1) Systemy liczbowe i konwersje pomiędzy nimi.

System liczbowy – zbiór reguł jednolitego zapisu i nazewnictwa liczb.

Podział systemów:

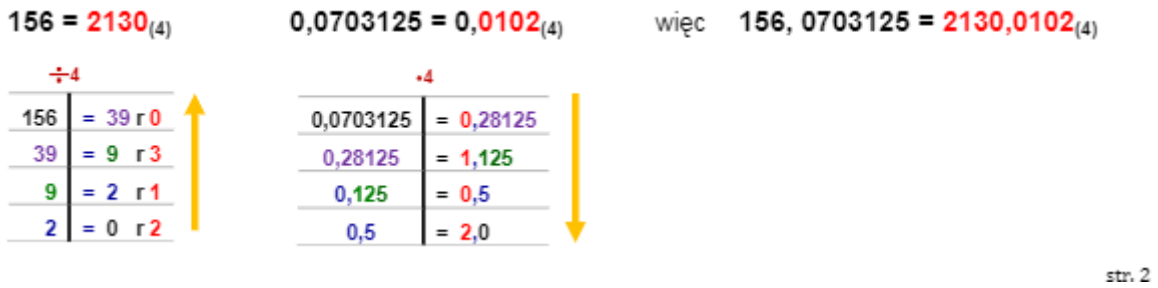
Addytywne, system w których wartośd liczby jest sumą wartości jej znaków cyfrowych.

Pozycyjne - System pozycyjny to metoda zapisywania liczb w taki sposób, że w zależności od pozycji danej cyfry w ciągu, oznacza ona wielokrotność potęgi pewnej liczby uznawanej za bazę danego systemu

Konsersja z systemy X na 10 ->

$$1614_{(5)} = 1 \cdot 5^3 + 6 \cdot 5^2 + 1 \cdot 5^1 + 4 \cdot 5^0 = 1 \cdot 125 + 6 \cdot 25 + 1 \cdot 5 + 4 = 125 + 150 + 5 + 4 = 284_{(10)}$$

Konwersja z systemu 10 na X ->



2)

3) Instrukcje sterujące w języku C.

Instrukcje sterujące pozwalają zmie

nić kolejność wykonywania poleceń zapisanych w skrypcie w zależności od spełnienia lub niespełnienia określonych warunków. Są nimi: IF, switch, pętle (for, while, foreach, do while), catch

4. Zarządzanie pamięcią w języku C.

alokowania pamięci na stercie służy operator new, a do zwalniania – delete

Operator new wymaga wskazania typu danej lub danych, które mają być w przydzielonej pamięci zapisane oraz informacji o ilości tych danych, czyli o wielkości obszaru pamięci, jaki ma być zarezerwowany.

Zarezerwowaną za pomocą new pamięć należy zwolnić, gdy nie będzie już potrzebna. Wykonuje się to za pomocą operatora delete.

5. Budowa, obsługa i formatowanie łańcuchów znakowych w języku C.

Char – pojedynczy znak, jeśli chcemy zrobić ciąg to tworzymy tablicę z rozmiarem n+1 w którym to ostatni wyraz tej tablicy będzie znakiem końcowym. Znaki te możemy konwertować na zmienne typu liczbowego jeśli jest to możliwe.

Formatowanie możemy uzyskać poprzez printf („%typzmiennej”, zmienna)

String

6. Założenia paradygmatu programowania obiektowego.

Program w języku obiektowym

przede wszystkim jest to rodzina imperatywna, więc struktura i logika kodu jest taka sama

zmiana następuje na poziomie struktury danych: dane grupowane są z operacjami, które można na nich wykonać, w nowy twór – obiekt

hermetyzacja (enkapsulacja) polega na ukryciu implementacji obiektu, zaś udostępnieniu jedynie niektórych danych i metod w postaci interfejsu

obiekty można grupować w hierarchicznie definiowane klasy, które mają pewne wspólne cechy (mechanizm dziedziczenia w ramach hierarchii)

hierarchia klas implikuje zawieranie się, co prowadzi do polimorfizmu dynamicznego – metody dobierane są automatycznie zgodnie z przynależnością do klasy

abstrakcja danych pozwala na definiowanie klas w postaci ogólnych szablonów, a na ich podstawie dopiero tworzenia szczegółowszych definicji

Główne pojęcia związane z paradygmatem obiektowym:

Klasa to abstrakcyjny typ danych, definiowany programowo poprzez podanie dopuszczalnych na nim operacji oraz jego reprezentację.

Obiekt to element należący do pewnej klasy.

Klasa potomna (pochodna, podklasa) jest klasą wywiedzioną poprzez dziedziczenie z innej klasy.

Klasa macierzysta (nadrzędna, nadklasa) jest klasą, z której wywodzi się klasa potomna.

Metoda to podprogram (operacja) działający na obiektach danej klasy zgodnie z jej i jego definicją.

Pole to zmienna zamknięta wewnątz obiektu. Komunikat to wywołanie metody.

Protokół obiektu to zbiór sposobów odwołania się do obiektu.

To, co jest w ramach danego obiektu ukryte, a co dostępne, określają kwalifikatory przy metodach i polach:

public oznacza nieograniczoną dostępność z zewnątrz

private oznacza ukrycie przed dostępem z zewnątrz i pozwala przez to na definiowanie typów abstrakcyjnych

protected oznacza ukrycie dla wszystkich za wyjątkiem klas potomnych

7. Idea dziedziczenia i polimorfizmu w programowaniu.

mechanizm współdzielenia funkcjonalności między klasami. Klasa może dziedziczyć po innej klasie, co oznacza, że oprócz swoich własnych atrybutów oraz zachowań, uzyskuje także te pochodzące z klasy, z której dziedziczy. Klasa dziedzicząca jest nazywana *klasą pochodną* lub *potomną* .

Hierarchia klas może przekładać się na hierarchię typów. Możliwe jest wtedy podstawienie pod zmienną (lub atrybut funkcji) typu T obiektu typu S będącego podtypem T i dalsze używanie go jakby był typu T.

Polimorfizm zapisywanie jednej funkcji (metody) pod różnymi postaciami.

Statyczny - Chodzi o to, że w 1 klasie może istnieć dużo metod o takiej samej nazwie, lecz różniących się tylko parametrami.

Dynamiczny jest powiązany z funkcjami wirtualnymi i abstrakcyjnymi, jest to tak zwane przesłanianie (nadpisywanie) funkcji. I

8. Zasięg i czas życia obiektów w języku C++.

9. Obsługa wyjątków w języku C++.

Polega on na tym, że funkcja która błąd wykryje i nie chce lub nie może go obsłużyć sama, rzuca wyjątek, który może być dowolnym obiektem. Rzucenie wyjątku powoduje natychmiastowe przerwanie wykonywania funkcji. Procedura wywołująca może ten wyjątek złapać. Wyjątek niezłapany prowadzi do zatrzymania programu, a więc wyjątki nie mogą zostać zignorowane

Pierwszym ze sposobów obsługi wyjątków jest zastosowanie bloku try catch. Opisowo konstrukcję tę można opisać jako „spróbuj przechwycić wyjątek w bloku try i jeśli on faktycznie wystąpi to wykonaj kod w bloku catch".

Możliwa jest obsługa również za pomocą instrukcji throw. Jeżeli w jakiejś metodzie wykorzystujemy fragmenty kodu mogące spowodować wygenerowanie wyjątku,

10. Definicje obiektu, klasy i szablonu klasy w języku C++.

Obiekt, czyli konkretny egzemplarz danej klasy

Klasa jest abstrakcją obiektów z modelowanej dziedziny problemu. Stanowi wzorzec, opis, na bazie którego będą tworzone obiekty.

Szablony zapewniają mechanizm, za pomocą którego funkcje lub klasy mogą być implementowane dla ogólnych typów danych

11. Algorytmy sortujące.

Bąbelkowe O(n^2) - cyklicznym porównywaniu par sąsiadujących elementów i zamianie ich kolejności w przypadku niespełnienia kryterium porządkowego zbioru. Operację tę wykonujemy dotąd, aż cały zbiór zostanie posortowany

Quick sort O(N log n)- Z tablicy wybiera się element rozdzielający, po czym tablica jest dzielona na dwa fragmenty: do początkowego przenoszone są wszystkie elementy nie większe od rozdzielającego, do końcowego wszystkie większe. Potem sortuje się osobno początkową i końcową część tablicy

Merge sort 2n log n – dzielenie tablicy na dwa wycinki

wywołanie rekurencyjne algorytmu sortującego na tych wycinkach

scalenie dwóch posortowanych wycinków

Zestawienie czasów działania:

- Przez wybór: $O(N^2)$ zawsze
- Bąbelkowe: $O(N^2)$ najgorszy przypadek; O(N) najlepszy przyp.
- Wstawianie: $O(N^2)$ średnio; O(N) najlepszy przypadek
- Shellsort: $O(N^{3/2})$
- Heapsort: O(NlogN) zawsze (jeszcze nie omówione)
- Mergesort: O(NlogN) zawsze
- Quicksort: O(NlogN) średnio; $O(N^2)$ najgorszy przypadek

12. Algorytmy zachłanne.

Algorytmem zachłannym nazywamy algorytm, który sprowadza rozwiazanie problemu do dokonania serii wyborów i kazdego z tych wyborów dokonuje, wybierajac – stosując pewne kryteria – lokalnie najlepsze rozwiązanie.

Ze względu na swoją specyfikę algorytmy zachłanne stosowane są często do rozwiązywania problemów optymalizacyjnych, w których osiągnięcie (możliwie) optymalnego rozwiązania wymaga podejmowania wielu decyzji.

Algorytmy zachłanne nie analizuja całosci problemu, ale w kolejnych swoich krokach próbują uzyskać najwięcej jak się da uzyskać w danym kroku – przy podejmowaniu jednej z wielu możliwych decyzji zawsze wykonuje działanie, które wydaje się w danej chwili najkorzystniejsze.

Zachłanny wybór lokalnie optymalnej możliwości sprawia że algorytmy zachłanne nie zawsze prowadzą do optymalnych rozwiązań

Mimio to są one często stosowane gdyż dają jakieś rozwiązanie trudnego problemu w rozsądnym czasie.

Dla niektórych problemów istnieją algorytmy zachłanne dające optymalne rozwiązanie

Algorytmy zachłanne sprawdzają się w wielu nietrywialnych zagadnieniach np. szeregowanie zadań, najkrótsza ścieżka z pojedynczego źródła w grafie o nieujemnych wagach krawędzi (Dijkstra), minimalne drzewo rozpinające (alg. Kruskala), zagadnienie wydawania reszty

- Problem wyboru zajęć, to problem przydzielenia dostępu do zasobu (np. sali) wykorzystywanego podczas wykonywania pewnych zajęć (zadań).
- Założenia:
  - Niech będzie dany zbiór proponowanych zajęć $S = \{1, ..., n\}$, do których ma być przydzielona sala wykładowa (zasób), w której może się odbywać w danej chwili tylko jedno z tych zajęć.
  - Każde zajęcie ma swój czas rozpoczęcia p_i oraz czas zakończenia k_i $(p_i \leqslant k_i)$. Jeżeli zadanie o numerze i zostanie wytypowane, to zajmuje zasób $[p_i, k_i)$.
  - Zajęcia o numerach i oraz j są zgodne, jeśli $[p_i, k_i) \cap [p_j, k_j) = 0$ tzn. godziny ich trwania na siebie nie zachodzą
- Problem: Wyznaczyć największy podzbiór parami zgodnych zajęć.

13. Zasady programowania dynamicznego.

Programowanie dynamiczne jest techniką lub strategią projektowania algorytmów stosowaną przeważnie do rozwiązywania wieloetapowych problemów decyzyjnych.

Programowanie dynamiczne opiera się na podziale rozwiązywanego problemu na mniejsze, **ZALEŻNE** od siebie podproblemy względem kilku parametrów

Każdy podproblem rozwiązywany jest tylko raz, po czym wynik zostaje zapamiętany w odpowiedniej tabeli, unikając zbędnych wielokrotnych obliczeń.

Zagadnienia odpowiednie dla programowania dynamicznego cechuje to że zastosowanie do nich metody siłowej (brute force) prowadzi do ponadwieomianowej liczby rozwiązań podproblemów (wiele kombinacji rozwiązań), podczas gdy sama liczba różnych podproblemów jest wielomianowa

Ponieważ podproblemy są zależne tj, zawierają powtarzające się fragmenty, zastosowanie algorytmu typu dziel i zwyciężaj wykonuje więcej pracy niż to w istocie konieczne, wielokrotnie rozwiązując ten sam problem

Wyodrębnione dla danego zagadnienia podproblemy MUSI cechować **własność optymalnej podstruktury**

Programowanie dynamiczne zwraca zawsze dobą odpowiedź przy wydawaniu reszty

14. Metoda „dziel i zwyciężaj" konstruowania algorytmów.

W podejuciu dziel i zwyciężaj każdy poziom rekursji składa się z następujicych trzech etapów:

DZIEL - problem główny zostaje podzielony na podproblemy

ZWYCIĘŻAJ - znajdujemy rozwiązanie podproblemów

POŁĄCZ - rozwiązania podproblemów zostają połączone w rozwiązanie problemu głównego

Przykłady:

Znajdowanie największego i najmniejszego elementu zbioru.

Wyszukiwanie binarne

Bisekcja miejsca zerowe funkcji

Potęgowanie liczb

Sortowanie szybkie

Sortowanie przez scalanie

Algorytm Euklidesa

15. Struktura kopców binarnych.

Kopiec binarny (ang. binary heap) jest kompletnym drzewem binarnym, co oznacza, że posada zapełnione wszystkie poziomy za wyjątkiem ostatniego, a ostatni poziom jest zapełniany bez przerw, poczynając od strony lewej do prawej. żaden z synów węzła nie jest od niego większy,

16. Algorytmy wyszukiwania najkrótszej ścieżki w grafie.

W uproszczeniu graf to zbiór wierzchołków, które mogą być połączone krawędziami w taki sposób, że każda krawędź kończy się i zaczyna od wierzchołka.

Drzewo to struktura danych składająca się z wierzchołków (węzłów) i krawędzi, przy czym krawędzie łączą wierzchołki w taki sposób, iż istnieje zawsze dokładnie jedna droga pomiędzy dowolnymi dwoma wierzchołkami

Graf bez wag

Tworzymy tablicę z indeksami wierzchołków

Wstaw do kolejki wierzchołek V w tablicy przypisujemy do tego wierzchołka 0

Dopuki kolejka nie jest pusta powtarzaj:

Bierzemy z klejki wierzchołek i dla każdego jego sąsiada wstawiamy dla którego w tablicy nie ma wartości przypisujemy wartość obecnego wierzchołka + 1

Graf z wagami

Tworzymy tablicę z wagami d

Dopóki istnieje nieodwiedzony wierzchołek powtarzaj:

Wybieramy nieodwiedzony wierzchołek W o najmniejszej wartości d[w]

Odwiedziamy W

Jeśli dla każdego q sąsiada w wykonujemy d[q] > d[w] + waga ; d[q] = d[q] + waga

17. Sposoby implementacji słownika.

- Listy lub tablice (naiwne)
  - nieuporządkowane
  - uporządkowane
- Drzewa
  - BST
  - AVL, czerwono-czarne
  - B-drzewa
  - samoorganizujące drzewa BST (*)
  - (a,b)-drzewa (w szczególności 2-3-trees) (*)
- Tablice haszujące

- Wykorzystujemy drzewa binarne, ale konstruowane tak aby wypisując przechwowyane klucze w kolejności INORDER otrzymać ciąg rosnący

B-drzew

- skrajnie lewy syn jest korzeniem poddrzewa zawierającego klucze mniejsze od kluczy przechowywanych w ojcu
- skrajnie prawy syn jest korzeniem poddrzewa zawierającego klucze większe od kluczy przechowywanych w ojcu
- i-ty syn jest korzeniem drzewa zawierającego klucze należące do przedziału pomiedzy (i-1)-tym a i-tym kluczem ojca

- Jeżeli popatrzymy na słownik jak na tablicę indeksowaną kluczami, stąd inne nazwy słownika – tablica asocjacyjna, mapa

- Dwie tablice – keys i values
- Tablica keys trzyma klucze, równoległa tablica values pod odpowiadającymi indeksami trzyma wartości

TABLICA HASZUJĄCA – idea

tab – n-elementowa tablica rekordów postaci (klucz, wartość)
h – funkcja haszująca
wstawienie pary (k,w) do tablicy haszującej tab → wykonanie przypisania tab[(k)]=(k,w)

Cel _ reprezentacja sªownika z liniowym porz¡dkiem na zbiorze kluczy
Wykorzystujemy drzewa binarne, ale konstruowane tak aby wypisuj¡c przechwowyane klucze w kolejno±ci INORDER otrzyma¢ ci¡g rosn¡cy
Czyli lewy syn (i wszystkie elementy lewego poddrzewa) jest zawsze mniejszy, a prawy syn (i wszystkie elementy prawego poddrzewa) wi¦kszy od ojca

18. Tablice mieszające.

Tablica mieszająca lub tablica z haszowaniem – struktura danych, która jest jednym ze sposobów realizacji tablicy asocjacyjnej, tj. abstrakcyjnego typu danych służącego do przechowywania informacji w taki sposób, aby możliwy był do nich szybki dostęp. Tablica mieszająca umożliwia również szybkie porównywanie danych, np. fragmentów tekstów, plików. Odwołania do przechowywanych obiektów dokonywane są na podstawie klucza, który dany obiekt (informację) identyfikuje.

pozwalają na szybkie wykonywanie operacji wstawiania, usuwania i wyszukiwania elementu o danym kluczu pesymistyczna złożoność każdej z tych operacji wynosi O(n) w praktyce, przy rozsądnych założeniach, oczekiwany czas tych operacji wynosi O(1)

W tablicy mieszającej stosuje się funkcję mieszającą, która dla danego klucza wyznacza indeks w tablicy; innymi słowy *przekształca* klucz w liczbę z zadanego zakresu

19. Algorytmy Monte Carlo oraz algorytmy Las Vegas.

Oba algorytmy są probalistyczne - Algorytm probabilistyczny polega na losowym przeszukiwaniu przestrzeni rozwiązań, przy czym kolejne próby znalezienia rozwiązania są wykonywane
na podstawie wskzań losowych

Las Vegas

Cel: Znajdź liczbę pierwszą w podanym przedziale

Działanie: Dopóki nie znajdziesz liczby pierwszej powtarzaj losowanie liczby z zakresu oraz weryfikacji jej pierwszości

Złożoność:

Oczekiwana liczba losowań - Log n gdzie n to rozmiar zakresu

Algorytm sprawdzania pierwszości to n^2 gdzie n to liczba bitów sprawdzanej liczby

Monte Carlo

Cel :Obliczanie pola figury

Działanie

-Wyznacz kwadrat K zawierający fiugrę F

- Wylostuj I punktów leżących wewnątrz kwadratu

- Dla każdego punktu sprawdz czy leży wewnątrz figury F

- zwróć Pole(K) * liczba pkt w figurze / liczba pkt poza nią

20. Pojęcia P, NP, NP-zupełne.

Problemami klasy NP nazywamy problemy decyzyjne w których **sprawdzenie** poprawności określonego rozwiązania wymaga złożoności obliczeniowej wielomianowej

Problemami klasy P nazywamy problemy decyzyjne w których **znalezienie** rozwiązania wymaga złożoności obliczeniowej wielomianowej

21. Główne paradygmaty programowania.

Programowanie imperatywne:

- główna koncepcja - instrukcja,
- program - sekwencja instrukcji,
- wykonanie programu- wykonanie instrukcji,
- wynik - końcowy stan pamięci komputera.

Programowanie obiektowe:

- główna koncepcja - obiekt,
- program - zbiór obiektów klas,
- wykonanie programu- wymiana wiadomości pomiędzy obiektami,
- wynik - końcowy stan stanów obiektów.

Programowanie funkcyjne:

- główna koncepcja - funkcja,
- program - zbiór funkcji,
- wykonanie programu - wyliczanie funkcji,
- wynik - wartość funkcji głównej.

Programowanie logiczne:

- główna koncepcja - predykat,
- program - formuły logiczne: aksjomaty i twierdzenie,
- wykonanie programu - kontrola poprawności twierdzenia,
- wynik - powodzenie bądź niepowodzenie kontroli poprawności twierdzenia.

22. Deterministyczne i niedeterministyczne automaty skończone.

Celem jest rozstrzygnięcie w skończonej ilości kroków,czy dowolne słowo należy,czy też nie należy do języka badanego przez automat

Automat skończony, w którym dla każdego wejścia (symbolu z języka) istnieje jedno i tylko jedno przejście z aktualnego stanu.

NASy rozszerzają definicję DAS o

dany stan nie musi mieć przejścia na każdy symbol wejściowy

możliwość wielu przejść po tym samym symbolu (niedeterminizm)

Każdy DAS jest NAS

Niedeterministyczny automat skończony różni się od deterministycznego automatu skończonego tym, że przeczytanie tego samego symbolu w danym stanie może powodować przejście do jednego z kilku różnych stanów.

Automat to abstrakcyjny model opisu pewnej klasy maszyn obliczeniowych i oprogramowania. Jest użyteczny w przypadku procesów, w których można wyodrębnić zbiór parametrów w całości je definiujący. W przypadku oprogramowania może symulować instrukcje warunkowe, skoki i pętle/rekurencje.

Definicja: deterministyczny automat skończony

Deterministyczny automat skończony (DAS) to automat skończony, deterministyczny, zdefiniowany poprzez $(Q, \Sigma, \delta, q_0, F)$, gdzie

- Q to skończony **zbiór stanów**
- Σ to skończony zbiór **symboli wejściowych** (alfabet)
- δ to **funkcje przejścia** między stanami, $\delta : Q \times \Sigma \to Q$
- $q_0 \in Q$ to **stan początkowy**
- $F \subset Q$ to zbiór **stanów akceptujących** (stanów końcowych)

Definicja: niedeterministyczny automat skończony

Niedeterministyczny automat skończony (NAS) to automat skończony, niedeterministyczny, zdefiniowany poprzez $(Q, \Sigma, \delta, q_0, F)$, gdzie

- Q to skończony **zbiór stanów**
- Σ to skończony zbiór **symboli wejściowych** (alfabet)
- δ to **„funkcje" przejścia** między stanami, $\delta : Q \times \Sigma \to Q', \quad Q' \subset Q$
- $q_0 \in Q$ to **stan początkowy**
- $F \subset Q$ to zbiór **stanów akceptujących** (stanów końcowych)

Jak działają odwzorowania δ w przypadku NAS?

- automat z pojedynczego stanu przechodzi do jednego lub kilku stanów
- oznacza to, że może znajdować się w kilku stanach jednocześnie

23. Automaty z epsilon przejściami, wyrażenia regularne.

Definicja: NAS z ε-przejściami

Niedeterministyczny automat skończony z ε-przejściami (ε-NAS) to automat skończony, niedeterministyczny, zdefiniowany poprzez $(Q, \Sigma, \delta, q_0, F)$, gdzie

- Q to skończony **zbiór stanów**
- Σ to skończony zbiór **symboli wejściowych** (alfabet)
- δ to **„funkcje" przejścia** między stanami, $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \{Q'\} \subset Q$
- $q_0 \in Q$ to **stan początkowy**
- $F \subset Q$ to zbiór **stanów akceptujących** (stanów końcowych)

UWAGA: Σ jest zbiorem symboli mogących pojawić się na wejściu automatu. Co prawda ε formalnie należy do każdego alfabetu, ale w przypadku ε-NAS wyklucza się ten symbol z Σ. Stąd zapis $\Sigma \cup \{\varepsilon\}$ w definicji funkcji przejścia.

Wyrażenia regularne (RE, regexp)

- język opisu wzorców czy wzorce opisujące język?
- są wykorzystywane w zadaniach dopasowania i rozpoznawania łańcuchów, m.in. w wyszukiwarkach, parserach, analizatorach treści...
- technicznie, wyrażenie regularne jest przez aplikację zamieniane na odpowiedni automat
- prowadzi to do (słusznego) podejrzenia, że automaty i RE nawet jeśli nie są sobie równoważne, to mają wiele wspólnego
- konkretne symbole używane do budowanie RE różnią się pomiędzy aplikacjami i systemami, ale podstawowe zasady są niezmienne

Na językach można wykonać trzy podstawowe operacje. Są to:

- **suma**, którą definiuje się jako sumę zbiorów łańcuchów należących do obu języków

$$\forall_a[a \in \mathcal{L} \vee a \in \mathcal{M}] \Leftrightarrow [a \in \mathcal{L} \cup \mathcal{M}]$$

- **konkatenacja**, którą definiuje się jako zbiór łańcuchów utworzonych jako konkatenacje łańcuchów należących do obu języków

$$\forall_{a,b}[a \in \mathcal{L} \wedge b \in \mathcal{M}] \Leftrightarrow [ab \in \mathcal{LM}]$$

- **domknięcie**, które jest definiowane jako suma łańcuchów należących do kolejnych konkatenacji języka z samym sobą (suma potęg języka)

$$\mathcal{L}^* = \bigcup_{i=0}^{\infty} \mathcal{L}^i = \mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2 \cup \cdots = \{\varepsilon\} \cup \mathcal{L} \cup \mathcal{LL} \cup \ldots$$

24. Kompilacja: gramatyka bezkontekstowa, skaner, parser, błędy.

Definicja: gramatyka bezkontekstowa

Gramatyką bezkontekstową G nazywamy czwórkę $G = (V, T, P, S)$, gdzie

- V to zbiór **zmiennych** – każda zmienna jest tożsama z jednym językiem; jedna ze zmiennych jest wyróżniona odpowiadając symbolowi początkowemu S, reszta to pomocnicze klasy łańcuchów
- T to zbiór **symboli końcowych** – jest to niepusty zbiór skończony, zawierający łańcuchy definiowanego języka
- P to zbiór **produkcji** – skończony zbiór reguł pozwalających na rekurencyjne definiowanie języka
- S to **symbol początkowy**

Można to w zwarty sposób zapisać następująco:

$$T \neq \emptyset \qquad S \in V$$
$$T \cap V = \emptyset \qquad P \subset V \times (T \cup V)^*$$

Skaner (analizator leksykalny) zazwyczaj posługuje się językiem regularnym, w którym alfabetem są znaki używane do zapisania kodu.

Generator parserów działa w oparciu o język bezkontekstowy, zdefiniowany na tokenach wygenerowanych wcześniej przez skaner.

25. Cechy programowania deklaratywnego.

wzorowany na językach naturalnych i ludzkich procesach myślowych

nieliniowość wykonania, komenda na końcu kodu może zmodyfikować wywołanie z jego początku

powszechne stosowanie rekurencji w miejsce pętli

definiuje problem (dane i reguły) i poszukuje odpowiedzi na zadane pytanie

Programowanie deklaratywne W przeciwieństwie do programów napisanych imperatywnie, programista opisuje warunki, jakie musi spełniać końcowe rozwiązanie (co chcemy osiągnąć), a nie szczegółową sekwencję kroków, które do niego prowadzą (jak to zrobić). Programowanie deklaratywne często traktuje programy jako pewne hipotezy wyrażone w logice formalnej, a wykonywanie obliczeń jako ich dowodzenie

Programowanie deklaratywne: opisujemy cel, który wykonawca (komputer) ma osiągnąć, w tym rodzaju programowaniu programista opisuje, co komputer ma osiągnąć.

Dwa główne nurty programowania deklaratywnego

- **funkcyjne** – zamiast instrukcji używa się *wyrażeń*, a więc nie ma pojęcia stanu programu; cały algorytm sprowadza się do obliczenia wartości pewnej (zazwyczaj złożonej) funkcji matematycznej; u podstaw programowania funkcyjnego leży rachunek lambda
- **w logice** – zamiast instrukcji używa się *zdań logicznych*, a więc każde działanie sprowadza się do określenia, czy rozważane stwierdzenie jest logicznie prawdziwe, czy fałszywe

26. Protokoły TCP i UDP - porównanie i zastosowanie.

	TCP	UPD
Prędkość	Niska	Wysokoa
Rozmiar	20 bajtów	8 bajtów
Niezawodność	Niezawody	Zawodny
Potwierdzenia	Wysyła potwierdzenie	Brak potiwerdzenia
Kontrola błędów	Tak	Nie
Powtóren wysłanie w przypadku utraty danych	TAK	NIE

UDP w datagramie dodaje tylko 8 bajtów danych sterujących. Bezpołączeniowość protokołu UDP polega na tym, że przed rozpoczęciem procesu komunikacji host źródłowy nie wysyła do hosta

docelowego żadnych informacji zestawiających to połączenie. Zasada jest taka, jeśli urządzenie źródłowe chce rozpocząć transmisję, robi to bez wcześniejszego ustalenia.

Protokół TCP działa w trybie klient – serwer. Serwer oczekuje na nawiązanie połączenia na określonym porcie, a klient inicjuje połączenie do serwera. TCP gwarantuje wyższym warstwom komunikacyjnym dostarczenie wszystkich pakietów w całości, z zachowaniem kolejności i bez duplikatów. Zapewnia to wiarygodne połączenie kosztem większego narzutu w postaci nagłówka i większej liczby przesyłanych pakietów. W tym protokole w celu nawiązania połączenia między dwoma hostami jest stosowana procedura nazwana three-way handshake.

27. Protokół IP.

W normalnych warunkach jego funkcja sprowadza się do wyboru optymalnej trasy i przesyłania nią pakietów. W przypadku wystąpienia awarii, na którymś z połączeń protokół będzie próbował dostarczyć pakiety trasami alternatywnymi (nie zawsze optymalnymi). Protokół IP jest podstawowym protokołem przesyłania pakietów w Internecie.

Protokół IP jest protokołem bezpołączeniowym. Oznacza to, że w celu przesłania pakietów nie jest nawiązywane połączenie z hostem docelowym. Pakiety mogą być przesyłane różnymi trasami do miejsca przeznaczenia, gdzie są następnie składane w całość.

28. Modele sieci komputerowych.

29. Porównanie protokołów IPv4 i IPv6.

14.1 Omówić różnice między protokołami IPv4 i IPv6 oraz opisać ich znaczenie

Główną różnicą pomiędzy protokołami IPv4 i IPv6 jest pula adresowa - IPv4 pozwala na zaadresowanie 2^32 adresów, gdzie IPv6 - 2^128 adresów. W dodatku w przypadku protokołu IPv6 nie jest konieczne korzystanie z NAT oraz DHCP, a także wyeliminowano możliwość kolizji adresów w sieciach prywatnych.

Liczba bitów:

- IPv4: długość 32 bitów, adres podzielony na cztery części 8-bitowe.
- IPv6: długość 128 bitów, adres podzielony na osiem części 16-bitowych.

Sposób adresowania:

- IPv4: numeryczny – poszczególne bity oddzielone kropkami. Same bity w zapisie dziesiętnym.
- IPv6: alfanumeryczny – poszczególne bity oddzielone dwukropkami. Same bity w zapisie szesnastkowym.

Liczba dostępnych adresów:

- IPv4: ok. 4,3 miliarda (wyczerpane).
- IPv6: ok. 340 sekstylionów.

Sposób przydzielania adresu do urządzenia:

- IPv4: ręcznie (przez APIPA lub DHCP).
- IPv6: samokonfiguracja (urządzenie generuje adres, gdy połączy się z siecią dzięki IRDP i NDP).

Sposób konfiguracji:

- IPv4: ręcznie albo przez DHCP.
- IPv6: samokonfiguracja.

IPSec (protokół zapewniający bezpieczeństwo i uwierzytelnianie danych):

- IPv4: nieobowiązkowy, może być w pełni zintegrowany.
- IPv6: obowiązkowy, w pełni zintegrowany.

30. Format pakietu IP (poszczególne pola, zastosowanie).

Wersja typ protokołu IP.

długość nagłówka (HLEN) - długość datagramu wyrażoną jako wielokrotność słów 32 bitowych.

typ usługi (TOS ang. Type-of-Service) - poziom ważności jaki został nadany przez protokół wyższej warstwy.

Długość całego pakietu wyrażona w bajtach.

Identyfikacja - numeru sekwencyjnego bieżącego datagramu.

Znaczniki - flagi wykorzystywane podczas fragmentacji datagramów. Zawierają dwa używane pola: DF, które wskazuje, czy pakiet może być fragmentowany oraz MF, które wskazuje, czy za danym datagramem znajdują się kolejne fragmenty.

Przesunięcie fragmentu - służące do składania fragmentów datagramu.

Czas życia (TTL, ang. Time To Live) - określające liczbę routerów przez które może być przesłany pakiet.

Protokół - 8który z protokołów warstwy wyższej odpowiada za przetworzenie pola Dane

Suma kontrolna nagłówka -

Adres IP nadawcy - 32-bitowe pole z adresem IP nadawcy pakietu

Adres IP odbiorcy - 32-bitowe pole z adresem IP odbiorcy pakietu

Opcje - pole to nie występuje we wszystkich pakietach. Szczegółowe wartości tego pola zostaną omówione na następnym slajdzie.

Uzupełnienie (Wypełnienie) - pole to jest wypełnione zerami i jest potrzebne, żeby długość nagłówka była wielokrotnością 32 bitów

Dane

31. Ethernet.

Ethernet – technika, w której zawarte są standardy wykorzystywane w budowie głównie lokalnych sieci komputerowych. Obejmuje ona specyfikację przewodów oraz przesyłanych nimi sygnałów. Ethernet opisuje również format ramek i protokoły z dwóch najniższych warstw Modelu OSI

Rozmiar pola w bajtach	7	1	6	6	2	46 - 1500	4
Nazwa pola	Preambuła	Znacznik początku ramki	Adres MAC odbiorcy	Adres MAC nadawcy	Długość/Typ	Dane i wypełnienie	Kod kontrolny ramki (FCS)

- **Preambuła** oraz **znacznik początku ramki** – te pola służą do poinformowania urządzenia docelowego, aby przygotował się na odbiór ramki;
- **Docelowy adres MAC**, czyli adres fizyczny odbiorcy ramki;
- **Źródłowy adres MAC**, czyli adres fizyczny hosta wysyłającego dane;
- **Długość / Typ** – pole długość określa wielkość ramki, natomiast **typ** określa, jaki został wykorzystany **protokół warstwy wyższej**, najczęściej jest to **IPv4**;
- **Dane** – to pakiet, który odebrany został z **warstwy sieciowej**. Minimalna wielkość tego pola to **46**, a maksymalna **1500 bajtów**. Jeśli pakiet jest mniejszy niż **46 bajtów**, to dopełnia się
  go, losowymi danymi, tak aby rozmiar całej ramki został zwiększony do wymaganego minimum, czyli do **64 bajtów**.
- **Kod kontrolny ramki** – pole zawierające **sumę kontrolną ramki**, służącą do wykrywania **ewentualnych błędów ramki**. Urządzenie wysyłające dane **oblicza sumę kontrolną** i umieszcza ją w ramce, odbiorca danych, po jej otrzymaniu **również taką sumę oblicza, jeśli obydwie sumy się zgadzają ramka jest akceptowana, jeśli się różnią, ramkę traktuje się jako uszkodzoną i odrzuca**.

Pierowtnie Ethernet był konentrykiem i magistralą

Potem były topologia gwazdy z hubami co wysyłały dane do wszystkich urządzeń w sieci

Potem znów gwiazada potem swicze wysyła do docelowego urządzenia

Pracuje w trybie full-duplex – komunikacja dwókierunkowa

32. Protokoły warstwy aplikacji.

DNS (ang. Domain Name System, port: 53) – protokół używany do odwzorowywania nazw w sieci Internet na adresy IP,

HTTP (ang. Hypertext Transfer Protocol, port: 80) – protokół używany do przesyłania plików tworzących strony WWW,

HTTPS (ang. Hypertext Transfer Protocol Secure, port: 443) – szyfrowana wersja protokołu HTTP, wykorzystująca szyfrowanie TLS,

SMTP (ang. Simple Mail Transfer Protocol, port: 25) – protokół używany do przesyłania wiadomości poczty elektronicznej wraz z załącznikami,

POP3 (ang. Post Office Protocol, , port: 110) — protokół używany do odbioru poczty elektronicznej,

IMAP (ang. Internet Message Access Protocol, port: 143) – protokół używany do odbioru poczty elektronicznej, oprócz funkcji POP3, pozwala również na zarządzanie zdalnymi folderami znajdującymi się na serwerze,

Telnet (ang. Telecommunication Network, port: 23) – protokół używany do emulacji terminala umożliwiający komunikację ze zdalnym urządzeniem,

FTP (ang. File Transfer Protocol, porty: 20, 21) – protokół używany do interaktywnego przesyłania plików pomiędzy systemami,

DHCP (ang. Dynamic Host Configuration Protocol, port: 67) – protokół używany do dynamicznej konfiguracji urządzeń, a dokładniej odpowiedzialny jest za przydzielanie adresów IP, adresu domyślnej bramy i adresów serwerów DNS,

SSH (ang. Secure Shell, port: 22) — protokół używany do emulacji terminala sieciowego zapewniający szyfrowanie połączenia,

33. Infrastruktura klucza publicznego - charakterystyka.

Struktura scentralizowana – na samym wierzchołku znajduje się CA

Struktura zdecentralizowna – peer to peer

Celem infrastruktury kluczy publicznych PKI (Public Key Infrastructure) jest zapewnienie zaufanego i wydajnego zarządzania kluczami oraz certyfikatami.

Infrastruktura klucza publicznego to zbiór osób, zasad, polityk, procedur i systemów komputerowych koniecznych do świadczenia usług szyfrowania i uwierzytelniania za pośrednictwem kryptografii asymetrycznej. W jego skład wchodzą głównie urzędy certyfikacyjne, urzędy rejestracyjne, użytkownicy certyfikatów klucza publicznego (subskrybenci), oprogramowanie oraz sprzęt. Jego podstawowym dokumentem jest certyfikat klucza publicznego. Podstawowymi funkcjami IKP są weryfikacja tożsamości użytkowników, wystawianie i weryfikacja certyfikatów oraz wymiana kluczy kryptograficznych.

34. Kryptografia symetryczna oraz asymetryczna - charakterystyka.

Kryptografia symetryczna – rodzaj kryptografii, w której wykorzystywany jest jeden klucz zarówno do szyfrowania jak i deszyfracji danych. Algorytm symetryczny jest znacznie szybszy niż algorytm asymetryczny, dzięki czemu może być wykorzystywany w szyfrowaniu danych na komputerze w czasie rzeczywistym. Algorytm symetryczny zapewnia również podobne bezpieczeństwo jak algorytm asymetryczny przy znacznie krótszej długości klucza. Główną słabością algorytmów symetrycznych jest konieczność bezpiecznej dystrybucji klucza

Kryptografia asymetryczna – rodzaj kryptografii, w której klucz szyfrujący, zazwyczaj dostępny publicznie jest tworzony oddzielnie od klucza deszyfrującego, który jest tajny. Dzięki zastosowaniu klucza publicznego z algorytmów asymetrycznych korzysta się często przy systemach, do których potrzebuje dostępu kilku użytkowników. Klucze asymetryczne stosuje się również w podpisach cyfrowych jako środek uwierzytelniający.

35. Bezpieczeństwo sieci w odniesieniu do warstw modelu TCP/IP.

36. Metody kontroli dostępu w systemach IT.

AAA- > uwierzytelnianie, autoryzacja i kontrola dostępu

Identyfikajca – czym będziemy się identyfikować w prcoesie AAA (nazwa użytkownika, nr telefonu itp.)

Authetication – stiwerdzenie że dany użytkownik jest jaktycznie tym używkownikiem (testowanie czy dane w idetyfikacji zostały poprawnie podane)

Authorization – dajemy dostęp do pewnych zasobów

Monitorowanie – monitorowanie kto co używał

Odpowiedzalność udowodnienie że dana osoba wykonywała daną czynność

- Types of identity:
  - User ID, Account Number, User Name, etc.
  - Unique, standard naming convention, non-descriptive of job function, secure & documented issuance process.

- Types of authentication:
  - Something the subject knows – Password, pass phrase, or PIN.
  - Something the subject has – Token, smart card, keys.
  - Something the subject is – Biometrics: fingerprints, voice, facial, or retina patterns, etc.

MAC – 4 poziomy dostępu : TOP SECRET, SECRET< CLASSIFED, UNCLASSIFED Przypisywane są one użytkownikom i obiektom

DAC właściciel katalgu ma dostęp do wszystkich pod katalogów

Lista dostęp do plików jak w linuxie

37. Atrybuty bezpieczeństwa informacji.

Integralnośc -to właściwość, która gwarantuje, że dane nie zostaną zmodyfikowane w sposób inny niż przez celowe działanie danej osoby

Dostępność- umożliwienie odpowiednim osobom dostępu do danych i dokonywanie na nich operacji.

Poufność - osoby upoważnione, mające dostęp do danych

niezaprzeczalność – właściwość umożliwiająca weryfikację, że strony biorące udział w wymianie informacji rzeczywiście brały w niej udział.

38. Charakterystyka modelu OSI i TCP/IP.

Model ISO-OSI składa się z 7 warstw:

Aplikacji - odpowiada za komunikację pomiędzy użytkownikiem, a pozostałymi warstwami modelu OSI.

Prezentacji - odpowiada za konwersję danych ze strumienia sieciowego do odpowiednich plików i żądań. Odpowiada również za szyfrowanie i deszyfrowanie pakietów.

Sesji - zarządza sesjami pomiędzy dwoma węzłami, pomiędzy którymi odbywa się komunikacja.

Transportowej - segmentuje dane i przesyła je do określonego węzła, a także wykrywa i naprawia błędy występujące podczas przesyłania.

Sieciowej - rozpoznaje, jakie trasy łączą poszczególne węzły i decyduje, jakie informacje należy przesłać do danego węzła.

Łącza danych - nadzoruje przekazywane przez warstwę fizyczną dane, oraz ma możliwość zmiany parametrów warstwy fizycznej.

Fizycznej - określa ona wszystkie składniki niezbędne do obsługi elektrycznej, radiowej i optycznej części sieci.

39. Rodzaje i przykłady nagłówków HTTP.

Nagłówki HTTP to wszelkie komendy używane do komunikacji między przeglądarką WWW (klientem) a serwerem. Nagłówki są to właściwości żądania i odpowiedzi przesyłane wraz z samą wiadomością. Służą one przede wszystkim do sterowania zachowaniem serwera oraz przeglądarki przez nadawcę wiadomości.

Date Zawiera datę mówiącą o czasie wygenerowania żądania/odwiedz

Host Zawiera domenę, do której wysyłane jest żądanie

Connection Zawiera informacje na temat połączenia pomiędzy klientem a serwerem

Accept Klient informuje serwer o tym jaki format jest w stanie zrozumieć, może to być na przykład JSON: application/json

User-Agent Identyfikuje klienta czyli przeglądarkę lub aplikację wysyłającą żądanie.

Server Serwer informuje klienty jakiego oprogramowania używa do obsługi odpowiedzi

WWW-Authenticate Określa sposób w jaki ma zostać przeprowadzone uwierzytelnienie użytkownika.

Niestandardowe:

Set-Cookie Nagłówek służący do ustawienia ciasteczka

Coockie W tym nagłówku przesyłane są wszystkie ciasteczka ustawione w przeglądarce.

Refresh Ustawia automatyczne przekierowanie w przeglądarce na podany adres po określonym czasie

40. Protokół WebSocket.

Zadaniem protokołu jest utworzenie zamkniętego kanału pomiędzy stronami uczestniczącymi w komunikacji, w ramach którego możliwe jest wysyłanie danych w dowolnej kolejności, niezależnie od innych uczestników transmisji. Przed przystąpieniem do wymiany danych, w trakcie inicjacji połączenia następuje tzw. uścisk dłoni (ang. handshake). Żądanie WebSocket zawiera kilka kluczowych nagłówków odróżniających je od zwykłej wiadomości http:

Connection: Upgrade oraz Upgrade: websocket informują serwer o chęci przeniesienia transmisji na protokół WebSocket;

Sec-WebSocket-Protocol zawiera listę możliwych do wykorzystania, obsługiwanych przez klienta protokołów warstwy aplikacji. Sec-WebSocket-Version określa wersję tych protokołów; • Sec-

WebSocket-Key jest kluczem służącym zapewnieniu integralności

W celu sprawdzenia poprawności danych, serwer zawiera w odpowiedzi nagłówek Sec-WebSocket-Accept, którego wartością jest zakodowany hasz. W celu zakończenia komunikacji, jedna ze stron zaprzestaje wymiany danych i wysyła ramkę z sekwencją kontrolną zawierającą informację o rozpoczęciu zamykania połączenia

41. Serwer zdarzeniowy, a wielowątkowy. Charakterystyka i porównanie.

42. Metody rozwiązywania rekurencji. Rekurencje Flawiusza i wieża w Hanoi.

Wierza Hanoi – Mamy wierze składjącyą się z krążków o różnej średnicy, którą trzeba przenieść na koniec miejsc do odkładania. Nie można przekładać krążka o większej średnicy na krązek o mniejszej średnicy, oraz nie można przekładać kilku krążków jednocześnie.

Flawiusz - na okręgu ustawiamy n obiektów, następnie eliminujemy co k-ty obiekt, tak długo, aż zostanie tylko jeden. Należy wskazać obiekt, który pozostanie.

Ogólna strategia przy rozwiązywaniu równań rekurencyjnych metodą repertuaru może być opisana następująco:

- Uogólnij równanie rekurencyjne uzależniając je od parametrów
- Ustal parametry dla których znasz rozwiązania
- Połącz przypadki szczególne aby znaleźć rozwiązanie ogólne.

Metoda czynnika sumacyjnego

Tylko dla tych rekurencji i dla tych, które da się sprowadzić do tej postaci

Rozważmy rekurencję

$$\begin{cases} a_o T_o &= c_o, \\ a_n T_n &= b_n T_{n-1} + c_n, \ \text{dla} \ n \geq 1, \end{cases}$$

43. Algorytmy Euklidesa. Algorytmy faktoryzacji.

Algorytm Euklidesa służy do obliczania NWD (największego wspólnego dzielnika) dwóch liczb całkowitych.

Dzielimy z resztą liczbę a przez liczbę b

jeżeli reszta jest równa 0, to NWD(a,b)=b

jeżeli reszta jest różna od 0, to przypisujemy liczbie a wartość liczby b, liczbie b wartość otrzymanej reszty, a następnie wykonujemy ponownie punkt 1.

Dzielimy z resztą liczba a przez b. Do momentu uzyskania 0 w tedy ostani dzielnik liczby jest wynikiem. Jeśli następuje reszta to wyniki dzielenia dzielimy przez tą resztę

zaufanie do RSA opiera si¦ na nieznajomo±ci algorytmu faktoryzacji,
bo gdyby±my majic n potra_li policzy¢ p i q, potra_liby±my z³ama¢
RSA

44. Metody reprezentacji grafów w komputerze.

Lista sąsiedztwa

W tej reprezentacji graf G=(V,R) przedstawiamy jako tablicę list tab taką, że lista tab[a] zawiera sąsiadów wierzchołka a.

By przedstawić krawędź wystarczy użyć tablicy dwuelementowej, zawierającej dwie liczby oznaczające wierzchołki lub tablicy obiektów, zawierającą informację o tym, które z wierzchołków są incydentne z daną krawędzią.

Macierz Sąsiedztwa

Dla grafu z |V||V|vertical bar, V, vertical bar wierzchołkami macierzą sąsiedztwa (ang. adjacency matrix) nazywamy macierz o wymiarach |V| \cdot |V||V|·|V|vertical bar, V, vertical bar, dot, vertical bar, V, vertical bar złożoną z zer i jedynek, w której wartość w iii–tego wiersza i jjj–tej kolumny jest równa 1 wtedy i tylko wtedy gdy krawędź (i,j)(i,j)left parenthesis, i, comma, j, right parenthesis występuje w grafie.

45. Droga i cykl Eulera. Droga i cykl Hamiltona.

jeśli istnieje zamknięta (Ten sam wiechołek początkowy i końcowy) ścieżka zawierająca każdą krawędź grafu G, wtedy cyklem Eulera nazywamy właśnie taką ścieżkę.

Graf spójny G jest hamiltonowski, jeśli istnieje zamknięta ścieżka przechodząca przez każdy wierzchołek tylko jeden raz i zawierająca wszystkie wierzchołki grafu G.

46. Drzewo spinające graf.

Drzewo spinające grafu G = (V, E) jest to podgraf spinający będący drzewem.

47. Standardowe metodyki procesu wytwórczego oprogramowania.

W opracowywaniu kaskadowym wynik jednego etapu staje się wejściem dla następnego, przy czym nie ma powrotu do poprzedniego etapu. Etapy w Waterfall'u: Analiza, Projektowanie, Implementacja, Testowanie, Wdrożenie i utrzymanie. Jest to zdecydowanie gorsza metodyka względem metody zwinnej. Waterfall jest nie odporny na zmiany, które często się zdarzają. Przez takie zmiany należy wykonać cały proces od samego początku.

48. Metodyki zwinne (agile).

Metodyki zwinne są niejako odpowiedzą na metodę wterfalla. Trakutją one cały proces wytwarzania oprogramowania bardziej elastycznie przez to klient może dokonywać zmian w każdym etapie procesu wytwarzania, które jest podzielone na kilka częśći. W tej metodyce istnieją tak zwane spriny, w których trwa określoną ilość czasu a na końcu niego powinia powstać mniejsza część oprogramowania. Na początku sprintu przypisywane zadania do członków zespołu. Zespoł powinien być samoorganizujący się i nie powinien posiadać takowego szefa, mówiącego co i jak kto ma robić. Osobą, która sprawdza czy zespół trzyma się zasad metod zwinnych jest scrum master. Zespół również spotyka się na codziennych spotkaniach, w których omawia swoje postępy oraz probmemy jeśli występują.

49. Metody testowania oprogramowania.

Testowanie ma na celu weryfikację oraz walidację oprogramowania.

a) testy integracyjne pozwalają sprawdzić jak współpracują ze sobą różne komponenty oprogramowania. Obecnie rzadko mamy do czynienia z monolitycznymi aplikacjami. Są one raczej tworzone modułowo, dlatego należy sprawdzić, czy wszystko razem działa poprawnie b) testy systemowe dotyczą działania aplikacji jako całości. Zazwyczaj na tym poziomie testujemy różnego rodzaju wymagania niefunkcjonalne, takie jak – szybkość działania, bezpieczeństwo, niezawodność, dobrą współpracę z innymi aplikacjami i sprzętem.

a) testy funkcjonalne – znane również jako testy czarnej skrzynki. Osoba testująca nie ma dostępu do informacji na temat budowy programu, który testuje. Wykonując testy nie opiera danych testowych na budowie wewnętrznej programu, lecz na założeniach funkcjonalnych, jakie powinien spełniać program zgodnie z dokumentacją

b) testy regresyjne – mają na celu sprawdzenie wpływu nowych funkcjonalności na działanie systemu

c) testy akceptacyjne z udziałem klienta – wykonywane w celu sprawdzenia na ile oprogramowanie działa zgodnie z wymaganiami klienta

d) testy dokumentacji, których celem jest wykrycie niespójności i niezgodności w dokumentacji analitycznej, technicznej oraz dokumentacji użytkownika, sporządzonej w ramach realizowanego projektu informatycznego

e) testy użyteczności, których celem jest weryfikacja interfejsu użytkownika w zakresie przystępności, wygody, szybkości oraz zgodności z oczekiwaniami przyszłych użytkowników.

50. Walidacja i weryfikacja oprogramowania.

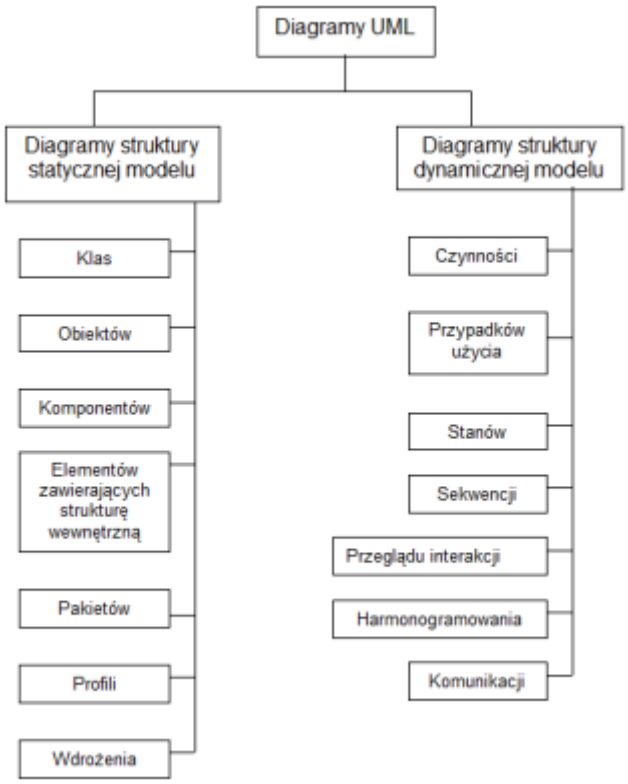
Weryfikacja oprogramowania - testowanie zgodności systemu z wymaganiami zdefiniowanymi w fazie określenia wymagań.

Walidacja (atestowanie) oprogramowania - ocena systemu lub komponentu podczas lub na końcu procesu jego rozwoju na zgodności z wyspecyfikowanymi wymaganiami.

Weryfikacja oprogramowania jest procesem sprawdzenia, czy wytwarzane oprogramowanie jest zgodnie z wytycznymi zasad programowania, z zastosowaniem najbardziej odpowiednich metod, języka programowania i najbardziej wydajnych algorytmów. Weryfikacja dokonywana jest podczas testów systemowych pojedynczego produktu, oraz testów integracyjnych, po wprowadzeniu produktu do istniejącego już i działającego środowiska informatycznego. Weryfikację z reguły przeprowadza się na dwa sposoby – statyczny oraz dynamiczny. Testowanie statyczne oznacza inspekcję kodu. Inspekcja kodu wykonywana jest przed skompilowaniem programu, i ma na celu wykrycie, czy tekst programu jest spójny, i czy nie zawiera ewidentnych błędów lub/i zbędnych fragmentów kodu mogących mieć wpływ na wydajność systemu. Weryfikacja dynamiczna przeprowadzana jest już po uruchomieniu skompilowanego programu, z użyciem testowych danych wejściowych, i kontroluje zachowanie systemu widoczne dla użytkownika.

Walidacja oprogramowania jest procesem sprawdzania czy oprogramowanie jest zgodne z oczekiwaniami użytkownika (potwierdzenie w sposób udokumentowany i zgodny z założeniami, że procedury, procesy, urządzenia, materiały, czynności i systemy rzeczywiście prowadzą do zaplanowanych wyników). Walidacja to sprawdzenie, czy produkt końcowy zawiera w sobie wszystkie wymagane przez klienta funkcjonalności, czy działa stabilnie i prawidłowo komunikuje się z pozostałymi systemami bądź użytkownikami. W walidacji produktu największe znaczenie mają analiza dokumentacji oraz testy akceptacyjne przeprowadzane przez użytkownika końcowego. W efekcie sprowadza się to do kontroli oprogramowania pod względem formalnym.

51. Diagramy UML (przypadków użycia, klas, aktywności, sekwencji, stanów, obiektów, wdrożenia).



Klas - Diagram klas obrazuje pewien zbiór klas, interfejsów i kooperacji oraz związki między nimi.

Obiektów - przedstawia egzemplarze elementów z diagramu klas. zawiera na ogół: obiekty, wiązania, notatki, ograniczenia, pakiety, podsystemy, klasy.

Komponentów – pokazują podział systemów programowych na mniejsze podsystemy. to wymienialny, wykonywalny fragment systemu, z ukrytymi szczegółami implementacyjnymi

Elementów –

Pakietów – przedstawiają pogrupowane elementy w pakiety.

Profilii –

Wdrożenia –

Czynności - modelowania przepływu sterowania między wykonywanymi czynnościami.

Przypadki użycia - wymagania stawiane systemowi przez użytkowników. Stanów- Diagramy stanów obrazują sposób, w jaki elementy modelu zmieniają w czasie swoje własności w odpowiedzi na różnego rodzaju zdarzenia i interakcje. Podstawowym elementem tego typu diagramów jest ikona stanu, która może być przedstawiona w sposób uproszczony, poprzez podanie nazwy stanu, lub w sposób szczegółowy, z dodatkowym wyszczególnieniem wartości atrybutów oraz podstawowych, aktualnie wykonywanych operacji i zdarzeń,

Sekwencji – pozwalają modelować wzajemną interakcję obiektów w funkcji czasu jej trwania. Interakcja traktowana jest jako ciąg zdarzeń występujących w czasie w określonym porządku. Diagram przebiegu składa się z przedstawianych w standardowej postaci obiektów, komunikatów oraz osi czasu (tzw. linii życia).

Przeglądu interakcji

Harmonogromowania

Komunikacji - umożliwiają pokazanie kolejności wysyłania komunikatów. Kolejność ta obrazowana jest poprzez numery umieszczane na początku etykiet zawierających nazwy i ew. listę argumentów (sygnaturę) odpowiedniego komunikatu.

52. Wzorce projektowe programowania obiektowego.

SOLID -Robert Cecil Martin zaproponował 5 zasad dobrego programowania, nazywane zasadami lub regułami SOLID, które stały się podstawą pisania dobrego, czystego i czytelnego kodu.

S – Zasada pojedynczej odpowiedzialności

O – Otwarte na rozbudowę zamknięte na modyfikację (Przy zmianie wymaga nie stary kod powinien działać)

L – Zasada podstawiania – możliwe jest podstawianie klas pochodnych za bazową

I – Zasada segregacji interfejsów – lepiej mieć kilka interfejsów niż jeden duży

D - Zasada odwracalności zależności – wysokopoziomowe moduły nie powinny zależeć o modółów niskopoziomowych.

Wzorce Kreacyjne - opisujące proces tworzenia nowych obiektów.

Strukturalne - opisujące struktury powiązanych ze sobą obiektów

Czynnościowe - opisujące zachowanie i odpowiedzialność współpracujących ze sobą obiektów.

53. Wzorce architektoniczne.

Określają ogólną strukturę danego systemu informatycznego, elementy z jakich się składa, zakres funkcji realizowany przez dany element jak również zasady komunikacji pomiędzy poszczególnymi elementami. Korzyści ze stosowania wzorców architektonicznych

- niezależność logiki od sposobu wyświetlania danych,

- niezależność kodu od technologii, w której wykonana jest warstwa prezentacji,

- łatwą zamianę GUI (brak sztywnych powiązań między GUI a dziedziną problemu).

MVC - odseparowanie logiki warstwy opisującej dziedzinę problemu od warstwy prezentacji.

MVVM

54. Wielowarstwowa organizacja oprogramowania komputera.

55. Procesy, zasoby i wątki.

56. Planowanie przydziału procesora, priorytety, wywłaszczanie oraz planowanie.

57. Zarządzanie pamięcią operacyjną.

58. Problem zakleszczenia, algorytm Bankiera.

Zakleszczenie (deadlock):

- Sytuacja, w której (co najmniej) dwa wątki/procesy czekają na siebie nawzajem. W prostym przypadku dwóch wątków:
  - ➔ pierwszy wątek czeka na zakończenie działania drugiego,
  - ➔ drugi wątek czeka na zakończenie pierwszego.
- Zjawisko występuje często przy współzawodnictwie o równoczesny dostęp do zasobów np. plików na dysku, rekordów w bazie danych.

Wywłaszczanie można zrealizować na przykład tak, że proces żądający zasobu, który nie jest obecnie dostępny, musi zwrócić wszystkie posiadane przez siebie zasoby. Następnie jest on umieszczamy w kolejce oczekiwania na zasób, którego zażądał, i zasoby, które przymusowo zwolnił.

Algorytm bankiera – algorytm służący do alokacji zasobów w taki sposób, aby uniknąć zakleszczeń

59. Model relacyjny baz danych i języki zapytań.

dane grupowane są w relacje, które reprezentowane są przez tabele. Relacje są pewnym zbiorem rekordów o identycznej strukturze wewnętrznie powiązanych za pomocą związków zachodzących pomiędzy danym

60. Model obiektowo-relacyjny baz danych, inne modele danych.

Model hierarchiczny

W tym modelu przechowywane dane są zorganizowane w postaci drzewa. Informacja jest zawarta w dokumentach oraz w strukturze drzewa (podobnej do drzewa folderów na dysku komputera).

Model sieciowy

Połączenia między dokumentami tworzą sieć. Informacja jest zawarta w dokumentach oraz w przebiegu połączeń sieci.

Model obiektowy

Model obiektowy łączy cechy programów komputerowych tworzonych w językach programowania obiektowego z cechami aplikacji bazodanowych. Obiekt w bazie reprezentuje obiekt w święcie rzeczywistym

Model relacyjny

W relacyjnym modelu baz danych informacja jest zapisywana w tabeli w formie wierszy. Tabele tworzą między sobą powiązania zwane relacjami.

61. Składnia podstawowych zapytań języka SQL.

SELECT -- określanie kształtu wyniku, selekcja pionowa (kolumn)

FROM -- określenie źródła (źródeł) i relacji między nimi

WHERE -- filtracja rekordów

GROUP BY -- grupowanie rekordów

HAVING -- filtrowanie grup

ORDER BY -- sortowanie wyniku

62. Projektowanie baz danych oraz model związków encji.

Cały proces projektowania bazy danych możemy podzielić na kilka etapów:

planowanie bazy danych,

Tworzenie diagramu ERD,

Normalizacja diagramu ERD

1NF - mówi o atomowości danych. Każde pole przechowuje jedną informację.

2NF - każda tabela powinna przechowywać dane dotyczące tylko konkretnej klasy obiektów.

3NF - kolumna informacyjna nie należąca do klucza nie zależy też od innej kolumny informacyjnej, nie należącej do klucza

Encją jest każdy przedmiot, zjawisko, stan lub pojęcie, czyli każdy obiekt, który potrafimy odróżnić od innych obiektów (na przykład: osoba, samochód, książka, stan pogody)

Encje podobne do siebie (opisywane za pomocą podobnych parametrów) grupujemy w zbiory encji. Projektując bazę danych, należy precyzyjnie zdefiniować encje i określić parametry, przy użyciu których będą opisywane

W modelu ER związek łączy encje. Związek z każdego końca posiada krótki opis ułatiwający interpertację związku.

Rodzaje związków:

1 do 1

1 do wielu

Wiele do wielu

63. Problemy indeksowania baz danych, rodzaje indeksów, indeksy typu B+ drzewo.

Problem z przeszukiwaniem baz danych polega na tym, że… tabele w MySQL (a także w innych RDBMS) nie są posortowane według kolumn, dzięki którym wyciągamy odpowiednie dane.

Indeks to struktura, która ma przyspieszyć wyszukiwanie danych. Indeks definiowany jest dla atrybutów, które nazywamy kluczami indeksu lub kluczami wyszukiwania. Indeksowanie jest sposobem sortowania wielu rekordów na wielu polach. Utworzenie indeksu na polu w tabeli tworzy inną strukturę danych, która przechowuje wartość pola i wskaźnik do rekordu, do którego się odnosi. Ta struktura indeksu jest następnie sortowana, umożliwiając wyszukiwanie binarne na to.

. Plik ten zawierałby rekordy odpowiadające poszukiwanym wartościom pierwszych rekordów w poszczególnych blokach pliku danych. Rekordy w dodatkowym pliku miałyby postać: , a plik

dodatkowy byłby uporządkowany według wartości poszukiwanych. Taki plik dodatkowy nazywa się indeksem

Wskazania do danych:

Indeks gęsty to taki, który posiada wpis dla każdej wartości klucza wyszukiwania.

Indeks rzadki posiada wpisy tylko dla niektórych wartości

Indeks podstawowy (primary index) – założony na atrybucie porządkującym unikalnym

Indeks zgrupowany (clustering index) – założony na atrybucie porządkującym nieunikalnym

Indeks wtórny (secondary index) – założony na atrybucie nieporządkującym

Liczba poziomów:

Jednopoziomowe dla pliku danych jest tworzony tylko jeden indeks (plik indeksu).

Wielopoziomowe, dla pliku danych tworzy się indeks (jak w przypadku pierwszym). Dodatkowo, do indeksu tworzy się dodatkowy indek

64. Przetwarzanie transakcyjne OLTP (On-Line Transaction Processing).

OLTP (Online Transaction Processing) to system przetwarzania danych polegający na wykonywaniu wielu transakcji odbywających się jednocześnie, co ma miejsce na przykład w bankowości internetowej, Transakcje te (tradycyjnie określane jako transakcje gospodarcze lub finansowe) są rejestrowane i zabezpieczane w taki sposób, aby przedsiębiorstwo mogło w każdej chwili uzyskać do nich dostęp

System OLTP	System OLAP
Umożliwia wykonywanie w czasie rzeczywistym dużej liczby transakcji bazodanowych przez dużą liczbę osób	Zazwyczaj wykonuje zapytania dotyczące wielu rekordów (nawet wszystkich rekordów) w bazie danych do celów analitycznych
Wymaga błyskawicznych czasów reakcji	Nie wymaga błyskawicznych czasów reakcji (reaguje o rząd wielkości wolniej niż system OLTP)
Często modyfikuje małe ilości danych i zazwyczaj dokonuje tyle samo odczytów, co zapisów danych	Nie modyfikuje danych w żadnym stopniu, występuje tu zdecydowana przewaga operacji odczytu
Używa indeksowanych danych w celu skrócenia czasu odpowiedzi	Przechowywanie danych w formacie kolumnowym, aby zapewnić łatwy dostęp do dużej liczby rekordów
Wymaga częstego lub równoczesnego wykonywania kopii zapasowych bazy danych	Wymaga znacznie rzadszego tworzenia kopii zapasowych bazy danych niż system OLTP
Wymaga stosunkowo niewiele miejsca do przechowywania danych	Zazwyczaj ma duże wymagania dotyczące miejsca do przechowywania danych, ponieważ ma duże ilości danych historycznych
Zazwyczaj obsługuje proste zapytania dotyczące tylko jednego lub kilku rekordów	Obsługuje złożone zapytania obejmujące dużą liczbę rekordów

Transakcja to ciąg operacji do wspólnego niepodzielnego wykonania. Współbieżne wykonywanie transakcji wymaga zachowania własności ACID (Atomicity, Consistency, Isolation, Durability) – cechy transakcji:

- Niepodzielności („wszystko-lub-nic") – transakcja nie może być wykonana częściowo.
- Integralności/Spójności – po zatwierdzeniu transakcji muszą być spełnione wszystkie warunki poprawności nałożone na bazę danych, po wykonaniu transakcji system będzie spójny, czyli nie zostaną naruszone zasady integralności.
- Izolacji – efekt równoległego wykonania dwu lub więcej transakcji musi być szeregowalny, jeśli dwie transakcje wykonują się współbieżnie, to zwykle (w zależności od poziomu izolacji) nie widzą wprowadzanych przez siebie zmian.
- Trwałości – po udanym zakończeniu transakcji jej efekty na stałe pozostają w bazie danych, system potrafi uruchomić się i udostępnić spójne, nienaruszone

i aktualne dane zapisane w ramach zatwierdzonych transakcji (np. po nagłej awarii zasilania).

Transakcja- Umożliwiają one współbieżny dostęp do zawartości bazy danych. Możliwość używania tego samego zasobu przez kilka osoób jednoczenie.

Transakcja składa się z 3 etapów: rozpoczęcia, wykonania zamknięcia. Powinny one trwać jak najkrócej. Każdy etap jest zapisywany (logi). Punkty pośrednie (save points) – zapamiętany w systemie etap transakcji, do którego można powrócić bez konieczności anulowania wykonanych działań.

65. Różnice pomiędzy obsługą zdarzeń w przerwaniach sprzętowych a obsługą zdarzeń w pętli programowej.

Przerwania wewnętrzne wykorzystywane są do obsługi komunikacji szeregowej, odliczania czasu, zliczania impulsów itp.

Przerwania zewnętrzne mogą być wykorzystywanie do komunikacji z różnymi układami podłączonymi do układu.

Są to zdarzenia, które mają na celu przerwanie wykonywania głównego programu i uruchomienie specjalnej funkcji obsługi przerwania. Gdy opcja ta obsłuży przerwanie, nastąpi powrót do głównego programu i wznowienie jego wykonywania od miejsca, w którym został przerwany. Przerwania mogą być wewnętrzne lub zewnętrzne. Pierwsze z wymienionych pochodzą od wewnętrznych części mikrokontrolera, natomiast drugie są generowane przez urządzenia zewnętrzne sterowane za pomocą mikrokontrolera.

Wymaga na pewno zmiany w rejestrze IE od licznika lub od zewnętrznego wyjścia

m. W wyniku tego zdarzenia jest ustawiana flaga zgłoszenia przerwania, IE

66. Stosowalność systemów opartych o mikrokontrolery vs stosowalność typowych komputerów (stacjonarnych i laptopów).

Mikrokontroler - do sterowania większością urządzeń elektronicznych w podstawowym zakresie. Często mikrokontrolery są montowane w silnikach samochodowych lub w zabawkach, w sprzęcie RTV i AGD, w przemysłowych układach automatyki, w podzespołach komputerowych, czy w układach kontrolno-pomiarowych.

Komputery wykonują większość zadań gdzie mikrokontrolery stawiają na konkretne zadnaia wykoywane cały czas

67. Dekoder, multiplekser i demultiplekser: budowa, zasada, działania, przeznaczenie/zastosowanie.

Multiplekser Posada 2^n wejść danych i n wejść adresowych. Za pomocą wejść zaresowych określamy które wejścia danych ma być przepisane na wyjściu. Budowa: Wejścia danych są podłączone do bramek AND które z kolei są podłączone do bramki NOT XOR. Wejścia adresowe również są podłączone do bramek AND. chcemy skorzystać ze znacznie większej liczby portów, niż jest dostępna w wybranym mikrokontrolerze.

Demultiplekser – Posiada jedno wejście które jest przepisywane jednego z 2^N wyjść sterowanych za pomocą n wejść adresowych. Bydowa jest podobna Wejścia adresowe są podłączone do bramek and

Dekoder – posiada n wejść i 2^n wyjść. Polega na zamianie kodu binarnego na kod 1 z 2^n Budowa to zazwyczaj bramki and i not

68. Podstawowe układy budujące system mikroprocesorowy i sposób wymiany informacji pomiędzy nimi.

W skład każdego systemu mikroprocesorowego wchodzi mikroprocesor, pamięć operacyjna (dane i program), układy wejścia wyjścia oraz oprogramowanie (rys. 1). Mikroprocesor pobiera kolejne instrukcje z pamięci programu, następnie je dekoduje i wykonuje. Adres komórki pamięci, która zostanie odczytana (z której zostanie pobrana instrukcja) przekazywany jest przez magistralę adresową A, odczytywana instrukcja wystawiana jest przez pamięć na magistralę danych D. Mikroprocesor informuje pomięć o tym, że ma zostać dokonany odczyt pamięci generując odpowiednie przebiegi na magistrali sterującej C. Podobnie wygląda odczyt danej z pamięci danych lub odczyt z układów wejścia wyjścia. O tym jaki zasób (pamięć programu, pamięć danych, układy wejścia wyjścia) będzie odczytywany decyduje to co pojawi się na magistrali sterującej C.
W przypadku, kiedy mikroprocesor zapisuje do pamięci danych lub układów wejścia

wyjścia na magistralę adresową wystawia adres A, na magistralę danych wystawia daną D, która ma zostać zapisana pod adres A, a następnie generuje odpowiednie przebiegi na magistrali sterującej C, które spowodują zapis danej D pod adres A, do zasobu określonego albo bezpośrednio przez stan magistrali sterującej C albo określonego przez wartość adresu (dekoder adresów) na magistrali adresowej A. Dekoder adresów w zależności od wartości adresu daje dostęp do pamięci programu, albo do pamięci danych