Лекція 11. Типи даних. Структури даних. Абстрактний тип даних

Тип даних – визначає множину припустимих значень, формат їхнього збереження, розмір виділяємої пам'яті та набір операцій, які можна робити над даними.

Концепція типу даних з'явилася в мовах програмування високого рівня як природне відбиття того факту, що оброблювані програмою дані можуть мати різні множини припустимих значень, зберігатися в пам'яті комп'ютера різним чином, займати різні обсяги пам'яті й оброблятися за допомогою різних команд процесора.

Кожна мова програмування підтримує один або кілька убудованих типів даних (базових типів), крім того, розвинені мови програмування надають програмістові можливість описувати власні типи даних, комбінуючи або розширюючи існуючі.

Простий тип даних – тип даних, об'єкти (змінні або константи) якого не мають доступної програмістові внутрішньої структури. Як правило, до простого відносяться числовий, символьний, логічний і деякі інші типи.

Складний тип даних– тип даних, об'єкти (змінні або константи) якого мають внутрішню структуру, доступну програмістові. Складний тип складається з елементів, що відносяться до простих типів. До складних типів даних відносяться: масиви, множини, рядки, записи, файли, динамічні змінні, вказівники; лінійні списки (стеки, черги), нелінійні списки (двійкові дерева, несиметричні дерева, тексти, графи), процедурний тип, об’єкти.

Поліморфічний тип даних – представлення набору типів як єдиного типу. Є мови, що не пов'язують зміні, константы, формальні параметры і повертаємі значения функцій з певними типами, підтримуючи єдиний поліморфний тип даних. У чистому вигляді таких мов не зустрічається, але близькі приклади – MS Visual Basic, Delphi – тип variant, Пролог, Лісп – списки. У цих мовах змінна може приймати значення будь-якого типу, у параметри функції можна передавати значення будь-яких типів, і повернути функція також може значення будь-якого типу. Зіставлення типів значень змінних і параметрів із застосовуваними до них операціями здійснюється безпосередньо при виконанні цих операцій. Наприклад, вираз a+b, може трактуватися як додавання чисел, якщо a і b мають числові значення, як конкатенация рядків, якщо a і b мають строкові значення, і як неприпустима (помилкова) операція, якщо типи значень a і b несумісні. Такий порядок називають «динамічною типізацією» (відповідає поняттю полиморфізм в ООП, поліморфний тип у теории типів). Мови, що підтримують тільки динамічну типізацію, називають іноді «безтиповими». Цю назву не слід розуміти як ознака відсутності поняття типів у мові – типи даних все одно є.

Прості типи даних

Цілий тип – числа без коми, можуть бути зі знаком, тобто приймати як позитивні, так і негативні значення; і без знака, тобто приймати тільки позитивні значення.

Дійсний тип – числа з комою (тобто зберігаються знак і цифри цілої й дробової частин) або з плаваючою комою (тобто число приводиться до виду $m*b^e$, де m – мантиса, b – підстанова показової функції, e – показник ступеня (порядок, або експонента).

Символьний тип – зберігає один символ, у певному кодуванні.

Логічний тип – може приймати два можливих значення, що іноді називаються «*істиною»* і «*хибою»* (також «*так»* й «*ні»*). При трійковій логіці може мати й третє значення – *«не визначено»* (або «невідомо»).

Вказівник – зберігає адресу змінної, на яку посилається, діапазон значень складається з адрес комірок пам'яті ком'ютера, нулюва адреса, не являється реальною і говорить про те, що в даний момент вказівник порожній.

Складні типи

Массив – індексований набор елементів одного типу. Одномірний массив – вектор, двумірний масив – матриця.

Рядковий тип – довільна послідовність символів алфавіту. Кожна змінна такого типу може бути представлена фіксованою кількістю байтів або мати довільну довжину.

Перелічений тип –явно вказані (перелічені) усі можливі значення. Переваги в тому, що крім заданих значень змінні цього типу не зможуть брати більше ніяких значень. Крім того, значення можна задавати цілком осмислені - наприклад слова. Це спростить розуміння коду і написання програми.

Множина – кінцева сукупність елементів, що належать деякому базовому типу.

Запис – набір різних елементів (полів запису), збережений як єдине ціле. Можливий доступ до окремих полів запису.

Файловий тип – зберігає тільки однотипні значення, доступ до яких здійснюється тільки послідовно (файл з довільним доступом, включений у деякі системи программирования, фактично є неявным массивом).

Абстрактні типи даних

Абстрактний тип даних (АТД) – це тип даних, що надає для роботи з елементами цього типу певний набір функций, а також можливість створювати елементи цього типу за допомогою спеціальних функцій. Вся внутрішня структура такого типу захована від розроблювача программного обеспечения – у цьому й полягає суть абстракції. Абстрактний тип даних визначає набір незалежних від конкретної реалізації типу функцій для оперування його значеннями. Конкретні реалізації АТД називаються структурами данных.

В програмуванні абстрактні типи даних звичайно представляються у вигляді інтерфейсів, які приховують відповідні реалізації типів. Програмісти працюють із абстрактними типами даних винятково через їхні інтерфейси, оскільки реалізація може в майбутньому змінитися. Такий підхід відповідає принципу інкапсуляції в об'єктно-орієнтованному програмуванні. Сильною стороною цієї методики є саме приховання реалізації. Раз зовні опублікований тільки інтерфейс, то поки структура даних підтримує цей інтерфейс, всі програми, що працюють із заданою структурою абстрактним типом даних, будуть продовжувати працювати. Розроблювачі структур даних намагаються,

не міняючи зовнішнього інтерфейсу й семантики функцій, поступово допрацьовувати реалізації, поліпшуючи алгоритми по швидкості, надійності й використовуваній пам'яті.

Розходження між абстрактними типами даних і структурами даних, які реалізують абстрактні типи, можна пояснити на наступному прикладі. Абстрактний тип даних список може бути реалізований за допомогою массива або лінійного списку, з використанням різних технік динамічного виділення пам'яті. Однак кожна реалізація визначає той самий набір функцій, що повинен працювати однаково (за результатом, а не за швидкістю) для всіх реалізації.

Абстрактні типи даних дозволяють досягти модульності програмних продуктів і мати декілька альтернативних взаємозамінних реалізацій окремого модуля.

Приклади АТД:

Список – множина (постійна або тимчасова) пов’язаних об’єктів, упорядкованих деяким логічним способом. Для позначення початку списку використовується елемент, який називається головою списку.

Стек – різновид лінійного списку, структура даних, яка працює за принципом (дисципліною) «останнім прийшов — першим пішов» (LIFO, англ. last in, first out). Всі операції (наприклад, видалення елементу) в стеку можна проводити тільки з одним елементом, який знаходиться на верхівці стеку та був введений в стек останнім.

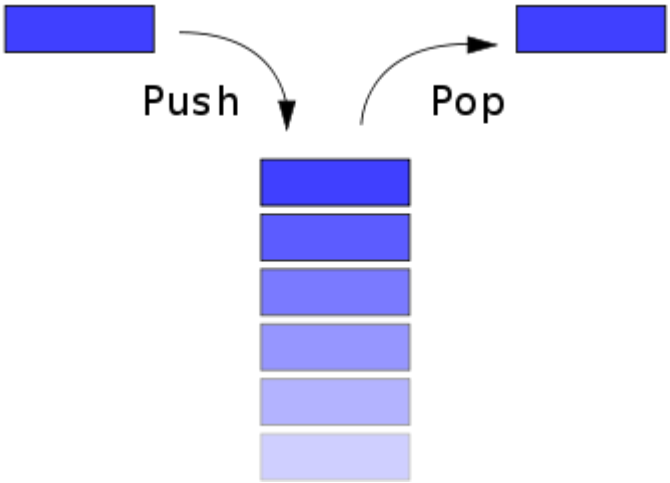

Рисунок 1 – Стек

Дек – різновид лінійного списку, в якому елементи можуть додаватись як на початок, так і в кінець.

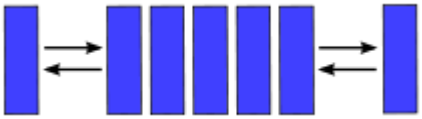

Рисунок 2 – Дек

Черга – динамічна структура даних, що працює за принципом "перший прийшов - перший пішов" (англ. FIFO — first in, first out). У черги є голова (англ. head) та хвіст (англ. tail). Елемент, що додається до черги, опиняється в її хвості. Елемент, що видаляється з черги, знаходиться в її голові.

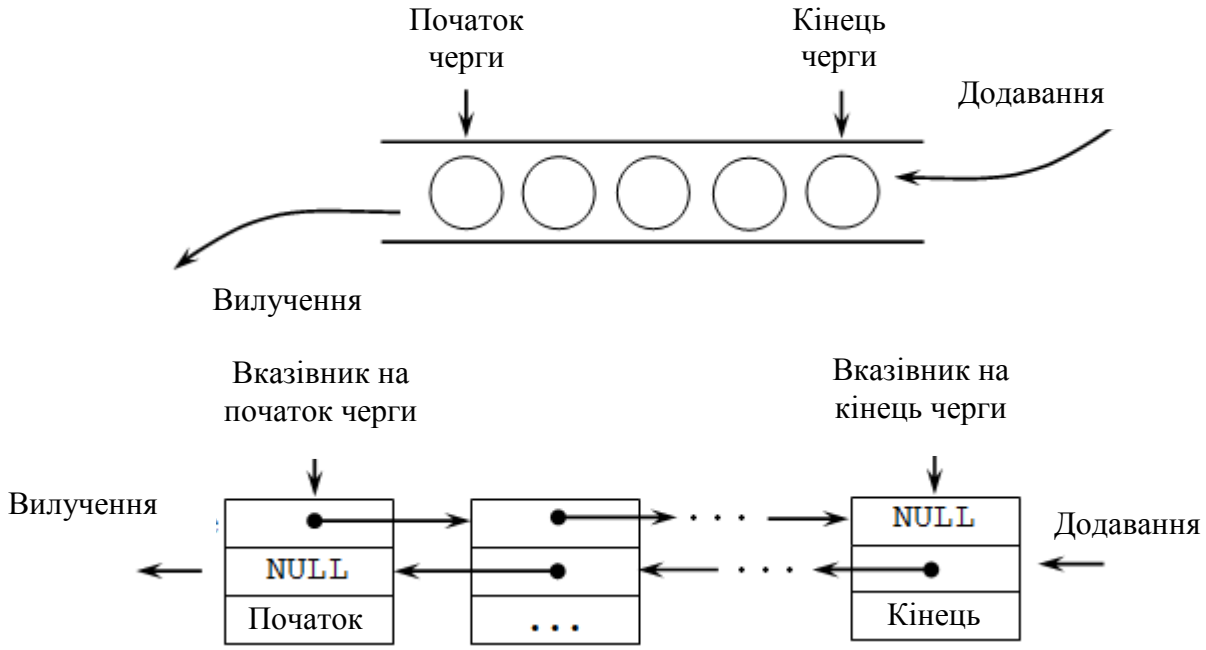

Рисунок 3 – Принцип роботи черги

Черга з пріоритетом – це структура даних, що призначена для обслуговування множини S, з кожним елементом якої пов'язано певне значення, що зветься ключем (англ. key). Можливі операції: вставка елементу x в множину S, повернення елементу множини *S* з найбільшим ключем, повернення елементу з найбільшим ключем, видаляючи його при цьому з множини *S,* зміна значення ключа для елемента *x*, шляхом заміни його ключем зі значенням *k.*

Асоціативний масив (також відомий як словник або карта) – абстрактний тип даних (інтерфейс до сховища даних), що дозволяє зберігати дані у вигляді набору пар ключ — значення та доступом до значень за їх ключем.

Граф – сукупність вузлів і ребер, що з'єднують ці вузли.

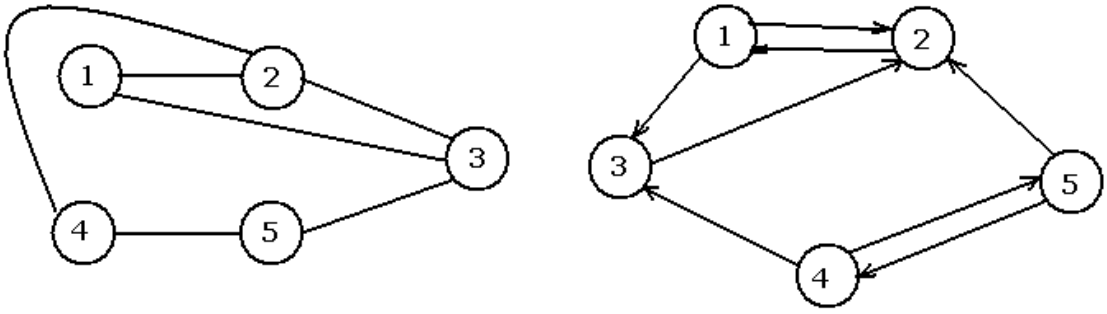

Рисунок 4 – Графи (неорієнтований зліва, орієнтований справа)

Дерево – зв’язаний граф, що не містить циклів, складається з кореневого вузла, вузлів та листків.

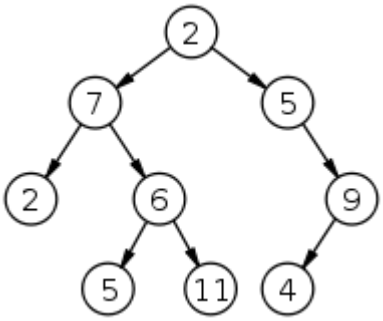

Рисунок 5 – Дерево

Купа (стіс або **піраміда) –** спеціалізована деревовидна структура даних, в якій існують певні властивості впорядкованості: якщо B – вузол-нащадок A, тоді ключ(A) ≥ ключ(B). З цього випливає, що елемент з найбільшим ключем завжди є кореневим вузлом.

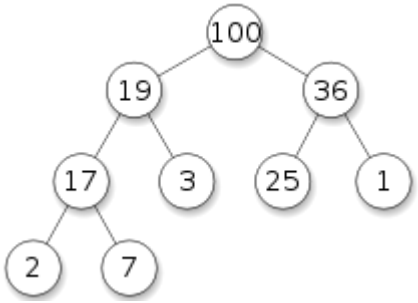

Рисунок 6 – Стіс