Oczywiście! Poniżej znajdziesz **bardziej szczegółowe** notatki z wybranych zagadnień przygotowane w prosty sposób i w formacie Markdown. Ze względu na ogromny zakres materiału, podzieliłem notatki na sekcje. Możesz poprosić o dalsze rozwinięcie konkretnych tematów według potrzeb.

ZAGADNIENIA DO EGZAMINU INŻYNIERSKIEGO 2024/2025

Wstęp do matematyki

1. Działania na liczbach i zbiorach.

2. Rozwiązywanie równań i nierówności.

3. Podstawowe własności funkcji.

4. Ciągi liczbowe (ciąg arytmetyczny i ciąg geometryczny, ciągi rekurencyjne).

5. Algebra wektorów (iloczyn skalarny, wektorowy, działania na wektorach).

Podstawy programowania

1. Typy danych.

2. Zmienne, operatory, wyrażenia, instrukcje.

3. Instrukcja złożona, instrukcje sterujące.

4. Funkcje i struktura programu.

5. Tablice jednowymiarowe oraz dwuwymiarowe.

6. Parametry funkcji typu wskaźnikowego. Tablice i wskaźniki.

7. Struktury.

Wprowadzenie do systemów operacyjnych

1. Rodzaje i mechanizmy działania systemu operacyjnego.

2. Zadania poinstalacyjne w systemach operacyjnych.

3. Zarządzanie użytkownikami, uprawnienia.

4. Instalacja i konfiguracja oprogramowania.

Teoretyczne podstawy informatyki

1. Znaczenie, działanie oraz najczęściej występujące typy kanałów informacyjnych.

2. Typy kodów oraz wielkości charakteryzujące kody.

3. Źródła informacji: bezpamięciowe i Markowa.

4. Ilość informacji, entropia, twierdzenie Shannona o kodowaniu kanałów bezszumowych.

5. Liczbowe systemy pozycyjne: system binarny i system U2.

6. Działanie maszyny Turinga.

Oprogramowanie użytkowe

1. Sekcje w dokumentach wielostronicowych.

2. Makropolecenia jako automatyzacja pracy użytkownika.

3. Adresowanie w arkuszach kalkulacyjnych: względne, bezwzględne i mieszane.

4. Narzędzia do analizy i przetwarzania danych w arkuszach kalkulacyjnych.

5. Projektowanie relacyjnej bazy danych (tabele, kwerendy, formularze, raporty, makra).

Algorytmy i struktury danych

1. Złożoność obliczeniowa. Notacja asymptotyczna (notacja O, notacja Θ).

2. Algorytmy wyszukiwania i sortowania.

3. Listy z dowiązaniami.

4. Stosy, kolejki, kopiec binarny, kolejki priorytetowe.

5. Drzewa (drzewa binarne, drzewa BST, drzewa AVL, B-drzewa).

6. Grafy i podstawowe algorytmy grafowe.

Matematyka 1

1. Tautologie i kontrtautologie klasycznego rachunku zdań oraz klasycznego rachunku kwantyfikatorów.

2. Funkcje zdaniowe.

3. Indukcja matematyczna i jej zastosowania.

4. Relacje. Relacje równoważności oraz klasy abstrakcji. Relacje porządku.

5. Funkcja jako relacja - własności funkcji (np. injekcja, surjekcja, bijekcja, składanie funkcji).

6. Kombinatoryka i rachunek prawdopodobieństwa. Zmienna losowa.

Organizacja i architektura komputerów

1. Reprezentacja informacji w komputerze - arytmetyka i logika.

2. Układy kombinacyjne i sekwencyjne (analiza działania i projektowanie).

3. Jednostka centralna i pamięć.

4. Organizacja równoległa.

Języki hipertekstowe i tworzenie stron WWW

1. Typy selektorów w CSS.

2. Sposoby pozycjonowanie elementów w dokumencie HTML.

3. Podstawowe konstrukcje w języku JS: zmienne, operatory, instrukcje warunkowe, pętle.

4. Semantyka w HTML5.

5. Metoda POST i GET przekazywania danych w formularzach.

6. CSS – dziedziczenie własności.

7. Drzewo dokumentu HTML i selektory odwołujące się do jego poszczególnych elementów (selektor brata, potomka, dziecka itp.).

Programowanie proceduralne

1. Wskaźniki i operacje na wskaźnikach w języku C.

2. Wskaźnik podwójny, tablice wskaźników w języku C.

3. Dynamiczna alokacja pamięci w języku C. Wycieki pamięci.

4. Łańcuchy znakowe w języku C.

5. Rekurencja w języku C.

6. Pliki w języku C. Pliki tekstowe i binarne. Dostęp sekwencyjny i swobodny do pliku.

7. Wskaźniki do funkcji.

8. Struktura programu w języku C. Klasy zmiennych. Czas trwania zmiennych, zasięg i łączność.

Matematyka 2

1. Rachunek różniczkowy i całkowy funkcji jednej zmiennej rzeczywistej.

2. Zastosowania pochodnych i całek.

3. Liczby zespolone.

4. Macierze i wyznaczniki. Zastosowania do rozwiązania układu równań liniowych.

Programowanie obiektowe

1. Pojęcie klasy i obiektu na przykładzie języka C++.

2. Mechanizm dziedziczenia klas i polimorfizm w C++. Koncepcja hermetyzacji (enkapsulacji) w programowaniu obiektowym.

3. Klasy abstrakcyjne w języku C++.

4. Wskaźniki i referencje oraz dynamiczne zarządzanie pamięcią w języku C++.

5. Przeciążenie operatorów, przeciążenie funkcji oraz koncepcja funkcji zaprzyjaźnionych w języku C++.

6. Poziomy dostępu do składników klasy oraz funkcje i zmienne statyczne w języku C++.

7. Konstruktory i destruktory w C++.

8. Obsługa wyjątków i biblioteka STL w języku C++.

Systemy operacyjne

1. Struktury systemów operacyjnych.

2. Zarządzanie procesami.

3. Synchronizacja procesów.

4. Zarządzanie zasobami pamięci operacyjnej.

5. Zarządzanie pamięcią masową.

6. System plików.

7. Bezpieczeństwo i ochrona w systemach operacyjnych.

Przetwarzanie dokumentów XML i zaawansowane techniki WWW

1. XML, elementy, atrybuty.

2. Definicja struktury dokumentu za pomocą DTD (ang. document type definition).

3. Renderowanie XML za pomocą CSS, transformacje XSLT.

4. XPath i DOM.

Sieci komputerowe

1. Model OSI oraz jego warstwy funkcjonalne.

2. Model TCP/IP i różnice w stosunku do OSI.

3. Protokół komunikacyjny i jego funkcje w poszczególnych warstwach.

4. Adresacja IP: IPv4 i IPv6 (struktura, klasy adresów, CIDR).

5. Podstawowe protokoły warstwy transportowej (TCP, UDP) i różnice między nimi.

6. Protokoły aplikacyjne (HTTP, FTP, DNS, SMTP) i ich funkcje w komunikacji.

7. Topologie sieci (gwiazda, magistrala, pierścień, siatka) i ich zastosowania.

8. Urządzenia sieciowe (routery, switche, huby, access pointy) i ich funkcje.

Języki skryptowe

1. Wbudowane typy, klasy i struktury danych oraz ich różne odmiany. Zakresy zmiennych (funkcja, moduł, domknięcie, itd.).

2. Sterowanie przebiegiem programu i różne aspekty wykorzystania pętli.

3. Funkcje i podstawowy mechanizm obsługi parametrów, słowa kluczowe. Funkcje rekurencyjne w języku Python.

4. Wyrażenia listowe i generatorowe. Filtrowanie i transformacja danych.

5. Wykorzystanie funkcji anonimowych (np. w sortowaniu danych).

6. Obsługa plików. Serializacja obiektów. Tworzenie modułów i pakietów.

7. Dekoratory funkcji i domknięcia, funkcje jako obiekty pierwszej klasy.

8. Podstawowa obsługa wyjątków (konstrukcja try-except).

9. Dopasowanie wzorców z użyciem match/case.

Przetwarzanie obrazów cyfrowych

1. Filtracja obrazu (filtry górno i dolnoprzepustowe).

2. Binaryzacja obrazu.

3. Segmentacja obrazu.

4. Metody wyszukania różnić w obrazach.

Projektowanie aplikacji internetowych

1. Zgodność z WCAG2.1.

2. Wymagania dotyczące ochrony danych osobowych w projektowaniu aplikacji.

Fizyczne podstawy działania urządzeń informatycznych

1. Zasada działania tranzystora w elektronice cyfrowej.

2. Wykorzystanie laserów w odczycie i zapisie danych.

3. Magnetyczne nośniki danych i ich mechanizmy zapisu.

4. Fale elektromagnetyczne w światłowodach i ich rola w transmisji danych.

5. Ekrany LCD i LED – zasady działania oraz wykorzystanie efektów optycznych w technologii wyświetlaczy.

Wstęp do programowania w języku Java

1. Podstawowe typy danych oraz instrukcje warunkowe.

2. Działanie różnych typów pętli w Java.

3. Zagadnienie tworzenia klas, obiektów oraz korzystanie z konstruktorów.

4. Mechanizm dziedziczenia i kompozycji.

5. Interfejsy, wyrażenia lambda i klasy wewnętrzne.

6. Obsługa wyjątków, asercji.

7. Zagadnienia związane z programowaniem generycznym.

8. Kolekcje w Java.

Relacyjne bazy danych

1. Podstawy modelu relacyjnego, diagramy ERD (Entity-Relationship Diagram), normalizacja danych.

2. Podstawy SQL, funkcje agregujące SQL.

3. Łączenie tabel.

4. Projektowanie baz danych dla aplikacji webowych.

5. Bezpieczeństwo baz danych w aplikacjach webowych.

6. Wzorzec projektowy MVC.

Inżynieria oprogramowania

1. Procesy wytwarzania oprogramowania i ich modele.

2. Inżynieria wymagań dla systemów oprogramowania.

3. Metody analizy i modelowania oprogramowania.

4. Projektowanie architektoniczne systemów oprogramowania.

5. Wzorce architektoniczne i ich zastosowania, ogólne architektury aplikacji.

6. Walidacja i testowanie oprogramowania.

Systemy czasu rzeczywistego

1. Rodzaje, klasy i przykłady systemów czasu rzeczywistego (SCR).

2. Opis jakości i skuteczności działania SCR za pomocą funkcji zysku.

3. Priorytety statyczne i dynamiczne w SCR.

4. Algorytmy szeregowania zadań z wywłaszczeniem stosowane w SCR.

5. Zjawisko inwersji priorytetów i sposoby zapobiegania w SCR.

Sztuczna inteligencja

1. Test Turinga, komunikacja Człowiek-Komputer (ELIZA).

2. Algorytmy wyszukiwania i rozwiązywania problemów w praktyce.

3. Przetwarzanie języka naturalnego i metody przygotowania tekstu do NLP.

4. Uczenie maszynowe - zadania klasyfikacyjne, drzewa decyzyjne, las losowy.

5. Uczenie maszynowe - zadania regresyjne, regresja Liniowa.

Metody statystyczne w projektach inżynierskich

1. Miary/statystyki opisowe.

2. Testowanie hipotez.

3. Analiza danych (analiza szeregów czasowych, analiza wariacji (ANOVA), analiza regresji).

4. Oprogramowanie statystyczne.

5. Eksploracja danych i interpretacja wyników.

Systemy wbudowane

1. Budowa i zasada działania mikrokontrolera.

2. Systemy wbudowane sterowane mikrokontrolerami.

3. Programowanie mikrokontrolerów.

4. Pamięci w systemach wbudowanych.

5. Układy peryferyjne i magistrale transmisji danych.

Programowanie robotów

1. Podstawy robotyki: definicja robota i robotyki; rodzaje robotów (mobilne, stacjonarne, przemysłowe, humanoidalne, serwisowe); podstawowe elementy budowy robota.

2. Algorytmy unikania przeszkód i poruszania się w przestrzeni.

3. Algorytmy planowania ścieżki (A*, Dijkstra, BFS, DFS).

4. Rodzaje sensorów: sensory kontaktowe, ultradźwiękowe, LIDAR, kamery, GPS, IMU.

5. Sensory do wykrywania przeszkód i nawigacji oraz zasady ich działania (np. sonar, czujniki podczerwieni).

Programowanie sieciowe

1. Wielowątkowość w Javie (Runnable i callable).

2. Aplikacje klient – serwer.

3. Wybrane mikro usługi i sposoby ich działania.

Podstawy programowania współbieżnego

1. Abstrakcja współbieżności.

2. Klasyczne problemy synchronizacji procesów.

3. Narzędzia synchronizacji procesów.

4. Procesy ciężkie i procesy lekkie w systemie Linux.

5. Komunikacja międzyprocesowa (IPC).

Programowanie aplikacji internetowych

1. Elementy niezbędne do wdrożenia i uruchomienia aplikacji stworzonej w technologii Java Enterprise Edition (Java EE).

2. Środowiska programistyczne wspierające programowanie w Java EE.

3. Serwlet: budowa i zasada działania.

4. Sposoby tworzenia i obsługi sesji na przykładzie aplikacji internetowej Java EE.

5. Baza danych w aplikacji Java EE: porównanie Java Database Connectivity (JDBC) i Java Persistance API (JPA).

6. Podstawy funkcjonowania i możliwości stron tworzonych przy użyciu frameworków: Java Server Pages (JSP), Java Server Faces (JSF).

Administracja serwerami WWW

1. Algorytm nawiązywania połączenia HTTPS.

2. Generowanie certyfikatów.

3. Główne typy serwerów proxy.

4. Funkcje wirtualnych hostów na serwerze HTTP/HTTPS.

E-biznes

1. Bezpieczeństwo w e-biznesie: zagrożenia w e-biznesie (phishing, malware, naruszenia danych). Certyfikaty SSL i ich znaczenie. RODO i ochrona danych osobowych w e-biznesie.

2. Definicja i charakterystyka e-biznesu oraz e-commerce. Modele biznesowe w Internecie. Relacje biznesowe (np. B2B, B2C, C2C, G2C).

Projektowanie wizualne i tworzenie interfejsów

1. Techniki UI/UX związane z użytkownikiem.

2. Prototypowanie interfejsów użytkownika.

3. Ewaluacja interfejsów użytkownika.

Kryptografia

1. Podstawowe elementy kryptografii.

2. Schematy algorytmów szyfrowania symetrycznego.

3. Szyfrowanie i schematy podpisów cyfrowych.

4. Kryptograficzne funkcje Skrótu.

5. Schematy identyfikacji i uwierzytelniania.

6. Kryptografia na krzywych eliptycznych.

Podstawy modelowania i symulacji

1. Etapy tworzenia modelu matematycznego. Kategorie modelu.

2. Układy I rzędu (np. układ RC).

3. Układy II rzędu (np. ruch w polu centralnym, drgania sprzężone).

4. Modelowanie liczebności populacji i modele epidemii.

Testowanie oprogramowania

1. Typy i poziomy testowania.

2. Metody testowania.

3. Projektowanie testów.

4. Automatyzacja testowania.

5. Zarządzanie testowaniem.

6. Dokumentacja testowa.

7. Narzędzia i środowiska testowe.

Wzorce projektowe

1. Ogólne określenie wzorca wg Christophera Alexandra.

2. Ogólne cechy wzorca projektowego.

3. Wybrane wzorce projektowe: behawioralne, kreacyjne, strukturalne oraz ich zastosowania.

Inżynieria i analiza danych

1. Podstawowe operacje na tablicach numpy oraz praca z obiektami DataFrame i Series w pandas.

2. Podstawy statystyki z wykorzystaniem Pythona (pomiar tendencji centralnej i dyspersji, współczynnik korelacji i kowariancja, centralne twierdzenie graniczne, testy statystyczne i rozkłady prawdopodobieństwa).

3. Techniki czyszczenia i przygotowywania danych do analizy.

4. Techniki odczytywania i zapisywania danych (m.in. tekstowych, binarnych, interfejs sieciowy).

5. Wizualizacja danych w języku Python – rodzaje wykresów i wizualizacja zależności.

6. Analiza regresyjna (regresja liniowa, wielomianowa, logistyczna), ocena skuteczności modelu (m.in. R^2, błąd średniokwadratowy) z językiem Python.

7. Klasyfikacja (naiwny klasyfikator Bayesa, drzewa decyzyjne, maszyny wektorów nośnych (SVM)), podział danych na zestaw uczący i testowy oraz ocena jakości modelu z językiem Python.

8. Redukcja wymiarowości za pomocą PCA oraz analiza skupień (K-means, DBSCAN) i ocena jakości.

9. Przetwarzanie i analiza dużych zbiorów danych w języku Python.

Fizyczne podstawy działania urządzeń informatycznych

1. Zasada działania tranzystora w elektronice cyfrowej.

2. Wykorzystanie laserów w odczycie i zapisie danych.

3. Magnetyczne nośniki danych i ich mechanizmy zapisu.

4. Fale elektromagnetyczne w światłowodach i ich rola w transmisji danych.

5. Ekrany LCD i LED – zasady działania oraz wykorzystanie efektów optycznych w technologii wyświetlaczy.

Wstęp do programowania w języku Java

1. Podstawowe typy danych oraz instrukcje warunkowe.

2. Działanie różnych typów pętli w Java.

3. Zagadnienie tworzenia klas, obiektów oraz korzystanie z konstruktorów.

4. Mechanizm dziedziczenia i kompozycji.

5. Interfejsy, wyrażenia lambda i klasy wewnętrzne.

6. Obsługa wyjątków, asercji.

7. Zagadnienia związane z programowaniem generycznym.

8. Kolekcje w Java.

Relacyjne bazy danych

1. Podstawy modelu relacyjnego, diagramy ERD (Entity-Relationship Diagram), normalizacja danych.

2. Podstawy SQL, funkcje agregujące SQL.

3. Łączenie tabel.

4. Projektowanie baz danych dla aplikacji webowych.

5. Bezpieczeństwo baz danych w aplikacjach webowych.

6. Wzorzec projektowy MVC.

Inżynieria oprogramowania

1. Procesy wytwarzania oprogramowania i ich modele.

2. Inżynieria wymagań dla systemów oprogramowania.

3. Metody analizy i modelowania oprogramowania.

4. Projektowanie architektoniczne systemów oprogramowania.

5. Wzorce architektoniczne i ich zastosowania, ogólne architektury aplikacji.

6. Walidacja i testowanie oprogramowania.

Systemy czasu rzeczywistego

1. Rodzaje, klasy i przykłady systemów czasu rzeczywistego (SCR).
2. Opis jakości i skuteczności działania SCR za pomocą funkcji zysku.
3. Priorytety statyczne i dynamiczne w SCR.
4. Algorytmy szeregowania zadań z wywłaszczeniem stosowane w SCR.
5. Zjawisko inwersji priorytetów i sposoby zapobiegania w SCR.

Sztuczna inteligencja

1. Test Turinga, komunikacja Człowiek-Komputer (ELIZA).
2. Algorytmy wyszukiwania i rozwiązywania problemów w praktyce.
3. Przetwarzanie języka naturalnego i metody przygotowania tekstu do NLP.
4. Uczenie maszynowe - zadania klasyfikacyjne, drzewa decyzyjne, las losowy.
5. Uczenie maszynowe - zadania regresyjne, regresja Liniowa.

Metody statystyczne w projektach inżynierskich

1. Miary/statystyki opisowe.
2. Testowanie hipotez.
3. Analiza danych (analiza szeregów czasowych, analiza wariacji (ANOVA), analiza regresji).
4. Oprogramowanie statystyczne.
5. Eksploracja danych i interpretacja wyników.

Systemy wbudowane

1. Budowa i zasada działania mikrokontrolera.
2. Systemy wbudowane sterowane mikrokontrolerami.
3. Programowanie mikrokontrolerów.
4. Pamięci w systemach wbudowanych.
5. Układy peryferyjne i magistrale transmisji danych.

Programowanie robotów

1. Podstawy robotyki: definicja robota i robotyki; rodzaje robotów (mobilne, stacjonarne, przemysłowe, humanoidalne, serwisowe); podstawowe elementy budowy robota.

2. Algorytmy unikania przeszkód i poruszania się w przestrzeni.

3. Algorytmy planowania ścieżki (A*, Dijkstra, BFS, DFS).

4. Rodzaje sensorów: sensory kontaktowe, ultradźwiękowe, LIDAR, kamery, GPS, IMU.

5. Sensory do wykrywania przeszkód i nawigacji oraz zasady ich działania (np. sonar, czujniki podczerwieni).

Programowanie sieciowe

1. Wielowątkowość w Javie (Runnable i callable).

2. Aplikacje klient – serwer.

3. Wybrane mikro usługi i sposoby ich działania.

Podstawy programowania współbieżnego

1. Abstrakcja współbieżności.

2. Klasyczne problemy synchronizacji procesów.

3. Narzędzia synchronizacji procesów.

4. Procesy ciężkie i procesy lekkie w systemie Linux.

5. Komunikacja międzyprocesowa (IPC).

Programowanie aplikacji internetowych

1. Elementy niezbędne do wdrożenia i uruchomienia aplikacji stworzonej w technologii Java Enterprise Edition (Java EE).

2. Środowiska programistyczne wspierające programowanie w Java EE.

3. Serwlet: budowa i zasada działania.

4. Sposoby tworzenia i obsługi sesji na przykładzie aplikacji internetowej Java EE.

5. Baza danych w aplikacji Java EE: porównanie Java Database Connectivity (JDBC) i Java Persistance API (JPA).

6. Podstawy funkcjonowania i możliwości stron tworzonych przy użyciu frameworków: Java Server Pages (JSP), Java Server Faces (JSF).

Administracja serwerami WWW

1. Algorytm nawiązywania połączenia HTTPS.

2. Generowanie certyfikatów.

3. Główne typy serwerów proxy.

4. Funkcje wirtualnych hostów na serwerze HTTP/HTTPS.

E-biznes

1. Bezpieczeństwo w e-biznesie: zagrożenia w e-biznesie (phishing, malware, naruszenia danych). Certyfikaty SSL i ich znaczenie. RODO i ochrona danych osobowych w e-biznesie.

2. Definicja i charakterystyka e-biznesu oraz e-commerce. Modele biznesowe w Internecie. Relacje biznesowe (np. B2B, B2C, C2C, G2C).

Projektowanie wizualne i tworzenie interfejsów

1. Techniki UI/UX związane z użytkownikiem.

2. Prototypowanie interfejsów użytkownika.

3. Ewaluacja interfejsów użytkownika.

Kryptografia

1. Podstawowe elementy kryptografii.

2. Schematy algorytmów szyfrowania symetrycznego.

3. Szyfrowanie i schematy podpisów cyfrowych.

4. Kryptograficzne funkcje Skrótu.

5. Schematy identyfikacji i uwierzytelniania.

6. Kryptografia na krzywych eliptycznych.

Podstawy modelowania i symulacji

1. Etapy tworzenia modelu matematycznego. Kategorie modelu.

2. Układy I rzędu (np. układ RC).

3. Układy II rzędu (np. ruch w polu centralnym, drgania sprzężone).

4. Modelowanie liczebności populacji i modele epidemii.

Testowanie oprogramowania

1. Typy i poziomy testowania.

2. Metody testowania.

3. Projektowanie testów.

4. Automatyzacja testowania.

5. Zarządzanie testowaniem.

6. Dokumentacja testowa.

7. Narzędzia i środowiska testowe.

Wzorce projektowe

1. Ogólne określenie wzorca wg Christophera Alexandra.

2. Ogólne cechy wzorca projektowego.

3. Wybrane wzorce projektowe: behawioralne, kreacyjne, strukturalne oraz ich zastosowania.

Inżynieria i analiza danych

1. Podstawowe operacje na tablicach numpy oraz praca z obiektami DataFrame i Series w pandas.

2. Podstawy statystyki z wykorzystaniem Pythona (pomiar tendencji centralnej i dyspersji, współczynnik korelacji i kowariancja, centralne twierdzenie graniczne, testy statystyczne i rozkłady prawdopodobieństwa).

3. Techniki czyszczenia i przygotowywania danych do analizy.

4. Techniki odczytywania i zapisywania danych (m.in. tekstowych, binarnych, interfejs sieciowy).

5. Wizualizacja danych w języku Python – rodzaje wykresów i wizualizacja zależności.

6. Analiza regresyjna (regresja liniowa, wielomianowa, logistyczna), ocena skuteczności modelu (m.in. R^2, błąd średniokwadratowy) z językiem Python.

7. Klasyfikacja (naiwny klasyfikator Bayesa, drzewa decyzyjne, maszyny wektorów nośnych (SVM)), podział danych na zestaw uczący i testowy oraz ocena jakości modelu z językiem Python.

8. Redukcja wymiarowości za pomocą PCA oraz analiza skupień (K-means, DBSCAN) i ocena jakości.

9. Przetwarzanie i analiza dużych zbiorów danych w języku Python.

I. Wstęp do Matematyki

1. Działania na Liczbach i Zbiorach

Liczby:

- **Naturalne ($\mathbb{N}$)**: Liczby używane do liczenia (1, 2, 3, ...).
- **Całkowite ($\mathbb{Z}$)**: Liczby naturalne, ich negacje i zero (..., -3, -2, -1, 0, 1, 2, 3, ...).
- **Wymierne ($\mathbb{Q}$)**: Liczby, które można przedstawić jako ułamek $\frac{a}{b}$, gdzie (a) i (b) są całkowite, a $b \neq 0$.
- **Niewymierne**: Liczby, które nie mogą być zapisane jako ułamek $\frac{a}{b}$ (np. $\sqrt{2}$, π).
- **Rzeczywiste ($\mathbb{R}$)**: Wszystkie liczby wymierne i niewymierne.

Operacje na liczbach:

- **Dodawanie (+)**: Łączenie dwóch liczb w jedną większą.
- **Odejmowanie (-)**: Znajdowanie różnicy między dwiema liczbami.
- **Mnożenie ($\times$)**: Powtarzane dodawanie.
- **Dzielenie ($\div$)**: Rozdzielanie liczby na równe części.
- **Potęgowanie**: Mnożenie liczby przez siebie określoną ilość razy (a^b).
- **Pierwiastkowanie**: Odwrotność potęgowania ($\sqrt{a}$).

Zbiory:

- **Definicja zbioru**: Kolekcja unikalnych elementów (np. $A = \{1, 2, 3\}$).
- **Elementy**: Pojedyncze obiekty w zbiorze.
- **Podzbiory**: Zbiory, które zawierają tylko elementy z większego zbioru (np. $\{1, 2\}$ jest podzbiorem $\{1, 2, 3\}$).

Operacje na zbiorach:

- **Suma ($A \cup B$)**: Wszystkie elementy, które są w (A) lub w (B) lub w obu.
- **Iloczyn ($A \cap B$)**: Elementy wspólne dla zbiorów (A) i (B).
- **Różnica ($A - B$)**: Elementy, które są w (A), ale nie ma ich w (B).
- **Dopełnienie**: Wszystkie elementy nie należące do danego zbioru.

Zbiory specjalne:

- **Zbiór pusty ($\emptyset$)**: Zbiór bez żadnych elementów.
- **Zbiór skończony**: Zbiór z ograniczoną liczbą elementów.
- **Zbiór nieskończony**: Zbiór z nieskończoną liczbą elementów.

---

2. Rozwiązywanie Równań i Nierówności

Równania:

- **Równania liniowe**: Postać $ax + b = 0$.
  - **Przykład**: $2x + 3 = 0 \Rightarrow x = -\frac{3}{2}$.
- **Równania kwadratowe**: Postać $ax^2 + bx + c = 0$.
  - **Rozwiązanie**: Wzór kwadratowy $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.
- **Równania wykładnicze**: Zmienna jest wykładnikiem potęgi (np. $2^x = 8$).
  - **Rozwiązanie**: $x = 3$.
- **Równania logarytmiczne**: Zmienna jest argumentem logarytmu (np. $\log_2(x) = 3$).
  - **Rozwiązanie**: $x = 8$.

Metody rozwiązywania równań:

1. **Podstawianie**: Zamiana jednej zmiennej na wyrażenie z innej zmiennej.
2. **Eliminacja**: Usunięcie jednej zmiennej z równań.
3. **Faktoryzacja**: Rozkładanie na czynniki wielomianów.

Nierówności:

- **Nierówności liniowe**: Postać (ax + b > 0).
  - **Przykład**: (2x - 4 > 0 \Rightarrow x > 2).
- **Nierówności kwadratowe**: Postać (ax^2 + bx + c < 0).
  - **Rozwiązanie**: Znalezienie przedziałów, w których nierówność jest prawdziwa poprzez analizę miejsc zerowych.

Metody rozwiązywania nierówności:

1. **Przekształcenia algebraiczne**: Podobne do rozwiązywania równań.
2. **Wykresy**: Rysowanie wykresów funkcji i sprawdzanie, gdzie funkcja spełnia nierówność.
3. **Przedziały**: Sprawdzanie wartości funkcji w różnych przedziałach.

---

3. Podstawowe Własności Funkcji

Definicja funkcji:

- **Funkcja** to przyporządkowanie każdemu elementowi z **dziedziny** dokładnie jeden element w **przeciwdziedzinie**.
  - **Notacja**: (f: A \rightarrow B), gdzie (A) to dziedzina, a (B) to przeciwdziedzina.

Rodzaje funkcji:

- **Liniowe**: (f(x) = ax + b)
  - **Przykład**: (f(x) = 2x + 3)
- **Kwadratowe**: (f(x) = ax^2 + bx + c)
  - **Przykład**: (f(x) = x^2 - 4x + 4)
- **Wykładnicze**: (f(x) = a \cdot b^x)
  - **Przykład**: (f(x) = 2^x)
- **Logarytmiczne**: (f(x) = a \cdot \log_b(x) + c)
  - **Przykład**: (f(x) = \log_2(x))
- **Trygonometryczne**: (f(x) = a \cdot \sin(x) + b), (f(x) = a \cdot \cos(x) + b), itd.
  - **Przykład**: (f(x) = \sin(x))

Własności funkcji:

- **Ciągłość**: Funkcja jest ciągła, jeśli jej wykres nie ma przerw.
  - **Przykład**: Funkcja liniowa (f(x) = 2x + 3) jest ciągła.
- **Różniczkowalność**: Funkcja jest różniczkowalna w punkcie, jeśli ma tam pochodną.
  - **Przykład**: (f(x) = x^2) jest różniczkowalna wszędzie.
- **Monotoniczność**:

    - **Rosnąca**: (f(x_1) < f(x_2)) dla (x_1 < x_2).
    - **Malejąca**: (f(x_1) > f(x_2)) dla (x_1 < x_2).
- **Symetria**:
    - **Parzysta**: (f(-x) = f(x))
        - **Przykład**: (f(x) = x^2)
    - **Nieparzysta**: (f(-x) = -f(x))
        - **Przykład**: (f(x) = x^3)

---

4. Ciągi Liczbowe

Definicja ciągu:

- **Ciąg** to uporządkowana sekwencja liczb, gdzie każda liczba nazywana jest **wyrazem** ciągu.

Ciąg Arytmetyczny:

- **Definicja**: Ciąg, w którym każdy wyraz jest sumą poprzedniego wyrazu i stałej różnicy (d).
    - **Wzór n-tego wyrazu**: (a_n = a_1 + (n-1)d)
    - **Suma n wyrazów**: (S_n = \frac{n}{2} \cdot (2a_1 + (n-1)d))
- **Przykład**:
    - (a_1 = 2), (d = 3)
    - Ciąg: 2, 5, 8, 11, 14, ...

Ciąg Geometryczny:

- **Definicja**: Ciąg, w którym każdy wyraz jest iloczynem poprzedniego wyrazu i stałego ilorazu (q).
    - **Wzór n-tego wyrazu**: (a_n = a_1 \cdot q^{n-1})
    - **Suma n wyrazów**: (S_n = a_1 \cdot \frac{1 - q^n}{1 - q}) dla (q \neq 1)
- **Przykład**:
    - (a_1 = 3), (q = 2)
    - Ciąg: 3, 6, 12, 24, 48, ...

Ciągi Rekurencyjne:

- **Definicja**: Ciąg, w którym każdy wyraz jest określany na podstawie poprzednich wyrazów.
- **Przykład**: Ciąg Fibonacciego
    - (a_1 = 0), (a_2 = 1)
    - (a_n = a_{n-1} + a_{n-2}) dla (n > 2)
    - Ciąg: 0, 1, 1, 2, 3, 5, 8, 13, ...

Metody rozwiązywania ciągów rekurencyjnych:

1. **Metoda charakterystyczna**: Tworzenie równania charakterystycznego i znajdowanie jego pierwiastków.

2. **Metoda iteracyjna**: Obliczanie wyrazów ciągu krok po kroku.

---

5. Algebra Wektorów

Definicja wektora:

- **Wektor** to obiekt mający zarówno **wielkość** (długość) jak i **kierunek**.
- **Przykład**: Wektor $\vec{v} = (3, 4)$ w przestrzeni 2D.

Iloczyn Skalarny ($\vec{u} \cdot \vec{v}$):

- **Definicja**: Suma iloczynów odpowiadających sobie współrzędnych wektorów.
  - $\vec{u} \cdot \vec{v} = u_1v_1 + u_2v_2 + \ldots + u_nv_n$
- **Własności**:
  - **Symetryczność**: $\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u}$
  - **Liniowość**: $\vec{u} \cdot (\vec{v} + \vec{w}) = \vec{u} \cdot \vec{v} + \vec{u} \cdot \vec{w}$
- **Zastosowania**:
  - Obliczanie kąta między wektorami: $\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| |\vec{v}|}$
  - Obliczanie długości wektora: $|\vec{u}| = \sqrt{\vec{u} \cdot \vec{u}}$

Iloczyn Wektorowy ($\vec{u} \times \vec{v}$):

- **Definicja**: Operacja wykonywana tylko w przestrzeni 3D, wynik jest wektorem prostopadłym do płaszczyzny wyznaczonej przez $\vec{u}$ i $\vec{v}$.
  - $\vec{u} \times \vec{v} = (u_2v_3 - u_3v_2, u_3v_1 - u_1v_3, u_1v_2 - u_2v_1)$
- **Własności**:
  - **Antysymetryczność**: $\vec{u} \times \vec{v} = -(\vec{v} \times \vec{u})$
  - **Brak komutatywności**: $\vec{u} \times \vec{v} \neq \vec{v} \times \vec{u}$
- **Zastosowania**:
  - Obliczanie momentu siły.
  - Znajdowanie prostopadłych wektorów.

Działania na Wektorach:

- **Dodawanie**:
  - Dodajemy odpowiadające sobie współrzędne.
  - $\vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n)$
- **Odejmowanie**:
  - Odejmujemy odpowiadające sobie współrzędne.

- $\vec{u} - \vec{v} = (u_1 - v_1, u_2 - v_2, \ldots, u_n - v_n)$

- **Mnożenie przez skalar**:
  - Mnożymy każdą współrzędną przez liczbę.
  - $k \cdot \vec{u} = (k \cdot u_1, k \cdot u_2, \ldots, k \cdot u_n)$

- **Przekształcenia liniowe**:
  - Mnożenie macierzą przez wektor, zmieniając jego kierunek i/lub długość.

- **Kombinacje liniowe**:
  - Dodawanie wektorów pomnożonych przez skalar.
  - $\vec{w} = a \cdot \vec{u} + b \cdot \vec{v}$

II. Podstawy Programowania

1. Typy Danych

Podstawowe Typy:

- `int`: Liczby całkowite (np. 1, -5, 42).
- `float`: Liczby zmiennoprzecinkowe (np. 3.14, -0.001).
- `char`: Pojedynczy znak (np. 'a', 'Z', '3').
- `bool`: Wartości logiczne (true, false).

Typy Złożone:

- **Tablice**: Kolekcje elementów tego samego typu (np. `int[] numbers = {1, 2, 3};`).
- **Struktury**: Grupy różnych typów danych (np. struktura `Student` zawierająca `int id`, `string name`).
- **Wskaźniki**: Zmienne przechowujące adresy innych zmiennych.

Typy Abstrakcyjne:

- **Klasy**: Szablony dla obiektów, zawierające atrybuty i metody.
- **Obiekty**: Instancje klas.

---

2. Zmienne, Operatory, Wyrażenia, Instrukcje

Zmienne:

- **Deklaracja**: Określenie typu i nazwy zmiennej (np. `int age;`).
- **Inicjalizacja**: Nadanie zmiennej początkowej wartości (np. `int age = 25;`).
- **Zakres**: Obszar, w którym zmienna jest dostępna (lokalny vs globalny).

Operatory:

- **Arytmetyczne**: `+`, `-`, `*`, `/`, `%`.

- **Logiczne**: && (AND), || (OR), ! (NOT).
- **Relacyjne**: ==, !=, <, >, <=, >=.
- **Bitowe**: &, |, ^, ~, <<, >>.

Wyrażenia:

- **Składnia**: Kombinacja zmiennych, operatorów i wartości, które obliczają wartość.
  - **Przykład**: `x + y * 2`

Instrukcje:

- **Przypisanie**: `x = 5;`
- **Wywołanie funkcji**: `print("Hello");`
- **Blok kodu**: Zbiór instrukcji w nawiasach klamrowych `{ ... }`

---

3. Instrukcja Złożona, Instrukcje Sterujące

Instrukcje Złożone:

- **Bloki kodu**: Grupowanie wielu instrukcji razem.
  - **Przykład**:

```
if (x > 0) {
    printf("x jest dodatnie");
    x = x - 1;
}
```

Instrukcje Sterujące:

- **Warunkowe**:
  - `if`: Wykonuje blok kodu, jeśli warunek jest prawdziwy.

```
if (x > 0) {
    // kod
}
```

  - `else`: Wykonuje blok kodu, jeśli poprzedni warunek jest fałszywy.

```
if (x > 0) {
    // kod
} else {
    // inny kod
}
```

- `switch`: Wybiera blok kodu do wykonania na podstawie wartości zmiennej.

```
switch (day) {
    case 1:
        printf("Poniedziałek");
        break;
    case 2:
        printf("Wtorek");
        break;
    // ...
}
```

- **Pętle**:
  - `for`: Powtarza blok kodu określoną liczbę razy.

```
for (int i = 0; i < 10; i++) {
    printf("%d\n", i);
}
```

  - `while`: Powtarza blok kodu, dopóki warunek jest prawdziwy.

```
while (x > 0) {
    printf("%d\n", x);
    x--;
}
```

  - `do-while`: Podobne do `while`, ale gwarantuje wykonanie bloku kodu przynajmniej raz.

```
do {
    printf("x jest: %d\n", x);
    x--;
} while (x > 0);
```

- **Skoki**:
  - `break`: Przerywa działanie pętli lub instrukcji `switch`.
  - `continue`: Pomija bieżącą iterację pętli i przechodzi do następnej.
  - `return`: Zwraca wartość z funkcji i kończy jej działanie.

---

4. Funkcje i Struktura Programu

Funkcje:

- **Definicja**: Blok kodu wykonujący określone zadanie.
  - **Przykład w C**:

```c
int dodaj(int a, int b) {
    return a + b;
}
```

- **Deklaracja**: Informuje kompilator o istnieniu funkcji.
  - **Przykład**:

```c
int dodaj(int a, int b);
```

- **Wywołanie**: Użycie funkcji w kodzie.
  - **Przykład**:

```c
int wynik = dodaj(3, 4);
```

Parametry:

- **Przekazywanie przez wartość**: Kopiowanie wartości zmiennej do funkcji.
  - **Przykład**:

```c
void zmien(int x) {
    x = 10;
}
```

- **Przekazywanie przez referencję**: Przekazywanie adresu zmiennej, umożliwiające modyfikację oryginalnej zmiennej.
  - **Przykład**:

```c
void zmien(int *x) {
    *x = 10;
}
```

Struktura Programu:

- **Funkcja `main`**: Punkt wejścia programu.
  - **Przykład**:

```c
int main() {
    // Kod programu
    return 0;
}
```

- **Funkcje pomocnicze**: Dodatkowe funkcje wykonywane przez `main` lub inne funkcje.
- **Modularność**: Podział programu na mniejsze, niezależne części (funkcje), co ułatwia zarządzanie kodem.

---

5. Tablice Jednowymiarowe oraz Dwuwymiarowe

Tablice Jednowymiarowe:

- **Deklaracja**:
  - **C**: `int numbers[5];`
  - **Python**: `numbers = [1, 2, 3, 4, 5]`
- **Indeksowanie**:
  - Rozpoczyna się od 0.
  - **Przykład**: `numbers[0]` odnosi się do pierwszego elementu.
- **Operacje**:
  - Dodawanie, usuwanie, modyfikowanie elementów.

Tablice Dwuwymiarowe:

- **Deklaracja**:
  - **C**: `int matrix[3][3];`
  - **Python**: `matrix = [[1,2,3], [4,5,6], [7,8,9]]`
- **Dostęp do elementów**:
  - **C**: `matrix[0][1]` odnosi się do elementu w pierwszym wierszu, drugiej kolumnie.
  - **Python**: `matrix[0][1]`
- **Operacje**:
  - Iteracja po wierszach i kolumnach.
  - Modyfikowanie elementów.

---

6. Parametry Funkcji Typu Wskaźnikowego. Tablice i Wskaźniki

Wskaźniki:

- **Definicja**: Zmienne przechowujące adresy pamięci innych zmiennych.

- **Przykład w C**:

```
int a = 10;
int *p = &a; // p przechowuje adres zmiennej a
```

- **Operacje**:
  - **Dereferencja**: Pobieranie wartości pod adresem.

```
int value = *p; // value = 10
```

  - **Arytmetyka wskaźników**: Zmiana adresu wskaźnika.

```
p = p + 1;
```

Tablice a Wskaźniki:

- **Związek**: Nazwa tablicy jest wskaźnikiem do jej pierwszego elementu.
  - **Przykład**:

```
int numbers[5] = {1, 2, 3, 4, 5};
int *p = numbers; // p wskazuje na numbers[0]
```

- **Przekazywanie do funkcji**:
  - Tablice są przekazywane jako wskaźniki, co pozwala na modyfikację oryginalnych danych.

```
void printFirst(int *arr) {
    printf("%d\n", arr[0]);
}
```

---

7. Struktury

Definicja:

- **Struktura** to typ danych, który pozwala na grupowanie różnych typów danych pod jedną nazwą.
- **Przykład w C**:

```

```

```
struct Student {
    int id;
    char name[50];
    float gpa;
};
```

Zastosowania:

- **Organizacja danych**: Ułatwia przechowywanie powiązanych informacji.
    - **Przykład**: Reprezentacja studenta z identyfikatorem, imieniem i średnią ocen.
- **Tworzenie złożonych typów**: Pozwala na tworzenie bardziej zaawansowanych struktur danych.

III. Algorytmy i Struktury Danych

1. Złożoność Obliczeniowa. Notacja Asymptotyczna (O, Θ)

Złożoność Czasowa:

- Określa, jak czas wykonania algorytmu rośnie wraz z wielkością danych wejściowych.

Notacja Asymptotyczna:

- **Big O (O)**: Górna granica złożoności. Opisuje najgorszy przypadek.
    - **Przykład**: Bubble Sort ma złożoność (O(n^2)).
- **Big Θ (Θ)**: Ścisła granica złożoności. Opisuje zarówno górną, jak i dolną granicę.
    - **Przykład**: Merge Sort ma złożoność (Θ(n \log n)).

---

2. Algorytmy Wyszukiwania i Sortowania

Wyszukiwanie:

- **Liniowe**:
    - Przeszukuje elementy jeden po drugim.
    - **Złożoność**: (O(n))
    - **Przykład**: Szukanie liczby 5 w tablicy `[1, 3, 5, 7, 9]`.
- **Binarne**:
    - Działa na posortowanej tablicy, dzieląc ją na pół w każdym kroku.
    - **Złożoność**: (O(\log n))
    - **Przykład**: Szukanie liczby 5 w tablicy `[1, 3, 5, 7, 9]`.

Sortowanie:

- **Quick Sort**:
  - Dzieli tablicę na mniejsze części wokół pivotu i sortuje je rekursywnie.
  - **Złożoność**: Średnio (O(n \log n)), najgorszy (O(n^2)).
- **Merge Sort**:
  - Dzieli tablicę na pół, sortuje każdą połowę i scala je.
  - **Złożoność**: (O(n \log n))
- **Bubble Sort**:
  - Porównuje sąsiednie elementy i zamienia je, jeśli są w złej kolejności.
  - **Złożoność**: (O(n^2))
- **Insertion Sort**:
  - Wstawia każdy element na właściwe miejsce w posortowanej części tablicy.
  - **Złożoność**: Średnio (O(n^2)), najlepiej (O(n))
- **Selection Sort**:
  - Znajduje najmniejszy element i zamienia go z pierwszym, następnie powtarza dla reszty tablicy.
  - **Złożoność**: (O(n^2))

---

3. Listy z Dowiązaniami

Listy Jednokierunkowe:

- **Definicja**: Węzły połączone wskaźnikami do następnego węzła.
- **Struktura w C**:

```c
struct Node {
    int data;
    struct Node* next;
};
```

- **Operacje**:
  - **Dodawanie**: Na początku, na końcu, w środku.
  - **Usuwanie**: Z początku, z końca, z określonej pozycji.
  - **Przeszukiwanie**: Przeglądanie listy w poszukiwaniu elementu.

Listy Dwukierunkowe:

- **Definicja**: Węzły połączone wskaźnikami do następnego i poprzedniego węzła.

- **Struktura w C**:

```c
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};
```

- **Operacje**:
    - **Dodawanie**: Na początku, na końcu, w środku.
    - **Usuwanie**: Z początku, z końca, z określonej pozycji.
    - **Przeszukiwanie**: Możliwość przeglądania listy w obu kierunkach.

---

4. Stosy, Kolejki, Kopiec Binarne, Kolejki Priorytetowe

Stos (LIFO - Last In, First Out):

- **Definicja**: Ostatni element dodany jest pierwszym usuwanym.
- **Operacje**:
    - `push`: Dodanie elementu na szczyt stosu.
    - `pop`: Usunięcie elementu ze szczytu stosu.
- **Przykład użycia**: Cofanie operacji (np. w edytorach tekstu).

Kolejka (FIFO - First In, First Out):

- **Definicja**: Pierwszy element dodany jest pierwszym usuwanym.
- **Operacje**:
    - `enqueue`: Dodanie elementu na końcu kolejki.
    - `dequeue`: Usunięcie elementu z początku kolejki.
- **Przykład użycia**: Kolejki w sklepach, drukarek.

Kopiec Binarne:

- **Definicja**: Struktura drzewkowa, w której każdy rodzic ma maksymalnie dwóch potomków, a wartość rodzica jest większa lub mniejsza od wartości potomków (w zależności od typu kopca).
- **Typy**:
    - **Min-Kopiec**: Rodzic ma mniejszą wartość niż jego potomkowie.
    - **Max-Kopiec**: Rodzic ma większą wartość niż jego potomkowie.
- **Operacje**:

- **Wstawianie**: Dodanie elementu i utrzymanie właściwości kopca.
- **Usuwanie**: Usunięcie korzenia i przywrócenie właściwości kopca.

Kolejki Priorytetowe:

- **Definicja**: Kolejka, w której każdy element ma przypisany priorytet i element z najwyższym priorytetem jest usuwany jako pierwszy.
- **Implementacja**: Najczęściej za pomocą kopca binarnego.
- **Przykład użycia**: Zarządzanie zadaniami w systemie operacyjnym.

---

5. Drzewa (Binarne, BST, AVL, B-Drzewa)

Drzewa Binarne:

- **Definicja**: Struktura danych składająca się z węzłów, gdzie każdy węzeł ma maksymalnie dwóch potomków (lewy i prawy).
- **Przykład**:

```
  A
 / \
B   C
```

Binary Search Tree (BST):

- **Definicja**: Drzewo binarne, w którym lewy potomek ma mniejszą wartość niż rodzic, a prawy potomek ma większą wartość.
- **Zalety**: Szybkie wyszukiwanie, dodawanie i usuwanie elementów.
- **Przykład**:

```
    5
   / \
  3   7
 / \   \
2   4   8
```

Drzewa AVL:

- **Definicja**: Samobalansujące BST, w którym różnica wysokości lewego i prawego poddrzewa każdego węzła nie przekracza 1.
- **Zalety**: Zapewnia (O(\log n)) złożoność operacji.

- **Operacje Balansowania**: Rotacje (prawo, lewo, podwójne).

B-Drzewa:

- **Definicja**: Drzewa o wielu dzieciach, używane głównie w bazach danych i systemach plików.
- **Zalety**: Efektywne przechowywanie i szybki dostęp do dużych zbiorów danych.
- **Charakterystyka**: Wszystkie liście znajdują się na tym samym poziomie, węzły mogą mieć wiele kluczy.

---

6. Grafy i Podstawowe Algorytmy Grafowe

Grafy:

- **Definicja**: Struktura danych składająca się z **wierzchołków** (nódów) i **krawędzi** (łączących wierzchołki).
- **Rodzaje**:
  - **Skierowane**: Krawędzie mają kierunek.
  - **Nieskierowane**: Krawędzie nie mają kierunku.
- **Przykład Grafu Nieskierowanego**:

```
A -- B
|    |
C -- D
```

Podstawowe Algorytmy Grafowe:

- **BFS (Breadth-First Search)**:
  - Przeszukiwanie grafu poziomami.
  - **Zastosowanie**: Znalezienie najkrótszej ścieżki w grafie nieskierowanym.
- **DFS (Depth-First Search)**:
  - Przeszukiwanie grafu zagłębiając się w głąb.
  - **Zastosowanie**: Wykrywanie cykli, topologiczne sortowanie.
- **Dijkstra**:
  - Algorytm znajdowania najkrótszej ścieżki w grafie z nieujemnymi wagami krawędzi.
  - **Zastosowanie**: Nawigacja GPS.
- **Kruskal**:
  - Algorytm znajdowania minimalnego drzewa rozpinającego.
  - **Zastosowanie**: Optymalizacja sieci, takich jak sieci drogowe czy telekomunikacyjne.
- **Prim**:

- Podobny do Kruskala, ale zaczyna od jednego wierzchołka i rozszerza drzewo.
- **Zastosowanie**: Podobne do Kruskala.

IV. Systemy Operacyjne

1. Rodzaje i Mechanizmy Działania Systemu Operacyjnego

Rodzaje Systemów Operacyjnych:

- **Jednoukładowe (Monolithic)**:
  - Wszystkie funkcje systemu operacyjnego są w jednym dużym bloku kodu.
  - **Przykład**: Linux.
- **Wieloukładowe (Microkernel)**:
  - Podstawowe funkcje w jądrze, reszta działa w przestrzeni użytkownika.
  - **Przykład**: MINIX, QNX.
- **Sieciowe**:
  - Zarządzają zasobami komputerów w sieci.
  - **Przykład**: Novell NetWare.
- **Czasu Rzeczywistego (RTOS)**:
  - Zapewniają szybką i przewidywalną reakcję na zdarzenia.
  - **Przykład**: VxWorks.

Mechanizmy Działania:

- **Zarządzanie Procesami**: Tworzenie, planowanie i kończenie procesów.
- **Zarządzanie Pamięcią**: Alokacja pamięci dla procesów, pamięć wirtualna.
- **System Plików**: Organizacja, przechowywanie i dostęp do plików.
- **Sterowniki Urządzeń**: Komunikacja z sprzętem komputerowym.

---

2. Zadania Instalacyjne w Systemach Operacyjnych

Instalacja Systemu Operacyjnego:

1. **Partycjonowanie Dysku**:
   - Podział dysku na mniejsze sekcje zwane partycjami.
   - **Przykład**: Partycja systemowa, partycja danych.
2. **Wybór Systemu Plików**:
   - Określenie struktury do przechowywania plików.
   - **Przykłady**: NTFS, ext4, FAT32.

3. **Konfiguracja Sprzętu**:
    - Ustawienia BIOS/UEFI, sterowniki urządzeń.

Konfiguracja Początkowa:

- **Ustawienia Sieciowe**: Konfiguracja połączeń internetowych, adresów IP.
- **Tworzenie Użytkowników**: Definiowanie kont użytkowników.
- **Bezpieczeństwo**: Ustawienie haseł, polityk bezpieczeństwa.

---

3. Zarządzanie Użytkownikami, Uprawnienia

Konta Użytkowników:

- **Tworzenie**: Dodawanie nowych użytkowników do systemu.
    - **Przykład w Linux**:

```
sudo adduser nazwisko
```

- **Usuwanie**: Usuwanie istniejących użytkowników.
    - **Przykład w Linux**:

```
sudo deluser nazwisko
```

Uprawnienia:

- **Rodzaje Uprawnień**:
    - **Odczyt (Read)**: Możliwość przeglądania pliku.
    - **Zapis (Write)**: Możliwość modyfikacji pliku.
    - **Wykonanie (Execute)**: Możliwość uruchomienia pliku jako programu.
- **Role Użytkowników**:
    - **Administrator**: Pełny dostęp do systemu.
    - **Standardowy**: Ograniczony dostęp do zasobów.

Mechanizmy Kontroli Dostępu:

- **ACL (Access Control List)**: Lista określająca, które użytkownicy mają jakie uprawnienia do pliku.
- **Grupy**: Umożliwiają zarządzanie uprawnieniami dla wielu użytkowników jednocześnie.
- **Polityki Bezpieczeństwa**: Reguły definiujące, co użytkownicy mogą robić w systemie.

---

4. Instalacja i Konfiguracja Oprogramowania

Metody Instalacji:

- **Pakiety**:
  - Używanie menedżerów pakietów (np. `apt`, `yum` w Linux) do instalacji.
  - **Przykład w Ubuntu**:

```
sudo apt install nazwa_pakietu
```

- **Instalatory**:
  - Programy graficzne lub tekstowe do instalacji oprogramowania (np. instalator Windows).
- **Kompilacja ze Źródeł**:
  - Pobieranie kodu źródłowego i kompilowanie go na własnym komputerze.
  - **Przykład**:

```
./configure
make
sudo make install
```

Zarządzanie Zależnościami:

- **Menedżery Pakietów**: Automatyczne pobieranie i instalacja zależności wymaganych przez oprogramowanie.
  - **Przykład**: `npm` dla Node.js, `pip` dla Pythona.

Aktualizacje:

- **Systemowe**: Aktualizacje samego systemu operacyjnego.
  - **Przykład w Linux**:

```
sudo apt update
sudo apt upgrade
```

- **Aplikacyjne**: Aktualizacje zainstalowanych aplikacji.
  - **Przykład**: Aktualizacja przeglądarki internetowej.

---

5. Zarządzanie Procesami

Proces:

- **Definicja**: Program w trakcie wykonywania, zawierający kod programu i jego aktualny stan.

- **Stany Procesu**:
    - **Nowy (New)**: Proces jest tworzony.
    - **Gotowy (Ready)**: Proces czeka na przydzielenie czasu procesora.
    - **Wykonywany (Running)**: Proces jest aktywnie wykonywany.
    - **Zakończony (Terminated)**: Proces zakończył działanie.

Planowanie Procesów:

- **Algorytmy Planowania**:
    - **FIFO (First-In, First-Out)**: Pierwszy przybył, pierwszy obsłużony.
    - **Round Robin**: Każdy proces otrzymuje określony kwant czasu.
    - **SJF (Shortest Job First)**: Procesy z najkrótszym czasem wykonania są obsługiwane jako pierwsze.
    - **Priority**: Procesy z wyższym priorytetem są obsługiwane przed niższymi.

Tworzenie i Zakończenie Procesów:

- **Tworzenie**:
    - `fork()`: Tworzy nowy proces jako kopię bieżącego.
    - `exec()`: Zastępuje kod bieżącego procesu innym programem.
- **Zakończenie**:
    - Proces kończy działanie samodzielnie lub przez inny proces.
    - `wait()`: Proces macierzysty oczekuje na zakończenie procesu potomnego.

---

6. Synchronizacja Procesów

Problemy Synchronizacji:

- **Krytyczne Sekcje**: Części kodu, które muszą być wykonywane przez jeden proces w danym czasie.
- **Wyścigi (Race Conditions)**: Sytuacje, w których wynik zależy od niekontrolowanego kolejności wykonywania procesów.

Mechanizmy Synchronizacji:

- **Semafory**:
    - Liczniki używane do kontrolowania dostępu do zasobów.
    - **Typy**: Binarny (0 lub 1), Liczbowy.
- **Muteksy (Mutexes)**:
    - Specjalny rodzaj semafora do ochrony pojedynczych zasobów.
    - Zapewniają, że tylko jeden proces może uzyskać dostęp do zasobu w danym czasie.
- **Monitory**:

    - Abstrakcje programistyczne łączące mutex i warunki.
    - Umożliwiają bardziej zaawansowane zarządzanie synchronizacją.

Przykłady:

- **Problem Producenta i Konsumenta**: Synchronizacja między procesem produkującym dane a procesem je konsumującym.

---

7. Zarządzanie Zasobami Pamięci Operacyjnej

Pamięć Wirtualna:

- **Definicja**: Technologia pozwalająca na używanie większej ilości pamięci niż fizycznie dostępna, poprzez korzystanie z dysku jako rozszerzenia RAM.
- **Mechanizmy**:
    - **Stronicowanie (Paging)**: Dzielenie pamięci na małe bloki (strony) i przechowywanie ich w pamięci fizycznej lub na dysku.
    - **Segmentacja (Segmentation)**: Podział pamięci na logiczne segmenty (np. kod, dane).

Alokacja Pamięci:

- **Dynamiczna Alokacja**: Przydzielanie pamięci w czasie wykonywania programu.
    - **Funkcje w C**: `malloc()`, `calloc()`, `realloc()`, `free()`.
- **Paging i Swapping**:
    - **Paging**: Przenoszenie stron między pamięcią RAM a dyskiem.
    - **Swapping**: Przenoszenie całych procesów między pamięcią RAM a dyskiem.

Mechanizmy Zarządzania Pamięcią:

- **Zarządzanie Stronami**: Tablice stron mapujące adresy wirtualne na fizyczne.
- **TLB (Translation Lookaside Buffer)**: Bufor szybkiego dostępu do mapowania stron.

---

8. Zarządzanie Pamięcią Masową

Systemy Plików:

- **FAT (File Allocation Table)**: Prosty system plików używany w starszych systemach.
- **NTFS (New Technology File System)**: Nowocześniejszy system plików używany w Windows.
- **ext4**: Powszechnie używany system plików w Linux.

Operacje na Plikach:

- **Tworzenie**: Inicjowanie nowego pliku.
- **Usuwanie**: Usuwanie istniejącego pliku.

- **Modyfikacja**: Zmiana zawartości pliku.

Zarządzanie Dyskiem:

- **Alokacja Przestrzeni**: Przydzielanie przestrzeni na dysku dla plików.
- **Defragmentacja**: Organizowanie fragmentów plików w celu optymalizacji dostępu.

---

9. System Plików

Struktura:

- **Katalogi (Foldery)**: Organizują pliki w hierarchię.
- **Metadane**: Informacje o pliku, takie jak rozmiar, data utworzenia, uprawnienia.
- **Uprawnienia**: Kontrola dostępu do plików i katalogów.

Operacje:

- **CRUD (Create, Read, Update, Delete)**:
  - **Create**: Tworzenie plików/katalogów.
  - **Read**: Odczytywanie zawartości.
  - **Update**: Modyfikowanie zawartości.
  - **Delete**: Usuwanie plików/katalogów.

Zarządzanie Dostępem:

- **ACL (Access Control List)**: Lista kontrolna określająca, które użytkownicy mają jakie uprawnienia.
- **Prawa Użytkowników**: Określenie, kto może czytać, pisać, wykonywać pliki.

---

10. Bezpieczeństwo i Ochrona w Systemach Operacyjnych

Mechanizmy Ochrony:

- **Uwierzytelnianie**: Proces potwierdzania tożsamości użytkownika (np. hasło, token).
- **Autoryzacja**: Określanie, do jakich zasobów użytkownik ma dostęp.
- **Szyfrowanie**: Przekształcanie danych w formę nieczytelną bez klucza.

Zarządzanie Uprawnieniami:

- **Role**: Definiowanie różnych poziomów dostępu dla różnych grup użytkowników.
- **Grupy**: Organizowanie użytkowników w grupy z określonymi uprawnieniami.

Ochrona Danych:

- **Kopie Zapasowe**: Tworzenie kopii danych na wypadek utraty.
- **Szyfrowanie Dysków**: Ochrona całych dysków poprzez szyfrowanie danych.

IV. Matematyka 1

1. Tautologie i Kontrtautologie w Rachunku Zdań i Kwantyfikatorów

Tautologie:

- **Definicja**: Formuły logiczne, które są zawsze prawdziwe niezależnie od wartości zmiennych.
  - **Przykład**: (p \lor \neg p) (Prawo wyłączonego środka).

Kontrtautologie:

- **Definicja**: Formuły logiczne, które są zawsze fałszywe niezależnie od wartości zmiennych.
  - **Przykład**: (p \land \neg p).

Rachunek Zdań:

- **Operatory Logiczne**:
  - **AND ((\land))**: Prawdziwe, gdy oba składniki są prawdziwe.
  - **OR ((\lor))**: Prawdziwe, gdy przynajmniej jeden składnik jest prawdziwy.
  - **NOT ((\neg))**: Odwraca wartość logiczną.
  - **IMPLIES ((\rightarrow))**: Fałszywe tylko wtedy, gdy pierwszy składnik jest prawdziwy, a drugi fałszywy.

Rachunek Kwantyfikatorów:

- **Uniwersalny ((\forall))**: "Dla każdego".
  - **Przykład**: (\forall x \in \mathbb{N}, x + 0 = x).
- **Egzystencjalny ((\exists))**: "Istnieje przynajmniej jeden".
  - **Przykład**: (\exists x \in \mathbb{N}, x^2 = 4).

---

2. Funkcje Zdaniowe

Definicja:

- **Funkcja zdaniowa** to funkcja logiczna, która przyjmuje wartości prawda/fałsz jako argumenty i zwraca wartość prawda/fałsz.

Przykłady:

- **Tautologie**: Zawsze prawdziwe (np. (p \lor \neg p)).
- **Sprzeczności**: Zawsze fałszywe (np. (p \land \neg p)).
- **Kontingencje**: Mogą być prawdziwe lub fałszywe w zależności od wartości zmiennych (np. (p \land q)).

---

3. Indukcja Matematyczna i Jej Zastosowania

Indukcja Matematyczna:

- **Zasady**:
    1. **Baza indukcyjna**: Dowodzenie twierdzenia dla pierwszego elementu (np. $n = 1$).
    2. **Krok indukcyjny**: Zakładanie, że twierdzenie jest prawdziwe dla n, i dowodzenie dla $n + 1$.

Przykład: Dowód sumy pierwszych n liczb naturalnych

- **Twierdzenie**: $1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$
- **Baza indukcyjna** ($n = 1$):
    - $1 = \frac{1(1+1)}{2} = 1$ – prawdziwe.
- **Krok indukcyjny**:
    - Zakładamy, że $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$.
    - Musimy pokazać, że $1 + 2 + \ldots + n + (n+1) = \frac{(n+1)(n+2)}{2}$.
    - Dodajemy $n + 1$ do obu stron: $\frac{n(n+1)}{2} + (n + 1) = \frac{n(n+1) + 2(n + 1)}{2} = \frac{(n + 2)(n + 1)}{2}$
    - Co potwierdza prawdziwość twierdzenia dla $n + 1$.

Zastosowania Indukcji Matematycznej:

- Dowodzenie własności ciągów.
- Dowodzenie formuł algebraicznych.
- Dowodzenie twierdzeń w teorii liczb.

---

4. Relacje

Definicja Relacji:

- **Relacja** to powiązanie między elementami jednego lub więcej zbiorów.
    - **Przykład**: Relacja "jest większy od" na zbiorze liczb naturalnych.

Relacje Równoważności:

- **Własności**:
    - **Refleksyjność**: Każdy element jest w relacji ze sobą (np. $a = a$).
    - **Symetryczność**: Jeśli a jest w relacji z b, to b jest w relacji z a.
    - **Przechodniość**: Jeśli a jest w relacji z b i b z c, to a jest w relacji z c.
- **Klasy Abstrakcji**:
    - Podziały zbioru na klasy równoważności, gdzie wszystkie elementy w klasie są ze sobą powiązane.

Relacje Porządku:

- **Definicja**: Relacja, która porządkuje elementy w sposób uporządkowany.

- **Rodzaje**:
    - **Częściowy Porządek**: Nie wszystkie pary elementów są porównywalne.
    - **Liniowy Porządek (Totalny)**: Każda para elementów jest porównywalna.
    - **Porządek Totalny**: Każdy element jest większy, mniejszy lub równy innemu.

---

5. Funkcja jako Relacja - Własności Funkcji

Funkcja jako Relacja:

- **Definicja**: Funkcja to specjalny przypadek relacji, gdzie każdy element dziedziny jest powiązany z dokładnie jednym elementem przeciwdziedziny.

Własności Funkcji:

- **Injekcja (Jednoznaczność)**:
    - Każdy element dziedziny mapuje się na unikalny element przeciwdziedziny.
    - $f: A \rightarrow B$ jest injekcją, jeśli $f(a_1) = f(a_2) \Rightarrow a_1 = a_2$.
- **Surjekcja (Pełność)**:
    - Każdy element przeciwdziedziny jest obrazem przynajmniej jednego elementu dziedziny.
    - $f: A \rightarrow B$ jest surjekcją, jeśli dla każdego $b \in B$ istnieje $a \in A$ taki, że $f(a) = b$.
- **Bijekcja**:
    - Funkcja jest zarówno injekcją, jak i surjekcją.
    - **Zastosowanie**: Umożliwia odwracalność funkcji (inwersja).
- **Składanie Funkcji**:
    - Łączenie dwóch funkcji w jedną.
    - Jeśli $f: A \rightarrow B$ i $g: B \rightarrow C$, to składanie $g \circ f: A \rightarrow C$ definiuje $(g \circ f)(a) = g(f(a))$.

---

6. Kombinatoryka i Rachunek Prawdopodobieństwa. Zmienna Losowa

Kombinatoryka:

- **Permutacje**:
    - Ułożenie wszystkich elementów w określonej kolejności.
    - **Wzór**: $P(n) = n!$
- **Kombinacje**:
    - Wybór elementów bez uwzględnienia kolejności.
    - **Wzór**: $C(n, k) = \frac{n!}{k!(n - k)!}$

- **Wariacje**:
    - Ułożenie części elementów w określonej kolejności.
    - **Wzór**: $V(n, k) = \frac{n!}{(n - k)!}$

Rachunek Prawdopodobieństwa:

- **Zasady Dodawania**:
    - $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- **Zasady Mnożenia**:
    - **Niezależne zdarzenia**: $P(A \cap B) = P(A) \cdot P(B)$
- **Prawdopodobieństwo Warunkowe**:
    - $P(A \mid B) = \frac{P(A \cap B)}{P(B)}$
- **Niezależność**:
    - Dwa zdarzenia A i B są niezależne, jeśli $P(A \cap B) = P(A) \cdot P(B)$.

Zmienna Losowa:

- **Definicja**: Funkcja przyporządkowująca zdarzeniom liczby rzeczywiste.
- **Rozkłady**:
    - **Dyskretne**: Przyjmuje określone, oddzielne wartości (np. liczba oczek na kostce).
    - **Ciągłe**: Przyjmuje dowolne wartości w przedziale (np. czas oczekiwania).
- **Wartość Oczekiwana ($E[X]$)**: Średnia wartość zmiennej losowej.
- **Wariancja ($Var(X)$)**: Miara rozproszenia wartości zmiennej losowej wokół średniej.

V. Matematyka 2

1. Rachunek Różniczkowy i Całkowy Funkcji Jednej Zmiennej Rzeczywistej

Rachunek Różniczkowy:

- **Pochodna**:
    - Mierzy, jak szybko zmienia się funkcja.
    - **Definicja**: $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$
- **Zastosowania**:
    - Znajdowanie ekstremów (maksimum, minimum).
    - Analiza nachylenia wykresu.

Rachunek Całkowy:

- **Całka Nierozpuszczalna (Indefinite Integral)**:
  - Odwrotność pochodnej.
  - **Definicja**: $\int f(x)\,dx = F(x) + C$, gdzie $F'(x) = f(x)$
- **Całka Ograniczona (Definite Integral)**:
  - Oblicza pole pod wykresem funkcji na danym przedziale.
  - **Definicja**: $\int_{a}^{b} f(x)\,dx$
- **Zastosowania**:
  - Obliczanie pól powierzchni.
  - Obliczanie objętości.

---

2. Zastosowania Pochodnych i Całek

Zastosowania Pochodnych:

- **Optymalizacja**: Znajdowanie punktów maksymalnych i minimalnych funkcji.
  - **Przykład**: Maksymalizacja zysku firmy.
- **Ruch**: Opis prędkości i przyspieszenia w fizyce.
  - **Przykład**: $v(t) = s'(t)$, $a(t) = v'(t)$.

Zastosowania Całek:

- **Obliczanie pól i objętości**:
  - **Przykład**: Pole pod krzywą prędkości daje przemieszczenie.
- **Całkowanie w ekonomii**:
  - **Przykład**: Obliczanie całkowitych kosztów lub przychodów.

---

3. Liczby Zespolone

Definicja:

- **Liczba Zespolona**: $z = a + bi$, gdzie a i b są liczbami rzeczywistymi, a i jest jednostką urojoną ($i^2 = -1$).
- **Części Liczby Zespolonej**:
  - **Część Rzeczywista**: a
  - **Część Urojona**: b

Działania na Liczbach Zespolonych:

- **Dodawanie**:
  - $(a + bi) + (c + di) = (a + c) + (b + d)i$
- **Mnożenie**:
  - $(a + bi) \times (c + di) = (ac - bd) + (ad + bc)i$
- **Moduł**:
  - $|z| = \sqrt{a^2 + b^2}$
- **Sprzężenie**:
  - $\overline{z} = a - bi$

Zastosowania:

- Analiza fal i sygnałów.
- Przeprowadzanie obliczeń w elektrotechnice.

---

4. Macierze i Wyznaczniki. Zastosowania do Rozwiązania Układu Równań Liniowych

Macierz:

- **Definicja**: Prostokątna tablica liczb ułożonych w wiersze i kolumny.
  - **Przykład**: $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ \end{pmatrix}$

Wyznacznik:

- **Definicja**: Liczba przypisana do kwadratowej macierzy, która określa jej właściwości.
- **Obliczanie Wyznacznika dla macierzy 2x2**: $det(A) = ad - bc$ Dla $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Zastosowania:

- **Rozwiązywanie Układów Równań Liniowych**:
  - **Metoda Cramera**: Używa wyznaczników do znalezienia rozwiązań układu równań.
- **Inwersja Macierzy**:
  - **Definicja**: Macierz odwrotna A^{-1} taka, że $A \times A^{-1} = I$ (macierz jednostkowa).

Przykład Rozwiązania Układu Równań za pomocą Metody Cramera:

Rozważmy układ: $\begin{cases} 2x + 3y = 5 \\ 4x + y = 6 \\ \end{cases}$

- **Macierz Główna (A)**: [A = \begin{pmatrix} 2 & 3 \ 4 & 1 \ \end{pmatrix}] (det(A) = 2 \times 1 - 3 \times 4 = 2 - 12 = -10)
- **Macierz dla x (A_x)**: [A_x = \begin{pmatrix} 5 & 3 \ 6 & 1 \ \end{pmatrix}] (det(A_x) = 5 \times 1 - 3 \times 6 = 5 - 18 = -13)
- **Macierz dla y (A_y)**: [A_y = \begin{pmatrix} 2 & 5 \ 4 & 6 \ \end{pmatrix}] (det(A_y) = 2 \times 6 - 5 \times 4 = 12 - 20 = -8)
- **Rozwiązania**: [x = \frac{det(A_x)}{det(A)} = \frac{-13}{-10} = 1.3] [y = \frac{det(A_y)}{det(A)} = \frac{-8}{-10} = 0.8]

VI. Organizacja i Architektura Komputerów

1. Reprezentacja Informacji w Komputerze - Arytmetyka i Logika

Systemy Liczbowe:

- **Binarny (Base-2)**:
  - Używa cyfr 0 i 1.
  - **Przykład**: 1010 (w dziesiętnym to 10).
- **Dziesiętny (Base-10)**:
  - Używa cyfr od 0 do 9.
  - Najbardziej znany system liczbowy.
- **Heksadecymalny (Base-16)**:
  - Używa cyfr od 0 do 9 oraz liter A-F.
  - **Przykład**: 1A3 (w dziesiętnym to 419).

Operacje Arytmetyczne w Systemie Binarnym:

- **Dodawanie**:

```
  1010
+  0111
-------
  10001
```

- **Odejmowanie**:

```
  1010
-  0111
-------
   0011
```

- **Mnożenie**:

```
  1010
x   11
-------
  1010
+1010
-------
  11110
```

- **Dzielenie**:

```
  1010 ÷ 10 = 101
  Reszta = 0
```

Logiczne Bramki:

- **AND (&&)**: Zwraca 1 tylko wtedy, gdy oba wejścia są 1.
- **OR (||)**: Zwraca 1, gdy przynajmniej jedno wejście jest 1.
- **NOT (!) **: Odwraca wartość wejścia (0 staje się 1, 1 staje się 0).
- **XOR (⊕)**: Zwraca 1, gdy dokładnie jedno z wejść jest 1.
- **NAND**: Negacja bramki AND.
- **NOR**: Negacja bramki OR.

---

2. Układy Kombinacyjne i Sekwencyjne

Układy Kombinacyjne:

- **Definicja**: Wyjścia zależą tylko od bieżących wejść.
- **Przykłady**:
  - **Adder**: Dodaje dwa bity i generuje sumę oraz przeniesienie.
  - **Multiplekser (MUX)**: Wybiera jeden z wielu wejść na podstawie sygnałów sterujących.
- **Cechy**:
  - Brak pamięci.
  - Reakcja natychmiastowa na zmiany wejść.

Układy Sekwencyjne:

- **Definicja**: Wyjścia zależą zarówno od bieżących wejść, jak i stanu poprzedniego.
- **Przykłady**:

- **Rejestr**: Przechowuje stan (bity) pomiędzy cyklami zegara.
- **Licznik**: Liczy impulsy zegara.
- **FSM (Finite State Machine)**: Maszyna stanów, która zmienia stan na podstawie wejść.

- **Cechy**:
  - Posiada pamięć.
  - Wymaga zegara do synchronizacji operacji.

---

3. Jednostka Centralna i Pamięć

CPU (Central Processing Unit):

- **Części CPU**:
  - **ALU (Arithmetic Logic Unit)**: Wykonuje operacje arytmetyczne i logiczne.
  - **Rejestry**: Szybka pamięć wewnętrzna do przechowywania danych tymczasowych.
  - **Jednostka Sterująca**: Zarządza przepływem danych w CPU.

Pamięć:

- **RAM (Random Access Memory)**:
  - Pamięć ulotna używana do przechowywania danych i kodu w trakcie wykonywania programu.
- **ROM (Read-Only Memory)**:
  - Pamięć trwała używana do przechowywania stałych danych i instrukcji rozruchowych.
- **Cache**:
  - Szybka pamięć pośrednia między RAM a CPU, zwiększa wydajność.
- **Pamięć Masowa**:
  - **HDD (Hard Disk Drive)**: Trwałe przechowywanie danych na talerzach magnetycznych.
  - **SSD (Solid State Drive)**: Trwałe przechowywanie danych na układach pamięci flash, szybsze niż HDD.

---

4. Organizacja Równoległa

Przetwarzanie Równoległe:

- **Definicja**: Wykonywanie wielu operacji jednocześnie.
- **Rodzaje**:
  - **Wielowątkowość**: Wykonywanie wielu wątków w ramach jednego procesu.
  - **Wieloprocesorowość**: Używanie wielu procesorów do wykonywania różnych zadań.

Modele Przetwarzania Równoległego:

- **SIMD (Single Instruction, Multiple Data)**:
    - Jedna instrukcja jest wykonywana na wielu danych jednocześnie.
- **MIMD (Multiple Instruction, Multiple Data)**:
    - Różne instrukcje są wykonywane na różnych danych jednocześnie.

Synchronizacja:

- **Bariery**: Punkty synchronizacji, gdzie wątki muszą się spotkać przed dalszym wykonywaniem.
- **Mutexy i Semafory**: Mechanizmy do kontroli dostępu do wspólnych zasobów.
