СТРУКТУРИ ДАНИХ

Лекція 5

Дисципліна «Алгоритми та структури даних»

Поняття структури даних

В комп’ютерних науках під структурою даних розуміють способи організації, зберігання та керування даними в комп'ютері; вони визначають не лише те, як дані зберігаються, а й те, як до них можна ефективно отримати доступ і маніпулювати ними

Структури даних – це сукупність елементів даних і відношень між ними

Елементи даних – як прості дані, так і структури даних

Відношення між даними – функціональні зв’язки між ними і покажчики на те, де знаходяться ці дані

Елемент відношень – це сукупність всіх зв’язків елемента з іншими елементами даних, що розглядається в структурі

Структури даних не прив'язані до якоїсь конкретної мови програмування

Рівні опису даних

1.Абстрактний (математичний) рівень – рівень, що визначає структуру як абстрактний тип даних; визначення структур через набір операцій та правил, без урахування того, як вони будуть реалізовані.

Наприклад, стек визначається лише двома операціями: **PUSH** (додати елемент) та **POP** (видалити елемент). Математично це послідовність, що підтримує принцип **LIFO** (Last-In, First-Out)

Абстрактний рівень відповідає на питання **ЩО** робить структура, конкретна структура (наприклад, зв’язаний список) відповідає на питання **ЯК** робиться

2. Логічний (концептуальний) рівень – пов'язує математичну концепцію з **конкретною логічною моделлю** для її реалізації

Вибір конкретної структури даних, яка найкраще підходить для реалізації абстрактного типу даних. Визначаються логічні зв'язки між елементами

Наприклад, стек може бути логічно реалізований як **зв'язаний список** (де кожен вузол має посилання на наступний) або як **динамічний масив** (вектор)

Рівні опису даних

3. Фізичний (внутрішній) рівень – описує, як логічна структура розміщується та організовується у фізичній пам'яті комп'ютера: деталізація розмірів елементів, використання покажчиків, суміжне чи несуміжне розміщення в пам'яті, методи індексування

Наприклад, якщо стек реалізований як динамічний масив, елементи розташовані у суміжних комірках пам'яті. Якщо як зв'язаний список, вузли можуть бути розкидані по пам'яті, а логічна послідовність підтримується за допомогою адрес (покажчиків)

4. Рівень реалізації/програмування – практичний рівень, на якому структура даних кодується конкретною мовою програмування.

Наприклад, реалізація стека в Python за допомогою вбудованого List (що є динамічним масивом). Реалізація зв'язаного списку в Java за допомогою класів та посилань

Класифікація структур даних

Простими називаються такі структури даних, які не можуть бути розділені на складові частини більші, ніж біти. З логічної точки зору, прості дані є неподільними одиницями.

Приклади: цілі числа (Integer), символи (Char), логічні значення (Boolean).

Інтегрованими називаються структури даних, складовими частинами яких є інші структури даних – прості або інтегровані. Інтегровані структури даних створюються програмістом із використанням засобів інтеграції даних, які надаються мовами програмування

Наприклад, граф: базова структура (вершини) − масив або список (для зберігання всіх вершин графа); інтегрований елемент (ребра) − кожен елемент масиву містить зв'язаний список сусідніх вершин

Класифікація структур даних

В залежності від характеру взаємного розташування елементів у пам'яті:
Лінійні структури поділяють на структури з **послідовним розподілом елементів у пам'яті** (вектори, рядки, масиви, стеки, черги) і **структури з довільним зв'язним розподілом** елементів у пам'яті (однозв'язні і двозв'язні лінійні списки)
Нелінійні структури – багатозв'язні списки, дерева, графи

За видом пам'яті, що використовується для збереження даних, існує поділ на структури даних для оперативної і для зовнішньої пам'яті
Структури даних для **оперативної пам'яті** – це дані, розміщені в статичній і динамічній пам'яті комп'ютера (якщо змінюється кількість елементів і/або відношення між ними)
Структури даних для **зовнішньої пам'яті** називають файловими структурами чи файлами

Статичні структури даних

Статичні структури відрізняються відсутністю змінюваності, пам’ять для них виділяється автоматично – як правило, на етапі компіляції, або при виконанні – в момент активізації того програмного блоку, в якому вони описані

Ключові характеристики статичних структур:

1. Фіксований розмір – не можна додавати більше елементів, ніж було визначено спочатку
2. Безперервне розміщення в пам’яті – елементи, як правило, зберігаються у суміжних комірках пам’яті
3. Ефективний доступ – доступ до будь-якого елемента за його індексом відбувається за константний час O(1)

Приклади: статичний масив, стек і черга, реалізовані на статичному масиві

Динамічні структури даних

Динамічна структура даних – це структура, розмір якої може змінюватися (зростати або зменшуватися) під час виконання програми

Ключові характеристики динамічних структур:

1. Гнучкість розміру – пам'ять виділяється або звільняється за потреби
2. Несуміжне розміщення – елементи можуть зберігатися у різних, несуміжних ділянках пам'яті (наприклад, у зв'язаних списках), поєднуючись за допомогою покажчиків або посилань
3. Ефективні модифікації – операції додавання або видалення елементів зазвичай є швидкими (O(1)) у порівнянні зі статичними структурами

Приклади: динамічний масив, зв’язаний список, дерева, графи

Лінійні структури даних

Лінійні структури даних - це такі структури, в яких елементи розташовані **послідовно** або **один за одним**, формуючи єдину пряму лінію або послідовність. Це означає, що кожен елемент, крім першого і останнього, має одного **попередника** і одного **наступника**

Вони є фундаментальними для програмування, оскільки забезпечують найпростіший спосіб організації та доступу до даних

Масиви

Масив в багатьох мовах програмування – впорядкований скінчений набір даних ***одного типу***, які зберігаються в *послідовно* розташованих комірках оперативної пам'яті і мають спільну назву, яку надає користувач (це визначення є **коректним** і описує фундаментальну структуру **статичного масиву**, яка використовується в наукових та системних обчисленнях)

Масив складається з **елементів**. Кожен елемент має **індекси**, за якими його можна знайти у масиві. Нумерація елементів масиву завжди починається з нуля

Кількість індексів визначає **розмірність** масиву. Розрізняють одно- та багатовимірні масиви. **Розмір** – це кількість елементів масиву

Властивості масивів

- **Елементи масиву розташовані безпосередньо один за одним у пам'яті.** Це є головною причиною його ефективності
- **Статичні масиви** (наприклад, у C/C++) мають фіксований розмір, визначений під час компіляції, і його не можна змінити
- **Динамічні масиви** (наприклад, Python list або Java ArrayList) можуть змінювати свій розмір під час виконання, але внутрішньо вони все одно зберігаються суцільним блоком (при зміні розміру створюється новий, більший масив, і дані копіюються)
- **Гомогенність (один тип даних) -** традиційно, масив зберігає елементи лише одного типу (наприклад, лише цілі числа або лише рядки)
- **Доступ за індексом -** доступ до будь-якого елемента здійснюється за його **індексом** (порядковим номером), який зазвичай починається з 0

Матриця

Матриця - двовимірний масив, що виглядає як список стовпців і рядків, на перетині яких знаходяться елементи даних

Це прямокутний масив, у якому кількість рядків та стовпців задає його розмір

	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	1	0
5	0	1	0	0	0

Зв'язаний список

У типовому **списку** (List) елементи розташовані в певному, послідовному порядку, і кожен елемент має чітко визначеного попередника (крім першого) і наступника (крім останнього)

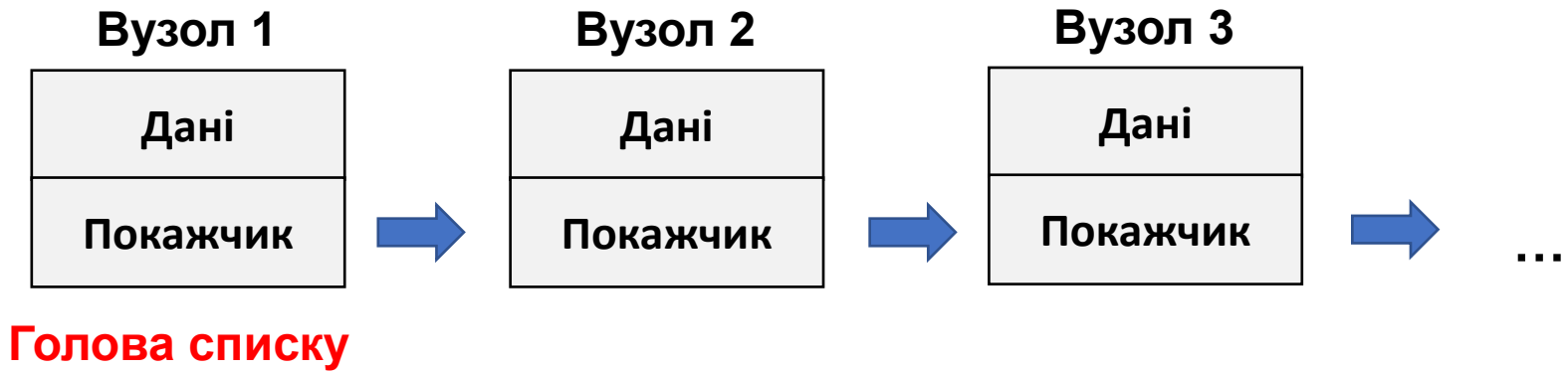
На відміну від масивів, зв’язаний список не зберігає елементи даних у суміжних місцях пам’яті. Зв'язаний список складається з окремих вузлів, які можуть бути **«розкидані» по пам'яті** і з'єднані між собою за допомогою **вказівників (покажчиків, посилань)**

Довжина списку дорівнює числу елементів у списку, список нульової довжини називається порожнім списком

Зв’язаний список

Зв'язний список є найпростішим типом даних динамічної структури, що складається з елементів (**вузлів**). Кожен вузол включає в себе в класичному варіанті два поля:

- дані (в якості даних може виступати змінна, об'єкт класу або структури і т. д.)
- покажчик на наступний вузол в списку

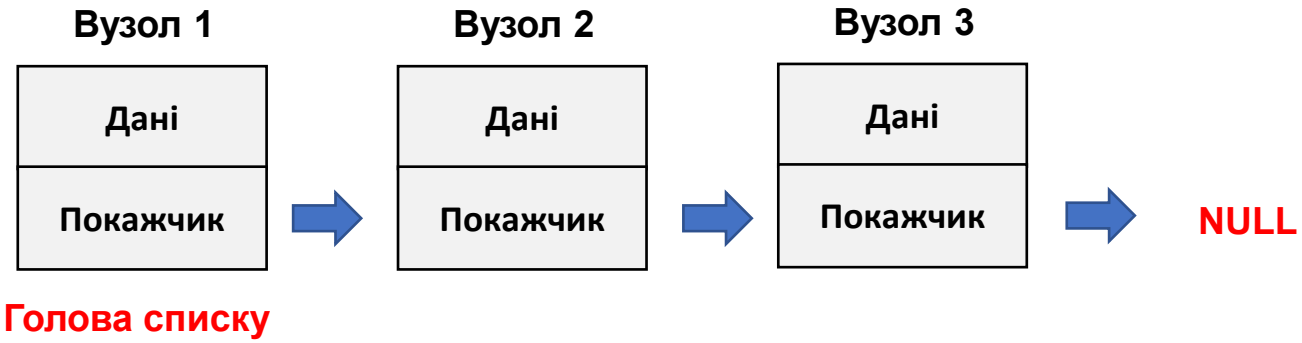


Доступ до списку здійснюється через покажчик, який містить адресу першого елемента списку - голова (head) списку

Класифікація списків

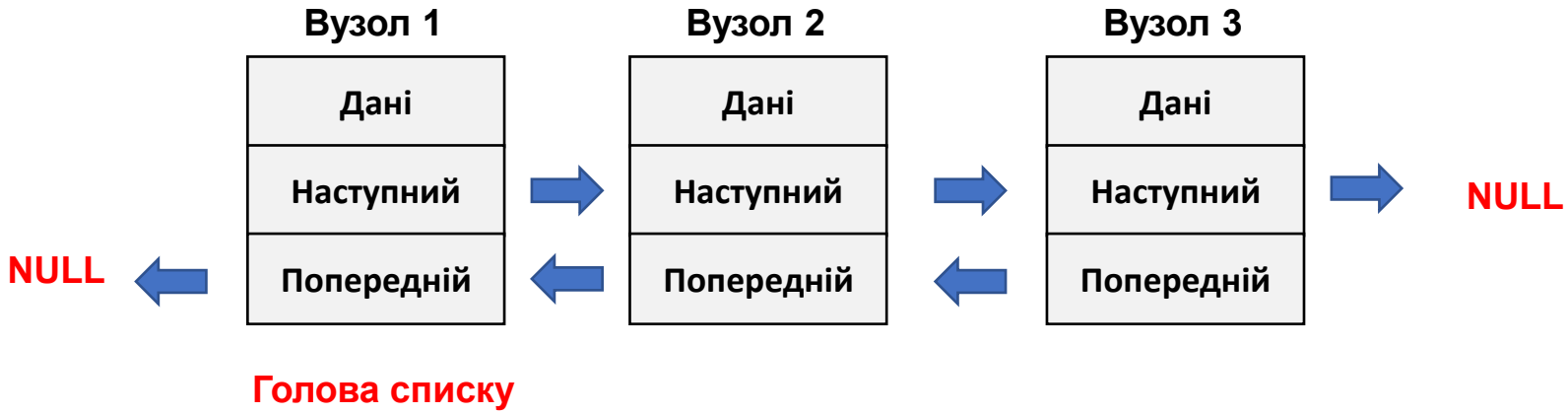
Однозв'язний лінійний список

Кожен вузол містить 1 поле покажчика на наступний вузол. Поле покажчика останнього вузла містить нульове значення (вказує на NULL)



Двозв’язаний лінійний список

Кожен вузол містить два поля покажчиків: на наступний і на попередній вузол. Поле покажчика на наступний вузол останнього вузла містить нульове значення (вказує на NULL). Поле покажчика на попередній вузол першого вузла (голови списку) також містить нульове значення (вказує на NULL)



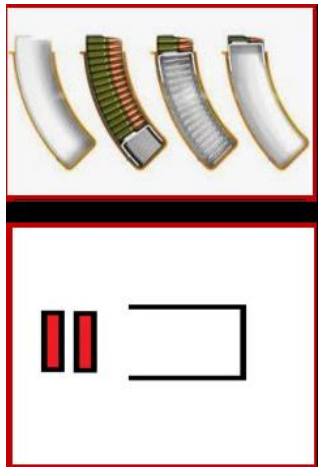
Операції над структурами даних

- **Операція створення** – це виділення пам’яті для зберігання структури даних. Пам’ять може резервуватися в процесі виконання програми або на етапі компіляції. Незалежно від мови програмування, що використовується, наявні в програмі структури даних явно або неявно **оголошуються** операторами створення
- **Операція знищення** структур даних протилежна до операції створення – дозволяє ефективно використовувати пам’ять; звільнює пам’ять, яку займала структура, для подальшого використання
- **Операція вибору** використовується для доступу до даних в самій структурі. Форма операції доступу залежить від типу структури даних, до якої здійснюється звертання. Метод доступу – одна з найбільш важливих властивостей структур, тому що вона має безпосереднє відношення до вибору конкретної структури даних
- **Операція поновлення** дозволяє змінити значення даних в структурі даних. Операцією поновлення є, наприклад, операція присвоєння, або передача параметрів

Основні операції зі списком

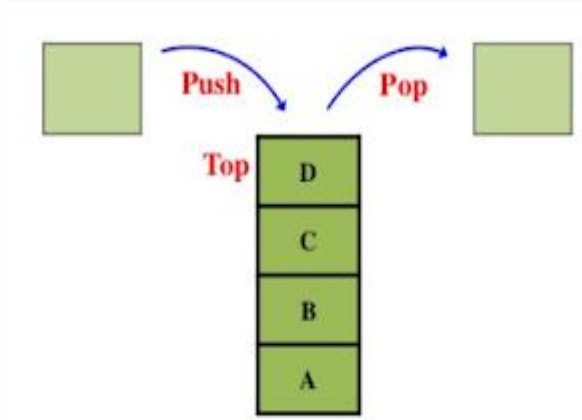
- створення списку;
- виведення (перегляд) списку;
- прохід списком;
- пошук елемента у списку;
- перевірка відсутності елементів у списку (NULL);
- долучення (вставлення) вузла;
- вилучення вузла;
- видалення всього списку.

Стек



Стек – це лінійна структура даних, яка дотримується певного порядку виконання операцій (як в магазині стрілецької зброї)
Порядок може бути LIFO (останній прийшов, перший вийшов) або FILO (першим прийшов останній вийшов)

В стеку виконуються основні три основні операції:
Push: додає елемент у стек. Якщо стек заповнений, то це називається умовою переповнення.
Pop: видаляє елемент зі стека. Елементи видаляються в зворотному порядку до того, у якому вони записувалися в стек. Якщо стек порожній, це називається умовою недостатнього наповнення.
Peek або **Top**: повертає верхній елемент стека
isEmpty: повертає true, якщо стек порожній, інакше повертає false



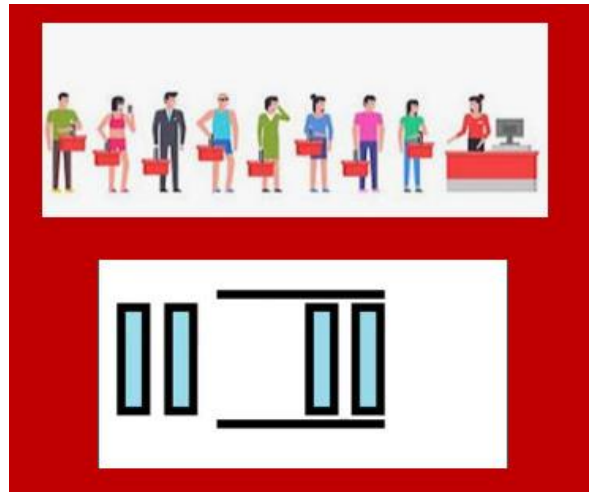
Часові складності операцій над стеком: push(), pop(), isEmpty() і peek() займають O(1) часу.

Застосування стека на ІТ-практиці:

- функції повторного скасування в багатьох програмах – редакторах, наприклад, Photoshop;
- функція переходу вперед і назад у веб-браузерах;
- в алгоритмах, таких як Ханойська вежа, обхід дерев, пошук шляху в лабіринті тощо;
- у графових алгоритмах;
- в управлінні пам’яттю будь-якого сучасного комп'ютера використовується стек, як основний спосіб керування роботою програм. Кожна програма, яка виконується в комп’ютерній системі, має власний розподіл пам’яті у стеці

Черга

Черга (англ. *queue*) в програмуванні – динамічна упорядкована структура даних, що працює за принципом «перший прийшов – перший пішов» (англ. *FIFO – first in, first out*)



Черга має *голову* (англ. *head*) та *хвіст* (англ. *tail*). Новий елемент черги додається у хвіст черги. З черги видаляють елемент, який знаходиться в її голові

Способи реалізації черги:

- за допомогою одновимірного масиву;
- за допомогою зв'язаного списку;
- за допомогою класу ООП

Основні операції з чергою

- додавання елемента x у кінець черги;
- видалення елемента з голови черги;
- виведення першого елемента черги;
- виведення елементів черги

Черги є критично важливими для управління ресурсами та подіями в асинхронних системах:

- Планування завдань CPU
- Обробка запитів на вебсерверах
- Черга друку
- Системи повідомлень
- …