



dr inż. Michał Malinowski
michal.malinowski@uth.edu.pl

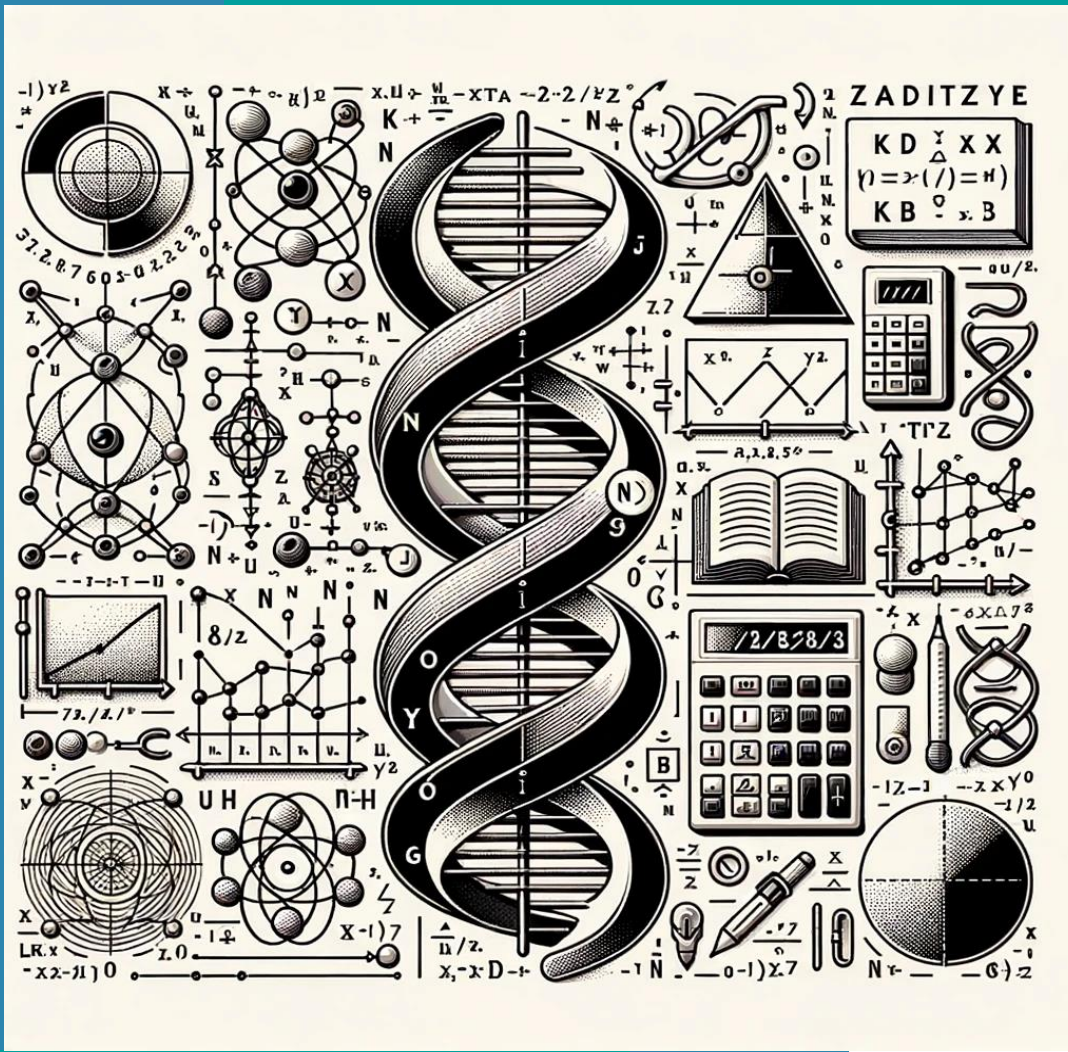


[https://www.linkedin.com/in/
michał-malinowski-malkon/](https://www.linkedin.com/in/michał-malinowski-malkon/)

Implementacja i korzystanie
z podstawowych struktur
danych.

Ćwiczenie 03

ALGORYTMY
I STRUKTURY DANYCH



Zagadnienia (2h)

- [Bezpieczne języki programowania](#)
- [Struktury proste](#)
 - [Zadanie: Struktury proste](#)
- [Struktury złożone liniowe](#)
 - [Zadanie: Tablica jednowymiarowa](#)
 - [Zadanie: Tablica dwuwymiarowa \(macierz\)](#)
 - [Zadanie: Wskaźniki](#)
- [Struktury złożone nielinowe](#)
- [Reprezentacja grafów](#)
 - [Zadanie: Drzewo binarne](#)

Biały Dom pilnie domaga się, aby programiści używali języków programowania bezpiecznych dla pamięci.

Luty 2024

Biuro Krajowego Dyrektora ds. Cyberbezpieczeństwa (ONCD)
w Białym Domu



Bezpieczne języki programowania

Automatyczne zarządzanie pamięcią

Eliminuje błędy związane z ręcznym zarządzaniem pamięcią, takie jak wycieki pamięci i podwójne zwalnianie pamięci, co zapobiega wielu potencjalnym problemom bezpieczeństwa.

Bezpieczne typowanie

Pomaga wykrywać błędy na wczesnym etapie (podczas kompilacji), zapobiegając nieprawidłowym operacjom na danych, które mogłyby prowadzić do błędów

Sprawdzanie granic

Dla tablic i innych struktur danych zapobiega przepełnieniom bufora, które mogą prowadzić do nadpisania danych i potencjalnych luk w bezpieczeństwie.

Zapobieganie wstrzyknięciu kodu

Uniemożliwia wstrzyknięcie i wykonanie nieautoryzowanego kodu, na przykład przez odpowiednie sanitowanie danych wejściowych i stosowanie bezpiecznych API.

Model własności i zarządzanie dostępem

Stosowanie modeli własności i pożyczania, które na poziomie kompilacji zapewniają, że dostęp do danych jest zarządzany w sposób bezpieczny, eliminując wyścigi danych i inne błędy związane z konkurencyjnym dostępem.

Immutability (niemutowalność)*

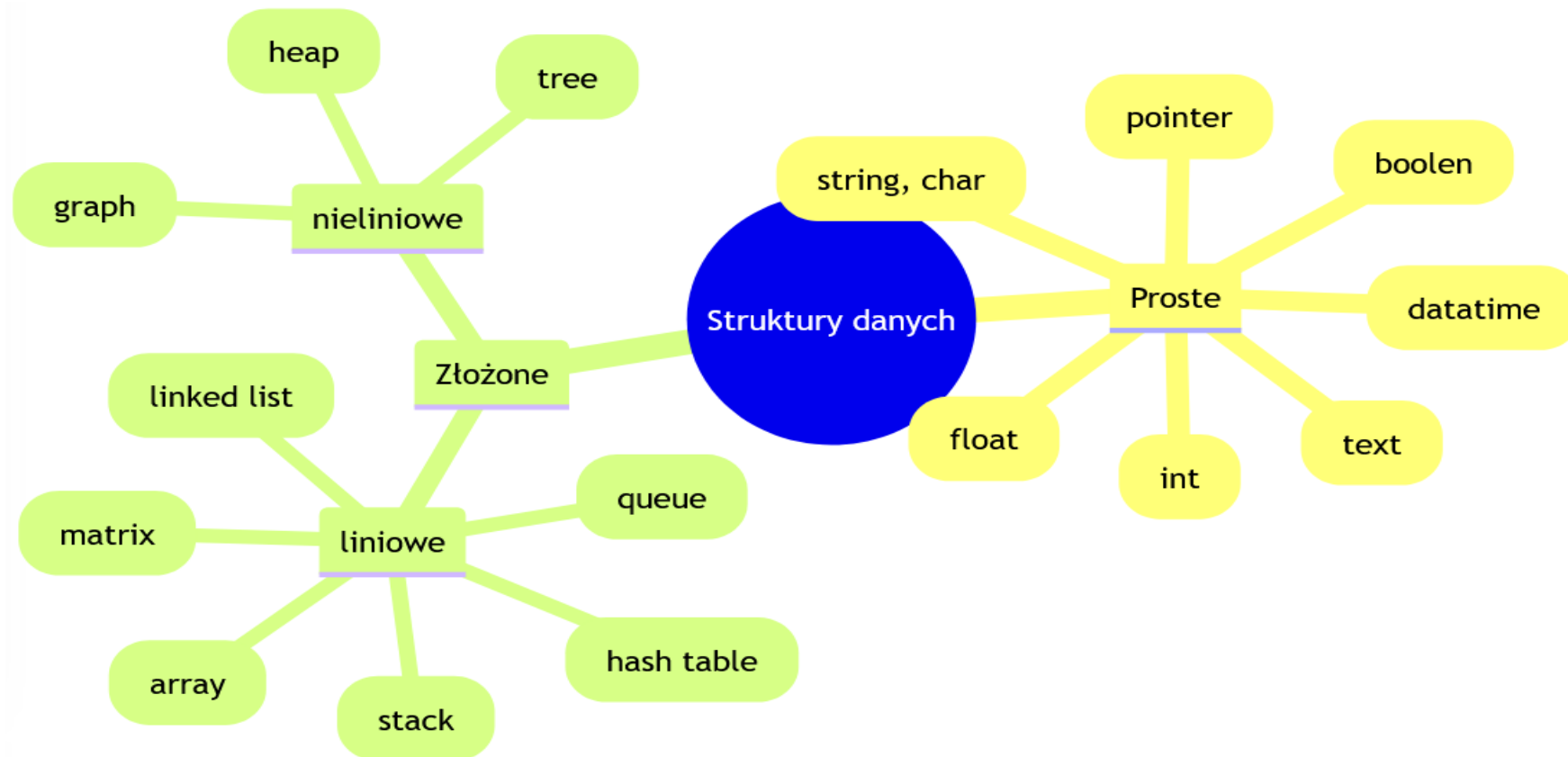
Wymuszanie stosowania niemutowalnych struktur danych pomaga w zapobieganiu błędom wynikającym ze zmian stanu, co jest szczególnie ważne w aplikacjach współbieżnych i rozproszonych.

*Immutability, czyli niemutowalność, to właściwość obiektów (zmiennych) w programowaniu, która nie pozwala na zmianę ich stanu po utworzeniu

Bezpieczne języki programowania

Bezpieczne języki	Pośrednie języki	Niebezpieczne języki
Rust	JavaScript	C
Swift	TypeScript	C++
Go	Kotlin	Assembly
Java	Scala	
C#	Python	
Haskell	Ruby	
Elixir	PHP	

Struktury proste



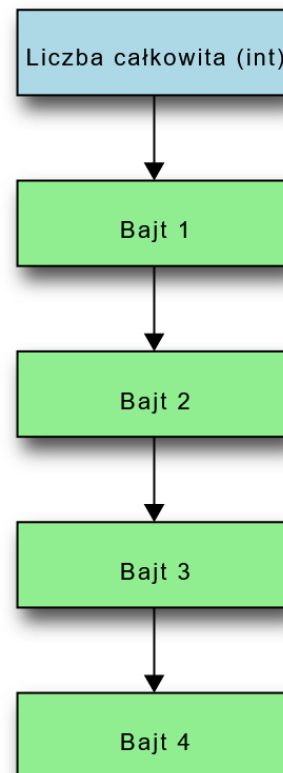
Struktury proste

Liczby całkowite (int)

Służą do przechowywania liczb całkowitych.

Zakres pamięci: 4 bajty (32 bity)

Przykład: `int x = 5;`



Struktury proste

Liczby zmiennoprzecinkowe (float)

Używane do przechowywania liczb rzeczywistych.

Zakres pamięci: 4 bajty (32 bity)

Przykład: `float y = 5.7;`

Mantysa

Mantysa to część liczby zmiennoprzecinkowej, która zawiera jej znaczące cyfry.

Wykładnik

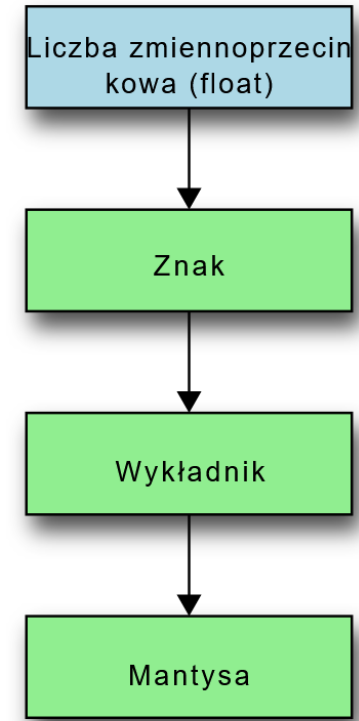
Wykładnik to część liczby zmiennoprzecinkowej, która określa potęgę, do której podniesiona jest liczba bazowa (zazwyczaj 2 w komputerach).

Przykład dla 2349.5678:

Znak: Plus (+), oznacza, że liczba jest dodatnia.

Wykładnik: 11, potęgę, do której podniesiona jest liczba bazowa.

Mantysa: 100100101101.1001, jest binarną reprezentacją części znaczącej liczby.



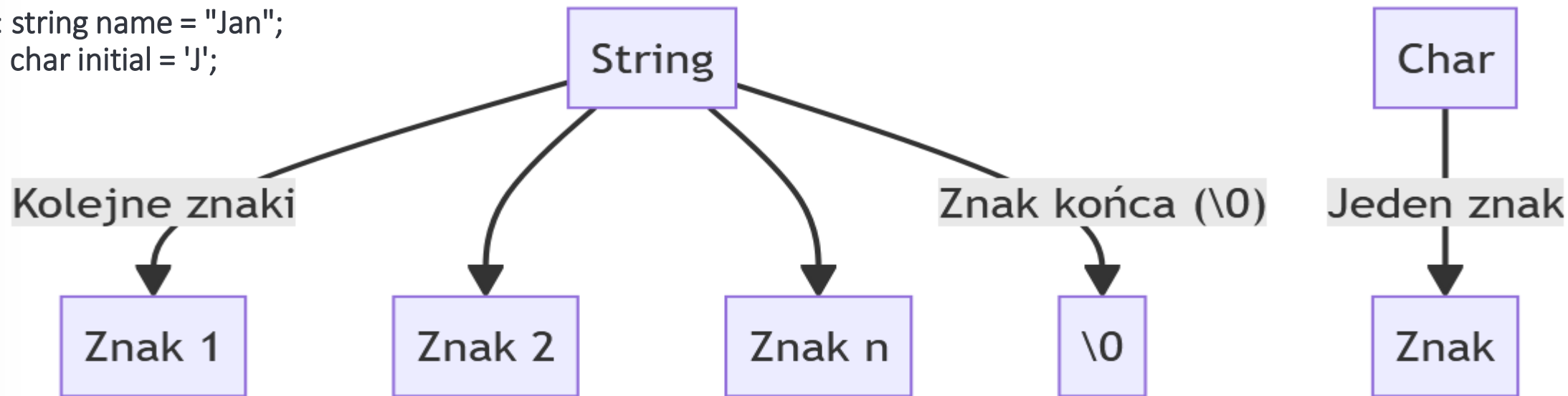
Struktury proste

Łańcuchy znaków (string, char)

string: służy do przechowywania ciągów znaków, zakres pamięci zależny od długości ciągu.

char: służy do przechowywania pojedynczych znaków, 1 bajt (8 bitów).

Przykłady: `string name = "Jan";`
`char initial = 'J';`



Znaki i wartości kodu ASCII

Standardowy Kod do Przetwarzania Informacji (ASCII) to zestaw znaków i ich odpowiadających liczbowych wartości.

- Obejmuje 128 znaków, reprezentowanych przez liczby od 0 do 127.
- Zawiera znaki kontrolne, cyfry, litery i znaki specjalne.

Znak	Wartość
['\n', '\t', ' ', '!', '"', '#', '\$', '%', '&', "'", '(', ')']	['10', '9', '32', '33', '34', '35', '36', '37', '38', '39', '40', '41']
['*', '+', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5']	['42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53']
['6', '7', '8', '9', ':', ';', '<', '=', '>', '?', '@', 'A']	['54', '55', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65']
['B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M']	['66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76', '77']
['N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y']	['78', '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89']
['Z', '[', '\', ']', '^', '_', '`', 'a', 'b', 'c', 'd', 'e']	['90', '91', '92', '93', '94', '95', '96', '97', '98', '99', '100', '101']
['f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q']	['102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '112', '113']
['r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '{', ' ', '}']	['114', '115', '116', '117', '118', '119', '120', '121', '122', '123', '124', '125']
['~']	['126']

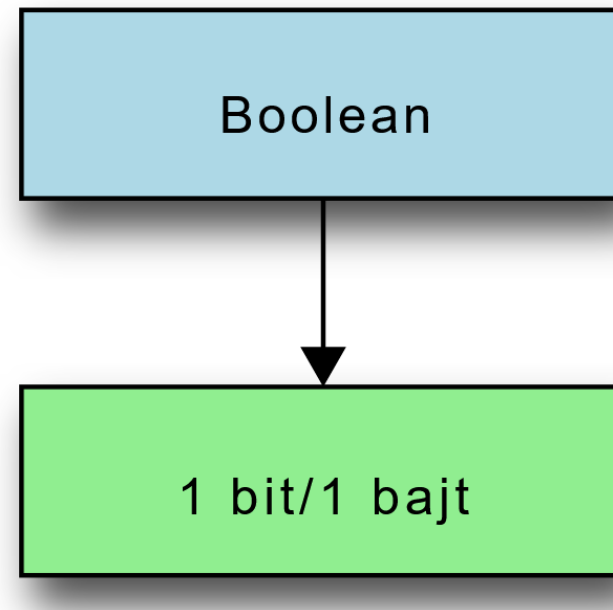
Struktury proste

Logiczne (boolean)

Przechowuje wartości logiczne: prawda/fałsz.

Zakres pamięci: 1 bajt (8 bitów)

Przykład: `bool isTrue = true;`



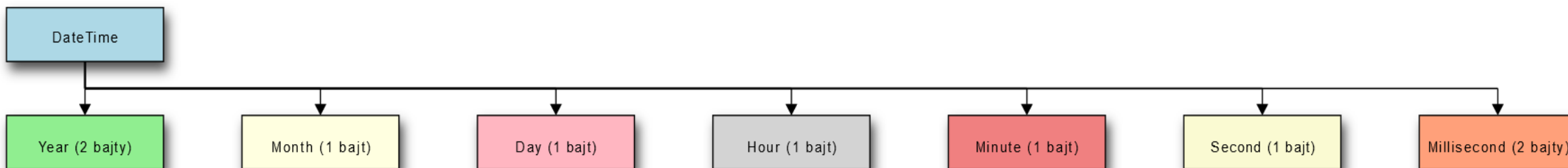
Struktury proste

Czas (datetime)

Używane do przechowywania dat i czasu.

Zakres pamięci: zależny od implementacji, często 8 bajtów (64 bity).

Przykład: `DateTime now = Now();`



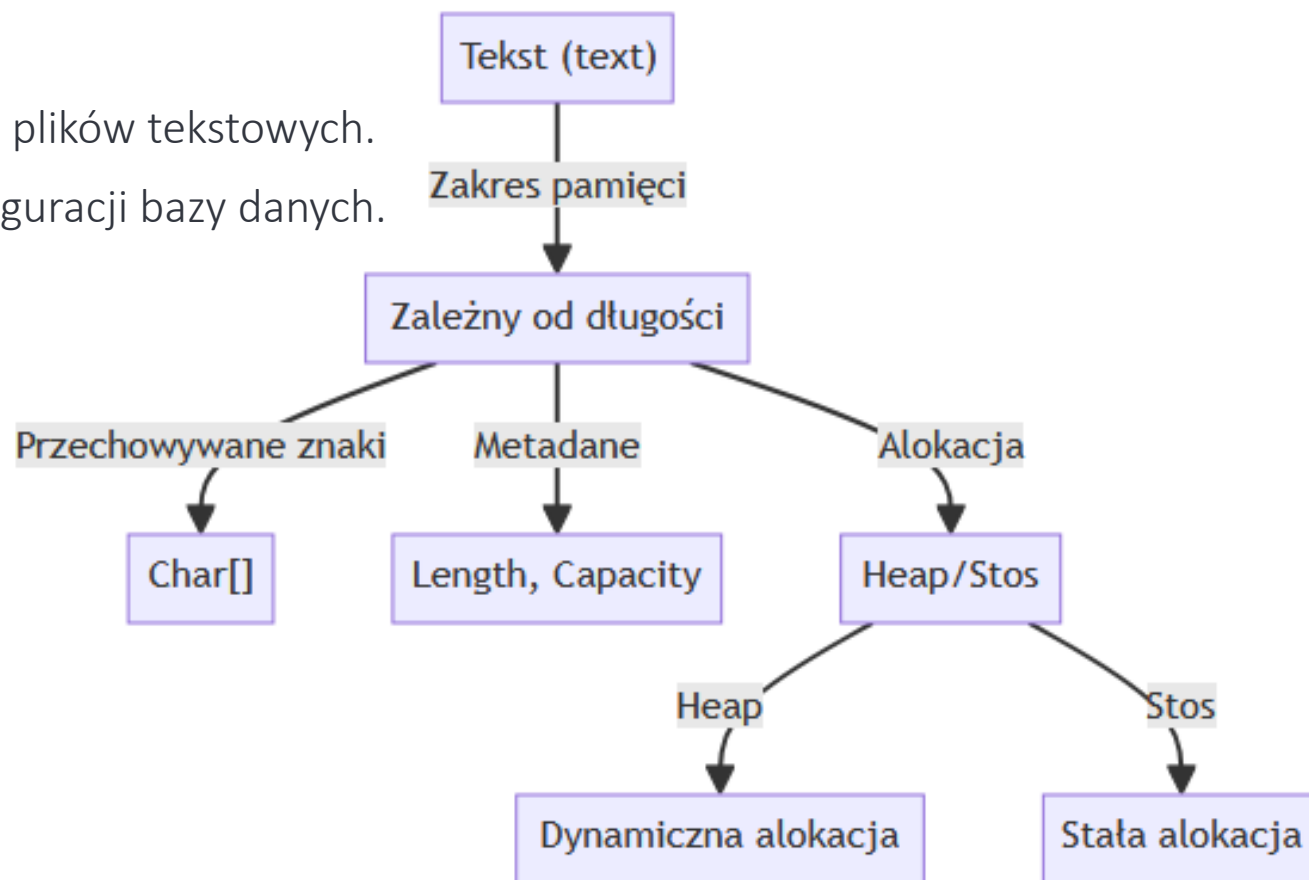
Struktury proste

Tekst (text)

Zazwyczaj używany w kontekście baz danych lub plików tekstowych.

Zakres pamięci: zależny od implementacji i konfiguracji bazy danych.

Przykład: `TEXT columnName;`



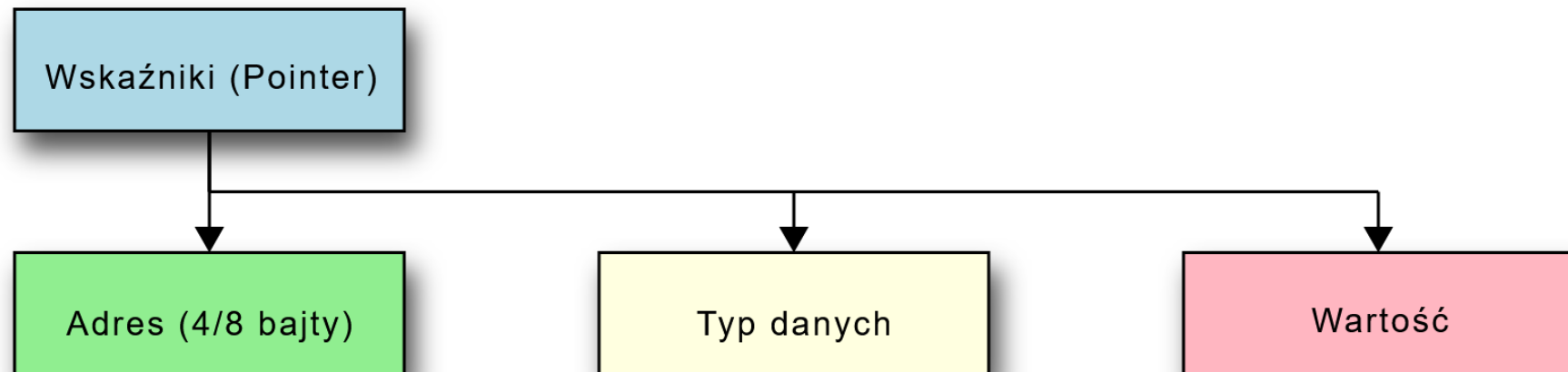
Struktury proste

Wskaźniki (pointer)

Zawierają adresy pamięci innych zmiennych.

Zakres pamięci: 4 bajty (32 bity) dla systemów 32-bitowych,
8 bajtów (64 bity) dla systemów 64-bitowych.

Przykład: `int *ptr;`
`ptr = &x;`



Zadanie: Struktury proste

Zadanie

Napisz funkcje w JavaScript, które wykonują poniższe operacje na różnych typach danych. Wyniki wyświetlaj w konsoli „console.log”:

1. Liczby całkowite (int): Oblicz sumę, różnicę, iloczyn i iloraz dwóch liczb całkowitych.
2. Liczby zmiennoprzecinkowe (float): Oblicz pole koła o zadanym promieniu.
3. Łańcuchy znaków (string, char): Połącz dwa łańcuchy znaków i wyświetl wynik.
4. Tekst (text): Zlicz ilość wystąpień danego słowa w tekście.
5. Logiczne (boolean): Sprawdź, czy liczba jest parzysta czy nieparzysta.
6. Czas (datetime): Oblicz różnicę między dwiema datami.

Zadanie: Struktury proste

Rozwiązanie

```
// Liczby całkowite (int)
function operacjeNaCalkowitych(a, b) {
  console.log("Suma: " + (a + b));
  console.log("Roznica: " + (a - b));
  console.log("Iloczyn: " + (a * b));
  console.log("Iloraz: " + (a / b));
}

// Liczby zmiennoprzecinkowe (float)
function poleKola(promien) {
  const pole = Math.PI * Math.pow(promien, 2);
  console.log("Pole kola o promieniu " + promien + " wynosi " + pole);
}
```

Zadanie: Struktury proste

Rozwiązanie

```
// Lancuchy znakow (string, char)
function polaczLancuchy(str1, str2) {
    const wynik = str1 + str2;
    console.log("Polaczone lancuchy: " + wynik);
}

// Tekst (text)
function zliczWystapieniaSlowa(tekst, slowo) {
    const regex = new RegExp("\\b" + slowo + "\\b", 'gi');
    const liczba = (tekst.match(regex) || []).length;
    console.log("Slowo \"" + slowo + "\" wystepuje " + liczba + " razy w tekście");
}

// Logiczne (boolean)
function czyParzysta(liczba) {
    const wynik = liczba % 2 === 0;
    console.log("Liczba " + liczba + " jest " + (wynik ? "parzysta" : "nieparzysta"));
}
```

Zadanie: Struktury proste

Rozwiązanie

```
// Czas (datetime)
function roznicaDat(data1, data2) {
  const roznica = Math.abs(data1 - data2);
  const dni = roznica / (1000 * 60 * 60 * 24);
  console.log("Roznica miedzy datami wynosi " + dni + " dni");
}

// Testowanie funkcji
operacjeNaCalkowitych(10, 5);
poleKola(5.0);
polaczLancuchy("Ala ", "ma kota");
zliczWystapieniaSlowa("Ala ma kota, a kot ma Ale", "ma");
czyParzysta(10);

const data1 = new Date("2021-01-01");
const data2 = new Date("2021-12-31");
roznicaDat(data1, data2);
```



Zadanie: Wskaźniki

Wskaźniki (pointer) (2 pkt)

1. Przygotuj dokument, w którym opiszesz, czym jest Wskaźniki (pointer).
2. W tym samym dokumencie opisz różne zastosowania Wskaźników (pointer) w programowaniu



Przedstawienie wyniku

- Dokument tekstowy (np.: PDF, Word)
- Umieszczenie wyniku w Moodle

Termin

- Podany przy zadaniu w Moodle (około 2 tygodnie)



"U mnie działa!"

Pierwsze prawo informatyka

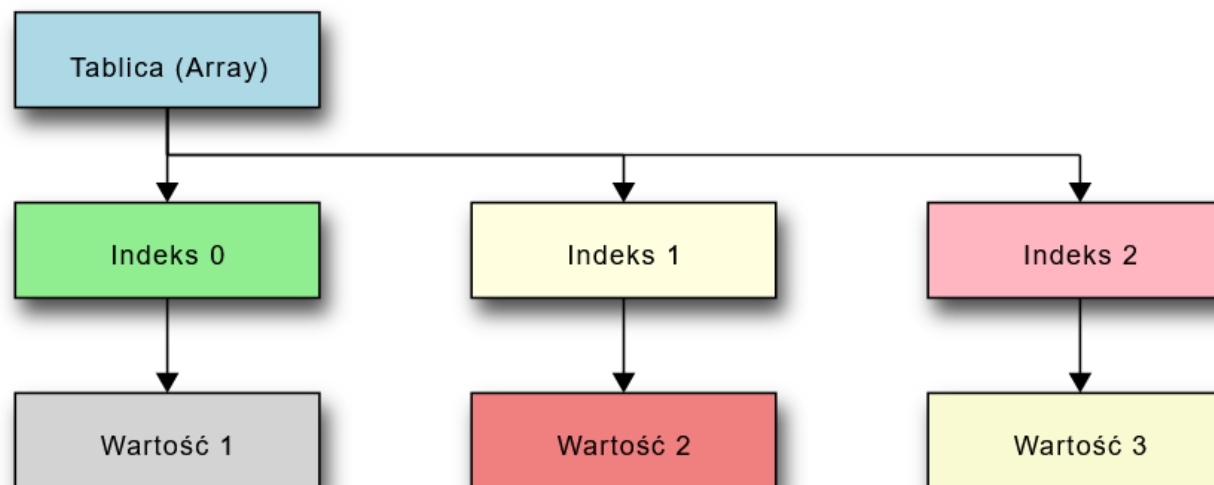
Struktury złożone liniowe

Tablica (Array)

Służy do przechowywania elementów w indeksowanych komórkach.

Zakres pamięci: zależny od typu elementu i wielkości tablicy.

Przykład: `$cena = array('jogurt'=>2, 'maslo'=>1.50, 'zapiekanka'=>4, 'gazeta'=>5);`



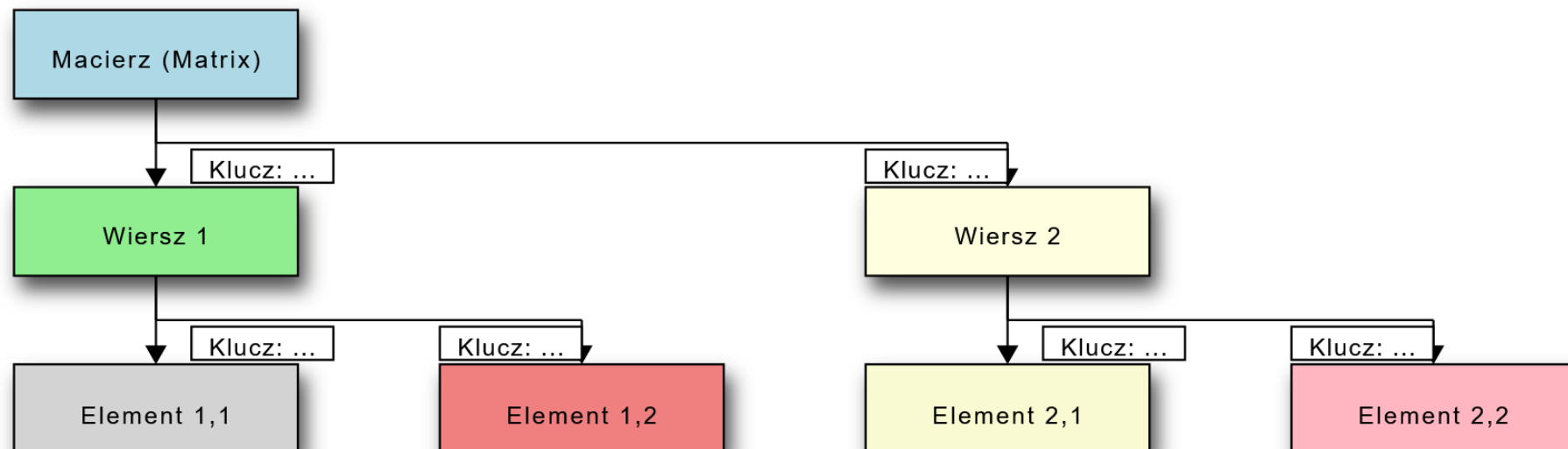
Struktury złożone liniowe

Macierz (Matrix)

Służy do przechowywania elementów w dwuwymiarowej tablicy.

Zakres pamięci: zależny od typu elementu i wymiarów macierzy.

Przykład: `int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};`



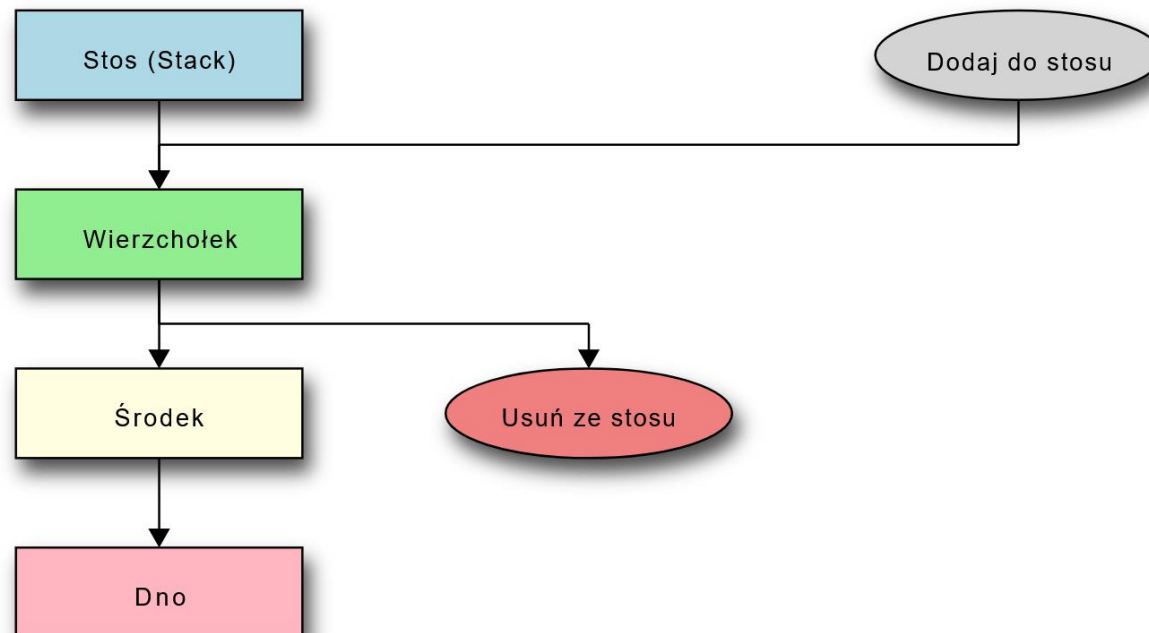
Struktury złożone liniowe

Stos (Stack)

Służy do przechowywania elementów w strukturze LIFO (Last In, First Out).

Zakres pamięci: dynamiczny.

Przykład: `std::stack<int> s;`



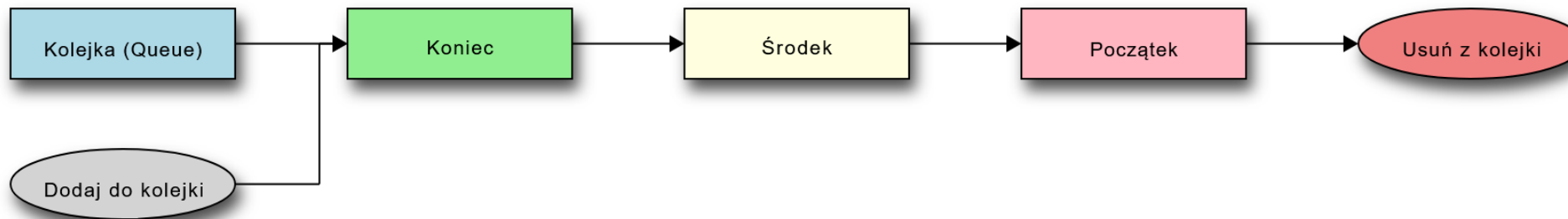
Struktury złożone liniowe

Kolejka (Queue)

Służy do przechowywania elementów w strukturze FIFO (First In, First Out).

Zakres pamięci: dynamiczny.

Przykład: `std::queue<int> q;`



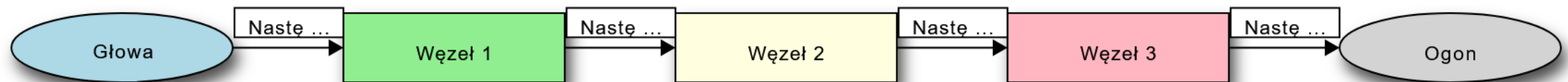
Struktury złożone liniowe

Lista łańcuchowa (Linked List)

Służy do przechowywania elementów w węzłach złożonych z danych i wskaźnika na następny element.

Zakres pamięci: dynamiczny.

Przykład: `std::list<int> l;`



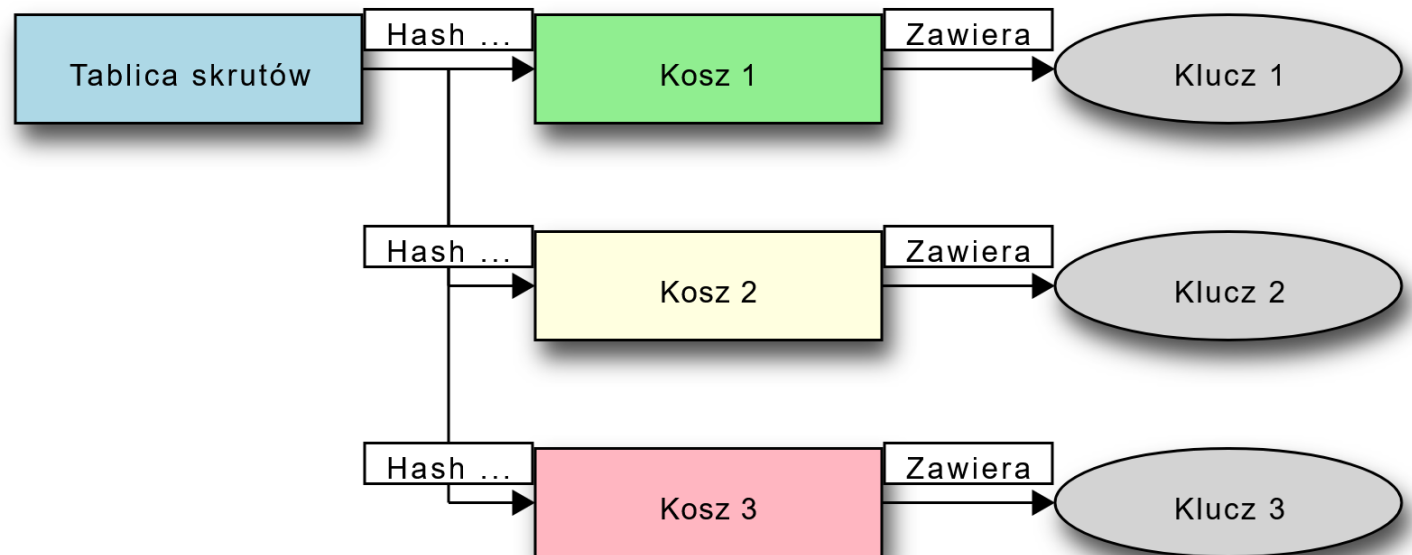
Struktury złożone liniowe

Tablica skrutów (Hash Table)

Służy do przechowywania par klucz-wartość.

Zakres pamięci: dynamiczny.

Przykład: `std::unordered_map<int, int> m;`



Zadanie: Tablica jednowymiarowa

Zadanie

Utwórz w JavaScript:

1. Tablicę jednowymiarową zawierającą 10 losowych liczb całkowitych z zakresu od 1 do 100.
2. Napisz funkcję, która obliczy sumę elementów tablicy.
3. Napisz funkcję, która znajdzie i zwróci największą liczbę w tablicy.

Zadanie: Tablica jednowymiarowa

Rozwiązanie

Utwórz w JavaScript:

1. Tablicę jednowymiarową zawierającą 10 losowych liczb całkowitych z zakresu od 1 do 100.

```
// Utworzenie pustej tablicy jednowymiarowej
losoweLiczby = [];

// Wypełnienie tablicy 10 losowymi liczbami całkowitymi od 1 do 100
for (i = 0; i < 10; i++) {
    losoweLiczby.push(Math.floor(Math.random() * 100) + 1);
}

console.log("Wygenerowana tablica: ", losoweLiczby);
```

Zadanie: Tablica jednowymiarowa

Rozwiązanie

Utwórz w JavaScript:

2. Napisz funkcję, która obliczy sumę elementów tablicy.

```
// Funkcja obliczająca sumę elementów tablicy
function obliczSume(tablica) {
    suma = 0;
    for (i = 0; i < tablica.length; i++) {
        suma += tablica[i];
    }
    return suma;
}
```

Zadanie: Tablica jednowymiarowa

Rozwiązanie

Utwórz w JavaScript:

3. Napisz funkcję, która znajdzie i zwróci największą liczbę w tablicy.

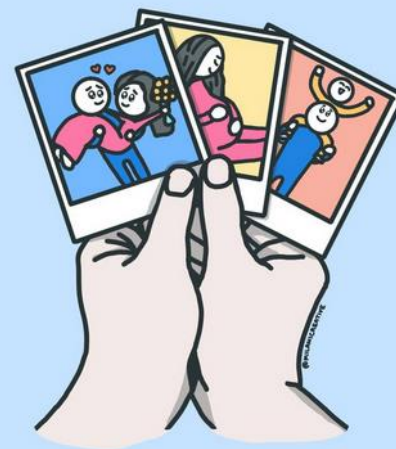
```
// Funkcja znajdujaca i zwracajaca najwieksza liczbe w tablicy
function znajdzMax(tablica) {
    max = tablica[0];
    for (i = 1; i < tablica.length; i++) {
        if (tablica[i] > max) {
            max = tablica[i];
        }
    }
    return max;
}
```

```
// Testowanie funkcji
suma = obliczSume(losoweLiczby);
max = znajdzMax(losoweLiczby);

console.log("Suma elementow tablicy: " + suma);
console.log("Najwieksza liczba w tablicy: " + max);
```



RICH



WEALTHY

  @milaniCREATIVE

Zadanie: Tablica dwuwymiarowa (macierz)

Zadanie

Utwórz w JavaScript:

1. Macierz (tablica dwuwymiarowa) o wymiarach 3x3, wypełnioną losowymi liczbami całkowitymi z zakresu od 1 do 10.
2. Napisz funkcję, która obliczy sumę elementów macierzy.
3. Napisz funkcję, która znajdzie i zwróci najmniejszą liczbę w macierzy.

Zadanie: Tablica dwuwymiarowa (macierz)

Rozwiązanie

Utwórz w JavaScript:

1. Macierz (tablica dwuwymiarowa) o wymiarach 3x3, wypełnioną losowymi liczbami całkowitymi z zakresu od 1 do 10.

```
// Utworzenie pustej macierzy
macierz = [];

// Wypełnienie macierzy losowymi liczbami od 1 do 10
for (i = 0; i < 3; i++) {
    wiersz = [];
    for (j = 0; j < 3; j++) {
        wiersz.push(Math.floor(Math.random() * 10) + 1);
    }
    macierz.push(wiersz);
}

console.log("Wygenerowana macierz: ", macierz);
```

Zadanie: Tablica dwuwymiarowa (macierz)

Rozwiązanie

Utwórz w JavaScript:

2. Napisz funkcję, która obliczy sumę elementów macierzy.

```
// Funkcja obliczająca sumę elementów macierzy
obliczSume = function(macierz) {
    suma = 0;
    for (i = 0; i < macierz.length; i++) {
        for (j = 0; j < macierz[i].length; j++) {
            suma += macierz[i][j];
        }
    }
    return suma;
};
```

Zadanie: Tablica dwuwymiarowa (macierz)

Rozwiązanie

Utwórz w JavaScript:

3. Napisz funkcję, która znajdzie i zwróci najmniejszą liczbę w macierzy.

```
// Funkcja znajdująca i zwracająca najmniejszą liczbę w macierzy
znajdzMin = function(macierz) {
    min = macierz[0][0];
    for (i = 0; i < macierz.length; i++) {
        for (j = 0; j < macierz[i].length; j++) {
            if (macierz[i][j] < min) {
                min = macierz[i][j];
            }
        }
    }
    return min;
};
```

```
suma = obliczSume(macierz);
min = znajdzMin(macierz);

console.log("Suma elementów macierzy: " + suma);
console.log("Najmniejsza liczba w macierzy: " + min);
```

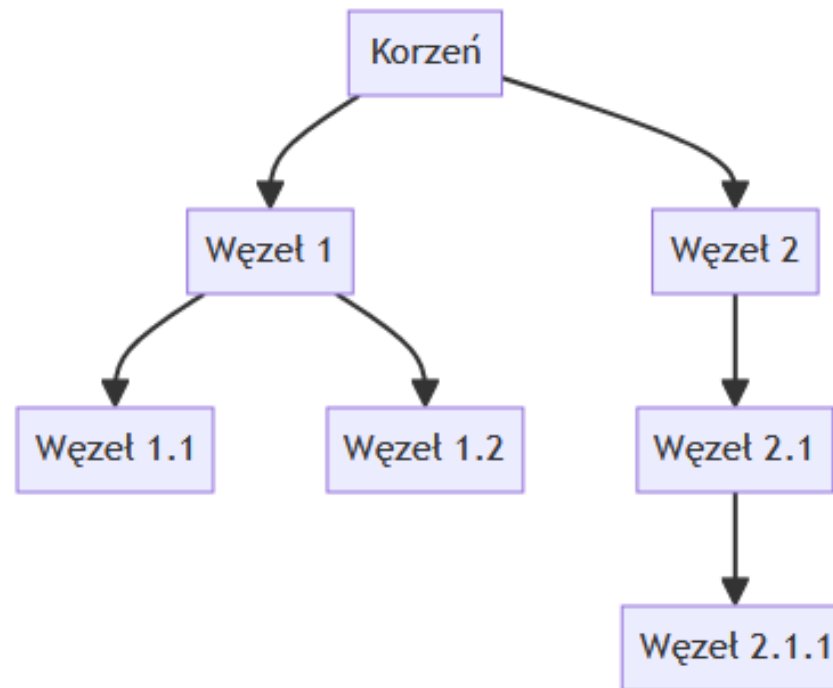
Struktury złożone nielinowe

Drzewo (Tree)

Służy do przechowywania elementów w hierarchicznej strukturze danych.

Zakres pamięci: dynamiczny.

Przykład: `std::set<int> tree;`



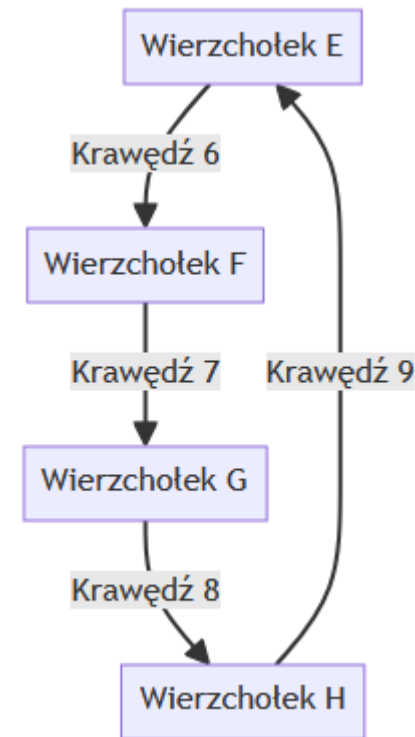
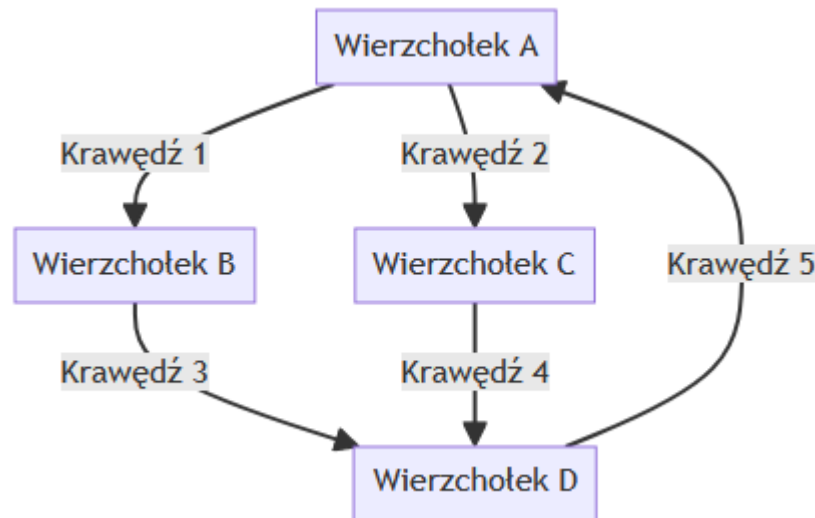
Struktury złożone nielinowe

Graf (Graph)

Służy do przechowywania elementów w strukturze sieciowej.

Zakres pamięci: dynamiczny.

Przykład: `std::vector<std::pair<int, int>> graph;`



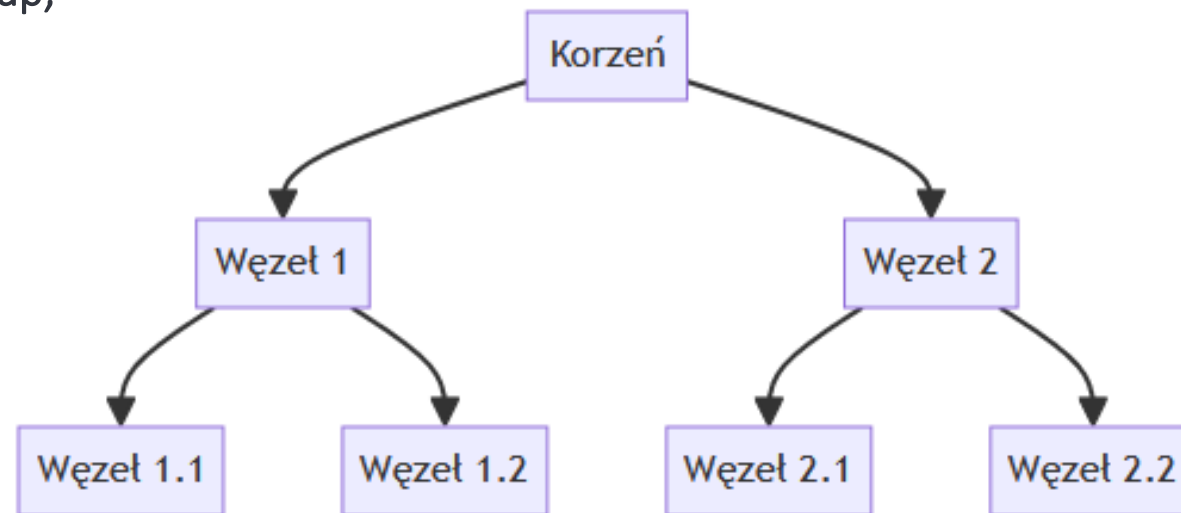
Struktury złożone nielinowe

Kopiec (Heap) - drzewo binarne

Służy do przechowywania elementów w strukturze drzewa binarnego.

Zakres pamięci: dynamiczny.

Przykład: `std::priority_queue<int> heap;`



Struktury złożone nielinowe

Kopiec (Heap) - drzewo binarne

Przeszukiwanie drzewa binarnego

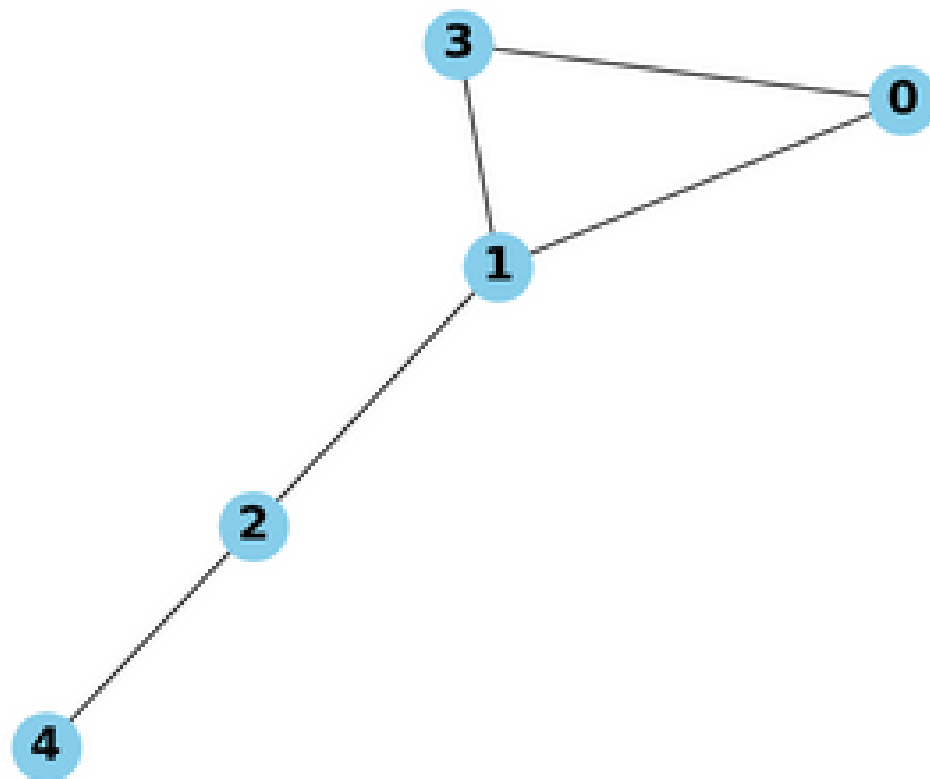
Liczba Kroków	Maksymalny Zakres (ilość elementów)
5	$2^5 = 32$
10	$2^{10} = 1024$
15	$2^{15} = 32768$
20	$2^{20} = 1048576$
25	$2^{25} = 33554432$
30	$2^{30} = 1073741824$

Złożoności czasowej to $O(\log n)$.

Dla liczby kroków w tabeli, możesz przeszukać 2^k elementów w k krokach przy użyciu drzewa binarnego.

Reprezentacja grafów

Graficzna reprezentacja



Reprezentacja grafów

macierz sąsiedztwa (adjacency matrix)

jest przedstawiana jako tablica dwuwymiarowa, gdzie indeksy wierszy i kolumn reprezentują numery wierzchołków, a wartości elementów równe 1 oznaczają krawędź łączącą wierzchołki określone numerem wiersza i kolumny;

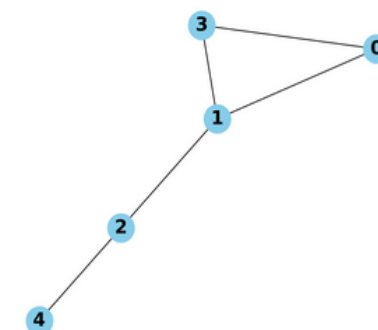
macierz incydencji (incidence matrix)

jest przedstawiana jako tablica dwuwymiarowa, gdzie indeksy wierszy reprezentują numery wierzchołków, a indeksy kolumn oznaczają numery krawędzi. Elementy równe 1 oznaczają krawędzie oznaczone numerami kolumn incydentnych z wierzchołkami oznaczonymi numerami wierszy;

Macierz sąsiedztwa

Wierzchołki	0	1	2	3	4
0	0	1	0	1	0
1	1	0	1	1	0
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	0
Wierzchołki	0	1	2	3	4

Graficzna reprezentacja



Macierz incydencji

Wierzchołki	0	1	2	3	4
0	1	1	0	0	0
1	1	0	1	1	0
2	0	0	1	0	1
3	0	1	0	1	0
4	0	0	0	0	1
Wierzchołki	0	1	2	3	4

Krawędzie

Reprezentacja grafów

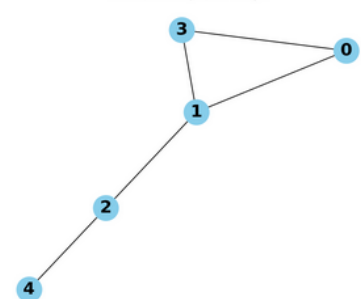
lista sąsiedztwa (adjacency list)

jest przedstawiana jako lista, gdzie indeksy reprezentują numery wierzchołków, a każdy element tej listy jest listą numerów sąsiednich wierzchołków z numerem wierzchołka będącym w indeksie;

Lista sąsiedztwa:

0: [1, 3]
1: [0, 2, 3]
2: [1, 4]
3: [0, 1]
4: [2]

Graficzna reprezentacja



lista incydencji (incidence list)

jest przedstawiana jako lista, gdzie indeksy reprezentują numery wierzchołków, a każdy element tej listy jest listą numerów krawędzi incydentnych z numerem wierzchołka będącym w indeksie;

Lista incydencji:

0: [(0, 1), (0, 3)]
1: [(1, 0), (1, 2), (1, 3)]
2: [(2, 1), (2, 4)]
3: [(3, 0), (3, 1)]
4: [(4, 2)]

lista krawędzi (list of edges)

jest przedstawiana jako lista par numerów wierzchołków dla każdej krawędzi.

Lista krawędzi:

(0, 1), (2, 4), (1, 2), (0, 3), (1, 3)



Zadanie: Drzewo binarne

Drzewo binarne (2 pkt)

1. Przygotuj dokument, w którym opiszesz, czym jest drzewo binarne.
2. W tym samym dokumencie opisz różne zastosowania drzew binarnych w informatyce



Przedstawienie wyniku

- Dokument tekstowy (np.: PDF, Word)
- Umieszczenie wyniku w Moodle

Termin

- Podany przy zadaniu w Moodle (około 2 tygodnie)