Лекція 2. Типи та структури даних

Структури даних

При використанні комп'ютера для збереження й обробки даних необхідно добре знати тип і структуру даних, і визначити спосіб їхнього представлення.

Як правило, структурування даних вимагає точного визначення типу кожного елемента, що входить у структуру.

Математичні принципи концепції типу, що покладені в основу мов програмування високого рівня такі [1]:

1. Будь-який тип даних визначає множину значень, до яких може відноситися деяка константа, що може приймати будь-яка змінна або вираз і яке може формуватися операцією чи функцією.

2. Тип будь-якої величини, що позначається константою, змінною чи виразом, може бути виведений з її виду або з її опису (без необхідності будь-яких обчислень).

3. Кожна операція чи функція вимагає аргументів певного типу і дасть результат також фіксованого типу.

В обчислювальній техніці існує ряд так званих **найпростіших стандартних типів** даних: цілий, логічний, символьний тощо.

Крім того, існує можливість визначення нових типів шляхом *переліку* множини їхніх значень. Тому ці типи ще звуть **перелічуванними**. Вони також є найпростішими, проте, звісно, не стандартними.

Структури даних інакше називають *складеними типами*, тому що вони являють собою сукупності компонентів, що відносяться до визначених раніше типів.

Якщо всі компоненти відносяться до одного типу, то такий тип називають *базовим*.

Дані, крім типу, що визначає їхню структуру, мають специфічні властивості.

При розгляді структур даних ми будемо оперувати таким поняттям як **множина**, яка є сукупністю даних певного типу, що володіють деякою властивістю. При цьому повинні бути встановлені *область визначення* даних і *правила визначення належності* даних до множини [20].

Імовірно, що найбільше широко відома структура даних — одновимірний чи багатовимірний **масив**. Ця структура являє собою відображення деякої кінцевої множини даних на множину даних іншого типу. Тут областю визначення є множина даних типу *індекс*, а областю значень — *множина елементів масиву*. Масив складається з елементів одного базового типу, тому структура масиву *однорідна*.

Найбільш загальний метод одержання складених типів полягає в об'єднанні елементів різних типів. Причому самі елементи можуть бути складеними.

Нехай у нас є такі елементи:

$$a_i \in A_1, a_j \in A_2.$$

Якщо розглянути множину виду:

$$\left\{ \left(a_i, a_j \right) \middle| a_i \in A_1, a_j \in A_2 \right\},$$

то ми дістанемо складений тип, що зветься **прямим** *(декартовим)* **добутком**. Кожен елемент такої структури зветься *кортежем*.

Назва „добуток" пояснюється тим, що множина значень, визначена таким типом, складається з усіляких комбінацій наборів a_i, a_j. Відповідно, число таких комбінацій дорівнює

добутку числа елементів у кожній зі складових множин. Приклад:

$$A_1 = \{1,3,8,9\}, A_2 = \{2,3\}$$

$$A_1 \times A_2 = \{(1,2),(3,2),(8,2),(9,2),(1,3),(3,3),(8,3),(9,3)\}.$$

При обробці даних такі складені типи, що описують людей або об'єкти, є найбільш застосовними у файлах і базах даних. Тому до даних такої природи стали використовувати назву **запис** і розглядати його як ще один (*третій*) тип даних.

Четвертим типом структурованих даних є **послідовності**. Типові приклади послідовного типу даних — *файл* і *стік*. Всі елементи послідовності мають той самий тип. Структура послідовності дуже схожа на структуру масиву. Істотна відмінність полягає в тому, що в масиві число елементів фіксується в його опису, а число елементів послідовності (довжина) скінченне, але не фіксовано.

Послідовним файлом називається послідовність, для якої визначені такі операції:

1. Формування порожньої послідовності.
2. Вибірка початкового елемента послідовності.
3. Додавання елемента в кінець послідовності.

Стік — послідовність, над якою можливе виконання таких операцій:

1. Створення порожньої послідовності.
2. Одержання першого елемента послідовності.
3. Додавання елемента в початок послідовності.

Спискові структури даних

Лінійні списки

Припустимо, що дані записані не в послідовні адреси пам'яті (як у випадку масивів), а в довільні місця. У цьому випадку найбільш простий спосіб об'єднати чи зв'язати деяку множину елементів — це витягнути їх у лінію [20]. І для того, щоб задати „зв'язок" між даними, необхідно кожному елементу поставити у відповідність посилання або *покажчик* на наступний елемент структури (рис. 1), *починаючи з порожнього*.

Формування списку може виконуватися 2-ма способами:

1. Додаванням елементів у кінець списку (рис. 1).

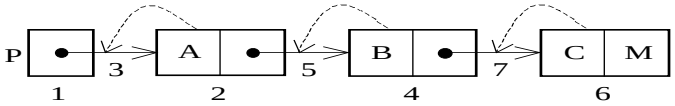


Рисунок 1 – Формування списку „з кінця"

2. Додаванням елементів у початок списку (рис. 2).

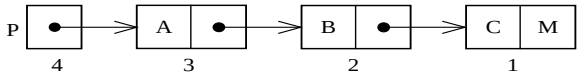


Рисунок 2 – Формування списку „з початку"

Додавання і виключення елементів може виконуватися просто зміною покажчика.

Двозв'язний список виглядає так (рис. 3).

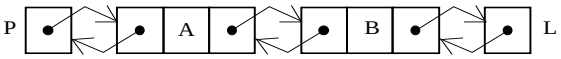


Рисунок 3 – Двозв'язний список

Таке представлення зручне з погляду обробки структур як з початку, так і з кінця списку. Недолік: додатковий покажчик. Якщо необхідно виконати обробку, як у прямому, так і в зворотному напрямку, то адреси повернення зручно зберігати в стеку: так організовані команди виклику/повернення з процедури.

Для забезпечення багаторазової обробки структури може використовуватися її представлення у вигляді *циклічного списку* (рис. 4).

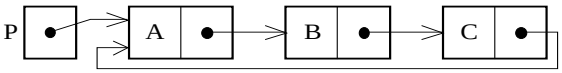


Рисунок 4 – Циклічний список

Деревоподібні структури

Припустимо, що в списковій структурі кожен елемент може містити або мітку кінця списку, або покажчик на не менше ніж 1 елемент. Дістанемо так званий *рекурсивний* тип даних (рис. 5). Подібні структури називаються **деревами** (вершини — вузли, зв'язки — гілки).

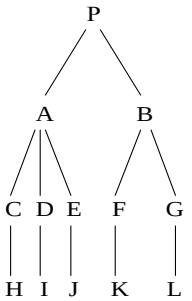


Рисунок 5. Дерево

До речі, лінійний список є *виродженим* деревом. У дереві можна виділити деякі рівні та вузли, що відносяться до них. Вузол *y*, що знаходиться на *i+1* рівні, називають *нащадком* вузла *x* *i-го* рівня, *x* — *предок y*. Вузол, що не має предків, називають *коренем*. Вузол, що не має нащадків — *термінальним* вузлом чи *листом*. Нетермінальну вершину називають *внутрішньою*.

Число безпосередніх нащадків вузла (фактично, число гілок, що виходять з нього) називають *ступенем вершини* (вузла). Максимальний ступінь, що мають усі вершини дерева — *ступінь дерева*.

Найбільш важливими в обчислювальній техніці є дерева ступеня 2, так звані *двійкові* (*бінарні*) дерева.

Дерева ступеня > 2 називають багатогілковими. Зберігати їх незручно через велике число посилань. Тому їх перетворюють у бінарні дерева.

Графи (графові структури)

Багато інформаційних задач зводяться до розгляду об'єктів, істотні властивості яких описуються зв'язками між ними. Наприклад, при розгляді електричних ланцюгів може цікавити наявність з'єднань між тими або іншими елементами, на карті авіаліній необхідно знати наявність

зв’язку між містами. Інтерес можуть становити різні зв’язки і відносини між людьми, подіями, станами й ін. У подібних випадках зручно об’єкти представляти у вигляді вершин, а зв’язки між ними - лініями, називаними ребрами.

Множина вершин X, зв’язки між якими визначені множиною ребер A, зветься **графом** (рис. 6):

$$\left\{ a_{ij} \middle| a_{ij} \cong \left(x_i, x_j \right), a_{ij} \in A, x_i \in X, x_j \in X \right\}$$

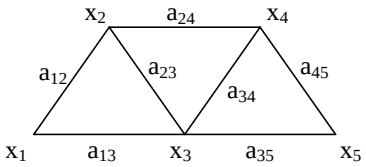


Рисунок 6 – Граф

Перша робота з графів була розглянута Ейлером для розв’язання задачі з топології. У середині минулого століття Кірхгоф застосував графи для аналізу електричних кіл.

До речі, розглянутий нами тип даних дерево — окремий випадок графу.

Часто зв’язки між об’єктами характеризуються певною орієнтацією. Наприклад, напрямок струму в електричних з’єднаннях, перехід деякої системи з одного стану в інший.

Для зазначення напрямку зв’язку ребро відзначається стрілкою (рис. 7). Орієнтоване таким чином ребро називається *дугою*, а граф з орієнтованими ребрами — *орієнтованим* графом чи *орграфом*.

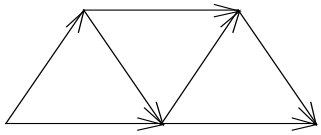


Рисунок 7 – Орграф

Якщо вузли з’єднуються більш ніж двома дугами, то ці дуги називаються *рівнобіжними*. Якщо вони мають один напрямок — *строго рівнобіжними*, якщо різний — *нестрого рівнобіжними*. Дві нестрого рівнобіжні дуги в графі можна замінити ребром, дістави тоді *змішаний* граф. Змінивши напрямки дуг орграфу на протилежні, дістанемо орграф, *зворотний* початковому.

Подальший розгляд графів складається в приписуванні ребрам і дугам деяких кількісних значень або характерних властивостей, називаних *вагами*. Вага може означати довжину шляху, пропускну здатність ліній зв’язку, величину струму в електричному колі тощо. Граф у цьому випадку називається *зваженим*. Особливе значення для моделювання фізичних систем мають зважені орграфи, названі графами потоків, сигнальними графами чи *сітками*.

Елементи і параметри графів

Кожне ребро, відповідно до визначення, з’єднує пару вершин $\left(x_i, x_j \right)$, іменованих *граничними* вузлами. Для дуги розрізняють *початковий* вузол, з якого дуга виходить, і *кінцевий,* в який вона входить.

Якщо граничні (чи початковий і кінцевий) вузли збігаються, ребро чи дуга називається *петлею*.

Поняття ступеня вузла аналогічно тому, що ми розглядали в деревах. В орграфі розрізняють *позитивний* (щодо вихідних з вузла дуг) і *негативний* (щодо вхідних) *ступені*. Граф,

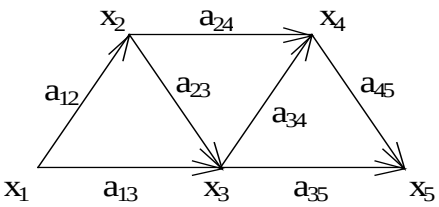
ступені усіх вузлів якого однакові і дорівнюють r, називається *однорідним* чи *регулярним* графом *r*-го ступеня.

Маршрут з вершини x_i до x_j дорівнює сумі ваг послідовних гілок (ребер), що з'єднують ці вузли. Замкнутий маршрут приводить у ту ж вершину, з якої почався, і називається *циклом*. Для орграфу маршрут визначається так, що початковий вузол наступної гілки повинен збігатися з кінцевим вузлом попередньої. Маршрут, що не містить повторюваних дуг, зветься *шляхом*.

Збереження топології графу

1. Матриця суміжності

Два вузли x_i, x_j називають *суміжними*, якщо вони є граничними вузлами однієї гілки. Рядки і стовпці матриці суміжності відповідають вузлам графу, а її *ij-й* елемент дорівнює числу ребер, що зв'язують вузли x_i і x_j чи дуг, що виходять з x_i у x_j. Для сітки значення елемента може відповідати вазі гілки. Приклад (рис. 8).



	x1	x2	x3	x4	x5
x1		1	1		
x2			1	1	
x3				1	1
x4					1
x5					

Рисунок 8 – Орієнтований граф і його матриця суміжності

2. Матриця інцидентності

Якщо вузол x_i — *граничний* для гілки a_{ij}, то вони *інцидентні*. Для орграфу розрізняють позитивну (для початкових вузлів) і негативну (для кінцевих) інцидентність.

Рядки матриці інцидентності відповідають вузлам, а стовпці — ребрам. *mn*-й елемент дорівнює *1*, якщо *m*-а вершина і *n*-е ребро інцидентні. Приклад (рис. 9).

	a12	a13	a23	a24	a34	a35	a45
x1	1	1					
x2	1		1	1			
x3		1	1		1	1	
x4				1	1		1
x5						1	1

Рисунок 9 – Матриця інцидентності графу з рис. 8

Для орграфу елементи дорівнюють *1* та *–1* для початкових і кінцевих вузлів відповідно.

3. За допомогою множини списків

І нарешті, існує ще один варіант представлення даних: *таблиці*. Зокрема, більшість сучасних баз даних можуть бути представлені у вигляді множини таблиць, що дозволяє досить легко вводити, шукати і виводити дані.