Notatki do egzaminu inżynierskiego – zebrane zagadnienia

Poniżej znajdziesz zagregowane **zagadnienia** i **najważniejsze informacje** pogrupowane według działów z bazy pytań. Materiał został opracowany w przyjaznej formie, tak aby wyjaśniać kluczowe pojęcia „po ludzku", z zachowaniem szczegółowości.

---

1. Projektowanie aplikacji internetowych (pyt. 1–12)

Timeout w aplikacjach webowych

- **Definicja**: Określa maksymalny czas, w którym pewna operacja (np. bezczynność użytkownika, czas ładowania komponentu) musi się zakończyć.
- **Ważny kontekst**:
  - *Bezpieczeństwo* (automatyczne wylogowanie po określonym czasie).
  - *Oszczędność zasobów* (niepotrzebne sesje nie utrzymują się w nieskończoność).
- **Przykłady**:
  - Czas bezczynności sesji (np. 20 minut).
  - Timeout przy wdrażaniu aplikacji (deployment).
  - Timeouty wątków/metod – ograniczenie, by nie „wisiały" bez końca.

HttpServlet (Java)

- **HttpServlet**: klasa abstrakcyjna z `javax.servlet.http`, po której dziedziczą konkretne serwlety.
- **Różnica**:
  - `Servlet` to **interfejs**.
  - `HttpServlet` to **klasa abstrakcyjna**, której możemy użyć jako bazy do własnych serwletów.
- **Zadanie serwletu**: obsługa żądań HTTP (GET, POST, itp.).
- **Odróżniaj**: obiekt `HttpServlet` (klasa serwletu) od obiektów `HttpServletRequest` (żądanie) i `HttpServletResponse` (odpowiedź).

Cykl życia serwletu

- **Kontener serwletów** (np. Tomcat) zarządza inicjalizacją (`init()`), obsługą żądań (`service()` → `doGet()`, `doPost()`) i usuwaniem (`destroy()`).
- Deskryptor wdrożenia (`web.xml`) jedynie konfiguruje, nie decyduje o uruchamianiu.
- Programista pisze serwlet, ale *cały cykl* kontroluje **kontener**.

Minimalne wymagania do uruchamiania aplikacji Java web

- **JDK (Java Development Kit)** – konieczny do kompilacji.
- Serwer aplikacji (np. **Tomcat** lub pełny **Jakarta EE**/Java EE**).
- **IDE** (np. Eclipse, IntelliJ) nie jest obowiązkowe, ale ułatwia pracę.

Język skryptowy vs. kompilowany

- **Java**: kompilowana do bajtkodu (nie jest skryptowa).
- **JavaScript, Python, PHP**: języki skryptowe.

Technologia Java do tworzenia interfejsu po stronie serwera

- **JSF (JavaServer Faces)** – komponenty UI generowane i zarządzane po stronie serwera (architektura MVC).
- EJB to warstwa logiki biznesowej, a **JavaFX / Swing** to biblioteki do aplikacji desktopowych.

ORM (Object-Relational Mapping) w Javie

- **JPA (Java Persistence API)**: standard do mapowania obiektowo-relacyjnego.
- **JDBC** to niskopoziomowa komunikacja z bazą (bez automatycznego mapowania).
- DI (Dependency Injection) i JTA to inne mechanizmy, nie ORM.

Serwlet – komponent działający w modelu żądanie-odpowiedź

- **Serwlet** to klasa Javy (działająca w kontenerze), rozszerzająca możliwości serwera HTTP.
- Nie mylić z apletami czy skryptletami w JSP.

Mechanizm utrzymywania sesji

- **Cookie** – standardowy sposób na identyfikację sesji (np. `JSESSIONID`).
- Inne: parametry URL, ale cookies są najbardziej popularne.

Bezpieczne przesyłanie danych z formularza

- Metoda **POST** w protokole HTTP (dane w *body*, nie w samym URL).
- Dodatkowo używa się **HTTPS**, żeby faktycznie były szyfrowane.

Filtry w Java EE

- Filtry mogą być przypisane do wielu zasobów i modyfikują lub przechwytują żądanie/odpowiedź.
- **Fałsz** byłoby stwierdzenie, że filtr można przypisać tylko do jednego zasobu.

Zakresy obiektów JavaBean w aplikacji web

- Najmniejszy to **page** (obowiązuje w obrębie jednej strony JSP).
- Kolejne to **request**, **session**, **application** (globalny).

---

2. Programowanie obiektowe 2 (Python) (pyt. 13–28)

`__init__()` w Pythonie

- Konstruktor obiektu – wywoływany przy tworzeniu instancji.
- Służy do inicjalizacji atrybutów instancji.

Tworzenie obiektu klasy

```
p = Person()
```

- `p` jest **referencją** do nowego obiektu klasy `Person`.
- W Pythonie wszystko jest „obiektem” i przekazywane przez referencję.

Protokół menadżera kontekstu (`with`)

- Wymaga metod `__enter__()` i `__exit__()`.
- Pozwala na automatyczne zwalnianie zasobów (np. zamykanie plików).

Wielokrotne dziedziczenie w Pythonie

- Python **pozwala** na dziedziczenie z wielu klas jednocześnie.
- Można też dziedziczyć po typach wbudowanych (`list`, `dict`, itd.).

Metoda specjalna `__repr__()`

- Powinna zwracać **łańcuch znaków** (`str`).
- Służy do reprezentowania obiektu (np. w konsoli, debugowaniu).

Klasa = obiekt typu `type`

- Wywołanie `NazwaKlasy(...)` działa jak wywołanie funkcji, bo klasa jest „callable” poprzez metaklasę `type`.

Sprawdzanie typu obiektu

- `isinstance(obj, A)` – wbudowana funkcja do weryfikacji, czy `obj` jest instancją klasy `A`.

Relacja `object` – `type` w Pythonie

- `object` to **podstawowa** klasa, z której dziedziczy wszystko.
- `type` to **metaklasa** dla wszystkich klas (łącznie z `object`).
- W uproszczeniu: `object` jest instancją `type`; `type` też jest instancją samego siebie.
- Fałszywe byłoby stwierdzenie „`object` jest podklasą `type`”.

Duck typing („kacze typowanie”)

- Polimorfizm oparty na **posiadanych metodach/atrybutach**, a nie na deklaracji typu.
- „Jeśli coś wygląda jak kaczka i kwacze jak kaczka, to jest kaczką.”

Przeciążanie operatora w Pythonie (np. `__sub__()`)

- Możemy zmienić zachowanie operatorów (`+`, `-`, itp.).
- Przykład z `__sub__()` może np. sumować zamiast odejmować, jeśli tak zdefiniujemy (uwaga na niespodzianki!).

Przeciążanie `__eq__()`

- Definiuje równość obiektów.

- `!=` domyślnie wywołuje odwrotność (`not __eq__`), chyba że zdefiniujemy `__ne__()` osobno.

Dziedziczenie i `__init__`

- Jeśli w klasie potomnej **nadpisujemy** `__init__()`, trzeba ręcznie wywołać inicjalizację klasy bazowej (np. `super().__init__(...)`), inaczej atrybuty bazy mogą nie być zainicjalizowane.

Iteratory vs. obiekty iterowalne

- Aby użyć `next(...)` bez błędu, obiekt musi mieć `__next__()`.
- Metoda `__iter__()` może zwracać generator; do `next(...)` potrzebujemy obiektu iteratora.
- Samo `with A():` to nie dotyczy, ale analogicznie – trzeba uważać, by klasa była poprawnym iteratorem.

`__rsub__()` (operatory odwrotne)

- Wywoływany, gdy lewy operand operatora `-` nie obsługuje tej operacji i Python próbuje prawego operandem (np. `x - obj` → `obj.__rsub__(x)`).

Monkey patching w Pythonie

- Dynamiczna modyfikacja/uzupełnianie klasy w runtime (np. `Deck.__setitem__ = set_card`).
- Bardzo elastyczne, ale może powodować trudne do wykrycia błędy.

Zmienne klasowe i `@classmethod`

- Atrybut klasy (np. `counter`) jest współdzielony przez wszystkie instancje.
- `@classmethod` ma pierwszy argument `cls`, wskazujący klasę, a nie instancję.
- Przykład: inkrementowanie `cls.counter` w konstruktorze powoduje zmianę widoczną we wszystkich obiektach.

---

3. Java – programowanie sieciowe (pyt. 29–40)

Strumienie kompresji i zapisu obiektów

- Konstrukcja najpierw otwiera się plik (`FileOutputStream`), potem nakłada strumień ZIP (`ZipOutputStream`), a na koniec strumień obiektowy (`ObjectOutputStream`).
  - Przykład (skracając):

```
new ObjectOutputStream(
    new ZipOutputStream(
        new FileOutputStream("plik.zip")
    )
);
```

Serializacja w Javie

- Klasa musi implementować `Serializable`.

- Nie jest wymagane posiadanie `serialVersionUID`, choć jest to zalecane dla kompatybilności wersji.

Wątki w Javie

- **`Runnable`** (metoda `run()`, nie zwraca wartości),
- **`Callable`** (metoda `call()`, może zwracać wartość i rzucać wyjątki).
- Klasa `Thread` implementuje `Runnable`.

Nawiązywanie połączeń sieciowych

- **`ServerSocket.accept()`** – do obsługi nowych połączeń po stronie serwera.
- **`Socket`** – po stronie klienta do łączenia się z serwerem.

`InetAddress` w Javie

- Uzyskujemy poprzez `InetAddress.getByName("www.example.com")`.
- Nie tworzymy bezpośrednio przez konstruktor.

Tworzenie gniazda (klienta / serwera)

- **`Socket`** – klient (nawiązanie połączenia).
- **`ServerSocket`** – serwer (nasłuchiwanie).

Wątki demony w Javie

- Można ustawić wątek jako demon: `thread.setDaemon(true)` przed `start()`.
- Wątek demon kończy się, gdy wszystkie wątki użytkownika zakończą działanie.

Zwracanie wartości z wątku

- **Runnable** – brak zwracanego wyniku.
- **Callable** – zwraca wartość przez `Future`.

Archiwizacja ZIP

- **`ZipOutputStream`** – do kompresji i tworzenia plików ZIP.

Synchronizacja wątków w Javie

- Metody **`wait()`** i **`notify()`** (lub `notifyAll()`) służą do zarządzania wstrzymywaniem i wybudzaniem wątków wewnątrz sekcji zsynchronizowanej.

Połączenie z bazą danych (JDBC)

- Wymagany jest **sterownik JDBC** dopasowany do konkretnej bazy (MySQL, PostgreSQL, itd.).

Odczyt strony internetowej w Javie

- Możliwy np. przez `URL.openStream()`, `HttpURLConnection`, lub zewnętrzne biblioteki (Apache HttpClient).

---

4. Podstawy programowania współbieżnego (pyt. 41–52)

Program współbieżny – definicja

- Program składający się z kilku strumieni wykonania (wątków/procesów), które mogą działać jednocześnie (wielordzeniowo) lub przeplatać się (jednordzeniowo).

Proces

- „Uruchomiony program” ze **własną** przestrzenią adresową i zasobami systemowymi.
- Może zawierać wątki.

Instrukcja atomowa

- Wykonywana w sposób „nieprzerywalny”: albo cała, albo wcale.
- Gwarantuje brak interferencji innych wątków w trakcie jej działania.

Równoległość a współbieżność

- **Równoległość**: fizyczne wykonywanie kodu w tym samym czasie na różnych rdzeniach.
- **Współbieżność**: logiczne przeplatanie zadań (na jednym lub wielu rdzeniach).

Sekcja krytyczna

- Fragment kodu, który może być wykonywany w danym momencie tylko przez **jeden** wątek/proces.
- Chroni wspólne zasoby (np. wspólne zmienne).

Poprawność programu współbieżnego

- Weryfikacja **własności bezpieczeństwa** (np. poprawna synchronizacja) i **własności żywotności** (np. brak zakleszczenia, brak zagłodzenia).

Semafory

- `wait(s)` (lub `P(s)`) i `signal(s)` (lub `V(s)`).
- Zapewniają kontrolę dostępu do zasobów i synchronizację wątków.
- Można z ich pomocą budować prymitywy do sekcji krytycznych czy bariery.

Bariera

- Mechanizm synchronizacyjny, w którym **wszystkie** wątki/procesy muszą „dotrzeć” do pewnego punktu, by mogły kontynuować.
- Stosowana np. w obliczeniach równoległych etapowych.

Zmienna warunkowa w monitorze

- Pozwala wstrzymać wątek do momentu spełnienia warunku.
- Typowe metody: `wait()`, `signal()`, `signalAll()` wywoływane z wnętrza monitora.

Problem „czytelnicy i pisarze”

- Naruszenie bezpieczeństwa: gdy co najmniej jeden pisarz **pisze** i choć jeden czytelnik **czyta** równocześnie.
- Czytelnicy mogą czytać równolegle, pisarz pisze sam, bez czytelników w tym czasie.

Algorytm Dekkera

- Jeden z najstarszych *„software'owych"* algorytmów zapewniających wzajemne wykluczanie (dla 2 procesów) *bez wsparcia sprzętowego*.

---

5. Przetwarzanie obrazów cyfrowych (pyt. 53–64)

Podstawowe przekształcenia geometryczne obrazu

- **Przesunięcie**, **obrót**, **odbicie (lustrzane)**, **skalowanie**, **ścięcie (shear)**.
- Często nazywane transformacjami afinicznymi.

Operacje punktowe na obrazie

- Nowa wartość piksela zależy **tylko** od wartości tego samego piksela w obrazie wejściowym.
- Przykłady:
  - **Operacje logiczne** (NOT, AND, OR itp.).
  - **Operacje arytmetyczne** (dodanie stałej, mnożenie).
  - **Progowanie (binaryzacja)**.
  - **Wyrównywanie histogramu**.
  - **LUT (look-up table)**.

Operacje logiczne (obrazy)

- **NOT** (negacja) – np. w skali szarości: `nowy = 255 - stary`.
- **AND**, **OR** – piksel po pikselu.

Operacje arytmetyczne (obrazy)

- **Dodawanie/odejmowanie** obrazów (piksel w piksel),
- **Mnożenie** przez stałą (rozjaśnienie/przyciemnienie),
- **Mieszanie** (np. blending dwóch obrazów).

Gradient Robertsa

- Maski w kierunkach **ukośnych (45° i 135°)**.
- Używane do wykrywania krawędzi (mimo że to filtr dość prosty).

Sposoby rozmieszczenia elementów obrazu (siatka)

- **Siatka prostokątna** (najpopularniejsza).
- **Siatka heksagonalna** (rzadziej używana).

Konwersja obrazu RGB na czarno-biały

- Najczęściej **skala szarości (grayscale)**.

- *Binaryzacja* to już sprowadzenie do 0/1, też „czarno-biały", ale w sensie bitowym.

Filtry górnoprzepustowe

- **Wyostrzają** obraz i wydobywają detale (krawędzie).
- Mogą jednocześnie wzmacniać szum.

Filtry dolnoprzepustowe

- **Wygładzają** obraz (rozmycie).
- Usuwają wysokie częstotliwości (w tym szum i drobne detale).

Operacje kontekstowe

- Wymagają sąsiedztwa piksela (np. **filtracja splotowa**, **mediana**).
- W przeciwieństwie do operacji punktowych, gdzie liczy się tylko pojedynczy piksel wejściowy.

Binaryzacja (progowanie)

- **Zamiana** wartości piksela na 0 lub 1 (czerń/biel) na podstawie progu (np. jeżeli piksel > T, to 1, w przeciwnym razie 0).
- Upraszcza obraz do dwóch wartości.

Wykrywanie krawędzi w obrazie

- **Obliczanie gradientu** (np. operatory Sobela, Canny, Prewitta).
- Szukanie miejsc o dużej zmianie intensywności piksela.

Podsumowanie

Powyższe zagadnienia stanowią esencję tematów mogących pojawić się na egzaminie inżynierskim – od **aplikacji webowych w Java**, przez **programowanie obiektowe w Pythonie**, **sieci** i **współbieżność** w Javie, aż po **przetwarzanie obrazów**. Warto zrozumieć nie tylko definicje, ale także *dlaczego* dane rozwiązania (np. semafory, filtry) działają w określony sposób i jakie problemy rozwiązują.

Notatki do egzaminu inżynierskiego 2024 – **zagadnienia i najważniejsze informacje**

Poniżej znajdują się **najważniejsze zagadnienia** oraz **kluczowe informacje** wyciągnięte z zestawu pytań egzaminacyjnych z różnych dziedzin. Materiał został przedstawiony w formie notatek, **posegregowanych tematycznie** i zebranych w przejrzysty sposób.

---

1. Elektronika

1.1. Półprzewodniki, diody i tranzystory

- **Napięcie przewodzenia (bariera potencjału)** krzemowego złącza półprzewodnikowego: (0{,}6\text{-}0{,}7;V).
- **Dioda Zenera** służy przede wszystkim do **stabilizowania napięcia** (np. w układach zasilania).
- **Dioda LED** (Light Emitting Diode) emituje światło, gdy jest **spolaryzowana w kierunku przewodzenia**.
- **Tranzystor bipolarny** umożliwia:
  - wzmacnianie natężenia prądu,
  - wzmacnianie napięcia,
  - przełączanie oraz wzmacnianie (zależnie od układu).
- **Przy wzroście temperatury** statyczny współczynnik wzmocnienia prądowego (beta, (h_{FE})) **tranzystora bipolarnego** zwykle **rośnie**, ale zależy to też od zakresu temperatur.

1.2. Zasilacze

- **Zasilacze impulsowe** (przetwornice) mają wysoką sprawność dzięki pracy z **wyższą częstotliwością** (np. powyżej 100 kHz).
- **LDO (Low Drop Out)** to liniowe stabilizatory o **zmniejszonym spadku napięcia** pomiędzy wejściem a wyjściem.
- Zasilacz do laptopa **musi mieć** zgodne napięcie i natężenie prądu.
- **Akumulatory Li-Ion** mają **największą gęstość energii** spośród wymienionych (w porównaniu np. z NiMH, NiCd, ołowiowymi).

1.3. Wzmacniacze operacyjne, generatory

- **Wzmacniacz operacyjny** może pracować m.in. w konfiguracjach:
  - wzmacniacza odwracającego fazę,
  - nieodwracającego,
  - całkującego,
  - komparatora.
- **Generator** sygnału elektrycznego stosuje **dodatnie sprzężenie zwrotne** do podtrzymania generacji.

---

2. Fizyczne podstawy działania urządzeń informatycznych

2.1. Promieniowanie elektromagnetyczne, efekt Comptona

- **Promieniowanie EM** powstaje, gdy **ładunek** nie jest w spoczynku ani nie porusza się jednostajnie po prostej (przyspieszenie, zmiana kierunku itp.).
- **Efekt Comptona** to **zmiana długości fali** promieniowania rentgenowskiego (lub wysokoenergetycznego) rozpraszanego na **swobodnych elektronach**.

2.2. Poziomy energetyczne, atom wodoru

- **Energia elektronu** w atomie wodoru (w modelu uproszczonym) zależy głównie od **głównej liczby kwantowej** (n).

2.3. Ładunki elektryczne, przyciąganie/odpychanie

- Zbliżenie kuli naładowanej do nienaładowanej powoduje **przyciąganie** (bo w kuli nienaładowanej indukuje się ładunek przeciwny).

- **Opór półprzewodnika** ze wzrostem temperatury **maleje**, ponieważ rośnie liczba nośników ładunku.
- **Przyspieszenie** ciała we wszystkich inercjalnych układach odniesienia ma **identyczną wartość** (o ile te układy nie przyspieszają względem siebie).

> Uwaga: Treść w pytaniach sugeruje, że przyspieszenie w różnych inercjalnych układach jest takie samo (wariant A – identyczna wartość).

2.4. Prędkość światła, pole magnetyczne

- **Prędkość światła w próżni** jest **stała** we wszystkich inercjalnych układach odniesienia.
- Rozcięcie **magnesu sztabkowego** daje dwa mniejsze magnesy, **każdy** z **dwoma biegunami** (N i S).
- W cewce płynący **prąd stały** powoduje **przyciąganie zwojów**, co skutkuje **skróceniem** cewki.
- **Pole magnetyczne** odchyla tor elektronu i protonu w **przeciwnych kierunkach** (ładunki mają różne znaki).
- **Zjawisko fotoelektryczne**: emisja elektronów z powierzchni metalu pod wpływem promieniowania EM.
- Wahadło w polu elektrostatycznym skierowanym pionowo w górę **nie zmienia** okresu drgań (jeśli ładunek jest niewielki i brak innych sił bocznych).

---

3. Inżynieria oprogramowania

3.1. Analiza klas, generalizacja, model przyrostowy

- **Generalizacja**: wspólne informacje w superklasie – można ograniczyć rozproszenie danych.
- **Model przyrostowy**: budowanie wstępnej implementacji i jej **stopniowe udoskonalanie**.

3.2. Testowanie, debugowanie, bug tracker

- **Testowanie**: uruchomienie programu i sprawdzenie poprawności działania.
- **Debugowanie**: poszukiwanie przyczyny błędu.
- **Bug tracker**: narzędzie do ewidencji błędów, wspiera zarządzanie defektami.

3.3. Walidacja, IDE, wytwarzanie przyrostowe

- **Walidacja**: sprawdzenie **zgodności programu z wymaganiami** użytkownika.
- **Zintegrowane środowisko programowania (IDE)**: wspólny zbiór narzędzi (edytor, debuger, kompilator itp.).
- **Wytwarzanie przyrostowe**: tańsze i łatwiejsze wprowadzanie zmian, możliwość wielokrotnego sprawdzania.

---

4. Systemy czasu rzeczywistego

4.1. Pojęcia podstawowe

- **System twardy czasu rzeczywistego**: musi **gwarantować** dotrzymanie ścisłych deadline'ów (np. system kontroli lotu).
- **Dziedziczenie priorytetu**: mechanizm **zapobiegania inwersji priorytetów** (zadanie blokujące zadanie o wyższym priorytecie może wykonywać się z priorytetem wyższym).

- **Algorytm szeregowania** EDF (Earliest Deadline First) jest optymalny w systemach dynamicznego przydziału priorytetów.
- **Stan zadania** w systemach RT, jeśli nie spełnia warunków czasowych do wykonywania, nazywa się **uśpiony** (lub inna nomenklatura w pytaniach).
- **System operacyjny czasu rzeczywistego** z dwoma jądrami (np. RTLinux) – jądro RT + jądro systemu.

4.2. EDF, determinizm

- **EDF** wymaga zdefiniowania terminu wykonania (deadline).
- Cechy systemu czasu rzeczywistego: **deterministyczność, niezawodność, reaktywność**. Niepożądana jest **nieprzewidywalność**.

---

5. Systemy wbudowane

5.1. Mikrokontrolery i peryferia

- System wbudowany jest oparty **na mikrokontrolerze**, wykonującym **ograniczoną liczbę zadań**.
- **Port mikrokontrolera AVR**: zwykle 8 pinów I/O.
- **Układy peryferyjne** to np. **DAC**, **ADC**, UART, SPI, I2C. Stabilizatory napięcia są elementami zasilania, nie zawsze wbudowane.
- **Pamięć mikrokontrolera**:
  - **Flash** – przechowuje program.
  - **EEPROM** – pamięć nieulotna na dane.
  - **SRAM** – pamięć ulotna na dane w trakcie działania.

5.2. Sygnały taktujące, transmisja

- **Wyprowadzenia generatora** mikrokontrolera: można dołączyć kwarc, rezonator.
- **I2C** w trybie Fast Mode – przepustowość do 400 kb/s.
- **SPI**: 4 linie: MOSI, MISO, SCK, SS.
- **BASCOM**: kompilator do mikrokontrolerów AVR (dialekt BASIC).
- **Pin INT** w AVR – przerwanie sprzętowe.

---

6. Wybrane aspekty automatyki i robotyki

6.1. Pojęcia

- **Stopnie swobody** w parach kinematycznych (np. klasa V → 1 st.).
- **Manipulator** – mechanizm do realizacji funkcji kończyny górnej.
- **Sensor** – element wejściowy systemu (czujnik).
- **Efektor** – urządzenie na końcu ramienia robota (np. chwytak).
- **Siłownik** – urządzenie wykonawcze (ruch postępowy/obrotowy).

6.2. Ogólna budowa robota, mikrokontrolera

- Układ robota: **mikrokontroler, czujniki, siłowniki, zasilanie/efektory**.
- Mikrokontroler: **CPU, RAM, FLASH, EEPROM, układy I/O**.
- **Przestrzeń robocza robota** – cały obszar, do którego może dotrzeć efektor.

- **Dokładność** (accuracy) – miara bliskości osiągania punktu.
- **Powtarzalność** (repeatability) – dokładność powrotu do nauczonego punktu.
- **Serwomechanizm** – układ nadążny, precyzyjna regulacja np. ramienia.
- **Serwosilnik** – składa się z **silnika** + **enkodera**.

---

7. Matematyka (różne działy)

7.1. Logika, rachunek zbiorów, relacje

- Zdanie logiczne to takie, któremu można przypisać wartość prawda/fałsz.
- Tautologia – formuła, której wartość logiczna to zawsze prawda.
- Rachunek zdań, rachunek zbiorów: np. własności relacji (zwrotność, symetryczność, przechodniość).
- Relacja równoważności: **zwrotna, symetryczna, przechodnia**.

7.2. Funkcje, ciągi, granice

- Funkcja injective, surjective, bijective.
- Granice ciągów i funkcji (czasem przykłady).
- Szeregi Maclaurina (np. cos(x)).
- Pochodne, całki oznaczone, własności całek.

7.3. Liczby zespolone, macierze

- Równania z liczbami zespolonymi.
- Macierz, jej wyznacznik, macierz odwrotna (detA ≠ 0 → nieosobliwa).
- Rząd macierzy (0 ≤ r ≤ n).

7.4. Metody numeryczne

- **Błąd względny** – (\frac{|x_{dokładne} - x_{obliczone}|}{|x_{dokładne}|}).
- **Metoda Newtona** do znajdowania pierwiastków równania wymaga znajomości pochodnej.
- **Metody całkowania**: metody trapezów, Simpsona.
- **Interpolacja** i **aproksymacja**.
- **Metoda Eulera** i **Runge-Kutty** – rozwiązywanie równań różniczkowych.

---

8. Podstawy modelowania i symulacji

- **Oscylator harmoniczny** (równania z częstością własną, tłumieniem, wymuszeniem).
- **Modele wzrostu populacji**: Malthusa, Verhulsta, Lotki-Volterry (drapieżnik-ofiara).
- **Modele probabilistyczne** vs. **deterministyczne**.
- Metoda Eulera w prostych układach (np. RC).
- Rozwój epidemii (model Malthusa).

---

9. Komputerowa grafika użytkowa

- **Grafika rastrowa** (np. BMP, JPG, PNG, GIF) vs. **grafika wektorowa** (np. SVG, CDR, AI).
- **GIF** umożliwia animacje.

- **Model kolorów** np. RGB.
- Tablice Ishihary – do diagnozy daltonizmu.

---

10. Komputerowe wspomaganie zadań inżynierskich (CAD)

- **AutoCAD**: format DWG (projekt), BAK (kopia zapasowa).
- Wprowadzanie współrzędnych: np. `@8<60` (długość=8, kąt=60°).
- **Warstwy** w AutoCAD: można usuwać warstwę bez obiektów, ale nie warstwę 0 (domyślną).
- **Inventor**: IAM – zespół, IPT – część, IDW – rysunek.

---

11. Inżynieria i analiza danych

11.1. Podstawowe operacje w Pandas, NumPy, scikit-learn

- `describe()` – statystyki opisowe.
- `train_test_split()` – podział na zbiór treningowy/testowy.
- `apply(lambda x: x.lower())` – operacja na kolumnie tekstowej.
- `pivot_table()` – tworzenie tabeli przestawnej.
- `corr()` – korelacja kolumn.
- NumPy: `tab[tab[:,2] > 10]` – wybór wierszy.

11.2. Regresja, korelacja, macierz konfuzji, PCA

- **Regresja liniowa**: minimalizacja sumy kwadratów błędów.
- **PCA** (Principal Component Analysis) – redukcja wymiarowości.
- **Macierz konfuzji** – ocena jakości modelu klasyfikacyjnego.
- **Analiza skupień** – grupowanie podobnych obserwacji.

---

12. Projekt multimedialny

- Zakres słyszalny dźwięku dla człowieka: ~20 Hz – 20 kHz (przykład pytania: 10 kHz jest słyszalne).
- **Próg absolutny słyszenia** – najniższy poziom dźwięku słyszalny w ciszy.
- Metody wizualizacji: wykresy, grafy, AR, mapy (nawigacja GPS to też wizualizacja, choć w pytaniu było, że jest to raczej osobna kategoria).
- Licencje Creative Commons (np. CC BY-SA 4.0 – pozwala na wykorzystanie pod warunkiem udostępnienia utworów pochodnych na tej samej licencji).
- **Audacity** – aplikacja do edycji dźwięku.

---

13. Oprogramowanie użytkowe (edytory, arkusze, bazy)

13.1. Dokumenty, sekcje, makra w edytorach tekstu

- **Sekcje** w dokumencie wielostronicowym pozwalają określić **różny układ strony** w różnych częściach.
- **Makra** – automatyzacja powtarzalnych czynności, integracja z językiem VBA (w MS Office) itp.

13.2. Arkusz kalkulacyjny: formuły tablicowe, tabele przestawne, konsolidacja

- **Formuły tablicowe** – wprowadzane do **zakresu** komórek.
- **Tabela przestawna** – grupowanie i podsumowanie złożonych danych.
- **Konsolidacja** – łączenie wartości z różnych arkuszy/zakresów i ich podsumowywanie.

13.3. SQL w aplikacjach

- **SELECT FROM** – kluczowe słowa dla kwerend wybierających.
- **ORDER BY** – porządkowanie rekordów.
- **Parametr WHERE** – filtrowanie.

---

14. Bazy danych w aplikacjach internetowych

14.1. Podstawy

- Tabele powinny być **znormalizowane** (redukcja redundancji, poprawa spójności).
- **Kursory** – iteracyjne przeglądanie wyników zapytań.
- **Triggery (wyzwalacze)** wykonują się automatycznie po wystąpieniu zdarzenia (INSERT, UPDATE, DELETE).
- **Transakcje** – albo wykonują się w całości, albo w ogóle (ACID: Atomicity, Consistency, Isolation, Durability).
- **Uprawnienia** w MySQL: GRANT / REVOKE.

14.2. Indeksy

- **Indeksy** przyśpieszają wyszukiwanie/sortowanie.
- Tworzenie indeksu (np. `CREATE INDEX indeksosoby ON osoby(imie, nazwisko);`).

---

15. Języki hipertekstowe i tworzenie stron WWW

15.1. Semantyka HTML, selektory CSS

- Znaczniki semantyczne: `<section>`, `<article>` itp.
- **Selektory**: np. `div > p`, `div p+p`, `tr:nth-child(2n)`.
- Formy i odbieranie danych w PHP: `$_POST["nazwa"]`.
- **Pseudo-klasy** CSS: `:nth-child()`, `:hover`, `:focus` itd.

15.2. Połączenie z bazą MySQL w PHP

- `mysqli_query($conn, $sql);` – wysyłanie zapytań.
- `SELECT * FROM users;` – pobranie wszystkich rekordów z tabeli.
- **Kaskada**: pewne właściwości w CSS się dziedziczą (`font-size`), a pewne nie (`border` nie jest dziedziczone).

---

16. Przetwarzanie dokumentów XML i techniki WWW

- **XML** = eXtensible Markup Language.
- Każdy element ma **znacznik otwierający i zamykający** (struktura poprawnie zagnieżdżona).

- **DTD (Document Type Definition)** – opis struktury XML.
- **XPath** – język zapytań do dokumentów XML: `//`, `/`.
- Atrybut nie może mieć wielu wartości jednocześnie (pojedynczy łańcuch znaków).
- **XSLT** – przekształcanie XML.

---

17. Administracja serwerami WWW

17.1. Bezpieczeństwo, TLS/SSL, proxy

- **TOR** – sieć anonimizująca, problem namierzania węzłów wejścia/wyjścia.
- **Handshake TCP** i **TLS** – kolejność zestawiania połączeń https.
- Certyfikat serwera https – **używany do szyfrowania klucza symetrycznego** przez klienta.
- Apache2 i porty: `Listen <port>`, wirtualne hosty `<VirtualHost IP:port>`.
- **Reverse Proxy**: publikacja serwera wewnątrz LAN na zewnątrz.

17.2. Logi, .htaccess

- Plik `access.log` nie loguje MAC klienckiego.
- Pliki `.htaccess`: sterowanie konfiguracją w wybranym katalogu.

---

18. Sieci komputerowe

18.1. Protokół, topologie, warstwy

- **NAT** – translacja adresów na wybranych interfejsach routera.
- **ARP** – odwzorowanie IP → MAC.
- **Topologia pierścienia** – uszkodzenie jednej stacji zatrzymuje ruch w całej sieci.
- **Skrętka** – 4 pary żył (8 przewodów).
- **Warstwa fizyczna**: sygnały, kable, transceivery.
- W warstwie łącza danych dodajemy **adres fizyczny** (MAC).

18.2. Adresowanie, podsieci, routing

- Jeśli host ma **brak** prawidłowej bramy domyślnej, nie wychodzi poza swoją sieć.
- Do urządzeń typu **drukarki sieciowe** przypisujemy zwykle **styczny (statyczny) adres IP**.
- **CSMA/CD** w sieciach przewodowych Ethernet, **CSMA/CA** w bezprzewodowych Wi-Fi.

18.3. Poczta e-mail, DNS

- Wysyłanie e-maila: MUA → SMTP → MTA → ... → POP → MUA (lub IMAP).
- **nslookup** – narzędzie diagnostyczne do DNS.

---

19. Integracja sieci i usług

19.1. Routing, VLAN

- **Trasa podsumowująca** (np. OSPF/EIGRP) – jeden wpis łączący kilka podsieci.

- **Zbieżność sieci** – czas aktualizacji tablic routingu po zmianach.
- **Routing między VLAN** – starszy router potrzebuje **wielu interfejsów** (po jednym na VLAN) lub trunk + router on a stick (jeśli protokół obsługiwany).
- **OSPF** – ID routera: jeśli brak pętli zwrotnej, używany najwyższy adres IP spośród aktywnych.

---

20. Języki skryptowe (Python)

- Python jest **interpretowany i dynamicznie typowany**.
- Struktury danych: listy (mutowalne), krotki (niemutowalne), zbiory, słowniki.
- `sorted()` sortuje listy, `.sort()` – metoda wbudowana list.
- Dziedziczenie w Pythonie może być **wielokrotne**.
- **Lambda** – funkcja anonimowa.
- **Destructor** w Pythonie (`__del__`) wywoływany, gdy obiekt jest usuwany przez garbage collector.

---

21. Systemy operacyjne

21.1. Procesy, pamięć, pliki

- **Stan procesu**: przejście `gotowy → aktywny` po decyzji planisty.
- Pamięć **pomocnicza**: zwykle dysk twardy (HDD/SSD).
- **Jądro (kernel)** – najważniejsza część systemu.
- Struktura katalogów: w Unix/Linux **drzewo** z root `/`, w Windows litera dysku `C:` i tak dalej.

21.2. Prawa dostępu, uprawnienia, potoki

- W Linuxie: `rwx` dla właściciela, grupy, reszty.
- Potok – przekierowanie **wyjścia jednego procesu** na **wejście drugiego** (`|`).
- Zadania w trybie **I/O** czekają w kolejce do urządzenia.
- **Problem głodzenia** – np. w planowaniu priorytetowym.
- **Zakleszczenie (deadlock)** – problem 5 filozofów.

21.3. System plików

- i-węzeł (inode) występuje w typowych FS Linuxa (np. ext4), **nie** w NTFS.

---

22. Organizacja i architektura komputerów

22.1. Podstawy CPU, rejestry, magistrale

- Pamięć wewnętrzna procesora: **rejestry**.
- Rdzeń GPU – wyspecjalizowany do obliczeń równoległych w grafice.
- Cykle zegara, taktowanie.
- **Magistrala systemowa** łączy CPU, pamięć, urządzenia.

22.2. Systemy liczbowe, bramki, przerzutniki

- Konwersje binarne/szesnastkowe (np. `110111100001` → sporo przykładów).

- **Bramka logiczna** – realizuje podstawowe operacje (AND, OR, NOT).
- **Przerzutnik** – element pamiętający bit.
- **Liczniki** – asynchroniczne i synchroniczne.

22.3. Różne architektury

- SPARC – stworzona przez Sun Microsystems.
- CUDA – stworzona przez NVIDIA.
- **Prawo Moore'a** – podwajanie liczby tranzystorów co ~2 lata.
- **Reguła Pollacka** – wydajność rośnie mniej więcej jak pierwiastek kwadratowy ze wzrostu złożoności.

22.4. Chmura

- **Model SaaS** – usługa w postaci oprogramowania w chmurze (np. Office 365).
- **IaaS** – infrastruktura jako usługa.
- **PaaS** – platforma jako usługa.

---

23. Algorytmy i struktury danych

- **Drzewo zrównoważone** – różnica wysokości poddrzew w węźle ≤ 1.
- **B-drzewa**, **B+ drzewa** – struktury w bazach, węzły wewnętrzne to indeksy, liście mają dane.
- **Sortowanie** – QuickSort, MergeSort, InsertSort, SelectionSort (różne złożoności).
- **Kod prefiksowy**: żaden symbol kodowy nie jest prefiksem innego (np. kod Huffmana).
- **Cykle** w grafach: eulerowski (przechodzimy każdą krawędź raz), hamiltonowski (przechodzimy każdy wierzchołek raz).
- **Problem NP** – problem decyzyjny, dla którego **weryfikacja** rozwiązania jest wielomianowa.

---

24. Programowanie (C, C++, Java) – wybrane aspekty

24.1. C, C++ – wskaźniki, funkcje, struktury

- Tablica przekazana do funkcji jest traktowana jako **wskaźnik** do jej pierwszego elementu.
- `malloc`, `free` – alokacja i zwalnianie.
- Struktury w C – definicja `struct`, dostęp do składowych przez `.` lub `->`.
- Wskaźnik `this` w C++ – pokazuje, na którym obiekcie działa metoda.
- Konstruktor, destruktor w C++ – do zarządzania zasobami.

24.2. Programowanie obiektowe (C++), wzorce projektowe

- Klasa abstrakcyjna – **metoda czysto wirtualna** (`=0`).
- Mechanizm **dziedziczenia**: C++ – pojedyncze, wielokrotne.
- **Przyjaźń** (`friend`) – dostęp do prywatnych składowych, nie jest przechodnia.
- Wzorce projektowe (m.in. Singleton, Fabryka Abstrakcyjna, Kompozyt, Dekorator, Strategia, Fasada, Pełnomocnik itp.).

24.3. Java

- Plik źródłowy może zawierać jedną publiczną klasę o tej samej nazwie co plik.

- Dziedziczenie po jednej klasie (`extends`), ale interfejsów może być wiele (`implements`).
- `final` – zmienna, której wartości nie można już zmienić.
- Klasa anonimowa – deklarowana bez nazwy (np. `new SomeClass() {...}`).
- Wyrażenia lambda w Javie – funkcje anonimowe (`(param) -> ciało`).

---

25. Informatyka kwantowa (wprowadzenie)

- **Operacje kwantowe**: unitarność → odwracalne, zachowują amplitudy.
- **Algorytm Shora**: faktoryzacja liczb pierwszych – zagrożenie dla klasycznych systemów kryptograficznych.
- **Paralelizm kwantowy** – superpozycja i splątanie daje równoległe wykonywanie obliczeń.
- **Zasada dodawania amplitud** – w mechanice kwantowej drogi nierozróżnialne sumują się na poziomie amplitud, a nie intensywności.

---

26. Testowanie oprogramowania

- **TDD (Test Driven Development)** – piszemy test **przed** implementacją kodu.
- **STLC** – Software Testing Life Cycle.
- **7 Zasad testowania** (m.in. testowanie ujawnia usterki, testowanie zależy od kontekstu, wczesne testowanie itd.).
- **Testy jednostkowe** – pisane zazwyczaj przez programistów, skupione na poszczególnych metodach/klasach.
- **Testy regresyjne** – sprawdzają, czy modyfikacje nie psują wcześniej działających fragmentów.
- **Smoke test** – szybka weryfikacja podstawowej funkcjonalności (czy w ogóle „działa”).

---

27. Wzorce projektowe (Design Patterns)

- **Kategorie wzorców**:
  - **Konstrukcyjne**: Factory Method, Abstract Factory, Builder, Prototype, Singleton.
  - **Strukturalne**: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy.
  - **Czynnościowe (operacyjne)**: Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor.
- **Most (Bridge)** – oddzielenie abstrakcji od implementacji.
- **Dekorator (Decorator)** – dynamiczne dodawanie zachowań/odpowiedzialności.
- **Strategia (Strategy)** – rodzina algorytmów, wymienialnych w czasie działania.
- **Budowniczy (Builder)** – tworzenie złożonego obiektu krok po kroku.
- **Mediator (Mediator)** – hermetyzacja interakcji między obiektami.
- **Fasada (Facade)** – ujednolicony interfejs do złożonego podsystemu.
- **Stan (State)** – zmiana zachowania w zależności od stanu wewnętrznego obiektu.
- **Kompozyt (Composite)** – traktowanie obiektów i ich złożeń w jednolity sposób (hierarchia drzewiasta).
- **Fabryka Abstrakcyjna (Abstract Factory)** – tworzenie rodzin powiązanych obiektów bez specyfikacji ich klas.
- **Pełnomocnik (Proxy)** – zastępca obiektu do kontrolowania dostępu.
- **Pula Obiektów (Object Pool)** – ponowne użycie istniejących obiektów, by unikać kosztownych tworzeń.

- **Pyłek (Flyweight)** – współdzielenie stanu w dużej liczbie małych obiektów.

---

28. Relacyjne bazy danych

- **SELECT ... ORDER BY ... LIMIT 3** – wybór np. trzech najlepszych rekordów.
- **Klucze**: główny (PRIMARY), obcy (FOREIGN KEY).
- **ALTER TABLE** – modyfikacja struktury tabeli.
- **WHERE** vs. **HAVING** – `WHERE` filtruje wiersze przed grupowaniem, `HAVING` filtruje grupy po `GROUP BY`.
- **GROUP BY** – grupowanie wyników; **ORDER BY** – sortowanie wyników.
- **Normalizacja** – eliminacja redundancji, poprawa integralności.
- **Indeksy** – przyspieszają wyszukiwanie i sortowanie w tabelach.

Koniec notatek.

> Powyższe zagadnienia stanowią **skondensowany przegląd** kluczowych tematów pojawiających się w pytaniach egzaminacyjnych.
> Warto uzupełnić je o **przykłady** i **szczegółowe definicje** podczas nauki. Powodzenia!