

Jak któreś pytanie jest wasze to sprawdźcie czy dobrze i ewentualnie coś dodajcie.

1. Algorytm. Własności algorytmu. Prezentacja algorytmu.

Algorytm – jest skończonym, uporządkowanym ciągiem jasno zdefiniowanych czynności, koniecznych do wykonania postawionego zadania.

Cechy algorytmów:

- poprawność (algorytm daje oczekiwane wyniki),
 - jednoznaczność (zawsze daje te same wyniki przy takich samych danych wejściowych),
 - skończoność (wykonuje się w skończonej liczbie kroków),
 - sprawność (czasowa - szybkość działania i pamięciowa)
- posiadanie wejścia i wyjścia (wejście oznacza zwykle pewne dane pobierane przez algorytm w celu ich przetworzenia, wyjście odnosi się do wyniku działania algorytmu)
- wykonalność (polecenia zawarte w algorytmie są wykonywalne, tzn. dostępne, a pisząc algorytm wystarczy się nimi tylko posłużyć.)

2. Złożoność algorytmów.

Złożoność algorytmu rozumiemy jako ilość niezbędnych zasobów do wykonania algorytmu. Złożoność dzielimy na czasową oraz pamięciową. W przypadku złożoności czasowej, z reguły wyróżnimy pewną operację dominującą, a czas będziemy traktować jako liczbę wykonanych operacji dominujących. Złożoność algorytmu może być rozumiana w sensie złożoności najgorszego przypadku lub złożoności średniej. Złożoność najgorszego przypadku nazywamy złożonością pesymistyczną - jest to maksymalna złożoność dla danych tego samego rozmiaru n . $g(n) = n$ liniowa, $g(n) = n^2$ kwadratowa, gdy $g(n)$ jest wielomianem to złożoność wielomianowa. Złożoność pamięciowa jest miarą ilości wykorzystanej pamięci. Jako tę ilość najczęściej przyjmuje się użytą pamięć maszyny abstrakcyjnej. Możliwe jest również obliczanie rozmiaru potrzebnej pamięci fizycznej wyrażonej w bitach lub bajtach.

3. Problemy sortowania. Przykłady algorytmów sortowania i ich złożoność.

Sortowanie – polega na uporządkowaniu zbioru danych względem pewnych cech charakterystycznych każdego elementu tego zbioru.

Złożoność algorytmów sortowania jest określana jako liczba wykonywanych porównań i zamian elementów, wyrażona w zależności od liczby porządkowanych elementów. Złożoność najlepszych algorytmów sortowania jest proporcjonalna do $n \log n$, gdzie n jest liczbą porządkowanych elementów.

Przykładowe algorytmy sortowania

- sortowanie kubełkowe (ang. bucket sort) $O(n^2)$
- sortowanie bąbelkowe (ang. bubblesort) $O(n^2)$
- sortowanie przez wstawianie (ang. insertion sort) $O(n^2)$
- sortowanie przez wybieranie (ang. selection sort) $O(n^2)$
- sortowanie przez scalanie (ang. merge sort) $O(n \log n)$
- sortowanie szybkie (ang. quicksort) $O(n \log n)$
- sortowanie przez kopcowanie (ang. heapsort) $O(n \log n)$
- sortowanie przez zliczanie (ang. counting sort) $O(n + k)$

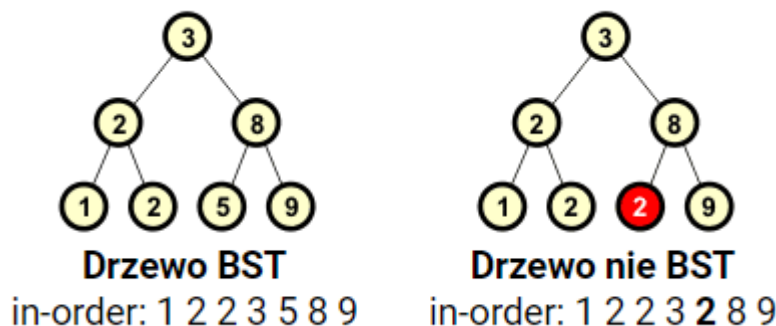
Sortowanie bąbelkowe

Polega na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności, jeżeli zaburza ona porządek, w jakim się sortuje tablicę. Sortowanie kończy się, gdy podczas kolejnego przejścia nie dokonano żadnej zmiany.

4. Drzewa poszukiwań binarnych. Sposób wykonywania na nich podstawowych operacji (dodawanie, wyszukiwanie, usuwanie).

Drzewo poszukiwań binarnych (ang. Binary Search Tree) jest drzewem binarnym, w którym każdy węzeł spełnia reguły:

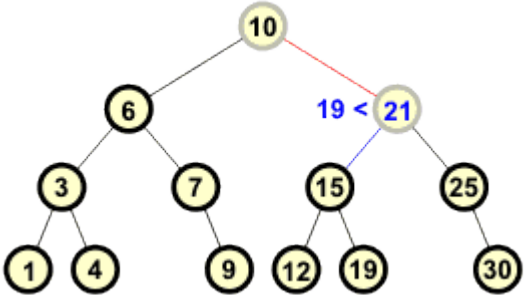
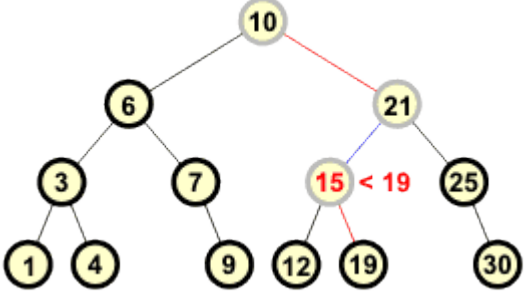
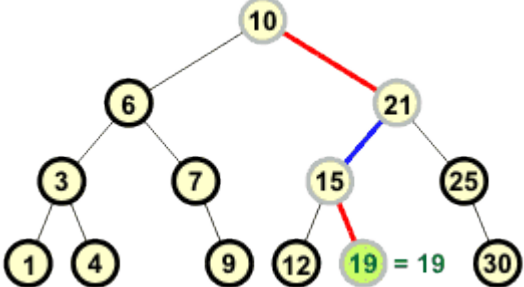
- Jeśli węzeł posiada lewe poddrzewo (drzewo, którego korzeniem jest lewy syn), to wszystkie węzły w tym poddrzewie mają wartość nie większą od wartości danego węzła.
- 1. Jeśli węzeł posiada prawe poddrzewo, to wszystkie węzły w tym poddrzewie są nie mniejsze od wartości danego węzła.



Wyszukiwanie:

Drzewa BST pozwalają wyszukiwać zawarte w nich elementy z klasą złożoności obliczeniowej $O(\log n)$, gdzie n oznacza liczbę węzłów. Prześledźmy sposób wyszukiwania węzła o kluczu 19 w drzewie BST:

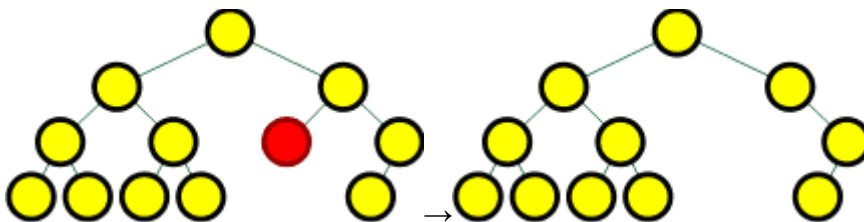
Stan	Opis
	Wyszukiwanie rozpoczynamy od korzenia drzewa. Porównujemy wartość węzła z wartością poszukiwaną. Ponieważ jest ona większa od wartości korzenia, idziemy wzdłuż prawej krawędzi do prawego syna (jeśli węzeł nie miałby prawego syna, to oznaczałoby to, że poszukiwanej wartości nie ma w drzewie BST.)

	<p>W węźle 21 ponownie dokonujemy porównania. Ponieważ poszukiwany węzeł jest mniejszy od 21, wybieramy gałąź lewą i przechodzimy do lewego syna 15.</p>
	<p>Porównujemy węzeł 15 z poszukiwanym 19. Ponieważ 19 jest większe, idziemy prawą krawędzią do prawego syna 19.</p>
	<p>Porównujemy węzeł 19 z poszukiwanym. Są równe. Wyszukiwanie zakończone.</p>

usuwanie:

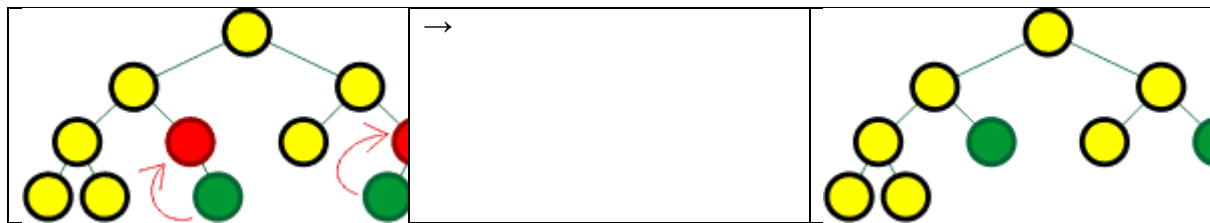
Usuwanie węzła w drzewie BST wymaga sprawdzenia kilku warunków.

Jeśli węzeł nie posiada dzieci, to po prostu odłączamy go od struktury drzewa:

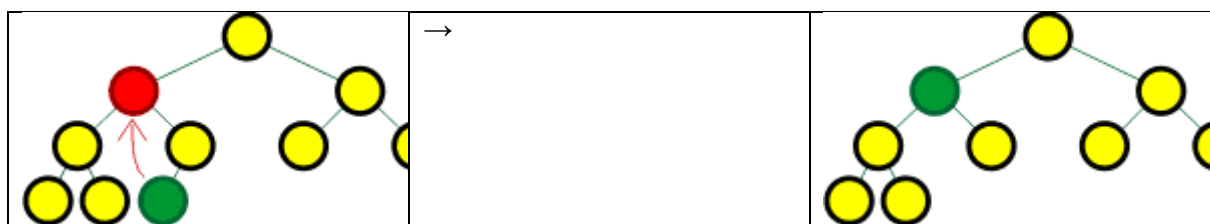


Jeśli węzeł posiada dzieci, to nie można go tak po prostu usunąć, gdyż w strukturze drzewa pojawiłyby się dziury. Mogą wystąpić dwa przypadki:

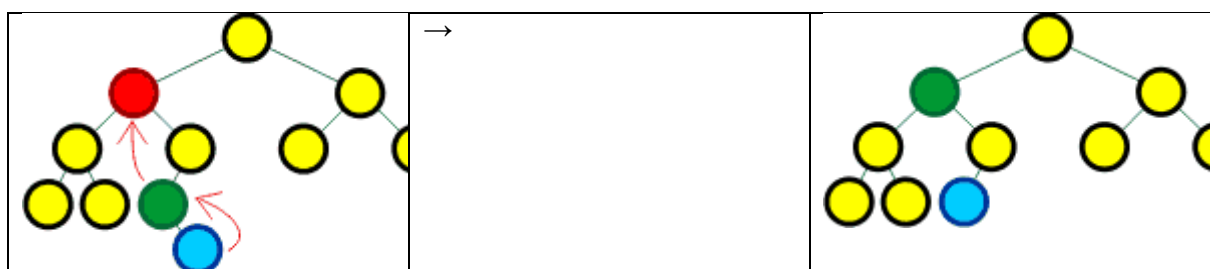
2. Węzeł posiada tylko lewego lub tylko prawego potomka - usuwany węzeł zastępujemy potomkiem



3. Węzeł posiada obu potomków - lewego i prawego. Aby nie degenerować struktury drzewa, usuwany węzeł zastępujemy losowo raz przez jego poprzednikiem, a raz przez jego następnikiem.



Jeśli następnik lub poprzednik posiada dziecko, to zostaje on nim zastąpiony w strukturze drzewa, co wynika z rekurencyjnego charakteru tej operacji.

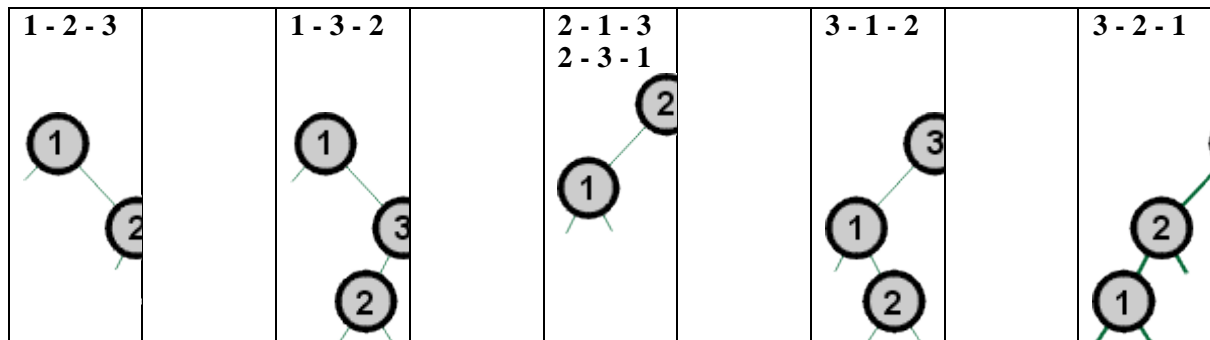


dodawanie:

Zasada pracy algorytmu dołączającego węzeł jest następująca:

Jeśli drzewo jest puste, to nowy węzeł staje się jego korzeniem.

W przeciwnym razie porównujemy klucz wstawianego węzła z kluczami kolejnych węzłów, idąc w dół drzewa. Jeśli klucz nowego węzła jest mniejszy od klucza aktualnie odwiedzonego węzła w drzewie, to przechodzimy do lewego syna. Inaczej przechodzimy do prawego syna. Całą procedurę kontynuujemy do momentu, aż dany syn nie istnieje. Wtedy dołączamy nowy węzeł na jego miejsce i kończymy.

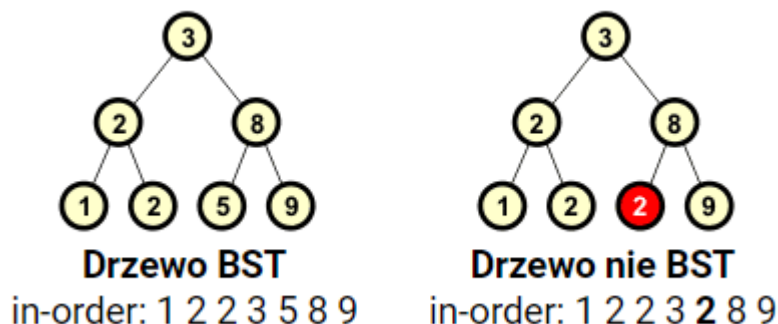


Szczególnie niekorzystne jest wprowadzanie wartości uporządkowanych rosnąco lub malejąco - w takim przypadku otrzymujemy drzewo BST zdegenerowane do listy liniowej (pierwszy i ostatni przykład). W takim drzewie operacje wyszukiwania wymagają czasu rzędu $O(n)$, a nie $O(\log n)$.

4. Drzewa poszukiwań binarnych. Sposób wykonywania na nich podstawowych operacji (dodawanie, wyszukiwanie, usuwanie).

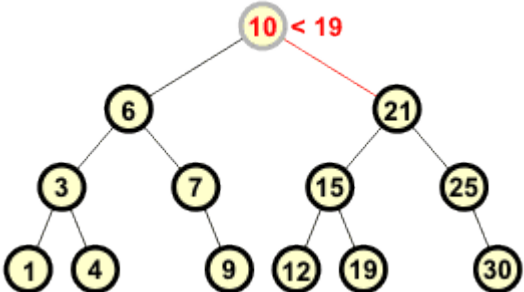
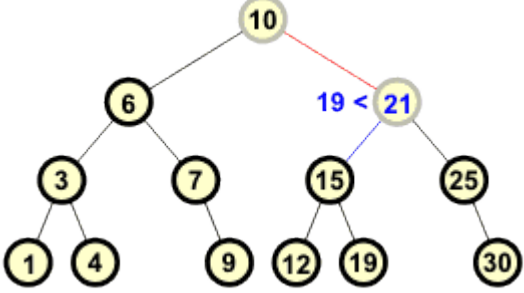
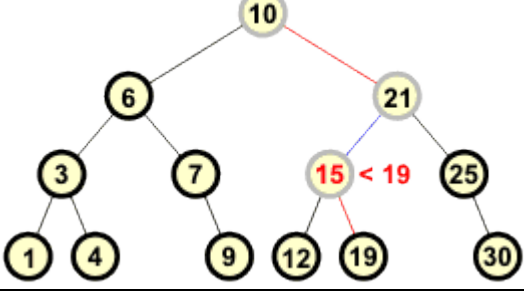
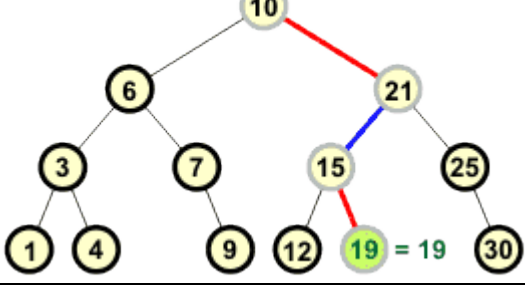
Drzewo poszukiwań binarnych (ang. Binary Search Tree) jest drzewem binarnym, w którym każdy węzeł spełnia reguły:

4. Jeśli węzeł posiada lewe poddrzewo (drzewo, którego korzeniem jest lewy syn), to wszystkie węzły w tym poddrzewie mają wartość nie większą od wartości danego węzła.
5. Jeśli węzeł posiada prawe poddrzewo, to wszystkie węzły w tym poddrzewie są nie mniejsze od wartości danego węzła.



Wyszukiwanie:

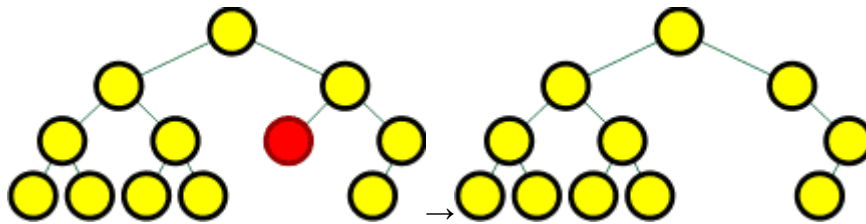
Drzewa BST pozwalają wyszukiwać zawarte w nich elementy z klasą złożoności obliczeniowej $O(\log n)$, gdzie n oznacza liczbę węzłów. Prześledźmy sposób wyszukiwania węzła o kluczu 19 w drzewie BST:

Stan	Opis
	Wyszukiwanie rozpoczynamy od korzenia drzewa. Porównujemy wartość węzła z wartością poszukiwaną. Ponieważ jest ona większa od wartości korzenia, idziemy wzdłuż prawej krawędzi do prawego syna (jeśli węzeł nie miałby prawego syna, to oznaczałoby to, że poszukiwanej wartości nie ma w drzewie BST.)
	W węźle 21 ponownie dokonujemy porównania. Ponieważ poszukiwany węzeł jest mniejszy od 21, wybieramy gałąź lewą i przechodzimy do lewego syna 15.
	Porównujemy węzeł 15 z poszukiwanym 19. Ponieważ 19 jest większe, idziemy prawą krawędzią do prawego syna 19.
	Porównujemy węzeł 19 z poszukiwanym. Są równe. Wyszukiwanie zakończone.

usuwanie:

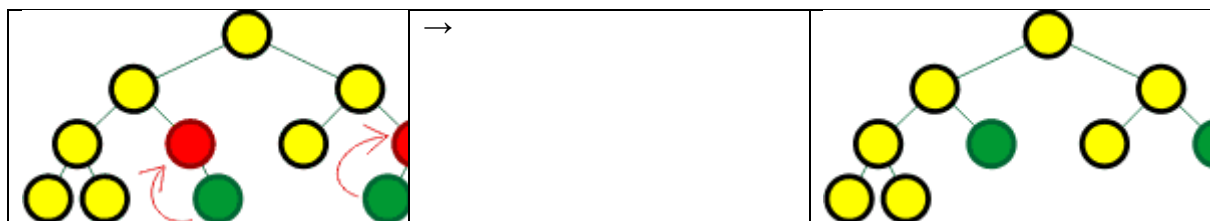
Usuwanie węzła w drzewie BST wymaga sprawdzenia kilku warunków.

Jeśli węzeł nie posiada dzieci, to po prostu odłączamy go od struktury drzewa:

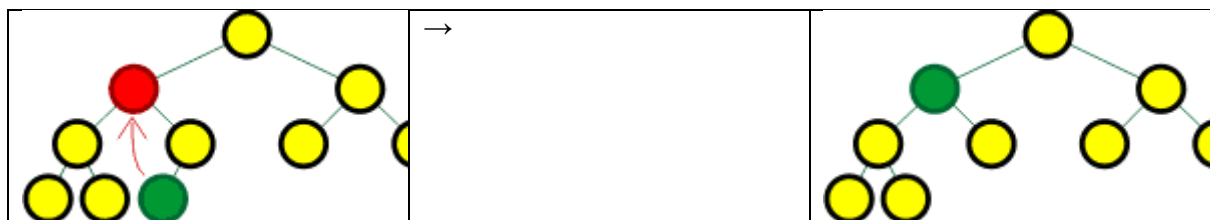


Jeśli węzeł posiada dzieci, to nie można go tak po prostu usunąć, gdyż w strukturze drzewa pojawiłyby się dziury. Mogą wystąpić dwa przypadki:

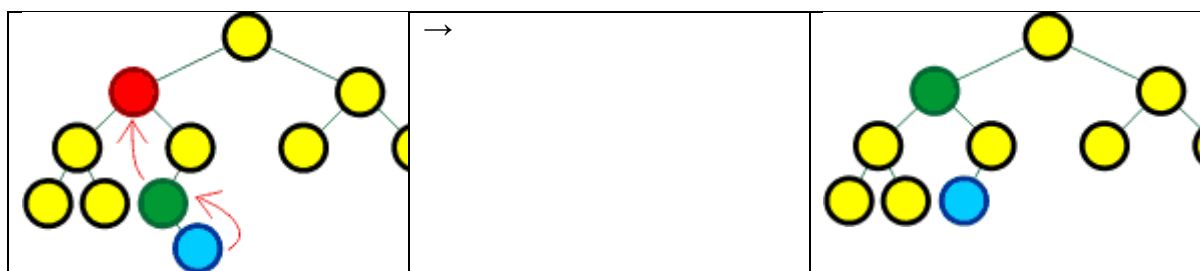
6. Węzeł posiada tylko lewego lub tylko prawego potomka - usuwany węzeł zastępujemy potomkiem



7. Węzeł posiada obu potomków - lewego i prawego. Aby nie degenerować struktury drzewa, usuwany węzeł zastępujemy losowo raz przez jego poprzednika, a raz przez jego następnika.



Jeśli następnik lub poprzednik posiada dziecko, to zostaje on nim zastąpiony w strukturze drzewa, co wynika z rekurencyjnego charakteru tej operacji.

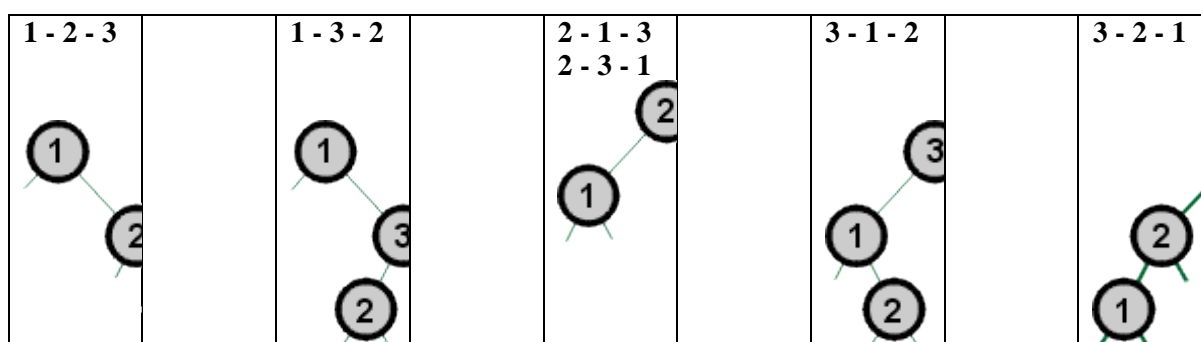


dodawanie:

Zasada pracy algorytmu dołączającego węzeł jest następująca:

Jeśli drzewo jest puste, to nowy węzeł staje się jego korzeniem.

W przeciwnym razie porównujemy klucz wstawianego węzła z kluczami kolejnych węzłów, idąc w dół drzewa. Jeśli klucz nowego węzła jest mniejszy od klucza aktualnie odwiedzonego węzła w drzewie, to przechodzimy do lewego syna. Inaczej przechodzimy do prawego syna. Całą procedurę kontynuujemy do momentu, aż dany syn nie istnieje. Wtedy dołączamy nowy węzeł na jego miejsce i kończymy.



Szczególnie niekorzystne jest wprowadzanie wartości uporządkowanych rosnąco lub malejąco - w takim przypadku otrzymujemy drzewo BST zdegenerowane do listy liniowej (pierwszy i ostatni przykład). W takim drzewie operacje wyszukiwania wymagają czasu rzędu $O(n)$, a nie $O(\log n)$.

5. Algorytmy przeszukiwania grafu.

Przeszukiwanie grafu to odwiedzenie w usystematyzowany sposób wszystkich wierzchołków grafu w celu zebrania potrzebnych informacji. Powszechnie stosuje się dwie podstawowe metody przeszukiwania grafów:

przeszukiwanie wszerz (BFS) - (ang. Breadth-first search, w skrócie BFS) – Algorytm zaczyna od korzenia i odwiedza wszystkie połączone z nim węzły. Następnie odwiedza węzły połączone z tymi

węzłami i tak dalej, aż do odnalezienia celu. przeszukiwanie w głąb (DFS) - (ang. Depth-first search, w skrócie DFS)

– Przeszukiwanie zaczyna się od korzenia i porusza się w dół do samego końca gałęzi, po czym wraca się o jeden poziom i próbuje kolejne gałęzie itd.

Przeszukiwanie w głąb (DFS):

Zaczynamy od wierzchołka 1, odwiedzamy go i wrzucamy na stos wszystkie jego następniki, w

kolejności od tego z największym indeksem: Odwiedzone: 1; Stos: 3, 2;

Bierzemy wierzchołek ze stosu, odwiedzamy go i znów wrzucamy wszystkie jego nieodwiedzone jeszcze następniki na stos:

Odwiedzone: 1, 2; Stos: 3, 5, 4;

Odwiedzone: 1, 2, 4; Stos: 3, 5;

Odwiedzone: 1, 2, 4, 5; Stos: 3;

Odwiedzone: 1, 2, 4, 5, 3; Stos: 6;

Odwiedzone: 1, 2, 4, 5, 3, 6; Stos: ;

Stos jest pusty zatem zakończyliśmy przeszukiwanie grafu w głąb.

Przeszukiwanie wszerz (BFS):

Zaczynamy od wierzchołka 1, odwiedzamy go i wrzucamy do kolejki wszystkie jego następniki, w

kolejności od tego z najmniejszym indeksem:

Odwiedzone: 1; Kolejka: 2, 3;

Bierzemy wierzchołek z kolejki, odwiedzamy go i znów wrzucamy wszystkie jego nieodwiedzone jeszcze następniki do kolejki:

Odwiedzone: 1, 2; Kolejka: 3, 4, 5;

Odwiedzone: 1, 2, 3; Kolejka: 4, 5, 6;

Odwiedzone: 1, 2, 3, 4; Kolejka: 5, 6;

Odwiedzone: 1, 2, 3, 4, 5; Kolejka: 6;

Odwiedzone: 1, 2, 3, 4, 5, 6; Kolejka: ;

Kolejka jest pusta zatem zakończyliśmy przeszukiwanie grafu wszerz.

6. Abstrakcyjne struktury danych: listy, kolejki, stosy. Ich implementacja komputerowa.

Stos

Liniowa struktura danych, w której dane dokładane są na wierzch stosu i z wierzchołka stosu są pobierane (bufor typu LIFO, Last In, First Out; ostatni na wejściu, pierwszy na wyjściu).

Ideę stosu danych można zilustrować jako stos ułożonych jedna na drugiej książek – nowy egzemplarz kładzie się na wierzch stosu i z wierzchu stosu zdejmują się kolejne egzemplarze. Elementy stosu poniżej wierzchołka stosu można wyłącznie obejrzeć, aby je ściągnąć, trzeba najpierw po kolei ściągnąć to, co jest nad nimi.

Implementacja

Strukturami danych służącymi do reprezentacji stosu mogą być tablice (gdy znamy maksymalny rozmiar stosu), tablice dynamiczne lub listy. Złożoność obliczeniowa operacji na stosie zależy od konkretnej implementacji, ale w większości przypadków jest to czas stały $O(1)$.

Listy

To rodzaj kontenera – dynamiczna struktura danych – składająca się z podstruktur wskazujących na następniki i/lub poprzedniki. Typowa lista jest łączona jednostronnie – komórki zawierają tylko odnośnik do kolejnej komórki. Innym przypadkiem jest lista dwustronna, gdzie komórki zawierają także odnośnik do poprzednika.

Implementacja

Tablicowa

Jak wskazuje nazwa, lista zaimplementowana w ten sposób opiera się na tablicy obiektów (lub rekordów) danego typu.

Wskaźnikowa

W tej implementacji każdy obiekt na liście musi (co nie było konieczne w wersji tablicowej) zawierać dodatkowy element: wskaźnik do innego obiektu tego typu. Wynika to z faktu, że to wskaźniki są podstawą nawigacji w tym typie listy, a dostęp do jej elementów jest możliwy wyłącznie przez wskaźnik.

Kolejki

Liniowa struktura danych, w której nowe dane dopisywane są na końcu kolejki, a z początku kolejki pobierane są dane do dalszego przetwarzania (bufor typu FIFO, First In, First Out; pierwszy na wejściu, pierwszy na wyjściu).

Implementacja

Tablicowa

Jak wskazuje nazwa, kolejka zaimplementowana w ten sposób opiera się na tablicy obiektów (lub rekordów) danego typu.

8. Idea algorytmu zachłannego. Przykład.

Algorytmy zachłanne są algorytmami, które w każdym kroku wybierają lokalnie najkorzystniejszą decyzję. Nie dają gwarancji znalezienia optymalnego globalnego rozwiązania oraz nie ma w nich losowości (należą do podzbioru algorytmów heurystycznych i są deterministyczne).

Przykład:

Problem wydawania reszty, tak aby użyć jak najmniejszej ilości monet. Dla monet o nominatach 1, 3 oraz 4. Chcemy wydać 6.

Algorytm w pierwszym kroku wybierze monetę o nominale 4(najkorzystniejsze rozwiązanie lokalne), pozostanie kwota 2, więc algorytm wybierze 1 i jeszcze raz 1. Użyto 3 monet, podczas gdy rozwiązanie optymalne to 2 monety o nominale 3.

Nie istnieje ogólna metoda dowodzenia, czy dla danego problemu rozwiązanie zachłanne (zawsze) odnajduje poprawny (i optymalny) wynik. Istnieją jednak stosujące się do samego problemu kryteria, pozwalające sądzić, iż dla owego problemu istnieje rozwiązanie zachłanne.

Własność zachłannego wyboru:

- za pomocą lokalnie optymalnych (zachłannych) wyborów można uzyskać optymalne rozwiązanie całego zadania.

Własność optymalnej podstruktury:

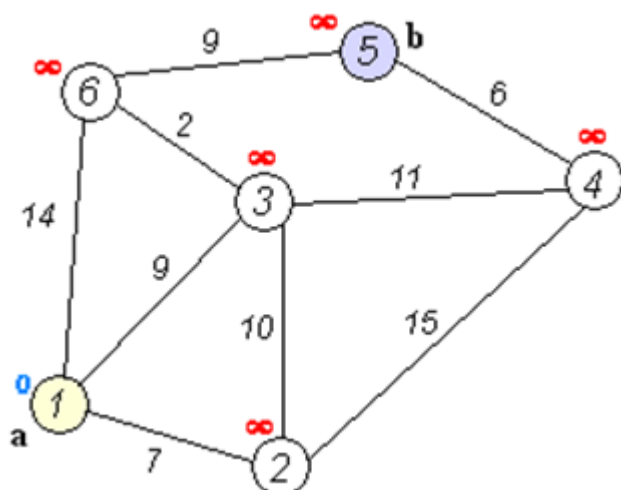
- optymalne rozwiązanie całego problemu jest możliwe tylko przy optymalnym rozwiązaniu podproblemów

1. Idea Dijkstry algorytmu znajdowania najkrótszej ścieżki.

Algorytm Dijkstry jest przykładem algorytmu zachłannego.

Algorytm Dijkstry służy do znajdowania najkrótszej ścieżki z pojedynczego źródła w grafie o nieujemnych wagach krawędzi. Mając dany graf z wyróżnionym wierzchołkiem (*źródłem*) algorytm znajduje odległości od źródła do wszystkich pozostałych wierzchołków. Łatwo zmodyfikować go tak, aby szukał wyłącznie (najkrótszej) ścieżki do jednego ustalonego wierzchołka, po prostu przerywając działanie w momencie dojścia do wierzchołka docelowego. Z algorytmu Dijkstry można skorzystać przy obliczaniu najkrótszej drogi do danej miejscowości. Wystarczy przyjąć, że każdy z punktów skrzyżowań dróg to jeden z wierzchołków grafu, a odległości między punktami to wagi krawędzi.

Jest często używany w sieciach komputerowych, np. przy trasowaniu



11. Budowa komputera

Komponenty, z których jest zbudowany komputer to:

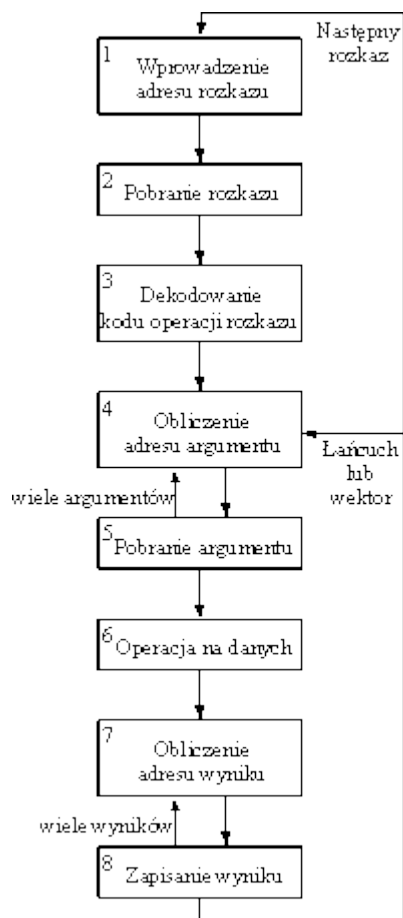
- **płyta główna** (łączy podzespoły i zapewnia komunikację między nimi),
- **procesor** (koordynuje pracę pozostałych podzespołów komputera, wykonuje instrukcje programów),
- **pamięć RAM** (pamięć tymczasowa, przechowuje dane dotyczące aktualnych programów i ich wykonywania),
- **dysk HDD/SSD** (urządzenie pamięci masowej, pozwala na zapisywanie dużej ilości danych i przechowywanie ich przez długi czas),
- **zasilacz** (dostarcza energię do poszczególnych podzespołów),
- **karta graficzna** (odpowiada za renderowanie grafiki i jej konwersję na sygnał zrozumiały dla wyświetlacza),
- **obudowa** (umożliwia zamontowanie w niej podzespołów komputerowych),
- opcjonalny **napęd optyczny** (umożliwia odczyt i/lub zapis danych na nośnikach optycznych)

12. Struktura procesora

W funkcjonalnej strukturze procesora można wyróżnić następujące elementy:

- Zespół rejestrów służących do przechowywania danych i wyników. Rejestry mogą być ogólnego lub specjalnego przeznaczenia.
- Jednostkę arytmetyczną (arytmometr) odpowiadającą za wykonywanie operacji arytmetycznych i logicznych na danych.
- Układ sterujący przebiegiem wykonywania programu.
- Wbudowaną pamięć podręczną cache, która usprawnia procesor o fakt, że nie musi czekać on na dane potrzebne mu do pracy.
- Inne układy wykorzystywane przez producentów w celu usprawnienia pracy procesora.

13. Cykl rozkazowy procesora



Cykl rozkazowy składa się z dwóch faz (pobranie i wykonanie rozkazu).

W fazie pobrania rozkazu na magistralę adresową wysyłana jest zawartość licznika rozkazów.

Licznik rozkazów zawiera adres komórki pamięci, która zawiera rozkaz, który ma być w danej chwili wykonany. Po odczytaniu z pamięci rozkaz wędruje magistralą danych do procesora i wpisuje się do rejestru rozkazów. Na końcu fazy pobrania rozkazów układ sterowania zwiększa zawartość licznika o 1.

W fazie wykonania rozkazów układ sterowania odczytuje z rejestru rozkazów rozkaz, dokonuje jego dekodowania i w zależności od rodzajów rozkazów generuje odpowiednie sygnały sterujące.

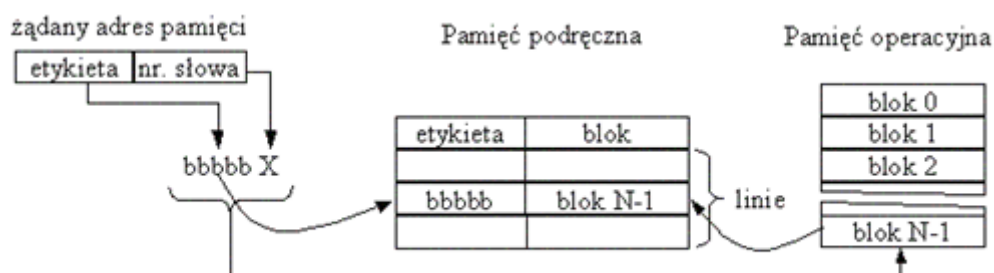
We współczesnych procesorach oba te cykle wykonywane są jednocześnie. W czasie wykonywania rozkazu pobierany jest już następny. Zbiór wszystkich możliwych do wykonania przez procesor rozkazów to lista rozkazów.

14. Metody odwzorowania pamięci głównej w pamięci podręcznej

Odwzorowanie asocjacyjne

Przy asocjacyjnym odwzorowaniu w pamięci podręcznej informacji pochodzącej z pamięci operacyjnej, adres słowa w pamięci asocjacyjnej zostaje podzielony na dwie części: etykieta i numer słowa.

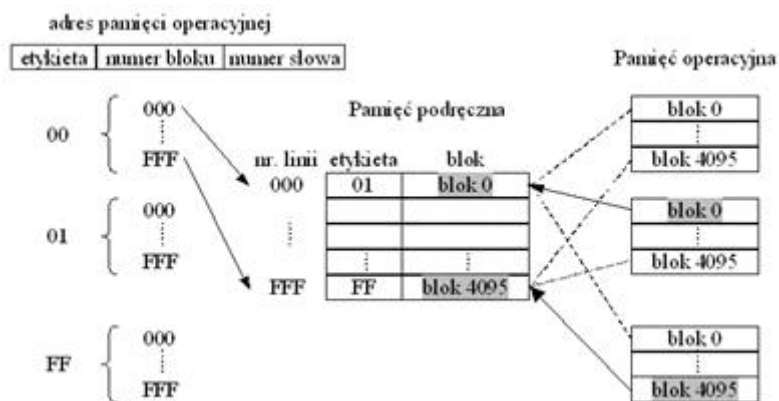
Informacja sprowadzana jest do pamięci podręcznej w blokach. Numer słowa określa położenie słowa w bloku. Pamięć podręczna jest podzielona na linie. W każdej linii może być zapisywany jeden blok, etykieta zawarta w jego adresie i czasami pewne bity sterujące.



Przy sprowadzaniu danego bloku do pamięci podręcznej, blok zapisywany jest w dowolnej wolnej linii. Jeśli wolnej linii nie ma, usuwany jest z pamięci podręcznej jeden blok informacji. Dla wspomagania wyboru bloku do usunięcia, każdy dostęp do bloku obecnego w pamięci podręcznej rejestruje się przez zmianę bitów sterujących w linii, w której ten blok się znajduje.

Odwzorowanie bezpośrednie

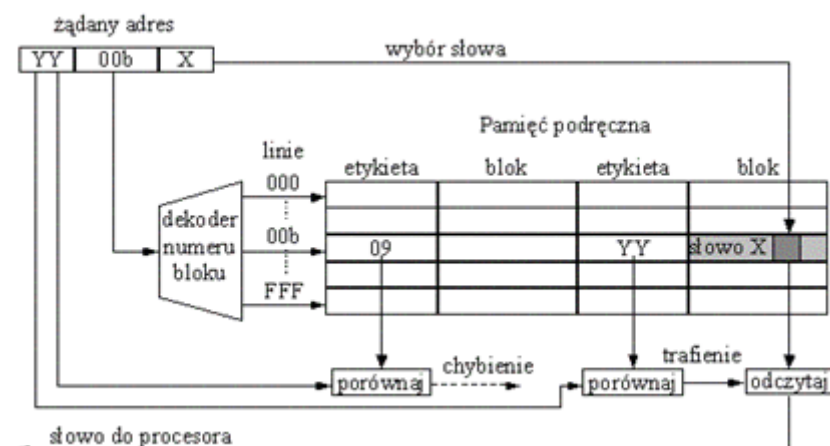
Nazwa tego typu odwzorowania pochodzi od tego, że część adresu żądanej informacji jest bezpośrednio odwzorowywana na numer linii pamięci podręcznej, w której ma być zapisany dany blok informacji. Przy odwzorowaniu bezpośrednim, adres pamięci operacyjnej jest podzielony na trzy części: etykieta, numer bloku, numer słowa w bloku. W danej linii pamięci podręcznej zapisuje się ten blok, który ma taki sam numer bloku w adresie jak numer linii. Razem z blokiem informacji zapisuje się etykietę bloku. Każdemu numerowi bloku odpowiada jedna linia w pamięci podręcznej, do której blok o danym numerze może być zapisany. W tej linii może jednak wymiennie rezydować po jednym bloku spośród tych, które mają numer odpowiadający numerowi linii ale różne etykiety.



Numeru bloku (środkowa część adresu potrzebnego słowa) jest dekodowana w dekodерze, który wybiera numer linii pamięci podręcznej. W linii wybranej przez dekodер następuje sprawdzenie czy zapisana tam etykieta jest równa etykietcie w adresie żadanego słowa. Jeśli jest zgodność, to znaczy, że w danej chwili, w linii odpowiadającej potrzebnemu numerowi bloku rezyduje dokładnie ten blok, który jest potrzebny. Jeżeli wynik porównania etykiet jest negatywny, to znaczy, że w danej linii nie ma żadnego bloku albo rezyduje tam inny blok niż potrzebny. W obu przypadkach następuje sprowadzenie potrzebnego bloku z pamięci podręcznej niższego poziomu lub z pamięci operacyjnej.

Odwzorowanie zbiorowo asocjacyjne

Odwzorowanie te ma ten sam podział na etykietę, numer bloku oraz numer słowa w bloku, oprócz tego zasada odwzorowania numeru bloku na linię pamięci podręcznej jest identyczna jak w metodzie bezpośredniej. Przy odwzorowaniu zbiorowo asocjacyjnym, w każdej linii pamięci podręcznej można zapisać kilka bloków z ich etykietami - czyli zbiór bloków. Dostęp do bloków zapisanych w danej linii następuje na zasadzie asocjacyjnej, przez porównanie z żadaną etykietą etykiet ze zbioru wszystkich bloków zapisanych w tej linii. Liczba bloków zapisywanych w jednej linii tego typu pamięci podręcznej wynosi zwykle od 2 do 6 przy długości bloku od 8 do 64 bajtów.



Linki:

- <https://edux.pjwstk.edu.pl/mat/198/lec/main84.html>
- <http://157.158.56.13/dyplomy/1/sposoby4.html>
- Dodatkowo pamięć podręczna <https://pclab.pl/art75257.html>

16. Porównanie różnych architektur sieci komputerowych

Architektura sieci – określa sposoby realizacji przekazu informacji pomiędzy urządzeniami końcowymi. Przeważnie organizowana warstwowo tworząc warstwową architekturę sieci.

Ważniejsze architektury warstwowe:

ISO-OSI (ISO - Open Systems Interconnection),
SNA (Systems Network Architecture) firmy IBM,

TCP/IP (Transmission Control Protocol/Internet Protocol)
DNA (Digital Network Architecture) firmy DEC

Model OSI dwa systemy mogą wymieniać informacje między sobą pod warunkiem, że w obu przypadkach zaimplementowano te same protokoły komunikacyjne.

Warstwy modelu ISO/OSI:

1 APLIKACJI - zajmuje się specyfikacją interfejsu, który wykorzystują aplikacje do przesyłania danych do sieci;

2 PREZENTACJI - zapewnia tłumaczenie danych, definiowanie ich formatu oraz odpowiednią składnię;

3 SESJI - zadaniem jest zarządzanie przebiegiem komunikacji, podczas połączenia między dwoma komputerami;

4 TRANSPORTOWA - segmentuje dane oraz składa je w tzw. strumień, zapewnia całościowe połączenie między stacjami;

5 SIECIOWA - podstawowe zadania to przesyłanie danych między węzłami sieci wraz z wyznaczaniem trasy przesyłu, łączenie bloków informacji w ramki na czas ich przesyłania a następnie stosowny ich podział;

6 ŁĄCZA DANYCH - odpowiada za obsługę ramek, czyli stałej długości pakietów danych, do jej zadań należy wykrywanie wszelkich błędów, które wystąpiły w warstwie fizycznej, oraz ich usuwanie;

7 FIZYCZNA - odpowiedzialna za przesyłanie sygnałów w elektryczny, elektromagnetyczny, mechaniczny, optyczny czy też inny sposób, wykorzystując do tego fizyczne medium komunikacyjne (przewody miedziane, światłowody)

Model TCP/IP - podział całego zagadnienia komunikacji sieciowej na szereg współpracujących ze sobą warstw (ang. *layers*). Każda z nich może być tworzona przez programistów niezależnie, jeżeli narzucimy pewne protokoły według których wymieniają się one informacjami. Założenia modelu TCP/IP są pod względem organizacji warstw zbliżone do modelu OSI. Jednak ilość warstw jest mniejsza i bardziej odzwierciedla prawdziwą strukturę Internetu.

Model TCP/IP składa się z czterech warstw:

1 APLIKACJI - najwyższy poziom, w którym pracują użyteczne dla człowieka aplikacje takie jak, np. serwer WWW czy przeglądarka internetowa. Obejmuje ona zestaw gotowych protokołów, które aplikacje wykorzystują do przesyłania różnego typu informacji w sieci.

2 TRANSPORTOWA - zapewnia pewność przesyłania danych oraz kieruje właściwe informacje do odpowiednich aplikacji.

3 INTERNETU - przetwarzane są datagramy posiadające adresy IP. Ustalana jest odpowiednia droga do docelowego komputera w sieci.

4 DOSTĘPU DO SIECI - zajmuje się przekazywaniem danych przez fizyczne połączenia między urządzeniami sieciowymi. Najczęściej są to karty sieciowe lub modemy. Dodatkowo warstwa ta jest czasami wyposażona w protokoły do dynamicznego określania adresów IP.

Model OSI	Model TCP/IP
został wymyślony przed wynalezieniem odpowiadających mu protokołów. Taka kolejność oznacza, że model nie był ukierunkowany na konkretny zbiór protokołów i stał się przez to bardziej ogólny.	Model jest opisem istniejących już protokołów. Jedyny problem jest taki że nie pasuje do żadnego innego stosu protokołów.
7 warstw	4 warstwy
Funkcjonalność warstw jest zbliżona. Np. w obu modelach warstwy od dołu aż do transportowej włącznie mają zapewnić dwupunktową, niezależną od sieci usługę transportową procesów, które chcą się komunikować.	
Warstwy powyżej transportowej w obu modelach są zorientowanymi na aplikację użytkownikami usługi transportowej.	

18. Funkcje warstwy sieci modelu OSI.

Warstwa sieci modelu OSI odpowiada za określenie trasy pomiędzy nadawcą a odbiorcą.

Kierowanie przepływem pakietów w sieci. Zapewnia ona, że pakiety przesyłane między komputerami nie łączącymi się bezpośrednio, będą przekazywane z komputera na komputer, aż osiągną adresata. Proces znajdowania drogi w sieci nazywa się trasowaniem lub routowaniem.

Nie wymaga się aby pakiety pomiędzy ustalonymi punktami poruszały się za każdym razem po tej samej drodze. Warstwa ta nie posiada żadnych mechanizmów wykrywania oraz korygowania błędów.

Dopuszcza się gubienie pakietów przez tę warstwę - jest jedynym wyjściem gdy router - węzeł dokonujący trasowania lub pewien segment sieci są przeciążone i nie mogą przyjmować nowych pakietów.

19. Dynamiczne przydzielanie adresów.

Protokół DHCP opisuje trzy techniki przydzielania adresów IP:

-przydzielanie ręczne oparte na tablicy adresów MAC oraz odpowiednich dla nich adresów IP. Jest ona tworzona przez administratora serwera DHCP. W takiej sytuacji prawo do pracy w sieci mają tylko komputery zarejestrowane wcześniej przez obsługę systemu.

-przydzielanie automatyczne, gdzie wolne adresy IP z zakresu ustalonego przez administratora są przydzielane kolejnym zgłaszającym się po nie klientom.

-przydzielanie dynamiczne, pozwalające na ponowne użycie adresów IP. Administrator sieci nadaje zakres adresów IP do rozdzielania. Wszyscy klienci po starcie systemu automatycznie pobierają swoje adresy na pewien czas (lease). Protokół DHCP minimalizuje również możliwe źródła błędów.

Niektóre serwery DHCP dodatkowo przydzielają każdemu klientowi własny adres DNS, przekazywany na serwer nazw protokołem zgodnym ze specyfikacją RFC 2136.

20. Protokoły połączeniowe i bezpołączeniowe.

Protokół połączeniowy to protokół komunikacyjny, w którym przed rozpoczęciem wymiany komunikatów strony muszą nawiązać połączenie, w którego ramach może zostać wynegocjowany sam protokół. Po zakończeniu łączności połączenie musi ulec likwidacji. W wyniku budowy połączenia zostaje utworzona pojedyncza ścieżka między stacją nadawczą a stacją odbiorczą. W fazie przesyłania dane są transmitowane przez utworzoną ścieżkę w sposób sekwencyjny. Dane docierają do stacji odbiorczej w kolejności ich wysyłania przez stację nadawczą.

Protokół bezpołączeniowy - typ protokołu sieciowego, umożliwia przesyłanie bez ustanawiania połączenia z odbiorcą. Nie buduje się jedynej ścieżki między stacją nadawczą i odbiorczą, pakiety do stacji odbiorczej mogą docierać w innej kolejności niż są wysyłane przez stację nadawczą, ze względu na to, że mogą być przesyłane różnymi trasami. W usłudze bezpołączeniowej dane przepływają przez trwałe połączenia między węzłami sieci, a każdy pakiet jest obsługiwany

indywidualnie i niezależnie od innych pakietów danego komunikatu. Jest to możliwe pod warunkiem, że każdy pakiet jest kompletnie zaadresowany, to znaczy, że każdy z nich ma swój adres stacji nadawczej i stacji odbiorczej.

Cecha	UDP	TCP
połączeniowy	nie	tak
nagłówek zawiera długość segmentu	tak	nie
sumy kontrolne	opcjonalne	tak
potwierdzenie pozytywne	nie	tak
retransmisje i kontrola czasu	nie	tak
wykrywanie powtórzeń	nie	tak
numery sekwencyjne	nie	tak
kontrola przepływu	nie	tak

21. Tryby FTP.

FTP działa w dwóch trybach: aktywnym i pasywnym, w zależności od tego, w jakim jest trybie, używa innych portów do komunikacji.

- jeżeli połączenie **FTP działa w trybie aktywnym** to używa portu 21 dla poleceń – zestawiane przez klienta i portu 20 do przesyłu danych – zestawiane przez serwer.

- jeżeli połączenie **FTP pracuje w trybie pasywnym** to wykorzystuje port 21 dla poleceń i port o numerze powyżej 1024 do transmisji danych – obydwa połączenia zestawiane są przez klienta.

Aktywny tryb FTP jest wygodny dla administratora serwera FTP, jednak kłopotliwy dla klienta. Serwer FTP stara się nawiązać połączenie z nieuprzywilejowanym portem klienta, co najprawdopodobniej zostanie udaremnione przez firewall po stronie klienta.

Tryb pasywny jest wygodny dla klienta, lecz kłopotliwy dla administratora serwera FTP. Oba połączenia nawiązuje klient, ale jedno z nich dotyczy nieuprzywilejowanego portu, i najprawdopodobniej będzie zablokowane przez firewall po stronie serwera.

22. Podstawowe pojęcia baz danych. Baza Danych. Funkcje bazy danych. Właściwości bazy danych. Modele baz danych.

Baza danych – zbiór informacji opisujący wybrany fragment rzeczywistości.

Właściwości baz danych:

- współdzielenie danych – wielu użytkowników tej samej bazy,
- integracja danych – baza nie powinna mieć powtarzających się, bądź zbędnych danych,
- niezależność danych – dane niezależnie od aplikacji wykorzystujących te dane.
- integralność danych – dokładne odzwierciedlenie rzeczywistości,
- trwałość danych – dane przechowywane przez pewien czas,
- bezpieczeństwo danych – dostęp do bazy lub jej części przez upoważnionych użytkowników,

Funkcje bazy danych:

- aktualizujące – dokonują zmian na danych,
- zapytań – wydobywają dane z bazy danych.

Model baz danych to zbiór zasad (specyfikacji), opisujących strukturę danych w bazie danych.

Określane są również dozwolone operacje.

Model hierarchiczny: W modelu hierarchicznym dane są przechowywane na zasadzie rekordów nadrzędnych-podrzędnych, tzn. rekordy przypominają strukturę drzewa. Każdy rekord (z wyjątkiem głównego) jest związany z dokładnie jednym rekordem nadrzędnym.

Model relacyjny: Dane w takim modelu przechowywane są w tabelach, z których każda ma stałą liczbę kolumn i dowolną liczbę wierszy. Każda tabela (relacja) ma zdefiniowany klucz danych (key) - wyróżniony atrybut lub kilka takich atrybutów, którego wartość jednoznacznie identyfikuje dany wiersz.

Model obiektowy: Dane w tym modelu przechowywane są jako instancje/obiekty zdefiniowanych klas. Każda klasa może mieć zdefiniowany unikalny zbiór atrybutów, a również super klasę (przodka) po którym dziedziczy atrybuty. Obiekty jednej klasy posiadają ten sam zbiór atrybutów a różnić się mogą tylko ich wartościami.

Model sieciowy: Model danych operujący pojęciami typów rekordów i typów kolekcji (opisów związków “jeden do wielu” między dwoma typami rekordów). Związki asocjacyjne pomiędzy danymi są reprezentowane poprzez powiązania wskaźnikowe. Struktura danych tworzy więc graf, czyli sieć.

23. System zarządzania bazami danych. Funkcje systemu. Przykłady SZDB.

System zarządzania bazą danych, SZBD (ang. Database Management System, DBMS) – oprogramowanie bądź system informatyczny służący do zarządzania bazą danych.

Funkcje SZBD:

- optymalizacja zapytań – takie przekształcenie zapytań kierowanych do bazy, aby oczekiwanie na odpowiedź było możliwie najkrotsze
- zapewnienie integralności danych – uniemożliwienie przejścia bazy do stanu, który nie istnieje w modelowanej rzeczywistości
- zarządzanie współbieżnym dostępem wielu użytkowników, tak aby ich działania nie kolidowały ze sobą
- odporność na awarie – możliwość odtworzenia poprawnego stanu bazy po wystąpieniu awarii
- ochrona danych – uniemożliwienie dostępu do danych bez autoryzacji

Przykłady SZBD:

- MySQL: silniki MyISAM, InnoDB
- Access: silnik Microsoft Jet
- Kexi (konkurent Access): silnik SQLite

24. Model relacyjny baz danych. Relacje, klucze główne i obce, integralność referencyjna.

Model relacyjny – model organizacji danych bazujący na matematycznej teorii mnogości, w szczególności na pojęciu relacji. Reprezentacją relacji jest dwuwymiarowa tabela złożona z kolumn (atrybutów) i wierszy (krotek).

W modelu relacyjnym przyjmuje się następujące założenia o tabeli:

- Liczba kolumn (atrybutów) jest z góry ustalona
- Z każdą kolumną (atrybutem) jest związana jej nazwa oraz dziedzina, określająca zbiór wartości, jakie mogą występować w danej kolumnie
- Na przecięciu wiersza i kolumny znajduje się pojedyncza (atomowa) wartość należąca do dziedziny kolumny
- Wiersz (krotka) reprezentuje jeden rekord informacji
- W modelu relacyjnym kolejność wierszy (krotek) może się zmieniać

Relacje - związki między tabelami.

- Jeden – do – wielu ($1 - \infty$) wierszowi (krotce) w tabeli A może odpowiadać wiele zgodnych wierszy (krotek) w tabeli B, ale wierszowi w tabeli B może odpowiadać tylko jeden zgodny wiersz w tabeli A
- Wiele – do – wielu ($\infty - \infty$) wierszowi (krotce) w tabeli A może odpowiadać wiele zgodnych wierszy (krotek) w tabeli B i na odwrot. Relację taką tworzy się definiując trzecią tabelę, zwaną tabelą skrzyżowań, której klucz podstawowy zawiera zarówno klucz obcy z tabeli A, jak i z tabeli B.
- Jeden – do – jednego ($1 - 1$) wierszowi (krotce) w tabeli A może odpowiadać nie więcej niż jeden zgodny wiersz (krotka) w tabeli B i na odwrot. Relacja jeden-do-jednego jest tworzona, jeśli obie powiązane kolumny są kluczami podstawowymi lub mają ograniczenia UNIQUE.

Klucz główny - dla każdej tabeli w modelu relacyjnym musi być określony, jednoznaczny identyfikator, nazywany kluczem głównym (podstawowym). Klucz główny składa się z jednej lub ze zbioru kolumn (atrybutów), w których wartości w jednoznaczny sposób identyfikują cały wiersz (krotkę). Oznacza to, że wartości znajdujące się w kolumnie będącej kluczem głównym nie mogą się powtarzać i muszą być unikatowe.

Klucz obcy - jest to zbiór złożony z jednej kolumny lub więcej kolumn, w których wartości występują, jako wartości ustalonego klucza głównego lub jednoznacznego w tej samej lub innej tabeli i są interpretowane, jako wskaźniki do wierszy w tej drugiej tabeli.

Zasada integralności referencyjnej w relacyjnej bazie danych wymaga, aby wartości klucza obcego tabeli podrzędnej były puste (null) lub odpowiadały wartościom klucza podstawowego tabeli nadrzędnej.

25. Modelowanie baz danych. Diagram związków encji.

Model bazy danych to zbiór zasad (specyfikacji), opisujących strukturę danych w bazie danych. Definiuje się strukturę danych poprzez specyfikację reprezentacji dozwolonych w modelu obiektów oraz ich związków

Hierarchiczny model danych W modelu hierarchicznym dane są przechowywane na zasadzie rekordów nadrzędnych-podrzędnych, tzn. rekordy przypominają strukturę drzewa. Każdy rekord (z wyjątkiem głównego) jest związany z dokładnie 1 rekordem nadrzędnym. Przykładem takiego modelu może być struktura katalogów na dysku twardym komputera.

Relacyjny model danych (RDBMS) Dane w takim modelu przechowywane są w tabelach, z których każda ma stałą liczbę kolumn i dowolną liczbę wierszy. Każda tabela (relacja) ma zdefiniowany klucz danych (key) - wyróżniony atrybut lub kilka takich atrybutów, którego wartość jednoznacznie identyfikuje dany wiersz.

Specyfikując schemat klasy obiektów w języku ODL, opisujemy trzy rodzaje właściwości obiektów:

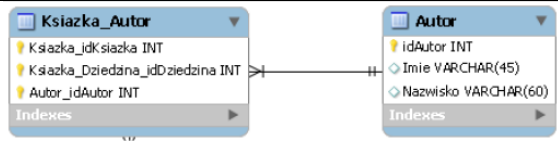
Atrybuty - przyjmują wartości typów pierwotnych takich jak całkowity lub tekstowy, albo typów złożonych powstających z pierwotnych;

Związki - referencje do obiektów pewnej klasy, albo kolekcje (zbiory) takich referencji;

Metody - funkcje operujące na obiektach danej klasy.

Diagram Związków Encji

Modele danych można opracowywać na różnych poziomach szczegółowości wykorzystując technikę diagram związkow encji.

Komponent	Opis
Encja	Rzecz mająca znaczenie, rzeczywista lub wymyślona, o której informacje należy znać lub przechowywać.
Atrybut	Element informacji służący do klasyfikowania, identyfikowania, kwalifikowania, określania ilości lub wyrażania stanu encji.
Związek	Znaczący sposób, w jaki mogą być ze sobą powiązane dwie rzeczy tego samego typu lub różnych typów.
Przykład	

Uwaga: encje opisuje się za pomocą rzeczowników lub wyrażeń rzeczownikowych w liczbie pojedynczej.

26. Język baz danych SQL. Podjęzyki DDL, DML, DCL.

SQL(Structured Query Language) – język zapytań, służący do komunikacji z relacyjną bazą danych. Za jego pomocą możliwe jest pobieranie, wstawianie i modyfikowanie danych oraz tworzenie i modyfikowanie baz danych. SQL jest językiem deklaratywnym tzn. programista opisuje warunki końcowego rozwiązania, a nie szczegółową sekwencję kroków, które do niego prowadzą. Składa się z zapytań.

Podstawowe pojęcia:

- Zapytanie – pojedyncza instrukcja, składająca się ze słów kluczowych oraz parametrów. Dla przykładu: SELECT nazwa_kolumny FROM nazwa_tabeli WHERE warunek (Słowa kluczowe: SELECT, FROM, WHERE i parametry: nazwa_kolumny, nazwa_tabeli, warunek).
- Tabela – zbiór wierszy(rekordów) o określonej kolumnami strukturze.
- Relacja – powiązanie między danymi np. dla tabel Kierowcy i Autobusy, możliwa jest relacja określająca kierowcę autobusu.

1. **SQL interakcyjny** wykorzystywany jest przez użytkowników w celu bezpośredniego pobierania lub wprowadzania informacji do bazy. Przykładem może być zapytanie prowadzące do uzyskania zestawienia aktywności kont w miesiącu. Wynik jest wówczas przekazywany na ekran, z ewentualną opcją przekierowania go do pliku lub drukarki.

2. Statyczny kod SQL (Static SQL) nie ulega zmianom i pisany jest wraz z całą aplikacją, podczas której pracy jest wykorzystywany. Nie ulega zmianom w sensie zachowania niezmiennej treści instrukcji, które jednak zawierać mogą odwołania do zmiennych lub parametrów przekazujących wartości z lub do aplikacji. Statyczny SQL występuje w dwóch odmianach.

a. **Embedded SQL (Osadzony SQL)** oznacza włączenie kodu SQL do kodu źródłowego innego języka. Większość aplikacji pisana jest w takich językach jak C++ czy Java, jedynie odwołania do bazy danych realizowane są w SQL. W tej odmianie statycznego SQL-a do przenoszenia wartości wykorzystywane są zmienne.

b. **Język modułów.** W tym podejściu moduły SQL łączone są z modułami kodu w innym języku. Moduły kodu SQL przenoszą wartości do i z parametrów, podobnie jak to się dzieje przy

wywoływaniu podprogramów w większości języków proceduralnych. Jest to pierwotne podejście, zaproponowane w standardzie SQL. Embedded SQL został do oficjalnej specyfikacji włączony nieco później.

3. Dynamiczny kod SQL (Dynamic SQL) generowany jest w trakcie pracy aplikacji. Wykorzystuje się go w miejsce podejścia statycznego, jeżeli w chwili pisania aplikacji nie jest możliwe określenie treści potrzebnych zapytań – powstaje ona w oparciu o decyzje użytkownika. Tę formę SQL generują przede wszystkim takie narzędzia jak graficzne języki zapytań. Utworzenie odpowiedniego zapytania jest tu odpowiedzią na działania użytkownika.

DML (Data Manipulation Language) służy do wykonywania operacji na danych – do ich umieszczania w bazie, kasowania, przeglądania, zmiany. Najważniejsze polecenia z tego zbioru to:

* SELECT * INSERT * UPDATE * DELETE

DDL (Data Definition Language) można operować na strukturach, w których dane są przechowywane – czyli np. dodawać, zmieniać i kasować tabele lub bazy. Najważniejsze polecenia tej grupy to:

* CREATE * DROP * ALTER

DCL (Data Control Language) ma zastosowanie do nadawania uprawnień do obiektów bazodanowych. Najważniejsze polecenia w tej grupie to:

* GRANT * REVOKE

27. Instrukcja SELECT.

Polecenie SELECT nazywa się także „zapytaniem”. Polecenie to pobiera krotki z relacyjnej bazy danych i zwraca w postaci zbioru odczytanych krotek (zbior taki może być też pusty), opcjonalnie może je przetworzyć przed zwroceniem. Mówiąc prościej polecenie SELECT służy do odczytywania danych z bazy danych i dokonywaniu na nich prostych obliczeń i przekształceń.

```
SELECT [DISTINCT] {atrybut1, atrybut2, ....}
```

```
FROM {nazwa relacji}
```

```
[WHERE {warunek}] [ORDER BY {{wyrażenie5 [ASC|DESC], alias1 [ASC|DESC]}}] [LIMIT {limit}];
```

28. Instrukcje DDL i DCL.

DDL (Data Definition Language - Język definiowania danych) definiuje struktury danych, na których operują instrukcje DML.

CREATE (np. CREATE TABLE, CREATE DATABASE, ...) – utworzenie struktury (bazy, tabeli, indeksu itp.),

DROP (np. DROP TABLE, DROP DATABASE, ...) – usunięcie struktury,

ALTER (np. ALTER TABLE ADD COLUMN ...) – zmiana struktury (dodanie kolumny do tabeli, zmiana typu danych w kolumnie tabeli).

DCL (Data Control Language - Język kontrolowania danych) ma zastosowanie do nadawania uprawnień do obiektów bazodanowych.

GRANT – (GRANT {ALL | lista_uprawnień} TO użytkownik})-przyznaje użytkownikowi uprawnienia

REVOKE – (REVOKE {ALL | lista_uprawnień} TO użytkownik})-zabiera uprawnienia, które zostały wcześniej przyznane

DENY- (DENY {ALL | lista_uprawnień} TO użytkownik})-Bezpośrednio zabiera uprawnienia

29. Indeksy w bazach danych, podział indeksów , B⁺-drzewo.

Indeks jest mechanizmem poprawienia szybkości dostępu do danych bez zmiany struktury ich przechowywania. Indeks zazwyczaj jest założony na jednym polu rekordu danych.

Rekord indeksu ma strukturę: <wartość pola, adres w pliku danych> i jest zapisywany do pliku indeksu w kolejności zgodnej z wartością pola indeksowanego.

Indeksy dzielimy na:

- indeks główny – założony na atrybucie porządkującym unikalnym,
- grupujący – założony na atrybucie porządkującym nieunikalnym,
- pomocniczy – założony na atrybucie nieporządkującym.
- Indeks gęsty - posiada rekord indeksu dla każdego rekordu indeksowanego pliku danych
- Indeks rzadki - posiada rekordy tylko dla wybranych rekordów indeksowanego pliku danych (wskazuje na blok rekordów, a nie poszczególne rekordy)

W miarę wzrostu rozmiarów indeksu wydłuża się czas wyszukiwania po tym indeksie, dlatego wiele systemów zarządzania bazą danych stosuje pewną formę indeksu wielopoziomowego:

- statyczne wielopoziomowe indeksy – modyfikowanie zawartości pliku z danymi powoduje, że struktura indeksu staje się nieefektywna
- dynamiczne wielopoziomowe indeksy – B+-Drzewo

B+ drzewo jest zrównoważoną strukturą drzewiastą, w której wierzchołki wewnętrzne służą do wspomagania wyszukiwania, natomiast wierzchołki liści zawierają rekordy indeksu ze wskaźnikami do rekordów w plikach danych. Zrównoważenie struktury oznacza, że odległość (liczba poziomów) od korzenia do dowolnego liścia jest zawsze taka sama. W celu zapewnienia odpowiedniej efektywności realizacji zapytań przedziałowych wierzchołki liści stanowią listę dwukierunkową.

```
CREATE INDEX <nazwa indeksu> ON <nazwa tabeli> (<nazwa kolumny>)
```

30. Normalizacja., cel normalizacji, postać normalna Boyce'a-Codda

Normalizacja – proces mający na celu eliminację powtarzających się danych w relacyjnej bazie danych. Dążymy w niej do zwiększenia elastyczności bazy danych poprzez wyeliminowanie nadmiarowości (jakieś dane są niepotrzebnie powtarzane w wielu miejscach bazy danych) i niespójnych zależności (np. sytuacja gdy adres klienta znajdowałby się w tabeli „Wynagrodzenie”). Pola powinny dodatkowo przechowywać dane atomowe (czyli wartości danego atrybutu nie mogą być listami, macierzami i innymi „bytami” posiadającymi jakąś szerszą strukturę). Wszystkie kolumny tabeli powinny być zależne tylko od jej klucza głównego (najczęściej stosowana jest kolumna z ID danego obiektu), a wykonywanie operacji na danych nie powinno skutkować anomaliami (np. sytuacja, że modyfikacja jednego rekordu wymusza na nas wprowadzenie zmian w innych rekordach).

Istnieją trzy główne reguły normalizacji (pierwsza, druga oraz trzecia forma normalna) oraz dodatkowo, wyróżnia się także czwartą oraz piątą formę normalną, które są z reguły bardzo rzadko wykorzystywane w praktycznych projektach.

Postać normalna Boyce’a Codda – relacja jest w postaci Boyce’a Codda wtedy i tylko wtedy, kiedy każdy jej atrybut zależy funkcjonalnie tylko od jej klucza głównego.

Relacja R jest w BCNF wtedy, i tylko wtedy gdy dla każdej nietrywialnej zależności zbioru $A_1 \dots A_n \rightarrow B$ zbiór ten jest nadkluczem R.

31. Transakcje, własności transakcji.

Transakcja jest sekwencją logicznie powiązanych operacji na bazie danych, która przeprowadza bazę danych z jednego stanu spójnego w inny stan spójny. Typy operacji na bazie danych obejmują: odczyt i zapis danych oraz zakończenie połączone z akceptacją (zatwierdzeniem) lub wycofaniem transakcji.

Atomicity - atomowość - zbior operacji wchodzących w skład transakcji jest niepodzielny, to znaczy albo zostaną wykonane wszystkie operacje transakcji albo żadna.

Consistency - spójność - transakcja przeprowadza bazę danych z jednego stanu spójnego do innego stanu spójnego. W trakcie wykonywania transakcji baza danych może być przejściowo niespójna. Transakcja nie może naruszać ograniczeń integralnościowych.

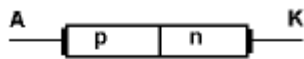
Isolation - izolacja - transakcje są od siebie logicznie odseparowane. Transakcje oddziałują na siebie poprzez dane. Mimo współbieżnego wykonywania, transakcje widzą stan bazy danych tak, jak gdyby były wykonywane w sposób sekwencyjny.

Durability - trwałość - wyniki zatwierdzonych transakcji nie mogą zostać utracone w wyniku wystąpienia awarii systemu. Zatwierdzone dane w bazie danych, w przypadku awarii, muszą być odtwarzalne.

32. Diody półprzewodnikowe. Tranzystory.

Diodą półprzewodnikową nazywamy element wykonany z półprzewodnika (np. krzem, german), zawierającego jedno złącze – najczęściej p-n z dwiema końcówkami wyprowadzeń.

Oznaczenia: A – anoda K – katoda n – warstwa ujemna(elektrony) p – warstwa dodatnia



rys. 1. schemat blokowy diody



rys. 2. symbol diody p-n

Dioda przepuszcza prąd w jednym kierunku (od anody do katody), natomiast w kierunku przeciwnym - w minimalnym stopniu. Diody stosowane są w układach analogowych i cyfrowych. Diody półprzewodnikowe stosuje się w układach prostowania prądu zmiennego, w układach modulacji i detekcji, przełączania, generacji i wzmacniania sygnałów elektrycznych.

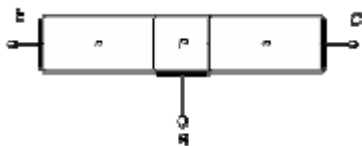
Każda dioda ma pewną częstotliwość graniczną, po przekroczeniu której nie zachowuje się jak dioda, lecz jak kondensator

Tranzystor - trójelektrodowy półprzewodnikowy element elektroniczny, posiadający zdolność wzmacniania sygnału elektrycznego

Tranzystory bipolarne, w których prąd wyjściowy jest funkcją prądu wejściowego (sterowanie prądowe).

Tranzystory unipolarne (tranzystory polowe), w których prąd wyjściowy jest funkcją napięcia (sterowanie napięciowe)

Tranzystor posiada trzy końcówki przyłączone do warstw półprzewodnika, nazywane: emiter (ozn. E), baza (ozn. B), kolektor (ozn. C)



Są wykorzystywane do budowy wzmacniaczy różnego rodzaju. Z tranzystorów buduje się także bramki logiczne realizujące podstawowe funkcje boolowskie, Tranzystory są także podstawowym budulcem wielu rodzajów pamięci półprzewodnikowych (RAM, ROM itp.)

33. Układy scalone.

Układ scalony jest zminiaturyzowanym układem elektronicznym, który może zawierać w sobie miliony elementów elektronicznych. Płytki krzemowe bo na nich najczęściej budowane są układy scalone stanowią podłoże półprzewodnikowe dla elementów elektronicznych jak diody, kondensatory, tranzystory lub rezystory.

Ze względu na sposób wykonania układy scalone dzieli się na główne grupy:

- **monolityczne**, w których wszystkie elementy, zarówno elementy czynne jak i bierne, wykonane są w monokrystalicznej strukturze półprzewodnika. Większość stosowanych obecnie układów scalonych jest wykonana właśnie w tej technologii.
- **hybrydowe** – na płytki wykonane z izolatora nanoszone są warstwy przewodnika oraz materiału rezystywnego, które następnie są wytrawiane, tworząc układ połączeń elektrycznych oraz rezystory. Do tak utworzonych połączeń dołącza się indywidualne, miniaturowe elementy elektroniczne (w tym układy monolityczne).

W najnowszych technologiach, w których między innymi produkowane są procesory Intel i AMD, minimalna długość bramki wynosi 10nm.

34. Układy impulsowe.

Układ sterowania lub regulacji, w którym sygnały wejściowe (sterujące), a często również sygnały wyjściowe (wielkości sterowane) mogą zmieniać swoje wartości tylko w pewnych chwilach czasu — uzależnionych od różnych czynników (np. gdy sygnał wejściowy przekracza pewną wartość).

Układy dyskretne opisywane są za pomocą równań różnicowych. Transmitancja operatorowa układów dyskretnych opiera się o przekształcenie Z. Transmitancją impulsową układu dyskretnego nazywa się stosunek transformaty Z odpowiedzi układu do transformaty Z sygnału wejściowego.

35. Układy cyfrowe.

Układy cyfrowe to rodzaj układów elektronicznych, w których sygnały napięciowe przyjmują tylko określoną liczbę poziomów, którym przypisywane są wartości liczbowe. Najczęściej (choć nie zawsze) liczba poziomów napięć jest równa dwa, a poziomom przypisywane są cyfry 0 i 1, wówczas układy cyfrowe realizują operacje zgodnie z algebrą Boole'a i z tego powodu nazywane są też układami logicznymi.

Obecnie układy cyfrowe budowane są w oparciu o bramki logiczne realizujące elementarne operacje znane z algebry Boola: iloczyn logiczny (AND, NAND), sumę logiczną (OR, NOR), negację NOT, różnicę symetryczną (XOR) itp.

Zalety układów cyfrowych:

- Możliwość bezstratnego kodowania i przesyłania informacji
- Zapis i przechowywanie informacji cyfrowej jest prostsze.
- Mniejsza wrażliwość na zakłócenia elektryczne.
- Możliwość tworzenia układów programowalnych, których działanie określa program

Wady układów cyfrowych:

- Są skomplikowane zarówno na poziomie elektrycznym, jak i logicznym i obecnie ich projektowanie wspomagają komputery
- Choć są bardziej odporne na zakłócenia, to wykrywanie przekłamań stanów logicznych wymaga dodatkowych zabezpieczeń (patrz: kod korekcyjny) i też nie zawsze jest możliwe wykrycie błędu.

Ze względu na sposób przetwarzania informacji rozróżnia się:

- układy kombinacyjne – układy „bez pamięci”, w których sygnały wyjściowe są zawsze takie same dla określonych sygnałów wejściowych.

- układy sekwencyjne – układy „z pamięcią”, w których stan wyjść zależy nie tylko od aktualnego stanu wejść, ale również od stanów poprzednich.

36. Pamięci półprzewodnikowe, magnetyczne, optyczne.

Rodzaj pamięci	Cechy
Pamięci półprzewodnikowe RAM, EEPROM	<ul style="list-style-type: none"> • Bardzo szybkie – czas dostępu rzędu ns (nanosekund) • Bardzo duży transfer danych – rzędu GB/s (Gigabajtów na sekundę) • Duża cena za jednostkę pojemności • Możliwość bezpośredniego sprzężenia z mikroprocesorami • Zapis, odczyt, zachowywanie danych: <ul style="list-style-type: none"> o Utrata informacji po odłączeniu zasilania (RAM) o Tylko do odczytu (ROM) o Trwały odczyt i zapis (EEPROM, Flash)
Pamięci dyskowe (magnetyczne - dysk twardy, taśma magnetyczna, optyczne - dysk CDROM, taśma filmowa)	<ul style="list-style-type: none"> • Wolne – duży czas dostępu rzędu ms (milisekund) • Średni transfer danych – rzędu dziesiątek MB/s (megabajtów na sekundę) • Niewielka cena za jednostkę pojemności • Wymagają specjalnych układów (interfejsowych – pośredniczących) • Podtrzymują dane bez zasilania • Możliwość realizacji jako pamięci wymiennych – nośnik jest oddzielony od czytnika

37. Przetworniki analogowo-cyfrowe i cyfrowo-analogowe.

Przetwornik C/A przetwarzający sygnał cyfrowy na sygnał analogowy w postaci prądu elektrycznego lub napięcia o wartości proporcjonalnej do tej liczby (rownoważny sygnał analogowy). Taki przetwornik ma n wejść i jedno wyjście.

Przetwornik A/C to układ służący do zamiany sygnału analogowego (ciągłego) na reprezentację cyfrową (sygnał cyfrowy). Dzięki temu możliwe jest przetwarzanie ich w urządzeniach elektronicznych opartych o architekturę zero-jedynkową oraz gromadzenie na dostosowanych do tej architektury nośnikach danych. Proces ten polega na uproszczeniu sygnału analogowego do postaci skwantowanej (dyskretnej), czyli zastąpieniu wartości zmieniających się płynnie do wartości zmieniających się skokowo w odpowiedniej skali (dokładności) odwzorowania.

38. Metody pomiarów wielkości elektrycznych.

Nazwa	Miernik	Podłączenie	Obwód
woltomierz	napięcia	rownolegle	
amperomierz	natężenia	szeregowo	zamknięty
omomierz	oporu	szeregowo	otwarty
watomierz	mocy	Zależy od rodzaju obwodu.	Podłączamy szeregowo gdy prądowy obwód podłączamy równolegle gdy napięciowy obwód

39. Metody pomiarów wielkości nieelektrycznych.

Wielkości nieelektryczne mierzy się dziś metodami elektrycznymi. Obecnie tylko te metody umożliwiają wykonywanie pomiarów wielkości szybko zmieniających się w czasie, prowadzenie tych pomiarów zdalnie i automatycznie. Obecnie metodami elektrycznymi mierzy się następujące wielkości: geometryczne (długość, kąt), kinetyczne (prędkość, przyspieszenie), dynamiczne (siłę, ciężar), akustyczne (natężenie dźwięku), optyczne (natężenie oświetlenia), cieplne (temperaturę), fizykochemiczne materii (lepkość, wilgotność) W pomiarach tych wielkości nieelektryczne muszą być zamienione na wielkości elektryczne. Do tego służą **przetworniki pomiarowe**. Wielkości elektryczne pojawiające się na wyjściu przetwornika są następnie mierzone metodami elektrycznymi, przy czym przyrządy pomiarowe mogą być wyskalowane w jednostkach mierzonych wielkości nieelektrycznych.

Pomiaru temperatury można dokonywać na dwa sposoby – **kontaktowo i bezkontaktowo**. Pierwsza z metod ze względu na niższy koszt oraz stosunkowo wysoką dokładność jest używana zdecydowanie częściej, a wykorzystuje się w niej czujniki rezystancyjne oraz termopary. Pomiaru bezkontaktowego można dokonywać

wykorzystując pirometry lub kamery termowizyjne. Jego zaletą jest możliwość kontroli temperatury w trudno dostępnych miejscach.

Pomiaru ciśnienia dokonuje się przy pomocy:

tensometrycznych czujników ciśnienia. Zasada działania tych czujników opiera się na wykorzystaniu właściwości drutu oporowego, który podczas przyłożenia siły odkształca się, w wyniku czego zmienia się jego rezystancja.

czujników piezorezystancyjnych. których zasada działania jest podobna do czujników tensometrycznych, jednak ciśnieniu danego medium poddawana jest powierzchnia krzemowa. Ich zaletami są większa czułość oraz dokładność.

czujników pojemnościowych. Cienka, krzemowa membrana jest umieszczana między okładzinami, w ten sposób tworząc kondensator. Ciśnienie danego medium wpływa na pojemność tak zbudowanego kondensatora, która jest przekształcana na sygnał wyjściowy.

41. Grafy. Grafy eulerowskie i hamiltonowskie.

Graf to zbiór wierzchołków, które mogą być połączone krawędziami, w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków.

Graf hamiltonowski to graf zawierający ścieżkę (drogę) przechodzącą przez każdy wierzchołek dokładnie jeden raz zwaną ścieżką Hamiltona. W szczególności grafem hamiltonowskim jest graf zawierający cykl Hamiltona, tj. zamkniętą ścieżkę Hamiltona. W niektórych źródłach graf zawierający tylko ścieżkę Hamiltona nazywany jest grafem półhamiltonowskim.

Graf eulerowski, graf Eulera – graf eulerowski odznacza się tym, że da się w nim skonstruować cykl Eulera, czyli drogę, która przechodzi przez każdą jego krawędź dokładnie raz i wraca do punktu wyjściowego.

Liczba krawędzi stykających się z danym wierzchołkiem nazywana jest jego stopniem. Jeżeli wszystkie wierzchołki grafu nieskierowanego mają stopień parzysty, to znaczy, że da się skonstruować zamkniętą ścieżkę Eulera nazywaną cyklem Eulera. Jeżeli najwyżej dwa wierzchołki mają nieparzysty stopień, to możliwe jest zbudowanie tylko takiej ścieżki Eulera, która nie jest zamknięta. Graf zawierający cykl Eulera jest nazywany grafem eulerowskim, a graf posiadający jedynie ścieżkę Eulera nazywany jest półeulerowskim.

Graf skierowany - ruch może odbywać się tylko w kierunkach wyznaczonych przez krawędzie. Poruszanie się "pod prąd" jest zabronione. Każdy wierzchołek posiada pewną liczbę krawędzi wejściowych nazywaną stopniem wchodzącym. Analogicznie ilość krawędzi wychodzących to stopień wychodzący. Skierowany graf eulerowski definiowany jest jako graf silnie spójny, w którym dla każdego wierzchołka grafu liczba krawędzi wchodzących jest równa ilości krawędzi wychodzących.

42. Kolorowanie grafów, definicja liczby chromatycznej grafu i indeksu chromatycznego. (faworyt)

Kolorowanie grafu polega na przyporządkowaniu koloru każdemu wierzchołkowi w grafie przy wykorzystaniu jak najmniejszej liczby kolorów (liczby chromatycznej). Jednakże przy założeniu, że dwa różne ale przyległe do siebie wierzchołki nie będą miały tego samego koloru. Takie kolorowanie stosuje się często na mapach do oznaczenia terenów lub do określenia jak składować chemikalia bo jak wiadomo niektóre substancje po zetknięciu się ze sobą mogą spowodować eksplozję.

Kolorowanie grafu polega w ogólności na przypisaniu określonym elementom składowym grafu (najczęściej wierzchołkom, rzadziej krawędziom lub ścianom) wybranych kolorów według ściśle określonych reguł. Klasyczne (czyli wierzchołkowe) kolorowanie grafu jest związane z przypisaniem wszystkim wierzchołkom w grafie jednej z wybranych barw w ten sposób, aby żadne dwa sąsiednie wierzchołki nie miały tego samego koloru. Innymi słowy, pewne pokolorowanie wierzchołkowe jest poprawne (legalne, dozwolone) wtedy, gdy końcom żadnej krawędzi nie przypisano tego samego koloru.

Podstawowe definicje

- **Klasyczne (wierzchołkowe) kolorowanie grafu** - przyporządkowywanie wierzchołkom grafu

liczb naturalnych w taki sposób, aby końce żadnej krawędzi nie miały przypisanej tej samej liczby. Ze względów historycznych oraz dla lepszego zobrazowania problemu mówi się o kolorowaniu, przy czym różnym kolorom odpowiadają różne liczby.

- **Pokolorowaniem wierzchołków grafu** nazywamy jedno konkretne przyporządkowanie kolorów wierzchołkom. Pokolorowanie jest legalne (dozwolone), gdy końcom żadnej krawędzi nie przyporządkowano tego samego koloru.

- **Optymalnym pokolorowaniem danego grafu** nazywamy legalne pokolorowanie zawierające najmniejszą możliwą liczbę kolorów.

Liczba chromatyczna - w teorii grafów jest to liczba kolorów (liczb) niezbędna do optymalnego klasycznego (wierzchołkowego) pokolorowania grafu, czyli najmniejsza możliwa liczba k taka, że możliwe jest legalne pokolorowanie wierzchołków grafu k kolorami. Oznacza się ją symbolem $\chi(G)$.

Algorytmy kolorowania grafów

Algorytm LF (largest first)

Kolorowanie grafu za pomocą algorytmu LF można opisać następująco:

- Uporządkuj wierzchołki grafu malejąco według ich stopni (liczby krawędzi z nich wychodzących).
- Koloruj wierzchołki zachłannie, zgodnie z ustaloną wcześniej kolejnością (zaczynając od wierzchołka o największym stopniu).

Algorytm LF jest algorytmem statycznym, gdyż raz ustalona kolejność wierzchołków nie zmienia się w trakcie jego działania. Najmniejszym dość trudnym grafem jest ścieżka P6.

Algorytm SL (smallest last)

Algorytm SL wygląda następująco:

- Znajdź wierzchołek o minimalnym stopniu i usuń go z grafu.
- Powtarzaj krok pierwszy tak długo, aż graf będzie pusty (zapamiętaj kolejność usuwanych wierzchołków).
- Koloruj wierzchołki zachłannie, zgodnie z ustaloną wcześniej kolejnością (zaczynając od wierzchołków usuniętych później).

Algorytm SL jest statyczny, jego złożoność wynosi $O(n + m)$, gdzie n - liczba wierzchołków, m - liczba krawędzi.

Algorytm SLF (saturated largest first)

Kolorowanie grafu przy pomocy algorytmu SLF polega na wykonaniu poniższych czynności:

Dopóki istnieją nie pokolorowane wierzchołki wykonuj operacje:

- znajdź wierzchołek o maksymalnym stopniu
- spośród wierzchołków o maksymalnym stopniu nasycenia pokoloruj znaleziony wierzchołek zachłannie

Stopniem nasycenia wierzchołka nazwiemy tu liczbę różnych kolorów sąsiednich z tym wierzchołkiem. Złożoność algorytmu SLF wynosi $O(m \log n)$.

Indeks chromatyczny grafu określa minimalną liczbę kolorów wystarczającą do prawidłowego pokolorowania krawędzi grafu. Indeks chromatyczny grafu jest równy liczbie chromatycznej jego

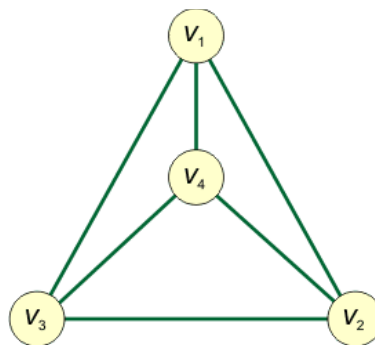
grafu krawędziowego.

Znalezienie indeksu chromatycznego jest problemem NP-trudnym, mimo że można bardzo dokładnie oszacować jego wartość.

44. Grafy planarne.

Graf nazywamy planarnym wtedy i tylko wtedy gdy posiada taką interpretację geometryczną na płaszczyźnie, że łuki reprezentujące krawędzie nie przecinają się (poza końcami reprezentującymi wierzchołki)

Na przykład dla grafu K_4 możliwa jest taka interpretacja graficzna, gdzie żadna z krawędzi się nie przecina, więc graf ten jest grafem planarnym.



Takiej interpretacji nie znajdziemy na przykład dla grafu K_5 , więc nie jest on grafem planarnym.

Twierdzenie Kuratowskiego:

1. Graf skończony jest planarny wtedy i tylko wtedy, gdy nie zawiera podgrafu homeomorficznego z grafem K_5 ani z grafem $K_{3,3}$

Grafy G i H są homeomorficzne, gdy istnieje graf F taki, że zarówno G jak i H można otrzymać z F poprzez zamianę niektórych krawędzi ścieżkami.

2. Graf jest planarny wtedy i tylko wtedy gdy nie zawiera K_5 ani $K_{3,3}$ jako minora.

Graf H jest minorem G , gdy możemy otrzymać H poprzez:

- usuwanie wierzchołków i krawędzi
- ściąganie krawędzi

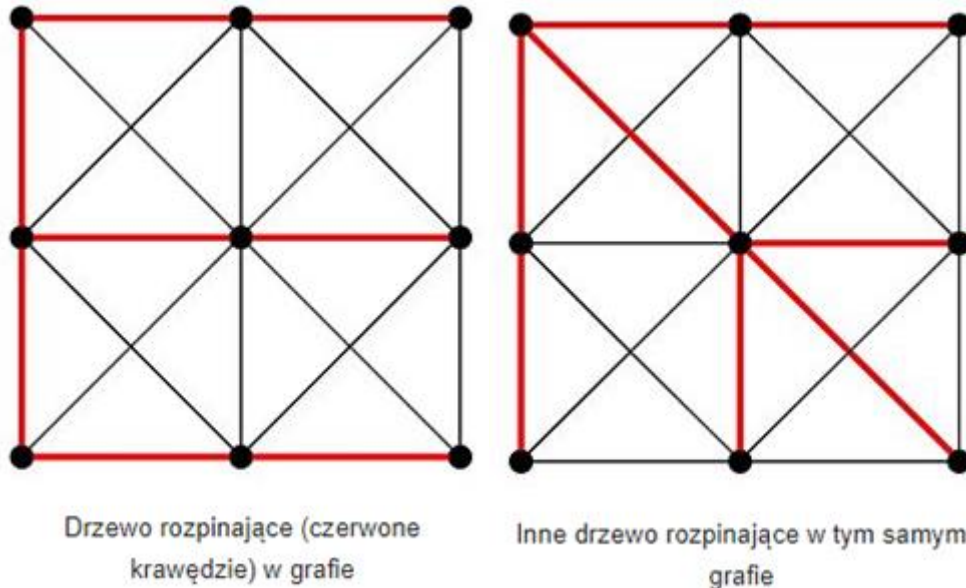
Twierdzenie Eulera:

Dowolny rysunek płaski grafu planarnego wyznacza spójne obszary płaszczyzny zwane ścianami. Dokładnie jeden z tych obszarów, zwany ścianą zewnętrzną, jest nieograniczony.

Zgodnie ze wzorem Eulera, jeżeli G jest grafem spójnym i planarnym, to $V - E + S = 2$, gdzie V – zbiór wierzchołków, E – zbiór krawędzi, S – zbiór ścian dowolnego rysunku płaskiego grafu G .

45. Drzewa spinające grafu. Minimalne drzewa spinające.

Drzewem spinającym grafu G nazywamy drzewo, które zawiera wszystkie wierzchołki grafu G , zaś zbiór krawędzi drzewa jest podzbiorem zbioru krawędzi grafu. Konstrukcja drzewa spinającego polega na usuwaniu z grafu tych krawędzi, które należą do cykli. Minimalne drzewo spinające jest to drzewo spinające danego grafu o najmniejszej z możliwych wag.



Algorytmy znajdujące dla zadanego grafu minimalne drzewo rozpinające:

- Algorytm Prima (nazywany też algorytmem Dijkstry-Prima)
- Algorytm Kruskala

47. Rodzaje pętli. Uwarunkowanie zastosowań. Problem równoważności pętli w wybranym języku programowania.

Pętla stanowi konstrukcję językową służącą do wielokrotnego powtarzania szeregu instrukcji. Rodzaje pętli (bardziej powszechne):

- **pętla ogólna** - kolejne iteracje wykonywane są, jeśli zdefiniowana zmienna sterująca zmieniająca się z każdą iteracją w wskazany przez programistę sposób spełnia zdefiniowany warunek, używana, przy znanej liczbie iteracji (pętla for...)
- **pętla "po kolekcji"** - kolejne iteracje są wykonywane dla każdego elementu tablicy, kolekcji, itd. (pętla foreach...)
- **pętla warunkowa** (repetycyjna) - wykonywanie kolejnych iteracji zależne jest od spełnienia zdefiniowanego warunku (pętle while... i do...while)

oraz rzadziej spotykane (występują w obecnie mniej popularnych językach):

- pętla powtórzeniowa
- pętla złożona
- pętla nieskończona

Przykład równoważnego zastosowania dwóch różnych pętli w języku python:

```

1 numbers = [4, 10, 6, 3]
2 total = 0
3
4 for num in numbers:
5     total += num
6
7 print(total) # 23

```

```

1 numbers = [4, 10, 6, 3]
2 total = 0
3 i = 0
4
5 while i < len(numbers):
6     total += numbers[i]
7     i += 1
8
9 print(total) # 23

```

48. Zmienne typu adresowego (wskaźniki).

Implementacja w wybranym języku programowania.

Zmienne typu wskaźnikowego przechowują w pamięci adres innej zmiennej.

W C# wskaźnik może trzymać adres jedynie dla typów wartościowych (bool, byte, char, decimal, double, float, int, long, sbyte, short, uint, ulong, ushort) i tablic.

Dodatkowo, wszelkie instrukcje na wskaźnikach muszą być w C# zawarte w bloku „unsafe”.

Przykład użycia:

```
int * ptr //deklaracja wskaźnika zmiennej ptr, która składowe adres dla zmiennej typu int.
```

```
int x = 100 //deklarujemy sobie jakąś zmienną typu int z wartością równą 100
```

```
//możemy wówczas użyć operatora '&' aby pozyskać jej adres w pamięci.
```

```
ptr = &x //wyrażenie &x zwraca nam adres pamięci zmiennej x, którą możemy
//przypisać do utworzonej wcześniej zmiennej wskaźnikowej
```

```
Console.WriteLine(int(ptr))
```

^ Zwróci nam więc adres pamięci zmiennej x (z pomocą rzutowania na typ int)

```
Console.WriteLine(*ptr)
```

^ Zwróci nam wartość zmiennej x

49. Funkcje (z wzmianką o procedurach). Przekazywanie parametrów przez wartość i referencję lub adres

Podprogram (częściej nazywany funkcją, rzadziej procedurą) to część programu, która przetwarza argumenty i ewentualnie zwraca wartość, która może być wykorzystana w innych działaniach lub funkcjach. Funkcja może posiadać własne zmienne lokalne.

W większości języków funkcja i procedura to to samo, ale w takich językach jak Pascal czy Ada podprogram dzieli się na:

Funkcje- ma wykonywać obliczenia i zwracać jakąś wartość, nie powinna natomiast mieć żadnego innego wpływu na działanie programu

Procedury- natomiast nie zwraca żadnej wartości, zamiast tego wykonuje pewne działania

Definicja funkcji – mówi jak działa funkcja tzn. nazwa, argumenty i return

Standardowo parametry (argumenty) funkcji są przekazywane przez wartość. Czyli funkcja operuje na kopii zmiennej która została do niej przekazana.(nie zmienia wartości w podprogramie głównym). Po przekazaniu parametru przez referencję (np. w języku C void func(int& x)) możemy zmieniać wartość zmiennej w miejscu, z którego funkcja została wezwana. Drugą opcją zmienienia wartości w funkcji matce jest przekazanie wskaźnika przez wartość (funkcja robi kopię wskaźnika i operuje na kopii (w języku C dodajemy & do argumentu przed przekazaniem go do funkcji i * przy określaniu parametrów w funkcji, jak również kiedy chcemy zmienić ich wartość, jeżeli tego nie zrobimy to będziemy modyfikować wskaźnik do parametru).

50. Cechy programowania strukturalnego. Kluczowe różnice między programowaniem strukturalnym a obiektowym.

Cechy programowanie strukturalnego –

- Podzielone bloki kodu mają jeden punkt wejścia (mogą mieć wiele punktów wyjścia)
- Wykonywanie wyrażeń w określonej kolejności
- Używanie instrukcji warunkowych (if , if else)
- Używanie pętli (for, while, do while)
- Unikanie instrukcji skoku (goto)
- Unikanie instrukcji break, continue

Różnice – W programowaniu strukturalnym bazuje się na podejściu top down (z góry na dół). Skupiamy się na pisaniu funkcji. Programowanie obiektowe skupia się stricte na danych. Programując obiektowo, programy budujemy od dołu, zaczynając od małych elementów, które składowy w całość. Skupiamy się na pisaniu klas i obiektów. Oprócz tego w programowaniu obiektowym uprościło się także samo testowanie kodu, ponieważ możemy wziąć pojedyncze obiekty i przetestować ich działanie. W przypadku programowania proceduralnego nie było to łatwe, kod był jedną całością i ciężko było testować pojedyncze fragmenty.

52. Zastosowanie składników statycznych w klasie. Statyczne funkcje składowe. Podać przykłady.

Czasami zachodzi potrzeba dodania elementu, który jest związany z *klasą*, ale nie z konkretną *instancją* tej klasy. Możemy wtedy stworzyć element **statyczny**. Element **statyczny** jest właśnie elementem, który jest powiązany z *klasą*, a nie z *obiekt*em tej klasy, czyli np. statyczna metoda nie może się odwołać do niestatycznej zmiennej lub funkcji.

Elementy **statyczne** poprzedza się podczas definicji słówkiem *static*. **Statyczne** mogą być zarówno *funkcje*, jak i *pola* należące do klasy.

```
class Klasa {
    protected:
        static int iloscInstancji; // pole statyczne
    public:
        Klasa() {
            iloscInstancji++;
        }
        virtual ~Klasa() {
            iloscInstancji--;
        }
        static int IloscInstancji() {
            return iloscInstancji;
        }
};
```

W powyższym przykładzie ponadto istnieje **metoda statyczna**. Z takiej metody nie można się odwołać do niestatycznych elementów klasy. Zarówno do **klasy statycznej** jak do **statycznego pola** możemy się odwołać nawet jeżeli nie został stworzony żaden obiekt klasy *Klasa*.

Odwołanie się do **metody statycznej** *IloscInstancji* z programu wymaga następująco:

```
int i=Klasa::IloscInstancji();
```

Gdyby zaś pole *iloscInstancji* było publiczne, a nie chronione, to moglibyśmy się do niego odwołać poprzez:

```
int i=Klasa.iloscInstancji;
```

Statyczne funkcje składowe

Takie funkcje można wywołać nawet wtedy, gdy nie istnieje jeszcze żaden obiekt klasy. Do jej nazwy z zewnątrz klasy odwołujemy się poprzez nazwę klasy za pomocą operatora zasięgu, czyli „czterokropka” (‘ ::’), albo za pomocą operatora wyboru składowej (kropka). Ponieważ funkcja statyczna *nie* jest wywoływana na rzecz obiektu, ale jak funkcja globalna, nie można w niej odwoływać się do **this** ani do żadnych składowych niestatycznych - te bowiem istnieją tylko wewnątrz konkretnych obiektów i w każdym z nich mogą być różne. Można natomiast w funkcjach statycznych klasy odwoływać się do składowych statycznych tej klasy: innych funkcji statycznych i zmiennych klasowych (określanych przez statyczne pola klasy).

54. Polimorfizm. Deklarowanie funkcji wirtualnych. Mechanizm wywołania. Podać przykłady.

Polimorfizm

praktyczne wykorzystanie łańcucha dziedziczenia klas. Pozwala na użyciu bardziej wyspecyfikowanego obiektu w miejscu, gdzie wymagany jest jego rodzic. Umożliwia to napisanie dwóch metod o różnym ciele i wywołanie ich metod w zależności od okoliczności.

Przykład w Javie:

```

public abstract class Animal {
    public abstract String speak();
}

public class Dog extends Animal {
    @Override
    public String speak() {
        return "Meow!";
    }
}

public class Cat extends Animal {
    @Override
    public String speak() {
        return "Woof!";
    }
}

List<Animal> animals = List.of(new Dog(), new Cat());
for (Animal animal : animals) {
    // Outputs:
    // Woof!
    // Meow!
    System.out.println(animal.speak());
}

```

Rolę funkcji wirtualnych w Javie przyjmują funkcje abstrakcyjne, czyli takie, które nie są zdefiniowane. Powyższy przykład ilustruje ich zastosowanie na przykładzie abstrakcyjnej metody `speak()` w klasie `Animal`, która jest następnie nadpisywana (adnotacja `@Override`) w klasach pochodnych.

Polimorfizm - mechanizmy pozwalające programiście używać wartości, zmiennych i podprogramów na kilka różnych sposobów. Inaczej mówiąc jest to możliwość wyodrębnienia wyrażeń od konkretnych typów. Podczas pisania programu wygodnie jest traktować nawet różne dane w jednolity sposób. Niezależnie czy należy wydrukować liczbę czy napis, czytelniej (zazwyczaj) jest gdy operacja taka nazywa się po prostu drukuj, a nie drukuj_liczbę i drukuj_napis. Jednak napis musi być drukowany inaczej niż liczba, dlatego będą istniały dwie implementacje polecenia drukuj ale nazwanie ich wspólną nazwą tworzy wygodny abstrakcyjny interfejs niezależny od typu drukowanej wartości.

Funkcje wirtualne to specjalne funkcje składowe, które przydają się szczególnie, gdy używamy obiektów posługując się wskaźnikami lub referencjami do nich. Dla zwykłych funkcji z identycznymi nazwami to, czy zostanie wywołana funkcja z klasy podstawowej, czy pochodnej, zależy od typu wskaźnika, a nie tego, na co faktycznie on wskazuje.

Dysponując funkcjami wirtualnymi będziemy mogli użyć prawdziwego polimorfizmu - używać metod klasy pochodnej wszędzie tam, gdzie spodziewana jest klasa podstawowa. W ten sposób będziemy mogli korzystać z metod klasy pochodnej korzystając ze wskaźnika, którego typ odnosi się do klasy podstawowej.

```

class Baza
{
public:
    void pisz()
    {
        std::cout << "Tu funkcja pisz z klasy Baza" << std::endl;
    }
};

class Baza2 : public Baza
{
public:
    void pisz()
    {
        std::cout << "Tu funkcja pisz z klasy Baza2" << std::endl;
    }
};

```

Jeżeli teraz w funkcji main stworzymy wskaźnik do obiektu typu Baza, to możemy ten wskaźnik ustawiać na dowolne obiekty tego typu. Można też ustawić go na obiekt typu pochodnego, czyli Baza2:

```

int main()
{
    Baza   *wsk;
    Baza   objB;
    Baza2  objB2;

    wsk = &objB;
    wsk -> pisz();

    // Teraz ustawiamy wskaźnik wsk na obiekt typu pochodnego

    wsk = &objB2;
    wsk -> pisz();
    return 0;
}

```

Po skompilowaniu na ekranie zobaczymy dwa wypisy: "Tu funkcja pisz z klasy Baza". Stało się tak dlatego, że wskaźnik jest do typu Baza.

Można jednak określić żeby kompilator nie sięgał po funkcję z klasy bazowej, ale sam się zorientował na co wskaźnik pokazuje. Do tego służy przydomek virtual, a funkcja składowa nim oznaczona nazywa się wirtualną. class Baza

```
class Baza
{
public:
    virtual void pisz()
    {
        std::cout << "Tu funkcja pisz z klasy baza" << std::endl;
    }
};

class Baza2 : public Baza
{
public:
    virtual void pisz()
    {
        std::cout << "Tu funkcja pisz z klasy Baza2" << std::endl;
    }
};
```

Gdy funkcja jest oznaczona jako wirtualna, kompilator nie przypisuje na stałe wywołania funkcji z tej klasy, na którą pokazuje wskaźnik, już podczas kompilacji.

55. Definiowanie szablonu klasy. Korzystanie z szablonu.

Szablony są niezwykle ważnym elementem programowania, umożliwiają one dynamiczne dostosowanie kodu klasy do naszych potrzeb, a to dynamiczne dostosowanie wykonywane jest w trakcie jego kompilacji. Umożliwiają one opisanie za pomocą jednego kodu klas, czy metod klas, które będą się różniły jedynie typem danych w nich zawartych.

Cechy:

- pisanie kodu, który może być używany w przyszłości, wpływa na bezpieczeństwo typów oraz wydajność;

- tworzenie generycznych kolekcji

- tworzenie własnych generycznych interfejsów, klas, metod, zdarzeń oraz delegatów;

- tworzenie generycznych klas, które pozwalają na dostęp do metod dla poszczególnych typów danych;

pobranie informacji dotyczących używanych typów w trakcie wykonywania programu – za pomocą mechanizmu refleksji.

Przykład:

```
class Program
{
    static void Main(string[] args)
    {
        KlasaGeneryczna<int> obiektKlasyGenerycznej = new
        KlasaGeneryczna<int>();
        obiektKlasyGenerycznej.Dodaj(12);
        KlasaGeneryczna<string> obiektKlasyGenerycznej1 = new
        KlasaGeneryczna<string>();
        obiektKlasyGenerycznej1.Dodaj("napis");
        Console.ReadKey();
    }
}
```

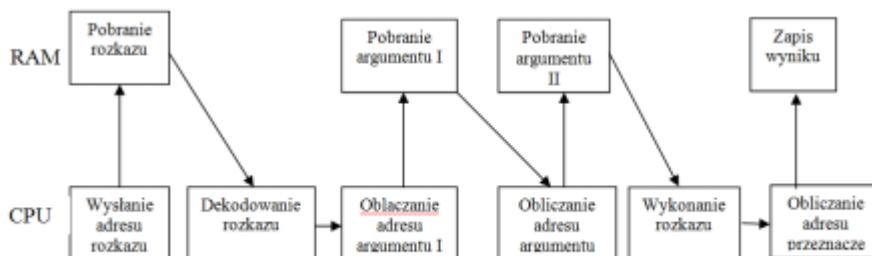
```

public class KlasaGeneryczna<T>
{
    public void Dodaj(T parametr)
    {
        Console.WriteLine("Wynik metody Dodaj: " + parametr);
    }
}
}

```

56. Cykl pracy procesora przy wykonaniu programu.

Typowy cykl rozkazowy w systemie o architekturze von Neumanna zaczyna się od pobrania rozkazu z pamięci i przesłania go do rejestru rozkazów (ang. instruction register). Rozkaz jest następnie dekodowany i realizowany (może spowodować pobranie argumentów z pamięci i umieszczenie ich w innym rejestrze wewnętrznym). Po wykonaniu rozkazu na argumentach jego wynik można z powrotem przechować w pamięci. Zauważmy, że jednostka pamięci „widzi” tylko strumień adresów pamięci. Nie jest jej znany sposób, w jaki one powstały (licznik rozkazów, indeksowanie, modyfikacje pośrednie, adresy literalne itp.) ani czemu służą.



57. Procesy, zarządzanie procesami.

Proces – egzemplarz wykonywanego programu. Aplikacja może składać się z większej liczby procesów. Każdy nowo powstały proces otrzymuje unikatowy numer, który go jednoznacznie identyfikuje, tzw. PID (od [ang.](#) process identifier). Należy odróżnić jednak proces od wątku. Każdy proces posiada własną przestrzeń adresową, natomiast wątki posiadają wspólną sekcję danych. Każdemu procesowi przydzielone zostają zasoby, takie jak: procesor, pamięć, dostęp do urządzeń wejścia-wyjścia, pliki.

Każdy proces posiada tzw. "rodzica". W ten sposób tworzy się swego rodzaju drzewo procesów. Proces może (ale nie musi) mieć swoje procesy potomne. Za zarządzanie procesami odpowiada jądro systemu operacyjnego. Wykonanie musi przebiegać sekwencyjnie.

Może przyjmować kilka stanów: działający, czekający na udostępnienie przez system operacyjny zasobów, przeznaczony do zniszczenia, właśnie tworzony lub proces zombie

W skład procesu wchodzi: kod programu, licznik rozkazów, stos, dane

58. Metody przydziału pamięci operacyjnej procesowi.

Organizacja pamięci wirtualnej.

First-Fit - Jest to najprostsza możliwa strategia. Wybierany jest pierwszy (pod względem adresów) wolny obszar, który jest wystarczająco duży.

Best-Fit - Wybieramy najmniejszy wolny obszar, który jest wystarczająco duży.

Worst-fit - Przydzielamy pamięć zawsze z największego wolnego obszaru (oczywiście, o ile jest on wystarczająco duży).

Segmentacja Pamięć wykorzystywana przez proces, z logicznego punktu widzenia, nie stanowi jednego spójnego obszaru. Zwykle składa się z kilku segmentów. Typowe segmenty to: kod programu, zmienne globalne, stos i sarta. System operacyjny może więc przydzielać procesom pamięć nie w postaci jednego spójnego bloku, ale kilku takich bloków, segmentów. Co więcej, proces może w trakcie działania prosić o przydzielenie lub zwolnienie segmentów.

Stronicowanie Podział wirtualnej przestrzeni adresowej procesu na strony. Strona to obszar ciągłej pamięci o stałym rozmiarze (zazwyczaj 4kB). Rzeczywista pamięć operacyjna podzielona jest na ramki, których rozmiar odpowiada wielkości stron. System operacyjny według uznania może przydzielać ramkom strony pamięci lub pozostawiać je puste.

Pamięć wirtualna jest techniką programową a także sprzętową gospodarowania pamięcią operacyjną RAM, pozwalającą na przydzielanie pamięci dla wielu procesów, zwalnianie jej i powtórne przydzielanie, w ilości większej niż rzeczywista ilość pamięci fizycznej zainstalowanej w komputerze poprzez przeniesienie danych z ostatnio nie używanej pamięci do pamięci masowej (np. twardego dysku), w sytuacji gdy procesor odwołuje się do danych z pamięci przeniesionej na dysk przesuwa się te dane do pamięci w wolne miejsce, a gdy brak wolnej pamięci zwalnia się ją przez wyżej opisane przerzucenie jej na dysk.

59. Funkcje systemowe – podstawowe kategorie, przykłady.

Programy użytkowników mają dostęp do usług systemu operacyjnego poprzez tzw. funkcje systemowe.

Funkcje te są widoczne w języku programowania podobnie jak funkcje z tego języka, a ich konkretny mechanizm wywoływania jest ukryty przed programistą.

Funkcje udostępniane przez system operacyjny możemy pogrupować w następujący sposób:

- operacje na procesach Ta grupa funkcji obejmuje m.in.: tworzenie nowych procesów, uruchamianie

programów, zakończenie się procesu, przerwanie działania innego procesu, pobranie informacji o procesie, zawieszenie i wznowienie procesu, oczekiwanie na określone zdarzenie, przydzielanie i zwalnianie pamięci,

- operacje na plikach Ta grupa funkcji obejmuje takie operacje, jak: utworzenie pliku, usunięcie pliku, otwarcie pliku, odczyt z pliku, zapis do pliku, pobranie lub zmiana atrybutów pliku.
- operacje na urządzeniach Zamówienie i zwolnienie urządzenia, odczyt z i zapis do urządzenia, pobranie lub zmiana atrybutów urządzenia, (logiczne) przyłączenie lub odłączenie urządzenia.
- informacje systemowe Funkcje te obejmują pobieranie i/lub ustawianie najrozmaitszych informacji systemowych, w tym: czasu i daty, wersji systemu operacyjnego, atrybutów użytkowników itp.
- komunikacja Ta grupa funkcji obejmuje przekazywanie informacji między procesami. Spotykane są dwa schematy komunikacji: przekazywanie komunikatów i pamięć współdzielona. Przekazywanie komunikatów polega na tym, że jeden proces może wysłać (za pośrednictwem systemu operacyjnego) pakiet informacji, który może być odebrany przez drugi proces. Mechanizm pamięci współdzielonej polega na tym, że dwa procesy uzyskują dostęp do wspólnego obszaru pamięci.

60. Szeregowanie procesów. Wybrane algorytmy szeregowania.

Algorytm szeregowania (ang. scheduler, planista) - część jądra wielozadaniowego systemu operacyjnego, odpowiedzialna za przydzielanie dostępu do procesora dla poszczególnych procesów. Planista musi także uwzględniać priorytety procesów i ich gotowość do wykonania oraz przeciwdziałać zagłodzeniu procesu poprzez przedłużający się brak dostępu do zasobów oraz tzw. inwersji priorytetów.

Rozróżniamy dwa typy planistów: krótkoterminowi i długoterminowi. Planista krótkoterminowy odpowiada za wybór procesów spośród gotowych do wykonania. Musi być on bardzo szybki, w przeciwieństwie do planisty długoterminowego, który wybiera procesy z pamięci masowej i ładuje do pamięci operacyjnej.

Najpopularniejsze rodzaje algorytmów szeregowania

FIFO - algorytm powszechnie stosowany, jeden z prostszych w realizacji, dający dobre efekty w systemach ogólnego przeznaczenia; zadanie wykonuje się aż nie zostanie wywłaszczone przez

siebie lub inne zadanie o wyższym priorytecie.

Planowanie priorytetowe - wybierany jest proces o najwyższym priorytecie. W tej metodzie występuje problem nieskończonego blokowania (procesu o niskim priorytecie przez procesy o wysokim priorytecie). Stosuje się tu postarzanie procesów, polegające na powolnym podnoszeniu priorytetu procesów zbyt długo oczekujących.

Planowanie rotacyjne ('round-robin') - Planista przydziela każdemu z procesów kwant czasu i jeżeli nie zdąży się wykonać to łąduje na końcu kolejki procesów gotowych.

W systemach operacyjnych czasu rzeczywistego (stosowanych m. in. w automatyce) najważniejszym zadaniem algorytmu szeregowania jest zapewnienie, by wykonanie danego procesu zakończyło się przed upływem zdefiniowanych dla niego ograniczeń czasowych. Opracowano w tym celu deterministyczne algorytmy szeregowania, takie jak RMS, EDF i inne.

Algorytm wywłaszczający - Algorytm, który może przerwać wykonanie procesu i przenieść go z powrotem do kolejki.

Algorytm niewywłaszczający - Algorytm, w którym procesy przełączają się dobrowolnie. Proces aktywny (wykonujący się) jest przenoszony do kolejki procesów oczekujących tylko wtedy, gdy sam przerwie (wstrzyma, zawiesi) swe działanie; dopóki tego nie uczyni (lub nie zakończy działania), żaden inny proces nie otrzyma dostępu do procesora.

62. Cykle projektowania i życia oprogramowania.

I. Tworzenie koncepcji:

- zdefiniowanie głównej idei projektu
- dokonanie analizy ewentualnych sprzecznych interesów, problemów
- określenie pożądanego stanu przyszłego
- przeprowadzenie rozeznania w zakresie: możliwości realizacyjnych, stanu aktualnego porównanego z preferowanym efektem końcowym, identyfikacji klienta i jego oczekiwań.

II. Definiowanie projektu:

- określenie struktury zarządzania projektem: kierownik projektu oraz członkowie zespołu realizującego projekt
- dokonanie analizy możliwości realizacyjnych koncepcji (wykonalności)
- wstępne oszacowanie kosztów

- sporządzenie wstępnego harmonogramu
- zaplanowanie parametrów jakości
- przygotowanie analizy otoczenia projektu, czyli czynników, które mogą mieć pozytywny lub negatywny wpływ na jego realizację.

III. Sporządzenie planu projektu:

- określenie niezbędnych zadań i czynności w realizacji projektu oraz ułożenie ich w logicznej kolejności
- zatwierdzenie struktury zarządzania projektem
- doprecyzowanie parametrów: czasu, kosztów i jakości oraz ich zoptymalizowanie w miarę potrzeb i możliwości
- dokonanie podziału obowiązków
- wyznaczenie osób odpowiedzialnych za kontrolę przebiegu planu projektu
- przygotowanie planów awaryjnych.

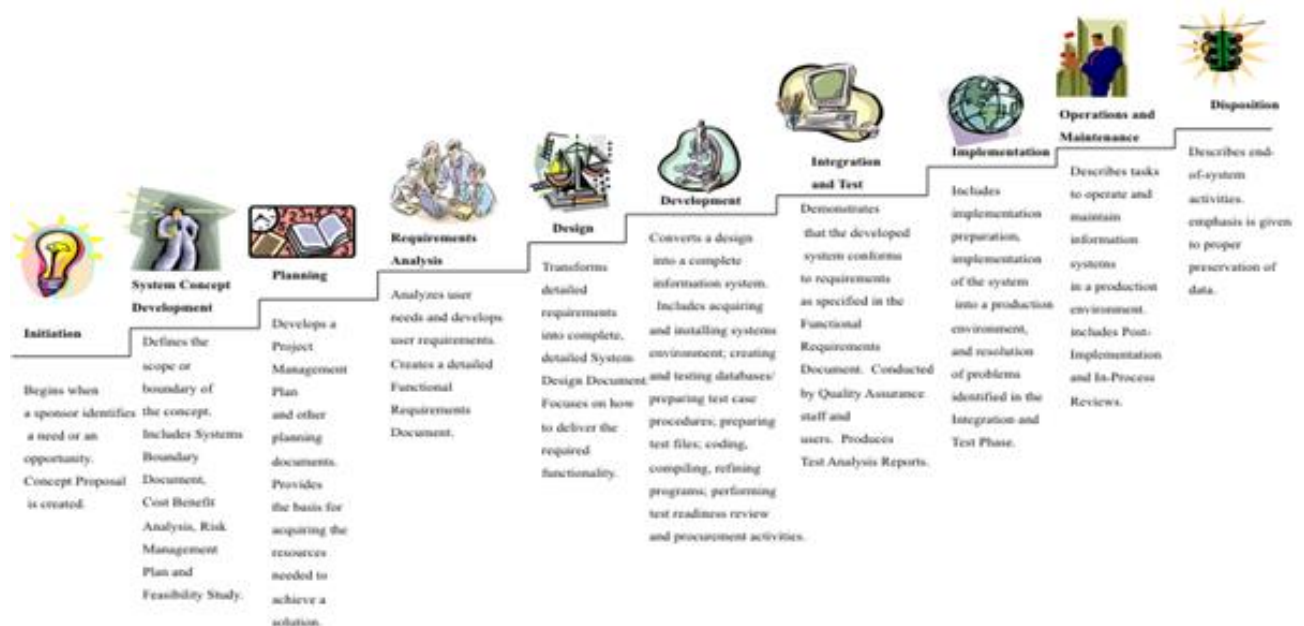
IV. Realizacja projektu:

- trzymanie się założonego planu
- wprowadzenie korekt i usprawnień czasowych, kosztowych i jakościowych
- śledzenie postępów w odniesieniu do planu projektu
- rozwiązywanie bieżących problemów
- identyfikacja ewentualnych odstępstw od planu projektu
- testowanie projektów
- oddanie i przekazanie odpowiedzialności za ich użytkowanie właścicielom
- uzyskanie pisemnej akceptacji produktów od klienta, użytkownika, osób i instytucji finansujących realizację projektu.

V. Zakończenie projektu:

- dokonanie oceny i opracowanie raportu dotyczącego parametrów: czasu, kosztów i jakości
- dokonanie oceny struktury zarządzania projektem i rozwiązanie zespołów
- zestawienie uzyskanego wyniku z założeniami planu projektu
- archiwizowanie dokumentacji projektowych.

Systems Development Life Cycle (SDLC) Life-Cycle Phases



63. Klasyfikacja narzędzi wspierających wytwarzanie oprogramowania.

1. Zintegrowane środowisko programistyczne- jest to aplikacja lub zespół aplikacji (środowisko) służących do tworzenia, modyfikowania, testowania i konserwacji oprogramowania.
 - pakiet Microsoft Visual Studio (popularny na systemach rodziny Windows)
 - Eclipse i NetBeans
2. CASE (Computer-Aided Software Engineering) - oprogramowanie używane do komputerowego wspomaganie projektowania oprogramowania.
 - narzędzia do modelowania w języku UML i podobnych
 - narzędzia do zarządzania konfiguracją zawierające system kontroli wersji
 - narzędzia do re factoring
3. Systemy kontroli wersji - służą do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu i modyfikacji zmian dokonanych przez wiele osób w różnych momentach, np. Git,
4. Kompilatory- programy służący do automatycznego tłumaczenia kodu napisanego w jednym języku (języku źródłowym) na równoważny kod w innym języku (języku wynikowym) Microsoft Visual Studio, GNU Compiler for Java
5. Narzędzia wspomagające kompilację
6. Narzędzia wspomagające budowę aplikacji

7. Narzędzia do analizy programów (Debuggery) programy komputerowy służący do dynamicznej analizy innych programów, w celu odnalezienia i identyfikacji zawartych w nich błędów, zwanych z angielskiego bugami (robakami).

65. Instalacja i konserwacja oprogramowania.

Instalacją oprogramowania nazywamy proces, przeprowadzany przez specjalny program komputerowy lub skrypt instalacyjny, polegający na skopiowaniu plików aplikacji na dysk komputera użytkownika z jednoczesnym dostosowaniem parametrów instalowanego programu do potrzeb i warunków sprzętowych oraz środowiska systemu operacyjnego, a także zwykle odpowiednią modyfikację parametrów tego środowiska.

Konserwacją oprogramowania nazywamy modyfikację oprogramowania po jego dostarczeniu w celu skorygowania błędów, aby poprawić wydajność lub inne własności.

66. Zagadnienia etyczne i prawne związane z procesem wytwarzania i użytkowania oprogramowania.

Różne problemy etyczne i prawne zauważalne w obszarze informatycznym:

- Własności oprogramowania – czy użytkownik ma licencję na korzystanie z danego produktu. Czy nie ukraść go, nielegalnie ściągając produkt z dostępnych źródeł.
- Dopuszczanie się plagiatów – korzystanie z czyichś, gotowych rozwiązań przy tworzeniu oprogramowania, a następnie przypisywanie ich do swojej pracy.
- Handel danymi – sprzedawanie danych użytkowników (np. adresów email) innym firmom, bez wiedzy samych userów.
- Celowe przeciążanie serwerów (ataki DDOS).
- Wykorzystywanie oprogramowania wirusowego (np. w celach zarobkowych: okup w zamian za rzekome odszyfrowanie danych).

Problemy związane z wytwarzaniem oprogramowania obejmują sytuacje, gdy:

- Dostawca oprogramowania nie ma doświadczenia w danej dziedzinie zastosowań produktu.
- Dostawca produktu nie poprawia zgłaszanych przez klienta błędów oprogramowania, bądź robi to z wielkimi opóźnieniami.
- Dostawca produktu nie dba należyście o walidację danych użytkowników.
- Dostawca produktu nie dostarcza gotowego oprogramowania w terminie, bądź robi to, lecz produkt okazuje się być niepełny.
- Klient wymaga więcej niż zakładał na początku.
- Klient mimo jasnych wytycznych oczekiwał innych rezultatów.

67. Etapy cyklu życia systemu informatycznego i ich charakterystyka

Cykl życia systemu informatycznego:

1. Specyfikacja wymagań – następuje zebranie potrzeb informacyjnych przyszłych użytkowników (tzn. wymagań dotyczących przyszłego systemu), co ma służyć ustaleniu zgodności celów systemu z celami użytkownika.
2. Projekt – utworzenie konceptu, na którym bazował będzie system informatyczny
3. Kodowanie- implementacja systemu
4. Testowanie- proces mający na celu zapewnienie jakości oprogramowania. Ma na celu również weryfikację oraz walidację systemu informatycznego.
5. Instalacja- wdrożenie systemu
6. Eksploatacja- sprzedaż obiektu i korzystanie z niego
7. Wycofanie- koniec życia projektu

W obecnych czasach, spotyka się dzisiaj wiele modyfikacji tradycyjnego modelu cyklu życia systemu. Modyfikacje te wynikają z dążenia do dopasowania się do celów i założeń danego, konkretnego systemu bądź organizacji.

68 Modele organizacyjne wytwarzania systemów informatycznych i ich charakterystyka – zalety i wady (wystarczą 2 modele – kaskadowy i drugi agilowy np. Przyrostowy)

Modele cyklu wytwarzania systemów internetowych porządkują przebieg prac, ułatwiają planowanie zadań oraz monitorowanie realizacji zadań. Dzielimy je na:

Model kaskadowy (wodospadu) - - W modelu tym, aby zbudować system informatyczny należy przejść przez kolejne etapy, których realizacja ma zapewnić zakończenie projektu. Wyjście z jednego etapu jest równocześnie wejściem w kolejny, bez możliwości powrotu.



- Określanie wymagań – określane są cele oraz szczegółowe wymagania wobec tworzonego systemu,
- Analiza – budowany jest logiczny model systemu
- Projektowanie - powstaje szczegółowy projekt systemu
- Implementacja/kodowanie oraz testowania modułów - projekt zostaje zaimplementowany w konkretnym środowisku programistycznym oraz wykonywane są testy poszczególnych modułów,

- Testowanie - następuje integracja poszczególnych modułów połączona z testowaniem poszczególnych podsystemów oraz całego oprogramowania,
- Wdrożenie - system jest wykorzystywany przez użytkownika(ów), a producent dokonuje konserwacji oprogramowania — wykonuje modyfikacje polegające na usuwaniu błędów, zmianach i rozszerzaniu funkcji systemu

Wady:

1. Wymagania klienta muszą być w dużym stopniu sprecyzowane. W przypadku ich zmiany koszty znacznie wzrastają a wszystkie etapy muszą być przechodzone od nowa.
2. Kolejność wykonywania prac musi być ściśle przestrzegana.
3. Weryfikacja zgodności produktu z wymaganiami i jego użyteczności następuje dopiero w końcowych krokach.
4. Błędy popełnione we wstępnych etapach (zbierania lub analizy wymagań) mogą być wykryte dopiero na etapie testów akceptacyjnych, bądź eksploatacji, a koszty ich naprawy są bardzo wysokie.
5. Próba dopasowania produktu do zmieniających się wymagań, powoduje znaczący wzrostu kosztów budowy systemu.
6. Długa przerwa w kontaktach z klientem może spowodować zmniejszenie zainteresowania produktem. Klient uczestniczy w projekcie na samym początku przy określaniu wymogów i analiz a następny jego udział jest dopiero na etapie wdrażania.

Zalety:

1. łatwe planowanie, harmonogramowanie oraz monitorowanie przedsięwzięcia
2. zmusza do zdyscyplinowanego podejścia

Model kaskadowy z iteracjami – zakłada się możliwość powrotu do poprzedniego etapu (wiąże się to z wydłużeniem czasu przedsięwzięcia)

Model przyrostowy

Model przyrostowy jest wariantem modelu ewolucyjnego. W tym modelu rozpoznanie i analiza dotyczą całego systemu. Dopiero po sformułowaniu problemu i określeniu pełnej koncepcji następuje podział systemu na moduły (przyrosty), które są z osobna projektowane, programowane a następnie wdrażane.

Zalety:

1. częste kontakty z klientem (skrócenie przerw w porównaniu z modelem kaskadowym)
2. brak konieczności zdefiniowania z góry całości wymagań
3. możliwość wczesnego wykorzystania przez klienta dostarczonych części systemu,
4. możliwość elastycznego reagowania na opóźnienia realizacji fragmentu – przyspieszenie prac nad inną/innymi częściami (sumarycznie - bez opóźnienia całości przedsięwzięcia projektowego)

Wady:

5. dodatkowy koszt związany z niezależną realizacją fragmentów systemu
6. potencjalne trudności z wycinaniem podzbioru funkcji w pełni niezależnych

Link:<http://mariusz.makuchowski.staff.iiar.pwr.wroc.pl/download/courses/komputerowe.wspomaganie.zarzadzania/wyk.slajdy/wyk09.cykl.zycia.pdf>

70. Metody identyfikacji wymagań na systemy informatyczne i ich charakterystyka.

Metodyki

- Postulują przebieg procesu projektowania
- Wyznaczają zadania i kolejność ich wykonywania - co, kiedy i dlaczego powinno być wykonane
- Wskazują zastosowanie odpowiednich technik

Strukturalne

- *Dane i procesy modelowane osobno
- *Wykorzystują w zasadzie tylko proste typy danych
- *Dobrze dostosowane do modelu relacyjnego danych
- *Podstawowe techniki
 - diagramy związków encji (ERD)
 - hierarchie funkcji (FHD)
 - diagramy przepływu danych (DFD)

Obiektowe

- *Dane i procesy są modelowane łącznie!!!
- *Wykorzystują złożone typy danych
- *Dostosowane do obiektowego modelu danych
- *W przypadku realizacji opartej na relacyjnej b.d. wynikowy obiektowy model danych musi być odpowiednio przekształcony
- *Podstawowe techniki
 - diagramy klas UML
 - przypadki użycia,
 - modele dynamiczne UML

72. Pożądana zawartość dokumentu „Wymagania na system informatyczny”.

1. **Wprowadzenie** – cele, zakres i kontekst systemu
2. **Opis wymagań funkcjonalnych** - opisują funkcje wykonywane przez system. Mogą być opisywane za pomocą: Języka naturalnego, Języka naturalnego strukturalnego, Tablic i Formularzy, Diagramów blokowych, Diagramów kontekstowych i DPU
3. **Opis wymagań niefunkcjonalnych** - Wymagania niefunkcjonalne – opisują ograniczenia, przy których system ma realizować swoje funkcje;
4. **Opis ewolucji systemu** – rozwój systemu
5. **Model logiczny systemu, opracowany przy użyciu np. języka UML** - Model logiczny to zbiór informacji określający zachowanie się systemu. elementy: lista funkcji systemu, określenie obiektów zew. systemu, określenie współzależności między funkcjami systemu
6. **Słownik terminów niejednoznacznych na udziałowców przedsięwzięcia projektowego** - Słownik zawiera terminy, które mogą być niejednoznacznie interpretowane przez uczestników przedsięwzięcia informatycznego- co może być przyczyną nieporozumień podczas jego realizacji;

73. Podstawowe rezultaty fazy modelowania systemu.

Faza analizy (modelowania)

Poprawiony dokument opisujący wymagania;

-Słownik danych zawierający specyfikację modelu;

-Dokument opisujący stworzony model, zawierający:

- *Diagram klas;
- *Diagram przypadków użycia;
- *Diagramy interakcji (dla wybranych sytuacji);
- *Diagramy aktywności;
- *Diagram stanów (dla wybranych sytuacji);
- * Raport zawierający definicje i opisy klas, atrybutów, związków, metod itd.;

-Harmonogram fazy projektowania;

-Wstępne przypisanie ludzi i zespołów do zadań

74. Podstawowe zadania realizowane w procesie budowy obiektowego modelu systemu informatycznego.

Proces tworzenia modelu obiektowego:

- Identyfikacja klas i obiektów
- Identyfikacja związków pomiędzy klasami
- Identyfikacja i definiowanie pól (atrybutów)
- Identyfikacja i definiowanie metod i komunikatów

Czynności te są wykonywane iteracyjnie. Kolejność ich wykonywania nie jest ustalona i zależy zarówno od stylu pracy, jak i od konkretnego problemu.

75. Rodzaje modyfikacji wprowadzanych w fazie pielęgnacji systemu informatycznego.

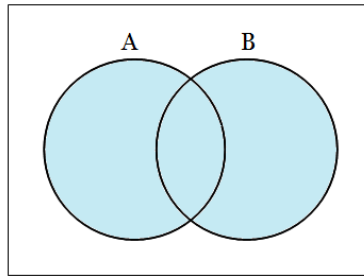
Faza pielęgnacji rozpoczyna się po fazie testowania, kiedy klient rozpoczyna użytkowanie systemu. Jest to najczęściej najdłuższa faza cyklu życia oprogramowania, gdyż obejmuje okres eksploatacji oprogramowania jak i jego utrzymania. Pielęgnacja oprogramowania polega na wprowadzeniu modyfikacji właściwości użytkowych. Istnieją trzy rodzaje takich modyfikacji:

- **Modyfikacje poprawiające** – polegają na usuwaniu z oprogramowania błędów powstałych bądź wykrytych w poprzednich fazach
- **Modyfikacje ulepszające** – polegają na poprawie jakości oprogramowania
- **Modyfikacje dostosowujące** – polegają na dostosowaniu oprogramowania do zmian zachodzących w wymaganiach użytkownika lub w środowisku komputerowym

76. Działania na zbiorach.

Suma:

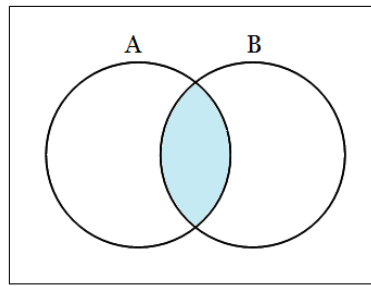
Suma zbiorów A i B to zbiór, który zawiera wszystkie elementy zbioru A i wszystkie elementy zbioru B.



$$A \cup B := \{x: x \in A \vee x \in B\}$$

Iloczyn:

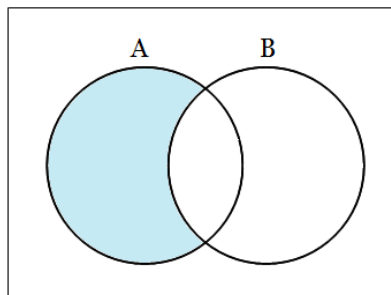
Iloczyn zbiorów A i B to zbiór, który zawiera elementy należące jednocześnie do zbioru A i do zbioru B, czyli iloczyn to część wspólna obu zbiorów.



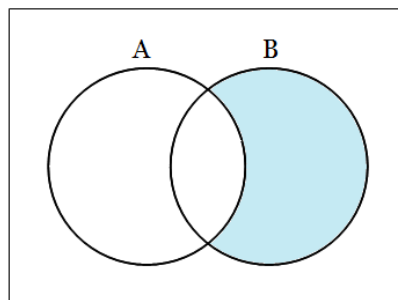
$$A \cap B := \{x: x \in A \wedge x \in B\}$$

Różnica:

Różnica zbiorów A i B to zbiór, który zawiera elementy należące do zbioru A i nie należące do zbioru B.



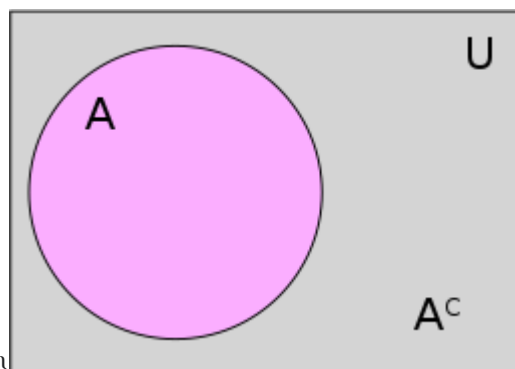
$$A \setminus B := \{x: x \in A \wedge x \notin B\}$$



analogicznie dla $B \setminus A$

Dopełnienie:

Dopełnieniem zbioru A z przestrzeni U nazywamy zbiór tych elementów przestrzeni U , które nie należą do zbioru A .



$$A' = \{x : x \in U \wedge x \notin A\}$$

77. Rachunek zdań.

W klasycznym rachunku zdań przyjmuje się założenie, że każdemu zdaniu można przypisać jedną z dwu wartości logicznych - prawdę albo fałsz, które umownie przyjęto oznaczać 1 i 0. Klasyczny rachunek zdań jest więc dwuwartościowym rachunkiem zdań.

Wartość logiczną zdań określa funkcja prawdy, związana z każdym spójnikiem zdaniowym. Wartość ta

zależy wyłącznie od prawdziwości lub fałszywości zdań składowych. Szczególną rolę w rachunku zdań odgrywają takie zdania złożone, dla których wartość logiczna jest równa 1, niezależnie od tego, jakie wartości logiczne mają zdania proste, z których się składają. Takie zdania nazywa się prawami rachunku zdań lub tautologiami.

Zmienne zdaniowe: p, q, r, s, itd.

Funktory: koniunkcja, alternatywa, równoważność, implikacja, itd.

Znaki pomocnicze: nawiasy: (,), [,], {, }, .

- a. prawo przemienności koniunkcji
 $(p \wedge q) \Leftrightarrow (q \wedge p)$
- b. prawo przemienności alternatywy
 $(p \vee q) \Leftrightarrow (q \vee p)$
- c. prawo łączności koniunkcji
 $[(p \wedge q) \wedge r] \Leftrightarrow [p \wedge (q \wedge r)]$
- d. prawo łączności alternatywy
 $[(p \vee q) \vee r] \Leftrightarrow [p \vee (q \vee r)]$
- e. prawo rozdzielności koniunkcji
względem alternatywy
 $[p \wedge (q \vee r)] \Leftrightarrow [(p \wedge q) \vee (p \wedge r)]$

- f. prawo rozdzielności alternatywy
względem koniunkcji
 $[p \vee (q \wedge r)] \Leftrightarrow [(p \vee q) \wedge (p \vee r)]$
- g. pierwsze prawo De Morgana
- h. (prawo zaprzeczenia koniunkcji)
 $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$
- i. drugie prawo De Morgana
- j. (prawo zaprzeczenia alternatywy)
 $\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$

78. Działania na macierzach. (faworyt)

Dodawanie i mnożenie przez skalar

Mnożenie macierzy przez stałą (skalar, element pierścienia) oraz dodawanie macierzy definiuje się następująco:

- $a \cdot (a_{ij}) = (a \cdot a_{ij})$
- $(a_{ij}) + (a_{ij} + b_{ij})$.

Słownie: mnożenie macierzy przez skalar polega na wymnożeniu przez niego wszystkich jej elementów, a dodawanie macierzy na dodaniu odpowiadających sobie elementów. Analogicznie definiuje się różnicę macierzy.

Zbiór R (bez kreski ułamkowej) z dodawaniem jest grupą przemenną: łączność i przemienność działań w pierścieniu przenosi się na działania na macierzach, elementem neutralnym jest macierz zerowa o wszystkich elementach równych 0, elementem przeciwnym do macierzy jest macierz $-A$. Zbiór R_n^m (bez kreski ułamkowej) z dodawaniem jest grupą przemenną: łączność i przemienność działań w pierścieniu przenosi się na działania na macierzach, elementem neutralnym jest macierz zerowa θ o wszystkich elementach równych 0 $\in R$, elementem przeciwnym do macierzy A jest

macierz, którą nazywa się **macierzą przeciwną** do A . Oczywiście jest o ile macierze te są zgodnego typu. macierz $-1 \cdot A = -A$, którą nazywa się **macierzą przeciwną** do A . Oczywiście jest $A - B = A + (-B)$ o ile macierze te są zgodnego typu.

Mnożenie przez skalar spełnia warunki:

- $(a + b) \cdot A = a \cdot A + b \cdot A,$
- $a \cdot (A + B) = a \cdot A + a \cdot B,$
- $(ab) \cdot A = a \cdot (b \cdot A),$
- $1 \cdot A = A,$

zatem zbiór R z dodawaniem i mnożeniem przez skalary jest modułem nad pierścieniem R .
zatem zbiór R_n^m z dodawaniem i mnożeniem przez skalary jest modułem nad pierścieniem R .

Przykład mnożenia przez skalar

$$3 \cdot \begin{bmatrix} 1 & 2 & 0 \\ -1 & -4 & 9 \end{bmatrix} = \begin{bmatrix} 3 \cdot 1 & 3 \cdot 2 & 3 \cdot 0 \\ 3 \cdot (-1) & 3 \cdot (-4) & 3 \cdot 9 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 0 \\ -3 & -12 & 27 \end{bmatrix}$$

Przykład dodawania macierzy

$$\begin{bmatrix} 1,3 & 2 & 3 \\ 1 & 2 & 9 \end{bmatrix} + \begin{bmatrix} 1,2 & 2 & 11 \\ 3 & -4 & 7 \end{bmatrix} = \begin{bmatrix} 2,5 & 4 & 14 \\ 4 & -2 & 16 \end{bmatrix}$$

Mnożenie

Działanie mnożenia macierzy można określić na wiele sposobów, jednak niżej określone, nazywane **mnożeniem Cauchy'ego**, ma wiele zastosowań w algebrze liniowej, w której macierze służą do zapisu przekształceń liniowych – odpowiada ono ich składaniu. Macierze można również mnożyć używając iloczynu Kroneckera.

Iloczyn macierzy (nazywanej w tym kontekście *lewym czynnikiem*) i macierzy (*prawy czynnik*) jest macierzą taką, że
Iloczyn $\cdot: R_n^m \times R_m^k \rightarrow R_n^k$ macierzy $A = (a_{ij}) \in R_n^m$ (nazywanej w tym kontekście *lewym czynnikiem*) i macierzy $B = (b_{ij}) \in R_m^k$ (*prawy czynnik*) jest macierzą $C = (c_{ij}) \in R_n^k$ taką, że $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{im}b_{mj}$.

80. Pojęcie relacji i funkcji.

Iloczyn kartezjański zbiorów A i B to zbiór wszystkich uporządkowanych par:

Dla $A = \{0,1,2\}$ i $B = \{1,3\}$

Relacja to dowolny podzbiór iloczynu kartezjańskiego skończonej liczby zbiorów. Jest to pewnego rodzaju zależność między elementami zbiorów.

Przykłady relacji:

- Relacja mniejszości $<$ w zbiorze liczb rzeczywistych
- Relacja podzielności $|$ w zbiorze liczb całkowitych

Typy relacji:

- zwrotna np.
- symetryczna np.

- c. przechodnia np.
- d. spójna np.
- e. asymetryczna np.
- f. równoważności jeżeli relacja jest zwrotna, symetryczna i przechodnia.

Funkcją nazywamy relację w , która jest prawostronnie jednoznaczna. Znaczy to, że każdemu elementowi przypisany jest dokładnie jeden element . Zbiór X nazywamy dziedziną, zbiór Y przeciwdziedziną.

81. Własności relacji: relacje porządkujące; relacje równoważności.

Relacje porządkujące

<p>Relację nazywamy relacją częściowego porządku, jeśli jest zwrotna, antysymetryczna i przechodnia.</p>	<p>Relację nazywamy relacją liniowego porządku, jeśli jest relacją częściowego porządku i dodatkowo jest spójna.</p>
<p>Relację określoną w zbiorze X nazywamy zwrotną, gdy każdy element zbioru X jest w relacji sam ze sobą, tzn. gdy:</p> $\bigwedge_{x \in X} x R x$ <p>Relację określoną w zbiorze X nazywamy przechodnią, gdy dla dowolnych elementów 'x', 'y', 'z' należących do zbioru X z faktu, że 'x' jest w relacji z 'y' i 'y' jest w relacji z 'z' wynika, że również 'x' jest w relacji z 'z', tzn. gdy:</p> $\bigwedge_{x \in X} \bigwedge_{y \in X} \bigwedge_{z \in X} [(x R y \wedge y R z) \Rightarrow x R z]$ <p>Relację określoną na zbiorze X nazywamy antysymetryczną, gdy dla dowolnych dwóch elementów 'x', 'y', z faktu, że 'x' jest w relacji z 'y' i 'y' jest w relacji z 'x' wynika, że 'x' jest równe 'y', tzn. gdy:</p> $\bigwedge_{x, y \in X} (x R y \wedge y R x \rightarrow x = y)$	<p>Relację określoną w zbiorze X nazywamy spójną, gdy dla dowolnych dwóch elementów 'x', 'y', 'x' jest w relacji z 'y' lub 'y' jest w relacji z 'x', tzn. gdy:</p> $\bigwedge_{x, y \in X} (x \neq y \rightarrow x R y \vee y R x)$

Relacje równoważności

Relację nazywamy relacją równoważności, jeśli jest zwrotna, symetryczna i przechodnia.

Relację określoną w zbiorze X nazywamy **zwrotną**, gdy każdy element zbioru X jest w relacji sam ze sobą, tzn. gdy:

$$\bigwedge_{x \in X} x R x$$

Relację określoną w zbiorze X nazywamy **przechodnią**, gdy dla dowolnych elementów x, y, z należących do zbioru X z faktu, że ' x ' jest w relacji z ' y ' i ' y ' jest w relacji z ' z ' wynika, że również ' x ' jest w relacji z ' z ', tzn. gdy:

$$\bigwedge_{x \in X} \bigwedge_{y \in X} \bigwedge_{z \in X} [(x R y \wedge y R z) \Rightarrow x R z]$$

Relację określoną w zbiorze X nazywamy **symetryczną**, gdy dla dowolnych elementów x i y , należących tego zbioru z faktu, że x jest w relacji z y wynika również, że y jest w relacji z x , tzn. gdy:

$$\bigwedge_{x \in X} \bigwedge_{y \in X} (x R y \Rightarrow y R x)$$

83. Zmienna losowa i jej charakterystyki liczbowe.

Zmienna losowa to funkcja przypisująca zdarzeniom elementarnym liczby. Rozróżniamy zmienną losową typu:

- **skokowego**, gdy jej zbiór możliwych wartości jest skończony lub co najwyżej przeliczalny, najczęściej wartościami są liczby naturalne lub całkowite,
- **ciągłego**, gdy jej zbiór możliwych wartości stanowią wszystkie liczby z pewnego przedziału liczb rzeczywistych

Zmienne losowe mogą być charakteryzowane za pomocą pewnych stałych które opisują ją pod względem np. rozrzutu jej wartości, wartości najbardziej prawdopodobnej, kształtu histogramu lub krzywej gęstości. Najważniejsze charakterystyki liczbowe to: **wartość oczekiwana (średnia)**, **wariancja** i pierwiastek z wariancji, czyli **odchylenie standardowe**.

84. Cele i rodzaje programowania deklaratywnego.

Programowanie deklaratywne w odróżnieniu od programowania imperatywnego polega na tym, że definiujemy raczej to „co chcemy uzyskać”, a nie „jakie kroki należy wykonać”. Nie ma więc w nim sekwencji instrukcji modyfikujących stan maszyny krok po kroku.

Zastosowania programowania deklaratywnego:

- W systemach ekspertowych.
- W automatyzacji dowodzenia twierdzeń.
- W rozwiązywaniu równań.
- W przetwarzaniu danych.
- W przetwarzaniu języka naturalnego.

Wyróżniamy dwie odmiany programowania deklaratywnego:

- Programowanie funkcyjne: brak stanu programu, nie ma instrukcji iteracyjnych – zastępuje je rekurencja, do rozwiązywania skomplikowanych problemów definiuje się liczne funkcje pomocnicze.

Przykładowe języki: Haskell.

- Programowanie w logice: stosowanie predykatów, wykonanie programu polega na znalezieniu takich danych, dla których można udowodnić spełnienie wymaganej własności.

Przykładowe języki: Prolog, SQL.

85. Definicje unifikatora (podstawienia uzgadniającego), najogólniejszego unifikatora, algorytm unifikacji i twierdzenie o unifikacji.

Podstawienie (unifikacja): Odwzorowanie zbioru zmiennych w zbiór termów (jedynie skończonej liczbie zmiennych przypisuje termy różne od nich samych).

Reprezentacja podstawienia: $\sigma = \{X_1/t_1, X_2/t_2, \dots, X_n/t_n\}$. ($t_i \notin X_i, X_i \notin X_j$)

Unifikacja wyrażeń: Podstawienie θ jest unifikatorem wyrażeń E_1 oraz E_2 wtw gdy $E_1\theta = E_2\theta$.

Najbardziej ogólne podstawienie unifikujące (mgu): Podstawienie θ jest mgu pewnych wyrażeń wtw gdy dla każdego podstawienia σ będącego unifikatorem istnieje podstawienie λ , takie że $\sigma = \theta\lambda$.

Algorytm:

Dane: W – zb. wyrażeń do unifikacji, Szukane: $\theta = \text{mgu}(W)$

8. Niech $k=0, W_k=W, \theta_k = \varepsilon$.
9. Jeżeli wszystkie literały W_k są równe to STOP; $\theta_k = \text{mgu}(W)$. W przeciwnym wypadku znajdź D_k zbiór niezgodności (podwyrażeń) dla W_k .
10. Jeśli w D_k mamy zmienną X_k i term $t_k \in D_k$ ($X_k \notin t_k$), to przechodzimy do następnego kroku. W przeciwnym wypadku STOP – W jest nieunifikowalne.
11. $\theta_{k+1} = \theta\{X_k/t_k\}, W_{k+1} = W_k\{X_k/t_k\}$.
12. $k = k+1$, przejdź do kroku 2.

Twierdzenie unifikacji: Jeśli zbiór literalów W jest unifikowalny, to algorytm unifikacji kończy pracę w trakcie wykonywania Ruchu 2 pewnego kroku $k + 1$. Wtedy σ_k jest najogólniejszym unifikatorem. Jeśli W_k nie jest unifikowalny, to algorytm unifikacji kończy pracę w trakcie wykonywania Ruchu 3 pewnego kroku $k + 1$.

86. Programy Horna, SLD-rezolucja, odpowiedzi poprawne, odpowiedzi obliczone, poprawność i zupełność SLD-rezolucji.

Klauzula Horna jest to zbiór formuł logicznych, którego wartość logiczna jest równa alternatywie tych formuł. Na przykład dla klauzuli $(\sim(n < 0), \sim(n > 12), \text{najwyzej_tuzin}(n))$ mamy alternatywę: $\sim(n < 0) \vee \sim(n > 12) \vee \text{najwyzej_tuzin}(n)$, która jest prawdziwa dla liczb od 1 do 12. Klauzula Horna to dowolna klauzula, zawierająca co najwyżej jeden literal pozytywny.

Def. (implikacyjna postać klauzuli Horna) Dowolną klauzulę Horna $A \vee \sim A_1 \vee \dots \vee \sim A_m$, gdzie A oraz A_1, \dots, A_m są formułami atomowymi, można zapisać w postaci $A \leftarrow A_1 \wedge \dots \wedge A_m$

Def. (SLD-rezolucja) jest to rezolucja źródłowa w zbiorze klauzul Horna z określoną strategią wyboru literalu.

Def. (nagłówek klauzuli, ciało klauzuli)

Niech $A \leftarrow A_1 \wedge \dots \wedge A_m$ będzie klauzulą Horna w postaci implikacyjnej. Formułę atomową A nazywa się nagłówkiem klauzuli, a koniunkcję atomów $A_1 \wedge \dots \wedge A_m$ określa się mianem ciała klauzuli.

Def. (fakt, reguła, klauzula pusta)

W szczególnym przypadku ciało klauzuli może być formułą pustą; $A \leftarrow$ klauzule takie nazywa się faktami. Dla rozróżnienia, klauzule o niepustych ciałach określa się mianem reguł. Klauzula o pustym ciele i pustym nagłówku reprezentuje klauzulę pustą, którą oznacza się symbolem \perp .

Def. (odpowiedź) Niech $\leftarrow B$ będzie dowolnym celem. Odpowiedzią dla zapytania B i programu P nazywa się najogólniejsze podstawienie θ , takie że $P(\forall)B\theta$

Def. (odpowiedź obliczona) Odpowiedzią obliczoną dla zapytania B i programu P , nazywa się podstawienie θ takie, że istnieje dowód SLD-rezolucyjny dla celu $\leftarrow B$ i programu P , którego ostatnim elementem jest para (\perp, θ)

Poprawność - jeżeli na podstawie ogólnej metody rezolucji można wyprowadzić klauzulę pustą, to zbiór klauzul jest niespełnialny.

Pełność (zupełność) - jeśli zbiór klauzul jest niespełnialny, to stosując ogólną metodę rezolucji można wyprowadzić klauzulę pustą.

90. Podaj rodzaje układów sekwencyjnych oraz różnice w procedurach ich projektowania.

Układ sekwencyjny

Charakteryzuje się tym, że stan wyjść y zależy od stanu wejść x oraz od poprzedniego stanu, zwanego stanem wewnętrznym, pamiętanego w zespole rejestrów (pamięci).



Jeżeli stan wewnętrzny nie ulega zmianie pod wpływem podania różnych sygnałów X , to taki stan nazywa się stabilnym.

Rozróżnia się dwa rodzaje układów sekwencyjnych:

1. asynchroniczne
2. synchroniczne

Układy asynchroniczne

W układach asynchronicznych zmiana sygnałów wejściowych X natychmiast powoduje zmianę wyjść Y . W związku z tym układy te są szybkie, ale jednocześnie podatne na zjawisko hazardu i wyścigu. Zjawisko wyścigu występuje, gdy co najmniej dwa sygnały wejściowe zmieniają swój stan w jednej chwili czasu (np. 11b -> 00b).

Układy synchroniczne

W układach synchronicznych zmiana stanu wewnętrznego następuje wyłącznie w określonych chwilach, które wyznacza sygnał zegarowy (ang. clock). Każdy układ synchroniczny posiada wejście zegarowe oznaczane zwyczajowo symbolami C , CLK lub $CLOCK$. Charakterystyczne dla układów synchronicznych, jest to, iż nawet gdy stan wejść się nie zmienia, to stan wewnętrzny - w kolejnych taktach zegara - może ulec zmianie.

Różnice polegają na konieczności eliminacji zjawisk szkodliwych w układach asynchronicznych (wyścigów i hazardów).

88. Budowa programu w Prologu: klauzule (fakty, reguły), definicje predykatów. Sposób realizacji programu.

PROGRAM zbiór procedur; kolejność procedur w programie nie jest istotna.

PROCEDURA ciąg klauzul definiujących jeden predykat kolejność klauzul w procedurze jest istotna

KLAUZULA fakt lub reguła.

FAKT Bezwarunkowo prawdziwe stwierdzenie o istnieniu pewnych powiązań (relacji) między obiektami.

Budowa: symbol_relacji (obiekt1, ..., obiektn)

Przykłady **student(marcin)**. Marcin jest studentem.

lubi(ewa, marek). Ewa lubi Marka

REGUŁA: Warunkowe stwierdzenie istnienia pewnych powiązań (relacji) między obiektami.

symbol_relacji (obiekt1, ..., obiektn) :- symbol_relacji_1 (obiekt1, ..., obiektn1),

symbol_relacji_2 (obiekt1, ..., obiektn2)...

Przykład: **powierzchnia (X, Y) :- dlugosc (X, D), szerokosc (X, S), Y is D*S.**

Powierzchnia Y prostokata X jest równa iloczynowi długości D i szerokości S tego prostokata.

Rozpoczęcie działania programu polega na wywołaniu dowolnej procedury, które jest nazywane w Prologu zadawaniem pytań lub podawaniem celu.

Celem działania programu jest uzyskanie odpowiedzi na Pytanie o prawdziwość podanych faktów lub polecenie znalezienia nazw obiektów będących w podanej relacji z innymi.

Postać ?- symbol_relacji_1 (obiekt1, ..., obiektn1),

symbol_relacji_2 (obiekt1, ..., obiektn2).

Przykład

?- **lubi (marta, jan)**. - Czy Marta lubi Jana?

lubi(marta, X). - Kogo lubi Marta?

lubi(X, marta) -Kto lubi Martę?

91. Jakie znasz elementy pamięciowe stosowane układach sekwencyjnych?

Układy synchroniczne projektuje się z przerzutników typu D, T i JK. Układy asynchroniczne projektuje się z przerzutników RS lub bez przerzutników (bramki ze sprzężeniem zwrotnym).

92. Co oznacza termin mikroprogramowanie? Do czego służy?

Układ mikroprogramowany to jedno z możliwych rozwiązań stosowanych w technice cyfrowej dla projektów o dużej złożoności. Projekt dekomponowany jest na dwie części: układ operacyjny (wykonawczy) i układ sterujący. Układ sterujący można zaprojektować jako układ mikroprogramowany. Bity sterujące, wraz z innymi odpowiednimi bitami służącymi do napisania mikroprogramu, umieszcza się w pamięci ROM.

93. Podaj znane zapisy liczbowe i zakres ich stosowania.

Zapisy można podzielić na **stałopozycyjne (liczby całkowite)** i **zmiennopozycyjne (liczby rzeczywiste)**.

Stałopozycyjne to: ZNAK-MODUŁ, U1 i U2. Najpowszechniejszy jest U2 ze względu na najprostsze algorytmy operacji arytmetycznych.

Zmiennopozycyjny zapis polega na reprezentowaniu liczby za pomocą trzech grup bitów: 1 bit znaku, 23-bitowej mantysy i np. 8-bitowego wykładnika.

8 bitów

- char -128 do +127
- unsigned char 0 do 255

16 bitów

- short int -32 768 do 32 767
- unsigned short int 0 do 65 535

32 bity

- int -2 147 483 648 do +2 147 483 647
- unsigned int 0 do 4 294 967 295
- float

64 bity

- double
- long int -9 223 372 036 854 775 808 do +9 223 372 036 854 775 807

94. Co to są mikrokontrolery?

Mikrokontroler - system mikroprocesorowy zrealizowany w postaci pojedynczego układu scalonego, zawierającego jednostkę centralną (CPU), pamięć RAM oraz na ogół pamięć programu i rozbudowane układy wejścia-wyjścia. Określenie mikrokontroler pochodzi od głównego obszaru zastosowań, jakim jest sterowanie urządzeniami elektronicznymi.

Mikrokontroler stanowi użyteczny i całkowicie autonomiczny system mikroprocesorowy, nie wymagający użycia dodatkowych elementów, których wymagałby do pracy tradycyjny mikroprocesor. Mikrokontrolery wykorzystuje się powszechnie w sprzęcie AGD, układach kontrolno-pomiarowych, w przemysłowych układach automatyki, w telekomunikacji itp.

95. Jakie parametry są charakterystyczne dla pamięci dynamicznych?

Podstawowymi parametrami opisującymi

każdy typ pamięci są:

Szybkość

Pojemność

Pobór mocy

Koszt

Dla pamięci dynamicznej charakterystycznymi parametrami są:

wymagany czas odświeżania (8-64 ms)

wymaganą liczbę cykli odświeżania

(pierwiastek z liczby bitów).

Czasem dostępu - access time. Jest to czas upływający od momentu w którym wysyłane jest żądanie dostępu do pamięci do czasu w którym informacja zwrotna ukaże się na jej wyjściu.

Czasem cyklu - cycle time. Jest to czas najkrótszy, który upływa między dwoma kolejnymi żądaniami dostępu do pamięci.

Szybkość transmisji - Szybkość transmisji mierzymy liczbą bitów, bądź bajtów, którą jesteśmy w stanie przesłać pomiędzy urządzeniem, a pamięcią.

Również pobór mocy stanowi bardzo ważny parametr, który staje się problemem przy budowaniu urządzeń zasilanych bateryjnie, jak komputery kieszonkowe, laptopy.

96. Podaj tryby adresowania dla rozkazów mikrokontrolera i odpowiadający im czas trwania cyklu instrukcyjnego wraz z omówieniem kolejnych cykli maszynowych.

Trybem adresowania nazywa się sposób określenia miejsca przechowywania argumentów rozkazu.

Rozróżniamy następujące tryby adresowania:

- adresowanie natychmiastowe, w którym argument rozkazu zawarty jest w kodzie rozkazu.

Cykl: fetch, odczytu;

- adresowanie bezpośrednie, w którym kod rozkazu zawiera adres komórki pamięci przechowującej argument rozkazu.

Cykl: fetch, 4xodczyt;

- adresowanie rejestrowe, w którym w kodzie rozkazu określony jest rejestr zawierający argument rozkazu.

Cykl: fetch;

- adresowanie pośrednie (rejestrowe pośrednie), w którym kod rozkazu zawiera określenie rejestru, bądź rejestrów zawierających adres komórki pamięci z argumentem rozkazu.

Cykl: fetch i odczytu;

Cykl instrukcyjny (rozkazowy) – czas potrzebny na odczytanie kodu rozkazu z pamięci, na pobranie argumentów, na wykonanie rozkazu i przesłanie wyniku operacji.

Rodzaje cykli:

- Pobranie kodu operacyjnego (fetch)
- Odczyt z pamięci (memory read)
- Zapis do pamięci (memory write)
- Odczyt z urządzenia wejściowego (IOread)
- Zapis do urządzenia wyjściowego (IOWrite)

- Cykl przyjęcia przerwania (interrupt)
- Cykl zatrzymania (HALT)

98. Porównaj pod względem szybkości znane rozwiązania operacji mnożenia w systemach wbudowanych

Mnożenie w układzie kombinacyjnym (układ iteracyjny, gdzie wynik pojawia się po czasie opóźnienia wnoszonym przez bramki) jest najszybsze.

Mnożenie w układzie sekwencyjnym (tyle kroków, ile jest bitów, a każdy krok to dodawanie warunkowe i przesunięcie) jest wolniejsze.

Mnożenie programowe (bez żadnego sprzętu) jest najwolniejsze.

99. Model obliczeniowy perceptronu – możliwości i ograniczenia.

Perceptron jest prostym matematycznym modelem neuronu. Posiada on n wejść przyjmujących wartości rzeczywiste lub binarne.

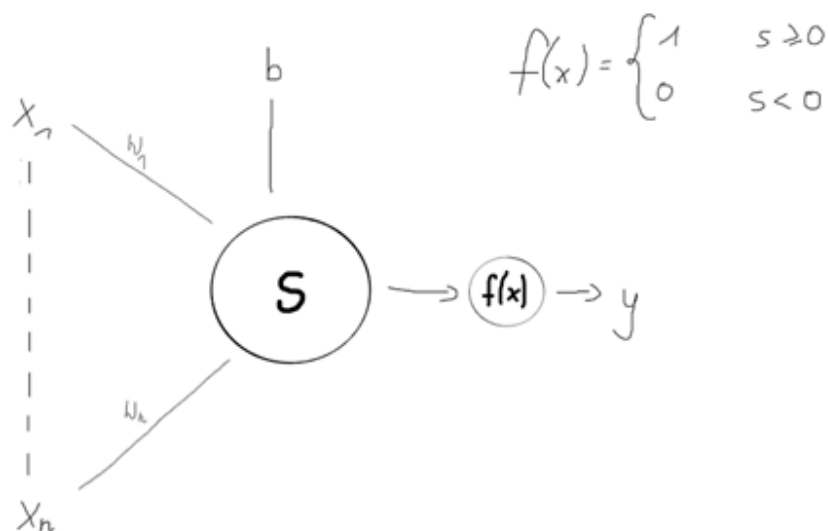
Parametry wewnętrzne perceptronu to: n wag, które przypisane są do odpowiednich wejść oraz wartość odchylenia b (ang. bias) odpowiadająca za nieliniowe przekształcenie wejść w wyjście.

ZASADA DZIAŁANIA opisana na przykładzie wejść o wartościach 0 lub 1

Do każdego i -tego wejścia perceptronu przypisana jest waga w_i . Dla danych stanów wejściowych x_1, \dots, x_n liczymy sumę ważoną:

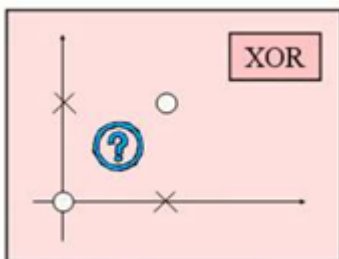
$$s = \sum_{i=1}^n x_i w_i + b$$

Jeżeli s jest większa lub równa 0, to ustawiamy wyjście $y=1$, zaś w przeciwnym przypadku ustawiamy $y=0$. Schemat działania:

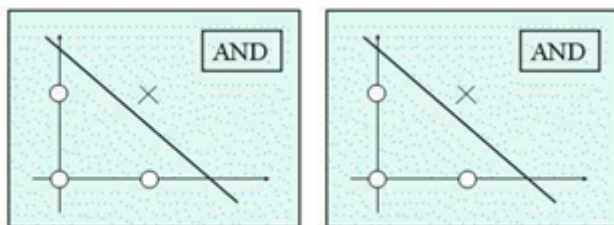


MOŻLIWOŚCI I OGRANICZENIA

Pojedynczy perceptron nie potrafi odróżniać zbiorów nieseparowalnych liniowo, tzn. takich, że między punktami z odpowiedzią pozytywną i negatywną nie da się poprowadzić prostej rozgraniczającej. Jednym z najprostszych przykładów takich zbiorów jest funkcja logiczna XOR (alternatywa wykluczająca).



Jak widać na poniższych rysunkach, funkcje AND i OR dają zbiory separowalne liniowo, a więc możliwe do realizacji perceptronem.



101. Mechanizm działania algorytmu genetycznego.

Algorytm oparty na mechanizmach dziedziczności i doboru naturalnego. Łączy w sobie zasadę przeżycia najlepiej przystosowanych osobników (osobnik reprezentuje rozwiązanie) i

zasadę losowej wymiany informacji. Mechanizm działania tego algorytmu wygląda następująco:

1. Losowana jest pewna populacja początkowa.
2. Populacja poddawana jest ocenie (selekcja) za pomocą ustalonej funkcji przystosowania. Najlepiej przystosowane osobniki biorą udział w procesie reprodukcji.
3. Genotypy wybranych osobników poddawane są operatorom ewolucyjnym:
 - 1) są ze sobą kojarzone poprzez łączenie genotypów rodziców (krzyżowanie),
 - 2) przeprowadzana jest mutacja, czyli wprowadzenie drobnych losowych zmian.
4. Rodzi się drugie (kolejne) pokolenie. Aby utrzymać stałą liczbę osobników w populacji te najlepsze (według funkcji oceniającej) są powielane, a najgorsze usuwane. Jeżeli nie znaleziono dostatecznie dobrego rozwiązania, algorytm powraca do kroku drugiego. W przeciwnym wypadku wybieramy najlepszego osobnika z populacji - jego genotyp to uzyskany wynik.

102. Definicja entropii informacji i wybrane zastosowanie tego pojęcia.

W pojęciu entropii słowo „wiadomość” nie oznacza tego samego co „informacja”.

Entropia – w ramach teorii informacji jest definiowana jako średnia ilość informacji, uzyskanej w wyniku obserwacji n niezależnych zdarzeń losowych.

$$H(X) = \sum_{i=1}^n p(x_i) \log_r \frac{1}{p(x_i)} = - \sum_{i=1}^n p(x_i) \log_r p(x_i),$$

Wzór: x_i – zdarzenie, p – prawdopodobieństwo zdarzenia, r – podstawa logarytmu

Własności entropii:

- Jest nieujemna.
- Jest maksymalna, gdy prawdopodobieństwa zdarzeń są takie same.
- Jest równa 0, gdy stany systemu przyjmują wartości albo tylko 0 albo tylko 1.
- Gdy dwa systemy są niezależne – entropia sumy systemów = sumie entropii.

Zastosowanie entropii informacji:

- Przy ocenie bezpieczeństwa systemów szyfrowania.
- W uczeniu maszynowym poprzez zastosowanie ukrytych modeli Markowa.
- Przy porównywaniu modeli językowych w lingwistyce komputerowej.

Przy rekonstrukcjach obrazu w tomografii komputerowej (zastosowanie medyczne).

103. Metody generacji reguł systemu decyzyjnego, minimalne, pokrywające, wyczerpujące.

Algorytm pokrywający obiekty: Szukamy w obiektach systemu decyzyjnego, począwszy od pierwszego, a skończywszy na ostatnim reguł długości jeden, które są niesprzeczne. Po

znalezieniu reguły niesprzecznej, dany obiekt wyrzucamy z rozważań, pamiętając o tym, że dalej bierze udział w sprawdzaniu sprzeczności i może wspierać inne reguły. Jeżeli po przeszukaniu wszystkich obiektów, pozostają obiekty nie wyrzucone z rozważań, szukamy w nich kombinacji niesprzecznej długości dwa i postępujemy analogicznie jak w przypadku reguł pierwszego rzędu. Wyszukiwanie reguł niesprzecznych jest kontynuowane do momentu wyeliminowania wszystkich obiektów niesprzecznych. Jeżeli w systemie pojawią się obiekty, które są sprzeczne na wszystkich deskryptorach, nie tworzymy z nich reguł.

Algorytm LEM2: Polega na tworzeniu pierwszej reguły przez sekwencyjny wybór "najlepszego" elementarnego warunku, przy zachowaniu ustalonych kryteriów. Przykłady treningowe pokryte przez regułę są usuwane z rozważań. Proces tworzenia reguł jest powtarzany iteracyjnie do momentu, gdy pozostają jakieś niepokryte obiekty w systemie treningowym. Wszelkie konflikty rozwiązywane są hierarchicznie (wybierana jest wartość pierwsza od góry z lewej strony)

Minimalne:

1. takie której liczba jest minimalna, a system jako-tako działa
2. minimalny zbiór reguł, który reaguje na wszystkie możliwe sygnały wejściowe (wszystkie możliwe próbki)

Pokrywające:

1. taki zbiór reguł, że reaguje na wszystkie możliwe sygnały wejściowe (wszystkie możliwe próbki)
2. zbiór reguł takich, że dla niektórych próbek kilka reguł (czasem nawet sprzecznych) jest aktywnych

Wyczerpujące:

1. zbiór reguł takich, że system działa zadowalająco
2. zbiór reguł takich, że system reaguje na wszystkie możliwe sygnały wejściowe.

104. Podaj znaczenie oraz omów wzajemne związki między terminami: przestrzeń urządzenia, przestrzeń operacyjna urządzenia, powierzchnia obrazowania, macierz adresowalna, jednostka rastru, krok kreślaka

Przestrzeń urządzenia (obrazującego) to obszar zdefiniowany przez układ współrzędnych ograniczony pojemnością rejestrów pozycji x i y w urządzeniu graficznym - określony przez skończony zbiór punktów adresowalnych urządzenia wizualizującego.

Przestrzeń operacyjna to obszar w przestrzeni urządzenia, którego zawartość została odwzorowana na powierzchni obrazowania.

- Odwzorowuje punkt P1 w Q1,
- Kierunek $P=P2-P1$ w kierunku $Q=Q2-Q1$,

- Transformuje płaszczyznę wyznaczoną przez : P1, P2, P3 w płaszczyznę wyznaczoną przez Q1, Q2, Q3

109. Podaj wzajemne położenie układów współrzędnych: danych i obserwatora.

Układ obserwatora nie jest określony jednoznacznie. Możemy przyjąć, że jego początek pokrywa się z początkiem układu danych 0. Przekształcenie układu danych do układu obserwatora będzie polegało na wykonaniu takich obrotów wokół osi układu, by wektor $n=[x_n, y_n, z_n]$ miał kierunek osi nowego układu i przeciwny do niej zwrot.

Aby przekształcić układ danych do układu obserwatora należy:

1. Określić kierunki osi w układzie obserwatora
2. Przedstawić dane we współrzędnych układu obserwatora

110. Wymień we właściwej kolejności operacje, jakie należy wykonać, aby obrócić punkt P o kąt φ dookoła prostej zadanej przez dwa punkty P1 i P2 .

- 1) Przesunięcie, aby punkt P1 znalazł się w początku układu współrzędnych.
- 2) Obrót wokół osi OX.
- 3) Obrót wokół osi OY. Obroty (etap 2. i 3.) zapewniają, że zadana oś obrotu (prosta) pokryje się (z uwzględnieniem zwrotów) z osią OX układu współrzędnych.
- 4) Realizacja zadanego obrotu o kąt wokół osi OX.
- 5) Obrót będący operacją odwrotną do operacji 3.
- 6) Obrót będący operacją odwrotną do operacji 2.
- 7) Przesunięcie odwrotne do przesunięcia 1.

111. Opisać 3 podstawowe obszary uzależnień komputerowych.

Środkami uzależniającymi są:

1. gry komputerowe
2. Internet
3. programowanie

Pozbawieni dostępu do komputera przeżywają stany identyczne z zespołem abstynenckim, są pobudzeni, cierpią na zaburzenia snu, popadają w stany depresyjne, fantazjują na temat Internetu. Uzależnienia komputerowe można podzielić na pierwotne i wtórne. W pierwszym przypadku mamy do czynienia z potrzebą przeżycia emocji, uzyskania efektu pobudzenia, sprawdzenia swoich umiejętności. W drugim przypadku komputer jest traktowany jako forma ucieczki od rzeczywistości. Ponadto należy pamiętać, że ten rodzaj uzależnienia ma negatywne konsekwencje dla zdrowia.

Uzależnienie od komputera u dziecka można stwierdzić, gdy:

- udaje, że się uczy, kiedy tymczasem godzinami siedzi przed ekranem monitora
- w czasie gry lub przeglądania stron WWW wpada w stan przypominający trans
- próby ograniczenia czasu spędzanego przy komputerze napotykają na silny opór, nawet agresję
- niechętnie spotyka się z kolegami, brak mu przyjaciół, woli komputer niż ruch na świeżym powietrzu
- nie potrafi odpoczywać, nie wzrusza się

112. Wymienić przynajmniej 3 polskie ustawy dotyczące środowiska informatycznego.

Najważniejsze aktualne obowiązujące regulacje prawne dotyczące informatyki:

- ustawa z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tekst jednolity Dz.U. z 2000 r. Nr 80, poz.904 z późn. zm.);
- ustawa z dnia 27 lipca 2001 r. o ochronie baz danych (Dz.U. Nr 128, poz.1402 z późn. zm.);
- ustawa z dnia 29 sierpnia 1997 r. o ochronie danych osobowych (tekst jednolity Dz.U. z 2002 r. Nr101, poz. 926 z późn. zm.);
- ustawa z dnia 18 września 2001 r. o podpisie elektronicznym (Dz.U. Nr 130, poz.1450 z późn. zm.)
- ustawa z dnia 18 lipca 2002 r. o świadczeniu usług drogą elektroniczną (Dz.U. Nr 144, poz.1204 z późn. zm.);
- ustawa z dnia 6 września 2001 r. o dostępie do informacji publicznej (Dz.U. Nr 122, poz.1198 z późn. zm.);
- ustawa z dnia 6 czerwca 1997 r. Kodeks karny (Dz.U. Nr 88, poz.553 z późn. zm.).

113. Jaka jest zasadnicza różnica między ochroną własności intelektualnej i ochroną patentową?

Podstawową różnicą prawa patentowego jako systemu ochrony partykularnej jest wyłączenie możliwości uzyskania patentu przez różne podmioty na wynalazki, które są do siebie podobne.

Czyli mając coś opatentowane nie możesz stworzyć czegoś podobnego posiadającego podobne algorytmy, funkcje itp a w prawie autorskim chronisz tylko program a nie algorytm itd.

Jak coś jest chronione patentem to jest chronione jako patent i własność intelektualna, a jak coś jest chronione jako własność intelektualna, to niekoniecznie jest chronione patentem.

Prawo autorskie chroni tylko formę (kod źródłowy i wynikowy). Nie chroni jednak odkryć, idei, procedur, metod, zasad działania, koncepcji matematycznych.

114. Opisać na czym polega szpiegostwo komputerowe. (faworyt)

Szpiegostwo definiuje się jako działanie przestępne dokonywane na szkodę określonego państwa, polegające na wykonywaniu odpowiednich czynności na rzecz obcego wywiadu, a w szczególności na zbieraniu, przekazywaniu informacji organom obcego wywiadu lub na gromadzeniu i

przechowywaniu informacji w celu przekazania obcemu wywiadowi.

Artykuł 130 – paragrafy 1 do 4 określają szpiegostwo komputerowe jako:

- Działanie w obcym wywiadzie przeciwko Rzeczypospolitej Polskiej
- Działanie w obcym wywiadzie lub biorąc w nim udział, udziela informacji, które mogą

wyrządzić szkodę Rzeczypospolitej Polskiej,

- Kto, w celu udzielenia obcemu wywiadowi wiadomości określonych w § 2, gromadzi je lub przechowuje, włącza się do sieci komputerowej w celu ich uzyskania albo zgłasza gotowość działania na rzecz obcego wywiadu przeciwko Rzeczypospolitej Polskiej,

- Kierowanie lub tworzenie organizacji obcego wywiadu