS i potrafi z niego korzystać.
- Zna obsługę przerwań, DMA, łącze szeregowe UART i komunikację przez to łącze.
- Potrafi obsługiwać komunikaty MIDI przez łącze szeregowe oraz wysyłać przez nie logi do komputera.
- Potrafi obsługiwać przetwornik cyfrowo-analogowy.
- Potrafi zaimplementować i uruchomić konwerter MIDI/CV.

Treści programowe dla zajęć:
- Zapoznanie się ze środowiskiem programistycznym STM32Cube, konfiguracja dla konkretnego mikrokontrolera, uruchomienie prostego programu (migająca dioda).
- Zapoznanie się ze środowiskiem programistycznym wyższego poziomu abstrakcji MBED OS, uruchomienie prostego programu (migająca dioda).
- Obsługa przerwań, obsługa DMA, obsługa łącza szeregowego UART, komunikacja z konsolą na komputerze przez łącze szeregowe.
- Odbiór komunikatów MIDI przez łącze szeregowe, wysyłanie logu przez łącze szeregowe do komputera.
- Obsługa przetwornika cyfrowo-analogowego.
- Implementacja i uruchomienie konwertera MIDI/CV.

Nazwa zajęć**: Programowanie obiektowe**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna podstawowe pojęcia i techniki obiektowego paradygmatu programowania.
- Potrafi czytać i analizować kod obiektowy napisany w języku Java.
- Zna zasadę enkapsulacji, umie ją zastosować i rozumie jej znaczenie w tworzeniu oprogramowania.
- Rozumie pojęcie polimorfizmu i umie wykorzystać techniki programowania polimorficznego w praktyce.
- Zna mechanizm dziedziczenia, potrafi definiować hierarchię klas.
- Rozumie czym są kolekcje i strumienie.
- Zna mechanizmy związane z obsługą wyjątków i potrafi je wykorzystać we własnych programach.
- Potrafi wykonać niewielki projekt programistyczny w metodyce obiektowej.
- Zna podstawy programowania współbieżnego i zasady projektowania interfejsu użytkownika.

Treści programowe dla zajęć:
- Obiektowy paradygmat programowania. Różnice pomiędzy paradygmatem obiektowym i proceduralnym.
- Podstawowe wiadomości o języku Java (zapis wartości liczbowych, znakowych i łańcuchowych w języku Java, typy zmiennych w języku Java, deklaracje zmiennych, tablic, wyrażenia i operatory, instrukcje sterujące, operacje na tablicach).
- Obiekty i klasy w Javie, pola, metody. Dziedziczenie i polimorfizm, klasy abstrakcyjne, interfejsy, tworzenie obiektów oraz korzystanie z ich pól i metod.
- Przechwytywanie, obsługa i zgłaszanie wyjątków w Javie.
- Wyrażenia lambda i strumienie; wątki i UI (Java).

Nazwa zajęć**: Podstawy programowania**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Potrafi wskazać wady i zalety poznanych języków programowania.
- Analizuje prosty kod napisany w języku programowania.
- Potrafi zaprojektować prosty algorytm i zaimplementować go w języku programowania.
- Potrafi wykorzystać w implementacji istniejące biblioteki danego języka programowania.
- Wykorzystuje własne funkcje w procesie programowania.
- Potrafi samodzielnie poprawiać błędy pojawiające się w procesie kompilacji.
- Potrafi samodzielnie rozpoznać i dokonać specyfikacji problemu algorytmicznego.

Treści programowe dla zajęć:

- Struktura programu w języku C. Instrukcje wejścia i wyjścia. Zmienne i wyrażenia arytmetyczne. Relacje i operatory logiczne. Instrukcja warunkowa if-else. Bloki instrukcji.
- Tablice. Instrukcja for. Zagnieżdżanie instrukcji. Pętla while i pętla do-while. Tablice znaków. Tablice wielowymiarowe. Obsługa plików tekstowych. Operacje na tekstach.
- Funkcje. Argumenty przekazywane przez wartość. Wskaźniki i argumenty funkcji. Wskaźniki i adresy. Zmienne dynamiczne.
- Struktura programu w języku Python. Łańcuchy, podstawowe metody łańcuchów. Listy, podstawowe metody listy. Listy składane.
- Instrukcje sterujące w Pythonie. Słowniki, podstawowe metody słowników. Krotki, podstawowe metody krotek. Zbiory i operacje na zbiorach.
- Funkcje w Pytonie i przekazywanie argumentów. Rekurencja. Obsługa plików tekstowych. Reprezentacja macierzy gęstych i rzadkich w Pythonie.

Nazwa zajęć**: Praktyka zawodowa**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Potrafi zastosować wiedzę zdobytą w toku studiów w praktyce.
- Potrafi wykonać zlecone zadania na podstawie przedstawionych przykładów. Samodzielnie znaleźć rozwiązanie problemów praktycznych.
- Potrafi samodzielnie poszukiwać źródeł wiedzy i rozwiązań problemów.
- Potrafi raportować wykonane zadania i postępy prac.
- Potrafi zaplanować pracę samodzielnie lub w grupie, w celu wykonania zleconego zadania.

Treści programowe dla zajęć:
- Realizacja praktyk zgodnie z programem praktyk, przy współpracy z pracodawcą.

Nazwa zajęć**: Inżynierski projekt zespołowy 1**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Potrafi przeprowadzić proces formowania zespołu projektowego.
- Potrafi opracować dokument wizji projektu informatycznego
- Potrafi wizualizować system informatyczny za pomocą makiety/prototypu.
- Potrafi zdefiniować kryteria akceptacji dla projektu informatycznego
- Potrafi przygotować się do publicznej prezentacji koncepcji systemu informatycznego.
- Potrafi uczestniczyć w projekcie zespołowym prowadzonym metodykami zwinnymi.
- Potrafi prowadzić dokumentować przebieg projektu.
- Potrafi kontaktować się z klientem/grupą docelową w celu określenia i weryfikacji zakresu projektu informatycznego.
- Potrafi opracować specyfikację zakresu systemu informatycznego.
- Potrafi organizować pracę w trakcie rozwoju systemu informatycznego.
- Potrafi zaprojektować użyteczny system informatyczny.
- Potrafi implementować fragmenty systemu informatycznego w celu realizacji wymagań projektowych.
- Potrafi uruchomić procesy prowadzące do pozyskania systemu informatycznego o wysokiej jakości.
- Potrafi planować zadania w projekcie informatycznym.
- Potrafi zaprezentować publicznie cele i działanie systemu informatycznego.
- Potrafi przedstawić cele i działanie systemu informatycznego jego interesariuszom.
- Potrafi przygotować demonstrację systemu informatycznego.
- Potrafi korzystać z narzędzi wspierających projekty informatyczne (repozytorium kodu źródłowego, system zarządzania projektem).

Treści programowe dla zajęć:
- Formowanie zespołu.
- Zajęcia kontaktowe (1 godzina).
- Maksymalnie w pierwszych dwóch tygodniach semestru studenci formują grupy projektowe składające się z 3-5 osób. Dobór członków grupy powinien uwzględniać potrzeby projektu związane z różnymi umiejętnościami członków zespołu, niezbędnymi zarówno do implementacji rozwiązania jak i do skutecznego zarządzania pracą zespołu. Na tym etapie grupa powinna ustalić jakiego rodzaju projekt chce realizować oraz czy posiada potrzebne w tym zakresie umiejętności.
- Wizja projektu.
- Zajęcia kontaktowe (2 godziny) lub kształcenie na odległość, z wykorzystaniem interaktywnych asynchronicznych i synchronicznych sposobów komunikowania się (konsultacje, prezentacja pracy studentów, praca nad dokumentacją podczas realizacji projektu grupowego), odpowiadające 2 godzinom pracy kontaktowej.

- W pierwszym miesiącu prac nad projektem grupa powinna zdefiniować cel i założenia projektu w formie dokumentu wizji projektu. Projekt może być realizowany we współpracy z klientem zewnętrznym (spoza zespołu), na potrzeby członków zespołu, czy też na potrzeby dobrze zdefiniowanej i sprofilowanej grupy docelowej. Elementem wizji projektu jest dokonanie analizy rynku i konkurencyjnych rozwiązań oraz porównanie ich z proponowanym przez zespół rozwiązaniem. Wizja projektu powinna być rozumiana jako umowa przedwstępna na wykonanie prac programistycznych. Jest ona opiniowana przez prowadzących przedmiot, ale akceptowana przez komisję przedmiotową. W przypadku zbyt małego lub zbyt szerokiego zakresu projektu prowadzący mogą zlecić wprowadzenie zmian w dokumencie wizji projektu.
- Prototyp produktu projektu.
- Zajęcia kontaktowe (2 godziny) lub kształcenie na odległość, z wykorzystaniem interaktywnych asynchronicznych i synchronicznych sposobów komunikowania się (konsultacje, prezentacja pracy studentów, praca nad dokumentacją podczas realizacji projektu grupowego), odpowiadające 2 godzinom pracy kontaktowej.
- W drugim miesiącu prac zespół projektowy dostarcza makietę/prototyp głównego produktu projektu. Prototyp powinien być skonsultowany z klientem/grupą docelową projektu. Zadaniem prototypu jest usprawnienie procesu pozyskiwania wymagań oraz wizualizacja budowanego systemu dla zespołu projektowego, prowadzącego oraz klienta/grupy docelowej.
- Zakres projektu.
- Zajęcia kontaktowe (2 godziny) lub kształcenie na odległość, z wykorzystaniem interaktywnych asynchronicznych i synchronicznych sposobów komunikowania się (konsultacje, prezentacja pracy studentów, praca nad dokumentacją podczas realizacji projektu grupowego), odpowiadające 2 godzinom pracy kontaktowej.
- Po drugim miesiącu prac zespół projektowy dostarcza prowadzącym dokument wymagań projektowych, w którym zawarte są między innymi kryteria akceptacji projektu. Kryteria akceptacji powinny być traktowane jako kontrakt między grupą a prowadzącym na temat oczekiwanych rezultatów projektu na koniec każdego semestru. Dokument ten powinien być aktualizowany, ale tylko w porozumieniu z prowadzącym.
- Wybór architektury systemu, narzędzi i metodyki pracy.
- Zajęcia kontaktowe (2 godziny) lub kształcenie na odległość, z wykorzystaniem interaktywnych asynchronicznych i synchronicznych sposobów komunikowania się (konsultacje, prezentacja pracy studentów, praca nad dokumentacją podczas realizacji projektu grupowego), odpowiadające 2 godzinom pracy kontaktowej.
- W czasie realizacji projektu zespół projektowy musi stosować metodykę pracy pozwalającą na: kontrolę systematyczności i stanu wykonania zadań w projekcie, reagowanie na zmiany w projekcie, kontrolę czasu spędzonego nad wykonaniem zadań projektowych, planowanie i harmonogramowanie prac, kontrolę jakości tworzonego rozwiązania. Wybór metodyki leży po stronie zespołu, jednak ocenie podlega to, czy realizuje ona wyżej wymienione aspekty. W celu wykonania projektu członkowie zespołu powinni dobrać narzędzia i technologie (języki programowania, biblioteki, oprogramowanie) względem zasadności zastosowania do realizowanego projektu oraz w ramach posiadanych umiejętności.
- Implementacja projektu.
- Zajęcia kontaktowe (5 godzin).
- Kluczowym aspektem projektu jest jego realizacja, czyli implementacja zgodna z procesem inżynierii oprogramowania (w tym testowanie). W procesie tworzenia rozwiązania muszą brać udział wszyscy członkowie zespołu, nawet jeżeli ich rola projektowa skupiona jest na aspektach nieprogramistycznych. Wymaga się przy tym, aby każdy członek zespołu, niezależnie od przyjętej roli, miał wkład w efekty programistyczne projektu. Jakość powstałego rozwiązania jest oceniana przez prowadzącego i komisję na podstawie prezentacji kolejnych przyrostów projektu oraz wyników konsultacji ze zdefiniowaną w dokumencie wizji projektu grupą odbiorców projektu i przedstawienie raportów z takich konsultacji.
- Zespół projektowy przeprowadza przynajmniej dwa przyrosty, które prezentowane są przed prowadzącym zajęcia.
- Publiczna prezentacja projektu.
- Zajęcia kontaktowe (1 godzina).
- Podsumowaniem realizacji pierwszej części zespołowego projektu inżynierskiego jest publiczna prezentacja osiągniętych rezultatów.

Nazwa zajęć**: Inżynierski projekt zespołowy 2**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Potrafi zorganizować proces ciągłej integracji.
- Potrafi zorganizować proces przeprowadzenia testów systemu informatycznego.

- Potrafi zarządzać harmonogramem przy zmieniających się wymaganiach projektowych.
- Potrafi wdrożyć/przygotować do wdrożenia system informatyczny.
- Potrafi zarządzać zakresem i kryteriami akceptacji projektu.
- Potrafi uczestniczyć w projekcie zespołowym prowadzonym metodykami zwinnymi.
- Potrafi prowadzić dokumentować przebieg projektu.
- Potrafi kontaktować się z klientem/grupą docelową w celu określenia i weryfikacji zakresu projektu informatycznego.
- Potrafi optymalizować (refaktoryzować) strukturę kodu źródłowego.
- Potrafi organizować pracę w trakcie rozwoju systemu informatycznego.
- Potrafi zaprojektować użyteczny system informatyczny.
- Potrafi implementować fragmenty systemu informatycznego w celu realizacji wymagań projektowych.
- Potrafi uruchomić procesy prowadzące do pozyskania systemu informatycznego o wysokiej jakości.
- Potrafi planować zadania w projekcie informatycznym.
- Potrafi zaprezentować publicznie cele i działanie systemu informatycznego.
- Potrafi przedstawić cele i działanie systemu informatycznego jego interesariuszom.
- Potrafi przygotować demonstrację systemu informatycznego.
- Potrafi korzystać z narzędzi wspierających projekty informatyczne (repozytorium kodu źródłowego, system zarządzania projektem).

Treści programowe dla zajęć:

- Refaktoryzacja.
- Zajęcia kontaktowe (2 godziny) lub kształcenie na odległość, z wykorzystaniem interaktywnych asynchronicznych i synchronicznych sposobów komunikowania się (konsultacje, prezentacja pracy studentów, praca nad dokumentacją podczas realizacji projektu grupowego), odpowiadające 2 godzinom pracy kontaktowej.
- Optymalizacja struktury wewnętrznej kodu źródłowego bez zmiany funkcjonalnej systemu, czyli refaktoryzacja, jest bardzo ważnym elementem zwinnego i przyrostowego wytwarzania oprogramowania. W pierwszych 2 tygodniach zajęć zespół w ramach pierwszego przyrostu zespół dokonuje ewaluacji obecnego poziomu złożoności kodu źródłowego, planuje a następnie przeprowadza stosowne optymalizacje.
- Implementacja i testy przygotowanego rozwiązania.
- Zajęcia kontaktowe (5 godzin) oraz zajęcia kontaktowe (5 godzin) lub kształcenie na odległość, z wykorzystaniem interaktywnych asynchronicznych i synchronicznych sposobów komunikowania się (konsultacje, prezentacja pracy studentów, praca nad dokumentacją podczas realizacji projektu grupowego), odpowiadające 5 godzinom pracy kontaktowej.
- Kluczowym aspektem projektu jest jego realizacja, czyli implementacja zgodna z procesem inżynierii oprogramowania (w tym testowanie). W procesie tworzenia rozwiązania muszą brać udział wszyscy członkowie zespołu, nawet jeżeli ich rola projektowa skupiona jest na aspektach nieprogramistycznych. Wymaga się przy tym, aby każdy członek zespołu, niezależnie od przyjętej roli, miał wkład w efekty programistyczne projektu. Jakość powstałego rozwiązania jest oceniana przez prowadzącego i komisję na podstawie prezentacji kolejnych przyrostów projektu, testów wykonanych przez użytkowników zewnętrznych, klienta oraz testów kodu. W przypadku braku klienta obowiązkiem grupy jest konsultacja na etapie implementacji ze zdefiniowaną w dokumencie wizji projektu grupą odbiorców projektu i przedstawienie raportów z takich konsultacji. Zespół projektowy przeprowadza przynajmniej cztery przyrosty.
- Wdrożenie produktów projektu.
- Zajęcia kontaktowe (2 godziny).
- Etapem końcowym projektu jest jego wdrożenie. Przez wdrożenie rozumie się przekazanie rozwiązania do użytku/testów klientowi lub publikację projektu w domenie publicznej i zebranie opinii od grupy docelowych użytkowników produktu. Elementem wdrożenia jest także dostarczenie dokumentacji projektu, opisującej jego cele, funkcje i architekturę.
- Publiczna prezentacja projektu.
- Zajęcia kontaktowe (1 godziny).
- Podsumowaniem realizacji pierwszej części zespołowego projektu inżynierskiego jest publiczna prezentacja osiągniętych rezultatów.

Nazwa zajęć**: Pracownia programowania**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna podstawowe narzędzia tworzenia projektów programistycznych, potrafi wykorzystywać podstawowe narzędzia informatyczne.
- Potrafi zbudować prosty system w architekturze wielowarstwowej lub rozproszonej.

- Potrafi zgodnie z zadaną specyfikacją zaprojektować oraz zrealizować prosty system informatyczny, używając właściwych metod, technik i narzędzi.
- Potrafi pisać, uruchamiać i testować programy w wybranym środowisku programistycznym.
- Zna budowę systemów wielowarstwowych i rozproszonych.
- Zna problemy zarządzania informacją, w tym dotyczące systemów baz danych, modelowania danych, składowania i wyszukiwania informacji.
- Zna technologie sieciowe, w tym podstawowe protokoły komunikacyjne, bezpieczeństwo i budowę aplikacji sieciowych (siedmiowarstwowy model ISO, protokoły komunikacyjne w tym TCP/IP, trasowanie, model klient-serwer, protokoły kryptograficzne.
- Zna metody projektowania i programowania obiektowego (kapsułkowanie i ukrywanie informacji, klasy i podklasy, dziedziczenie, polimorfizm, hierarchie klas).
- Zna zagadnienia inżynierii oprogramowania, w tym projektowania (wzorce projektowe, architektura oprogramowania, analiza i projektowanie obiektowe), wykorzystania API, narzędzi i środowisk wytwarzania oprogramowania (narzędzia do analizy wymagań i modelowania).
- Zna narzędzia, technologie i urządzenia informatyczne właściwe dla wybranych obszarów zastosowań oraz podstawy ich działania.
- Zna ograniczenia własnej wiedzy i rozumie potrzebę dalszego kształcenia.
- Potrafi posługiwać się przynajmniej jednym z najbardziej popularnych systemów zarządzania wersjami.
- Potrafi stosować techniki prowadzące do otrzymania oprogramowania wysokiej jakości.
- Potrafi zgodnie z zadaną specyfikacją zaprojektować oraz zrealizować prosty system informatyczny, używając właściwych metod, technik i narzędzi.
- Potrafi budować proste systemy bazodanowe wykorzystujące przynajmniej jeden z najbardziej popularnych systemów zarządzania bazą danych.
- Potrafi pisać, uruchamiać i testować programy w wybranym środowisku programistycznym.
- Potrafi pozyskiwać informacje z literatury, baz wiedzy, Internetu oraz innych wiarygodnych źródeł, integrować je, dokonywać ich interpretacji oraz wyciągać wnioski i formułować opinie.

Treści programowe dla zajęć:

- Przygotowanie środowiska programistycznego, wybór edytora, repozytorium kodu, sposobu budowania aplikacji oraz logowania komunikatów. Sugerowana ścieżka: język Java, środowisko IntelliJ, repozytorium Git, manadżer budowania Maven, Loger komunikatów log4J.
- Debuggowanie i testowanie kodu w wybranym środowisku programistycznym. Sugerowana ścieżka: JUnit + IntelliJ.
- Przetwarzanie strumieniowe, paradygmat programowania funkcyjnego, kolekcje. Sugerowana ścieżka: Java Streams, Java Collections.
- Serializacja i deserializacja danych w postaci JSON i XML. Sugerowana ścieżka: Jackson.
- Mapowanie relacyjno - obiektowe (ORM), praca na modelu obiektowym, zapytania SQL w modelu ORM. Sugerowana ścieżka: Hibernate.
- Tworzenie i praca z REST API. Postman jako narzędzie komunikacji z restowym API. Sugerowana ścieżka: Postman + Tomcat.
- Framework do tworzenia serwisu opartego na REST API z przykładową implementacją. Sugerowana ścieżka: Spring + Spring Boot.
- Serwery aplikacji, osadzanie projektu na serwerze aplikacji, konfiguracja serwera, monitorowanie i debuggowanie osadzonej aplikacji. Sugerowana ścieżka: Tomcat.
- Tworzenie serwisu WWW dla serwera korzystającego z REST API, przykładowa implementacja. Sugerowana ścieżka: React.
- Integracja wielu serwisów – autoryzacja za pomocą OAuth i zewnętrznego API. Sugerowana ścieżka JWT i GoogleAPI.
- Indywidualna obrona projektu podsumowującego zagadnienia 1-3, 4-7 i 8-10.
- Praca nad projektem, konsultacje sytuacji problemowych.

Nazwa zajęć**: Przetwarzanie obrazów**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Zna tematykę badawczą dziedziny widzenia komputerowego. Potrafi przygotować środowisko programistyczne do przetwarzania obrazów i wideo oraz wykonywać w nim podstawowe operacje.
- Rozumie zasady postrzegania wzrokowego i potrafi konstruować algorytmy zgodnie z nimi.
- Zna klasyczne typy metod służących do wydobywania i przetwarzania informacji zawartej w obrazach cyfrowych.
- Potrafi stosować algorytmy binaryzacji na obrazach.
- Potrafi stosować wzmacnianie i filtrowanie obrazów.

- Potrafi analizować materiały wideo w celu śledzenia obiektów.
- Potrafi wykonywać transformacje geometryczne i operować cechami obrazów.
- Potrafi dokonywać segmentacji i rozpoznawania obrazów.

Treści programowe dla zajęć:

- Teoria widzenia koloru, modele koloru, formaty graficzne. Podstawowe operacje punktowe na kolorach. Histogram, jego zastosowania - konstrukcja, rozciągniecie i wyrównanie histogramu.
- Filtry liniowe i ich zastosowania.
- Wybrane algorytmy binaryzacji obrazu.
- Algorytmy wykrywania krawędzi - klasyczne rozwiązania przez filtry liniowe i morfologię. Algorytm Canny'ego wykrywania krawędzi.
- Wykrywanie linii - transformata Hougha oraz Ransac Transformata Hougha jako narzędzie wykrywanie prostokątów.
- Wybrane zagadnienia segmentacji obrazu.
- Wykrywanie narożników - wybrane algorytmy. Śledzenie w materiale wideo.

Nazwa zajęć**: Seminarium dyplomowe**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Potrafi przedstawić ustnie kilkudziesięciominutową prezentację na zadany temat na odpowiednim poziomie merytorycznym; potrafi zredagować szczegółowy konspekt prezentacji oraz przygotowywanej pracy inżynierskiej.
- Sprawnie posługuje się matematycznym językiem i notacją, rozumie ich specyfikę.
- Umie wyszukiwać materiały w bazach danych i zasobach bibliotecznych niezbędne do przygotowania prezentacji oraz pracy inżynierskiej.
- Dokonuje właściwej i krytycznej oceny oraz selekcji materiału zebranego do prezentacji i pracy inżynierskiej.
- Rozumie konieczność systematycznej pracy, stałego uzupełniania i aktualizowania posiadanej wiedzy.
- Potrafi formułować i objaśniać najważniejsze pojęcia i twierdzenia z działów informatyki bezpośrednio związanych z tematyką seminarium oraz pracą inżynierską.

Treści programowe dla zajęć:

- Treści kształcenia ustala prowadzący seminarium w zależności od problematyki seminarium powiązanej z tematami prac inżynierskich.

Nazwa zajęć**: Sieci komputerowe**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Rozumie czym są sieci fizyczne i intersieć.
- Wie jak pakiety znajdują trasę do celu (routing).
- Zna podstawowe protokoły niższych i wyższych warstw.
- Wie jaka jest struktura Internetu.
- Wie co to jest system automatyczny.
- Rozróżnia routing wewnętrzny i zewnętrzny.
- Wie jak działają sieci fizyczne przewodowe i bezprzewodowe, zarówno LAN jak i WAN.
- Rozumie jak Internet radzi sobie z przesyłaniem multimediów.
- Wie jak zabezpieczać sieć i komunikację sieciową.

Treści programowe dla zajęć:

- Omówienie pojęć: sieć fizyczna i intersieć, węzeł sieci fizycznej, interfejs sieciowy, pakiet, router w intersieci, przekazywanie pakietów, typy sieci fizycznych:
- LAN/MAN/WAN oraz przykłady (eth, wifi i inne), rodzaje adresów, przydzielanie adresów w sieci fizycznej i intersieci, adresy IP klasowe/bezklasowe, maska, adresy specjalne (np. prywatne), routery w intersieci vs
- przełączniki ethernetowe.
- Laboratorium: polecenia sieciowe (z użyciem np. VBox i Linuxa); adresacja IP, maski, adresy klasowe/bezklasowe.
- Protokoły, warstwy protokołów, enkapsulacja pakietów, warstwy ISO vs TCP/IP, krótkie omówienie czym się zajmują prot: IP, TCP, UDP oraz ARP, DHCP; filtrowanie pakietów (zapory i NAT) przy pomocy iptables; gniazdka BSD w j. C dla prot. TCP i UDP, rola nr portu w połączeniach TCP i w UDP. Krótkie omówienie usług i prot. nad warstwą 4 (transport): TELNET/SSH, FTP, HTTP, SSL/TLS, prot. mailowe (SMTP, POP3, IMAP), usługa DNS, LDAP/X.500; szerzej o HTTP i jego zastosowaniach; więcej o zabezpieczeniach połączeń SSL/TLS.
- Laboratorium: gniazda BSD w j. C (prot. TCP), gniazda BSD w jęz. C (prot. UDP).

- Dokładniejsze omówienie prot. IP, omówienie pól w nagłówku, fragmentacja, przekazywanie pkg, rola ICMP. Szczegółowe omówienie prot. TCP, zasada działania poł. TCP, segmenty, okno nadawcze, różne wersje TCP, strategie używane w TCP, "zrywanie się" poł. TCP.
- Laboratorium: gniazda BSD w innych językach, w tym skryptowych, przekazywanie struktur danych przez sieć (różne rozwiązania),
- Warstwy 1 i 2: dokładniejsze omówienie sieci Ethernet: format ramki, zasada dostępu do medium wczoraj i dziś, topologia sieci Eth, typy kabli, zasada działania switcha, VLAN, przyszłość Eth (carrier eth, metro eth), sygnał w kablu eth, metody wykrywania i naprawiania błędów w ramkach.
- Laboratorium: obserwowanie działania prot. pod wireshark: Ethernet/IP/TCP/UDP oraz wyższe warstwy.
- Warstwy 1 i 2: dokładniejsze omówienie sieci bezprzewodowych: Wifi, infrastruktura sieci Wifi, format ramki, dostęp do medium w sieci wifi, NAV, RTS/CTS itp, zabezpieczenia sieci wifi: WEP, WPA-Personal/ Enterprise, warstwa fizyczna sieci wifi, wifi w Linuxie: wpa_supplicant, hostapd, iwconfig, iwlist, itp. Bluetooth, zasada działania, piconet, scatternet; polecenia linuxowe dotyczące BT; łącze szeregowe nad bt (rfcomm).
- Laboratorium: eksperymenty z protokołami nad warstwą transp.: FTP, HTTP, mailowe, DNS, DDNS, SSL, tworzenie cert SSL itp.
- Routing, algorytmy ustalania tablic routingowych, typy tych alg (DV i stanu łącza), prot. RIP, OSPF, …; struktura Internetu: rola ISP, sieci szkieletowe, historia rozwoju Internetu, pojęcie systemu autonomicznego (AS), routing wewnętrzny i zewnętrzny, prot. EGP, BGP-4, usprawnienia routingu: MPLS.
- Laboratorium: ataki sieciowe, bezpieczeństwo w sieci, "hackowanie" wifi, wykradanie session_id w http, itp.
- W stronę telekomunikacji i sieci WAN: typy sieci: "z przełączaniem obwodów", "z przełączaniem pakietów", sieci telefoniczne, multipleksowanie: TDM/FDM, sieci optyczne SONET/SDH, sieć ATM, prot PPP na łączem szeregowym. Multimedia w Internecie, prot. RTP/RTCP, voip czyli prot. SIP, IAX2, H.323; jak zapewnić namiastkę(?) QoS w Internecie? Walka z "best effort": rola prot. UDP, Intserv, Diffserv, RSVP; multicasting, prot. IGMP, narzędzia mbone; software-owa centrala telefoniczna "asterisk".
- Laboratorium: budowanie małych sieci LAN i ich konfigurowanie (ifconfig, route, iptables), emulować sieci przy pomocy VBox, małego Linuxa.
- Prot IPv6 jako następca IPv4, adresy IPv6, łączenie IPv6 i IPv4, dlaczego IPv6 nie zdominowało świata? Automat. konfiguracja sieci DHCP (szczegóły działania), prot. wspierający zarządzanie siecią: SNMP, zmienne MIB, programy do zarzadzania siecią (np. program "scotty").
- Laboratorium: prezentacje projektów przez studentów.

Nazwa zajęć**: Systemy operacyjne**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Posiada podstawową wiedzę na temat idei oraz algorytmów wykorzystywanych w systemach operacyjnych w przeszłości i obecnie.
- Zna podstawowe fakty dotyczące budowy wybranych przedstawicieli systemów operacyjnych z rodziny Unix oraz Windows.
- Umie posługiwać się podstawowymi poleceniami systemów operacyjnych z rodziny Unix oraz Windows, służącymi do
- wykonywania operacji na plikach, procesach/wątkach i urządzeniach wejścia/wyjścia.
- Potrafi przeczytać ze zrozumieniem/napisać skrypt w języku powłoki BASH, wykorzystujący podstawowe konstrukcje sterujące tej powłoki dostępne w systemach z rodziny Unix.
- Potrafi przeczytać ze zrozumieniem/napisać program w języku C zawierający wywołania funkcji systemowych Unixa dotyczących procesów, plików i sygnałów.
- Rozumie/ umie wykonać podstawowe czynności związane z administrowaniem systemem operacyjnym z rodziny Unix i Windows.
- Zna podstawowe fakty oraz algorytmy związane ze współbieżnym wykonywaniem procesów/wątków.
- Zna/potrafi zastosować w praktyce podstawowe algorytmy przydziału procesów do procesorów.
- Ma świadomość znaczenia i roli systemów operacyjnych w informatyce, rozumie potrzebę dalszego kształcenia w tym zakresie.

Treści programowe dla zajęć:
- Wprowadzenie do tematyki SO, Podstawowe definicje i pojęcia: funkcje systemu operacyjnego; unix z perspektywy użytkownika: pojęcia i polecenia dotyczące systemu plików (plik, katalog, montowanie, prawa do plików).

- UNIX z perspektywy użytkownika c.d.: pojęcia dotyczące procesów (proces, polecenie ps, sygnały, deskryptory, stdin/out, procesy macierzysty/potomny, pierwszo/drugo-planowy, potoki, łącza) powłoka bash - krótki przegląd.
- Instalacja i konfiguracja systemu LINUX, rozruch systemu, podstawowe usługi systemowe i sieciowe.
- UNIX z perspektywy programisty: krótki przegląd funkcji systemowych; UNIX z perspektywy administratora: pliki z kat. /etc., poziomy działania, konfiguracja podstawowych usług.
- Historia rozwoju systemów operacyjnych (motywacja idei wieloprogramowości i podziału czasu); klasyfikacja architektur systemów operacyjnych (monolityczne, warstwowe, z mikrojądrem itp); założenia sprzętowe (pojęcia: magistrala, kontroler, port, przerwanie, DMA, ...), architektura x86.
- Zarządzanie procesami oraz wątki (diagram stanów
- procesu, kolejki BKP, planiści, SJF, RR), zagadnienia systemów wieloprocesorowych.
- Systemy plików (atrybuty pliku, katalogi, dowiązania twarde i symb. + ich impl.) implementacja systemu plików, przydział listowy, FAT, indeksowy; struktury danych kernela związane z plikami (w tym omówienie idei VFS); omówienie i porównanie konkretnych systemów plików Linuxa i Windows: ext2/3/4, xfs, ntfs, siecowe systemy plików: CIFS/SMB, SSHFS, NFS ; woluminy linuxowe (lvm2) i windowsowe; dyski RAID.
- Obsługa urządzeń wejścia i wyjścia, pliki specjalne blokowe i znakowe, pojęcie sterownika sprzętowego i programowego, zasada działania sys. we/wy; struktura warstwowa sterowników (sieci i sys. plików), struktury danych dotyczące we/wy w linuxie (tabl. deskr., tabl. plików, tabl. i-węzłów) nietypowe operacje we/wy (asynchr., nieblokujące, fun. select(), itp.) obsługa sieci (interfejsy sieciowe); obsługa urządzeń USB.
- Współbieżność, synchronizacja procesów: semafory, sem. binarne, monitory, problemy współbieżności (sekcja krytyczna, producent/konsument, czytelnicy i pisarze, n-filozofów itp.).
- Rodzaje pamięci, hierarchia pamięci, cache; ochrona sprzętowa; przegląd składników sys. op.
- Zarządzanie pamięcią operacyjną; w tym także: tworzenie programów, bibl. dynamiczne, przydzielanie pamięci procesom i jakie to rodzi problemy; sposoby zarządzania pamięcią: rej. przesunięcia, stronicowanie, segmentacja, rozwiązania stosowane w procesorach x86 i późniejszych, pamięć wirtualna, algorytmy wyszukiwania "ramki ofiary" (FIFO, LRU).
- Wirtualizacja: pojęcie i typy wirtualizacji, pojęcie hyperwizora, przegląd oprogramowania służącego do wirtualizacji.
- Bezpieczeństwo systemu komputerowego - standardy, uprawnienia, zabezpieczanie logowania i kont użytkowników, zabezpieczenia kryptograficzne, bezpieczeństwo sieciowe, moduły autoryzacyjne (np. PAM).
- Omówienie architektury LINUX i Android
- Omówienie architektury Windows.
- Obsługa systemu LINUX - Powłoka BASH, obsługa systemu pików, zmienne środowiskowe, std-in/stdout.
- Obsługa systemu LINUX - Powłoka BASH, obsługa procesów, sygnały, łącza, potoki, filtry, skrypty.
- Instalacja i konfiguracja maszyny wirtualnej, instalacja systemu LINUX.
- Konfiguracja podstawowych usług sieciowych i systemowych.
- Instalacja modułów, instalacja oprogramowania ze źródeł, gcc, make, biblioteki linkowane statycznie i dynamicznie.
- LINUX - wprowadzenie do funkcji systemowych - zrządzanie procesami.
- LINUX - wprowadzenie do funkcji systemowych - zarządzanie systemem plików.
- LINUX - moduły jądra, implementacja prostego sterownika.
- Współbieżność, problem sekcji krytycznej, semafory (BACI).
- Klasyczne problemy współbieżności (BACI).
- Współbieżność, monitory (BACI).
- LINUX - współbieżność – semafory.
- LINUX - współbieżność - IPC, pamięć dzielona, łącza nazwane i nienazwane.
- Windows, zarządzanie systemem, PowerShell/TWAPI.
- Windows, współbieżność, mutexy, zarządzanie procesami, POSIX.

Nazwa zajęć**: Statystyka**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna podstawy języka programowania R. Zna podstawowe konstrukcje programistyczne oraz składnie tego języka.
- Umie opisać rozkład empiryczny badanej cechu za pomocą odpowiednich tabel, wykresów oraz statystyk opisowych.

- Potrafi dobrać odpowiedni do danego zagadnienia model statystyczny. Potrafi dokonać estymacji parametrów przyjętego modelu.
- Zna konstrukcje testów statystycznych. Potrafi dobrać odpowiedni test do rozważanego zagadnienia.
- Potrafi wykonać analizę wariancji, sprawdzić jej założenia. Zna pojęcie układu doświadczalnego.
- Zna podstawowe modele regresji: regresja liniowa prosta, regresja liniowa wielokrotna, regresja nieliniowa, regresja logistyczna. Potrafi dobrać odpowiedni model oraz sprawdzić jego założenia.
- Umie zastosować wybrane procedury statystyki wielowymiarowej: analizę składowych głównych, analizę skupień, klasyfikację.

Treści programowe dla zajęć:

- Podstawy języka programowania R. Podstawowe konstrukcje programistyczne (przypisanie, instrukcje sterujące, pętle, wywoływanie podprogramów i przekazywanie parametrów) oraz składnia i struktury danych tego języka.
- Opis rozkładu empirycznego badanej cechu za pomocą odpowiednich tabel, wykresów oraz statystyk opisowych, np. szereg rozdzielczy, histogram, wykres słupkowy, wykres kołowy, średnia, mediana, wariancja, odchylenie standardowe, współczynnik zmienności.
- Model statystyczny (model normalny, wykładniczy, dwumianowy, Poissona, jednostajny). Estymacja punktowa i przedziałowa parametrów modelu.
- Testy statystyczne. Hipotezy statystyczne, obszar krytyczny, błędy pierwszego i drugiego rodzaju, poziom istotności testu, p-wartość, test ilorazu wiarogodności. Testy t Studenta, testy chi-kwadrat Pearsona.
- Analiza wariancji: założenia, hipoteza ogólna, testy POST HOC. Układ doświadczalny.
- Podstawowe modele regresji: regresja liniowa prosta, regresja liniowa wielokrotna, regresja nieliniowa, regresja logistyczna. Metody szacowania parametrów. Dobór modelu oraz jego założenia.
- Wybrane procedury statystyki wielowymiarowej: analiza składowych głównych, analiza skupień, klasyfikacja.

Nazwa zajęć**: Sztuczna inteligencja**

Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:

- Rozumie pojęcie sztuczna inteligencja oraz potrafi wskazać jej działy. Zna historię sztucznej inteligencji.
- Zna założenia podejścia agentowego w definiowaniu problemów sztucznej inteligencji.
- Zna metody rozwiązywania problemów poprzez przeszukiwanie przestrzeni stanów i potrafi je zastosować.
- Rozumie zasady prowadzenia wnioskowania w języku logiki i umie je zastosować.
- Zna metody wnioskowania nieprecyzyjnego i potrafi je wykorzystać.
- Zna podstawowe narzędzia i techniki reprezentacji wiedzy oraz umie reprezentować wiedzę z ich wykorzystaniem.
- Zna metody uczenia maszynowego i umie je zastosować.
- Zna sposoby wykorzystania sztucznych sieci neuronowych.
- Zna obszary zastosowania sztucznej inteligencji.
- Rozumie odpowiedzialność twórców systemów sztucznej inteligencji. Rozumie problemy etyczne sztucznej inteligencji.

Treści programowe dla zajęć:

- Czym jest sztuczna inteligencja. Różne sposoby jej definiowania. Rys historyczny. Dziedziny sztucznej inteligencji.
- Podejście agentowe w definiowaniu problemów sztucznej inteligencji. Implementacja środowiska agentowego.
- Rozwiązywanie problemów poprzez przeszukiwanie przestrzeni stanów. Reprezentacja przestrzeni stanów. Niepoinformowane i poinformowane strategie przeszukiwania. Implementacja przestrzeni stanów i strategii przeszukiwania przy rozwiązywaniu problemów.
- Wnioskowanie w języku logiki. Wykorzystanie w sztucznej inteligencji klasycznego rachunku zdań, logiki pierwszego rzędu, logiki rozmytej. Wnioskowanie w systemach sztucznej inteligencji.
- Wnioskowanie nieprecyzyjne (reguły rozmyte, przybliżone, Baysowskie).
- Reprezentacja wiedzy. Wybrane formalizmy reprezentacji wiedzy i związane z nimi struktury danych.
- Uczenie maszynowe nadzorowane i nienadzorowane.
- Klasyfikacja i predykcja. Miary w uczeniu maszynowym. Wybrane metody uczenia maszynowego: drzewa decyzyjne, regresja liniowa, regresja logistyczna.
- Sztuczne sieci neuronowe. Głębokie sieci neuronowe.
- Wykorzystanie sieci neuronowych w systemach sztucznej inteligencji.
- Wybrane obszary zastosowań sztucznej inteligencji.

- Odpowiedzialności twórców systemów sztucznej inteligencji, problemy etyczne, przyszłość sztucznej inteligencji.

Nazwa zajęć**: Technologie internetowe**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna podstawowe pojęcia związane z Internetem.
- Potrafi skonfigurować domenę internetową.
- Potrafi wykorzystać podstawowe narzędzia przydatne w pracy z Internetem.
- Zna podstawowe protokoły poczty elektronicznej.
- Potrafi korzystać z podpisów elektronicznych oraz szyfrowania wiadomości e-mail.
- Posiada wiedzę na temat bezpieczeństwa usług w Internecie.
- Posiada wiedzę na temat tworzenia stron WWW.
- Potrafi stworzyć stronę internetową korzystając z HTML oraz CSS.
- Rozumie zasadę działania protokołu http.
- Potrafi stworzyć dynamiczną stronę internetową.
- Zna zasadę działania stron typu Single Page Application.
- Potrafi stworzyć stronę internetową korzystającą z JavaScript.
- Zna podstawowe aspekty związane z komercyjnym wykorzystaniem stron WWW.
- Potrafi opublikować stronę internetową.
- Potrafi zainstalować i skonfigurować System Zarządzania Treścią (CMS).
- Potrafi skonfigurować stronę WWW korzystającą z protokołu HTTPS.
- Rozumie zagadnienie wykorzystania ciasteczek i innych danych przechowywanych w przeglądarce.
- Rozumie problem prywatności w Internecie.
- Potrafi korzystać z narzędzi zwiększających prywatność w Internecie.
- Rozumie idee oprogramowania typu firewall.
- Zna architekturę klient-serwer.
- Zna pojęcie webserwisów.
- Potrafi operować danymi w formacie JSON.
- Rozumie potrzebę integracji z systemami zewnętrznymi.
- Potrafi wykorzystać system zewnętrzny do uwierzytelniania użytkownika na stronie internetowej.
- Rozumie pojęcie chmury oraz model infrastructure as a service.
- Potrafi stworzyć prostą aplikację internetową opartą o chmurę.

Treści programowe dla zajęć:
- Pojęcia podstawowe: Internet, domena, DNS, serwis, protokół, dostawca usług internetowych. Zadania do zrealizowania: Konfiguracja serwera DNS. Whois, lokalizacja po IP, traceroute.
- Poczta internetowa: protokoły e-mailowe, własności poczty internetowej, operacje, bezpieczeństwo, szyfrowanie i podpisywanie wiadomości (PGP), dostawcy usług, następcy email - ProtonMail.com. Zadania do zrealizowania: Wysyłanie (SMTP) i odbieranie (POP3) wiadomości przez Telnet. Konfiguracja szyfrowania i podpisów PGP/GPG.
- Prywatność i bezpieczeństwo w Internecie, VPN. Zadania do zrealizowania: Omówienie ciasteczek, webstorage, trybu incognito, AdBlocka, konfiguracja proxy, firewall, ssh Tora oraz UFW.
- Przesyłanie danych w Internecie: architektura klient-serwer, MVC, webserwisy. Zadania do zrealizowania: Omówienie pojęć JSON, RPC, REST. Obsługa narzędzi cURL oraz jq.
- Witryny internetowe: typy witryn: statyczne i dynamiczne, elementy witryny, zasada działania protokołu HTTP. Tworzenie stron internetowych: HTML i CSS. Zadania do zrealizowania: HTML, CSS – podstawy; przygotowanie średnio złożonej statycznej strony WWW.
- Systemy Zarządzania Treścią (CMS). Zajęcia kontaktowe (1 godzina) oraz kształcenie na odległość, z wykorzystaniem interaktywnych asynchronicznych i synchronicznych sposobów komunikowania się (praca z laboratorium cyfrowym), odpowiadające 1 godzinie pracy kontaktowej. Zadania do zrealizowania: Wordpress - konfiguracja, bezpieczeństwo, komunikacja z bazą danych pgAdmin/phpMyAdmin, stworzenie prostego serwisu
- Tworzenie i publikowanie witryn internetowych: hosting, bezpieczeństwo, SEO, aspekty komercyjne, Google Analytics. Zadania do zrealizowania: Obsługa FTP, SCP, konfiguracja nginx/apache, plik .htaccess.
- Dynamiczne witryny internetowe. Zadania do zrealizowania: Programowanie witryn internetowych z wykorzystaniem Python/Django.
- Witryny typu Single Page Application (SPA). Zadania do zrealizowania: Wprowadzenie do JavaScript, Angular, NodeJS.

- Chmura (cloud): model infrastructure as a service. Zadania do zrealizowania: Stworzenie prostej aplikacji w chmurze: Google Firebase / Amazon WS / MS Azure / Google Cloud Platform.
- Bezpieczeństwo stron www - protokół HTTPS. Zadania na zajęciach: Konfiguracja certyfikatu SSL - Lets Encrypt, Certbot.

Nazwa zajęć**: Uczenie maszynowe**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Rozumie rolę i znaczenie uczenia maszynowego we współczesnej informatyce, potrafi wskazać przykłady zastosowań uczenia maszynowego.
- Potrafi wyróżnić podstawowe typy zadań uczenia maszynowego i wskazać ich przykłady.
- Umie korzystać z podstawowych narzędzi bibliotek NumPy i PyTorch oraz elementów języka Python przydatnych do implementowania rozwiązań z dziedziny uczenia maszynowego.
- Umie przetwarzać dane przechowywane w tekstowych formatach tabelarycznych (CSV/TSV).
- Umie wizualizować dane, korzystając z bibliotek Matplotlib i Seaborn.
- Rozumie zagadnienie regresji liniowej jednej i wielu zmiennych.
- Rozumie metodę gradientu prostego.
- Umie zaimplementować algorytm gradientu prostego do znalezienia rozwiązania problemu regresji liniowej.
- Rozumie zagadnienie regresji logistycznej.
- Umie zaimplementować algorytm gradientu prostego do znalezienia rozwiązania problemu regresji logistycznej.
- Rozumie znaczenie ewaluacji algorytmów uczenia maszynowego i zna jej podstawowe metody.
- Rozumie rolę zbiorów danych: uczącego, walidacyjnego i testowego, i potrafi z nich korzystać.
- Zna podstawowe miary jakości stosowane przy ewaluacji algorytmów uczenia maszynowego.
- Potrafi korzystać z modułów pakietu Scikit-Learn do implementacji rozwiązań uczenia maszynowego.
- Potrafi dokonać ewaluacji zaimplementowanego rozwiązania.
- Rozumie zjawiska nadmiernego i niedostatecznego dopasowania.
- Zna metody regularyzacji.
- Umie zapobiegać nadmiernemu i niedostatecznemu dopasowaniu w implementowanych przez siebie rozwiązaniach.
- Umie poprawnie reprezentować dane różnych typów i korzystać z nich do rozwiązywania problemów metodami uczenia maszynowego.
- Rozumie znaczenie optymalizacji i zna jej podstawowe metody.
- Umie stosować metody optymalizacji uczenia maszynowego.
- Rozumie ideę uczenia nienadzorowanego i zna najważniejsze algorytmy uczenia nienadzorowanego.
- Potrafi zaimplementować przykładowy algorytm uczenia nienadzorowanego.
- Rozumie zasadę działania naiwnego klasyfikatora bayesowskiego.
- Rozumie zasadę działania algorytmu k najbliższych sąsiadów.
- Rozumie zasadę działania drzew decyzyjnych.
- Rozumie zasadę działania maszyn wektorów nośnych.
- Rozumie zasadę działania sztucznych sieci neuronowych, w tym wielowarstwowych.
- Potrafi wykorzystywać metodę propagacji wstecznej do uczenia wielowarstwowych sieci neuronowych.
- Potrafi implementować sieci neuronowe z wykorzystaniem odpowiednich bibliotek.
- Rozumie zasadę działania i potrafi wskazać zastosowania splotowych sieci neuronowych.
- Rozumie zasadę działania i potrafi wskazać zastosowania rekurencyjnych sieci neuronowych.
- Rozumie zasadę działania i potrafi wskazać zastosowania modeli typu encoder-decoder, w szczególności do tworzenia modeli języka i tłumaczenia maszynowego.
- Rozumie ideę uczenia przez wzmacnianie i zna podstawowe paradygmaty uczenia przez wzmacnianie.
- Potrafi zaprojektować, zaimplementować i zewaluować system wykorzystujący uczenie maszynowe.

Treści programowe dla zajęć:
- Wprowadzenie do uczenia maszynowego. Czym jest uczenie maszynowe? Rola i miejsce uczenia maszynowego we współczesnej informatyce. Przegląd zastosowań i metod uczenia maszynowego. Podstawowe pojęcia związane z uczeniem maszynowym.
- Podstawowe narzędzia uczenia maszynowego. Elementy języka Python przydatne przy implementowaniu algorytmów uczenia maszynowego. Biblioteki NumPy i PyTorch
- Narzędzia przetwarzania i wizualizacji danych w języku Python. Format CSV/TSV. Biblioteki Matplotlib i Seaborn.
- Regresja liniowa jednej zmiennej. Funkcja kosztu. Metoda gradientu prostego. Regresja liniowa wielu zmiennych.

- Implementacja regresji liniowej jednej zmiennej w języku Python.
- Regresja logistyczna. Metoda gradientu prostego dla regresji logistycznej.
- Implementacja regresji logistycznej w języku Python.
- Ewaluacja algorytmów uczenia maszynowego. Podział na zbiory: uczący, testowy i walidacyjny. Walidacja krzyżowa. Miary jakości.
- Pakiet Scikit-Learn. Implementacja regresji liniowej i regresji logistycznej z wykorzystaniem gotowych modułów. Implementacja wybranych metod ewaluacji.
- Sposoby reprezentacji danych. Implementacja algorytmów regresji dla danych różnych typów, w tym dla danych nieliczbowych, oraz dla danych niepełnych.
- Nadmierne i niedostateczne dopasowanie. Obciążenie i wariancja. Ilustracja problemu nadmiernego dopasowania na przykładzie regresji wielomianowej. Metody regularyzacji.
- Nadmierne i niedostateczne dopasowanie w praktyce. Implementacja metod zapobiegających nadmiernemu dopasowaniu.
- Stochastic Gradient Descent. Przegląd metod optymalizacji.
- Porównanie różnych metod optymalizacji na przykładach.
- Uczenie nienadzorowane. Algorytm k średnich. Algorytm analizy głównych składowych.
- Implementacja metod uczenia nienadzorowanego na przykładzie algorytmu k średnich.
- Przegląd metod uczenia nadzorowanego. Naiwny klasyfikator bayesowski. Algorytm k najbliższych sąsiadów. Drzewa decyzyjne. Maszyny wektorów nośnych.
- Wprowadzenie do sztucznych sieci neuronowych. Prosty perceptron. Funkcje aktywacji. Wielowarstwowe sieci neuronowe.
- Propagacja wsteczna. Uczenie wielowarstwowych sieci neuronowych.
- Implementacja sieci neuronowych.
- Splotowe sieci neuronowe – idea, przegląd najpopularniejszych architektur, przegląd zastosowań. Czym jest uczenie głębokie?
- Rekurencyjne sieci neuronowe – idea, przegląd najpopularniejszych architektur, przegląd zastosowań. Modele typu encoder-decoder. Neuronowe tłumaczenie maszynowe. Autoencoder. Word embeddings.
- Wprowadzenie do uczenia przez wzmacnianie.
- Indywidualny projekt programistyczny – implementacja wybranych metod uczenia maszynowego.

Nazwa zajęć**: Wstęp do informatyki**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna metody reprezentacji informacji i danych za pomocą liczb binarnych. Potrafi zapisywać liczby zmiennoprzecinkowe i wykonywać na nich operacje arytmetyczne.
- Zna podstawowe pojęcia teorii informacji, potrafi obliczyć ilość informacji w komunikacie, potrafi wyznaczyć entropię źródła informacji.
- Zna pojęcia średniej długości słowa kodowego, redundancji, różnych rodzajów kodów. Potrafi wyznaczać kod zwarty metodą Huffmana.
- Zna techniki kompresji danych stratnej i bezstratnej, potrafi obliczyć stopień kompresji danych.
- Zna metody przechowywania danych na poziomie logicznym oraz sprzętowym, zna podstawowe elementy formatu XML, potrafi zaprojektować prosty schemat XML.
- Zna metody walidacji plików XML, potrafi zapisać schemat XML przy użyciu języka DTD.
- Zna podstawowe techniki przetwarzania niskopoziomowego, pojęcie bramki logicznej, sumatora. Potrafi zapisać schematy funkcji logicznych przy użyciu bramek logicznych.
- Zna budowę lokalnych, średnich oraz rozległych sieci komputerowych, zna podstawowe urządzenia sieciowe, rozumie ogólne zasady działania sieci Internet, potrafi diagnozować urządzenia sieciowe.
- Zna podstawowe protokoły sieciowe, potrafi opisać algorytm komunikacji w protokołach sieciowych.
- Zna zasady formatowania tekstu przy użyciu LaTeX, potrafi przygotować w pełni sformatowaną książkę LaTeX oraz jej wersje w MathJax, beamer oraz Jupyter.
- Zna ideę repozytorium danych, zna komendy pakietu Git na poziomie operacji na gałęziach, potrafi stworzyć i zarządzać repozytorium danych Git.

Treści programowe dla zajęć:
- System liczbowe i reprezentacje liczb w komputerze ze szczególnym naciskiem na szczegóły IEEE 754 single/double.
- Teoretyczne wyprowadzenie pojęcia entropii informacyjnej i pochodnych dla niej pojęć teorii informacji, podstawowe idee kompresji danych.
- Model warstwowy sieci komputerowej i związane z nim protokoły, algorytmika podpisu elektronicznego, konfiguracja serwera Apache z SSL i zabezpieczenia stron WWW przez protokół HTTP/SSL, generowanie certyfikatów.

- Teoretyczne podstawy budowy komputera przez konstrukcję kilku bramek na bazie NAND, układy sekwencyjne.
- GIT jako przykład systemu kontroli wersji do poziomu tworzenia i zarzadzania gałęziami.
- Edycja tekstów z nauk ścisłych: do druku - Latex na poziomie pełnej książki (okładka, struktura (część/rozdział/sekcja itd., dodatki), spis treści i inne spisy, grafika TikZ/PGF, indeksy, bibliografia (bez Bibtexa)); dla przeglądarki – MathJax; jako prezentacja – Beamer; jako materiał interaktywny – Jupyter.
- XML jako język prezentacji danych na poziomie XPath oraz XSLT.

Nazwa zajęć**: Wstęp do rachunku prawdopodobieństwa**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna podstawowe definicje, własności i twierdzenia dotyczące przestrzeni probabilistycznych. Rozumie pojęcie niezależności zdarzeń oraz prawdopodobieństwa w przestrzeniach kartezjańskich.
- Rozumie pojęcie prawdopodobieństwa warunkowego, zna i potrafi stosować wzory z nim związane. Zna wybrane schematy doświadczalne.
- Zna pojęcia zmiennej losowej jednowymiarowej i wektora losowego ora ich rozkładów prawdopodobieństwa. Potrafi posługiwać się pojęciem niezależności zmiennych losowych i wektorów losowych.
- Zna pojęcia momentów zmiennych losowych, w szczególności wartości oczekiwanej, wariancji i kowariancji. Potrafi te wielkości interpretować obliczać dla konkretnych rozkładów prawdopodobieństwa i stosować w praktyce.
- Zna metody szacowania prawdopodobieństwa z wykorzystaniem znanych nierówności probabilistycznych oraz Centralnego Twierdzenia
- Granicznego. Zna klasyczne przybliżenia rozkładu dwumianowego i umie stosować je w praktyce.
- Potrafi podać zastosowania rachunku prawdopodobieństwa w informatyce i/lub statystyce.
- Potrafi wykorzystać wybrane języki programowania do wyznaczania i szacowania prawdopodobieństwa oraz momentów zmiennych losowych także symulowania rozkładów prawdopodobieństwa.

Treści programowe dla zajęć:
- Eksperymenty losowe. Klasyczna definicja prawdopodobieństwa. Aksjomatyczna definicja przestrzeni probabilistycznej. Podstawowe własności prawdopodobieństwa.
- Prawdopodobieństwo geometryczne. Niezależność zdarzeń. Prawdopodobieństwa na iloczynach kartezjańskich
- Prawdopodobieństwo warunkowe. Wzór na prawdopodobieństwo całkowite, twierdzenie Bayesa i wzór łańcuchowy.
- Jednowymiarowe zmienne losowe. Dystrybuanta i jej własności. Podstawowe rozkłady dyskretne (dwumianowy, Poissona, geometryczny Pascala, hipergeometryczny).
- Zmienne losowe wielowymiarowe. Rozkłady brzegowe. Niezależność zmiennych losowych. Wartość oczekiwana, wariancja, kowariancja. Funkcje zmiennej losowej.
- Zmienne losowe ciągłe jedno- i wielowymiarowe. Funkcja gęstości i dystrybuanta rozkładu ciągłego. Podstawowe rozkłady ciągłe (jednostajny, normalny, wykładniczy).
- Odchylenie od średniej. Nierówność Markowa, nierówność Czebyszewa.
- Prawo wielkich liczb. Centralne Twierdzenie Graniczne.
- Wybrane zastosowania rachunku prawdopodobieństwa w informatyce statystyce.

Nazwa zajęć**: Zarządzanie produktem**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Rozumie pojęcie „produktu” i „zarządzania produktem”, charakter pracy product managera i zna jego podstawowe techniki i narzędzia.

Treści programowe dla zajęć:
- Produkt, zarządzanie produktem – wprowadzenie, definicje i historia, wspaniałe produkty.
- Cykl życia produktu.
- Product Manager a Product Owner: produkt w świecie metodologii zwinnych (Scrum, skalowany Scrum np. SAFE).
- Praca z zespołem, role w zespole we współczesnej firmie informatycznej, tzw. “empowered product teams”.
- Business Cases oraz elementy marketingu.
- Użytkownicy produktu: segmentacja i persony.
- Wymagania: pochodzenia, analiza, zarządzanie.
- Mapy drogowe (roadmaps).
- Prototypowanie: cele, rodzaje prototypów.
- Dostępność (accessibility).

- Analiza zachowań użytkowników w skali.
- Zarządzanie produktami różnego typu (backend, frontend, produkty nieinformatyczne).
- Narzędzia i systemy Product Managera.
- Etyka w świecie produktu.
- Rekrutacja w świecie produktu.

Nazwa zajęć**: Szkolenie BHP**
Po zakończeniu zajęć i potwierdzeniu osiągnięcia efektów uczenia się student:
- Zna zasady udzielania pierwszej pomocy przedmedycznej.
- Zna zasady dotyczące bezpieczeństwa i higieny pracy.
- Zna treść wybranych zagadnień z zakresu prawa pracy.
- Zna zasady ochrony przeciwpożarowej.

Treści programowe dla zajęć:
- Pierwsza pomoc.
- Bezpieczeństwo i higiena pracy.
- Elementy prawa pracy.
- Ochrona przeciwpożarowa.