**Лекція 6. Концепція типу даних, організація даних та структури даних**

1. Концепція типу даних
2. Класифікація мов програмування за способом визначення типів даних
3. Практичне застосування тіпів у мовах програмування
4. Основні принципи концепції типу в мові програмування
5. Поняття базових скалярних типівмови
6. Стандартні прості типи даних у мові С ++
7. Одиниці фізичної структури даних.
8. Типи структур даних
9. Основні моделі даних
10. Лінійні структури даних.
11. Нелінійні структури даних.
12. Складені типи даних.

**Концепція типу даних**

Типи даних розрізняються, починаючи з нижніх рівнів системи. В усіх мовах програмування розрізняються типи даних. Пояснюється це тим, що для чисел цих типів відводяться різні обсяги пам'яті, використовуються різні регістри мікропроцесора, а для операцій з ними застосовуються різні команди і різні ядра мікропроцесора.

Концепція типу даних з'явилася в мовах програмування високого рівня як відображення того, що оброблювані дані можуть мати:

* + 1. Різні безлічі допустимих значень,
    2. Зберігатися в пам'яті комп'ютера різним чином,
    3. Займати різні обсяги пам'яті
    4. Оброблятися за допомогою різних команд процесора.

Чи існують мови без типів? Теоретично не може існувати мов, в яких відсутні типи даних. Це випливає з того, що всі мови засновані на машині Тьюринга або на лямбда-численні. Практично це не можливо.

**Класифікація мов програмування за способом визначення типів даних**

Можна класифікувати мови програмування за способом визначення типів даних:

1. *Мови з поліморфним типом даних*. Вони не пов'язують змінні, константи, формальні параметри, які повертаються значення функцій з певними типами, підтримуючи єдиний поліморфний тип даних.

У чистому вигляді таких мов не зустрічається, але близькі приклади - MS Visual Basic - тип variant, Пролог. У цих мовах змінна може приймати значення будь-якого типу, в параметри функції можна передавати значення будь-яких типів, і повернути функція також може значення будь-якого типу.

Зіставлення типів значень змінних і параметрів з застосовуваними до них операціями здійснюється безпосередньо при виконанні цих операцій.

Наприклад. вираз a + b, може означати

* + - Складання чисел, якщо a і b мають числові значення.
    - З'єднання рядків в одну, якщо a і b мають строкові значення.
    - Неприпустима (помилкова) операція, якщо типи значень a і b несумісні.

Такий порядок називають динамічною типізацією. Мови, що підтримують тільки динамічну типізацію, називають іноді безтипових. однак, типи даних в них все одно є.

2. *Мови з неявним визначенням типів*. BASIC є прикладом мови без типів, але це строго типізований мову в ньому розрізняються:

* + - Строкові типи (додається символ $),
    - Числові типи (нічого не додається).

3. *Мови з типом, визначеним користувачем*. Відомі мови, в яких типи даних визначаються автоматично, а не задаються користувачем.

Кожній змінній, параметру, функції приписується певний тип даних. В цьому випадку для будь-якого виразу можливість його виконання та тип отриманого значення можуть бути визначені без виконання програми. Такий підхід називають статичною типізацією. При цьому правила поводження зі змінними, виразами і параметрами різних типів можуть бути як дуже суворими (С ++), так і досить ліберальними (Сі). Наприклад. в класичноою мовою Сі практично усі типи даних сумісні - їх можна застосовувати спільно в будь-яких виразах, привласнювати значення змінної одного типу змінної іншого майже без обмежень. Компілятор генерує код, що забезпечує перетворення типів, а логічна коректність такого перетворення залишається на сумлінні програміста.

Подібні мови називають мовами зі слабкою типізацією. Протилежністю є мови з сильною типізацією. Ада, де кожна операція вимагає операндів строго заданих типів. Ніякі автоматичні перетворення типів не підтримуються. На практиці мови програмування підтримують кілька моделей визначення типів одночасно.

**Практичне застосування типів у мовах програмування**.

Тіпи мов програмування не завжди строго відповідають подібним математичним типам. Наприклад, тип ціле число більшості мов програмування не відповідає прийнятому в математиці. Оскільки в математиці даний тип не має обмежень ні зверху, ні знизу, а в мовах програмування дані обмеження є.

У мовах і системах є безліч цілих типів, що відрізняються визначеним обсягом займаної пам'яті.

У мовах програмування класифікують змінні відповідно до деяких характеристиками і виділяють:

1. Що представляють собою окремі значення, безлічі значень або безлічі множин.
2. Будь-яка константа, змінна, вираз чи функція відноситься до деякого типу.

**Основні принципи концепції типу в мові програмування**

*Основні принципи концепції типу наступні*:

1. Будь-який тип даних визначає безліч значень, які може приймати константа, змінна або вираз і які можуть формуватися функцією або операцією

2. Тип будь-якої константи, змінної або виразу може бути виведений з їх вигляду або по їх опису. Для цього немає необхідності проводити будь-які обчислення.

3. Кожна операція або функція вимагає аргументів для її визначення і дає результат певного типу. Якщо операція використовує аргументи декількох типів, наприклад «+» використовується як для складання дійсних чисел, так і для складання цілих чисел, то тип результату регламентується цілком певними правилами.

4. Найважливіші основні операції - порівняння і присвоювання. Виконується перевірка відносини рівності (і порядку в разі упорядкованних типів), а потім визначається "установці рівності".

Основна мета будь-якої програми полягає в обробці даних. Дані різного типу зберігаються і обробляються по-різному. У будь-якій алгоритмічній мові кожна константа, змінна, результат обчислення виразу або функції повинні мати певний тип.

*Тип даних визначає*:

* + - Внутрішнє представлення даних в пам'яті комп'ютера;
    - Безліч значень, які можуть приймати величини цього типу;
    - Операції і функції, які можна застосовувати до величинам цього типу.

Обов'язковий опис типу дозволяє компілятору проводити перевірку допустимості різних конструкцій програми. Від вибору типу величини залежить послідовність машинних команд, побудована компілятором.

**Поняття базових скалярних типів мови**

Всі типи мови можна розділити на: прості (скалярні), можуть бути стандартними, і визначені певним програмістом.

**Тип даних** – це множина значень разом із множиною застосовних до них операцій. Множина значень називається **носієм** типу, множина операцій – **сигнатурою**. У програмуванні використовуються типи чисел, символьних і логічних значень. Ці значення розглядаються як цілісні елементи, що не мають окремих складових частин, тому називаються **скалярними** на відміну від **структурних**, складених з окремих компонентів. Типи скалярних даних називаються **скалярними**. Кожна мова програмування забезпечує цілу сім'ю скалярних типів. Вони мають певні, означені наперед (стандартні) імена й називаються **базовими скалярними типами** мови.

**Стандартні прості типи даних у мові С ++**

У мові С ++ визначено шість стандартних простих типів даних для представлення цілих, дійсних, символьних і логічних величин. На основі цих типів, а також масивів і покажчиків (покажчик не є самостійним типом, він завжди пов'язаний з будь-яким іншим конкретним типом), можна вводити опис власних простих або структурованих типів.

Для опису стандартних типів визначені наступні ключові слова:

**іnt** (цілий); Розмір типу int стандартом ANSI[[1]](#footnote-1) не визначається. Він залежить від реалізації. Для 16-розрядного процесора під величини цього типу відводиться 2 байти, для 32-розрядної - 4 байта.

**char** (символьний); Символьний тип (char). Під величину символьного типу відводиться кількість байт, достатню для розміщення будь-якого символу з набору символів для даного комп'ютера. Як правило, це 1 байт. Тип char, як і інші цілі типи, може бути зі знаком або без знаку.

У величинах зі знаком можна зберігати значення в діапазоні від -128 до 127. При використанні специфікатор unsigned значенія можуть перебувати в межах від 0 до 255. Величини типу char застосовуються також для зберігання цілих чисел, що не перевищують межі зазначених діапазонів.

Окрім того є тип **wchar\_t** (дабл-ю-чар-підкреслення-ти) - розширений символьний; Тип wchar\_t призначений для роботи з набором символів, для кодування яких недостатньо 1 байта, наприклад, Unicode. Розмір цього типу залежить від реалізації; як правило, він відповідає типу short.

**bool** (логічний); Логічний тип (bool). Величини логічного типу можуть приймати тільки значення true і false. Внутрішня форма подання значення false - 0 (нуль). Будь-яке інше значення інтерпретується як true. При перетворенні до цілого типу true має значення 1.

**float** (дійсний); типу даних для зберігання речових значень: Внутрішнє подання дійсного числа складається з мантиси і порядку. Довжина мантиси визначає точність числа, а довжина близько - його діапазон.

**double** дійсний з подвійною точністю. Константи з плаваючою точкою мають за замовчуванням тип **double**.

**void** порожній, не має значення.

Існує чотири специфікатор типу, уточнюючих внутрішнє уявлення і діапазон значень стандартних типів:

**short** (короткий); Специфікатор short перед ім'ям типу вказує компілятору, що під число потрібно відвести 2 байта.

**long** (довгий); Специфікатор long означає, що ціла величина буде займати 4 байти.

**signed** (зі знаком); При використанні специфікатор signed старший біт числа інтерпретується як знаковий (0 - позитивне число, 1 - негативне).

**unsigned** (без знака). Константам, яке трапляється в програмі, приписується тип відповідно до їх виду.

**Одиниці фізичної структури даних**

З точки зору подання інформації на певних носіях (фізичне структурування) відповідні одиниці визначаються залежно від носія інформації та способу її фіксації. Це пов'язано з розміщенням масивів даних у пам'яті ПК. Як правило, виділяють такі одиниці фізичної структури даних (від найнижчої до найвищої): символ, поле, агрегат даних, запис, файл, база даних.

Поняття символу наведено вище.

**Поле** - множина символів, яка створює мінімальний семантичний елемент масиву.

**Агрегат даних** - це пойменована сукупність двох і більше елементів нижчого рівня, яка має окремий зміст . До агрегату даних можуть належати як елементи, так і інші агрегати даних. Прикладом агрегату даних можуть бути групи елементів, які утворюють "Адресу" або "Дату народження".

**Запис** - пойменована сукупність полів, об'єднаних за змістовним принципом.

Агрегати даних і записи реалізуються на практиці шляхом організації списків, черг, стеків, таблиць.

**Файл** - іменована сукупність записів про об'єкти одного типу. Як правило, записи, що входять у файл, мають однакову структуру. Прикладом файлу може бути сукупність записів про колір пікселів, що складають зображення.

**База даних** - це іменована сукупність взаємозв'язаних даних, що відображає стан об'єктів і їх відношення в даній предметній області. Наприклад, база "Студент" містить інформацію про вік, стать, домашню адресу, успішність та інші дані про студента.

Організація даних у базі характеризується певною структурою, тобто формою ї способом їх впорядкування.

**Типи структур даних**

За характером взаємозв'язків елементів усі структури даних можна поділити на лінійні та нелінійні.

До лінійних структур належать послідовні структури, в яких елементи розміщуються в тому порядку, який необхідний під час їх обробки, наприклад, список, елементами якого є записи (рядкова структура).

До нелінійних структур даних належать складні списки, дерева, мережі, табличні та гібридні структури:

* складні списки містять вкладені елементи (підсписки) меншого обсягу;
* деревоподібні структури - елементи розміщуються на різних рівнях і сполучаються за допомогою адрес;
* сіткові структури являють собою розширення дерева за рахунок нових адрес зв'язку;
* табличні структури даних призначені для зберігання інформації про ключові ознаки даної інформаційної сукупності;
* гібридні структури даних містять фрагменти різних структур.

Зі структурою пов'язана **модель даних** - фіксована система понять і правил для представлення структури даних.

**Основні моделі даних**

Існує декілька моделей даних, які становлять основу інформаційних систем. Найчастіше використовуються три: ієрархічна, мережева (сіткова) та реляційна. Останнім часом з'явився четвертий тип: об'єктно-орієнтовані моделі даних, які використовують поєднання традиційної та об'єктної моделей. В моделі сіткових баз використовують представлення даних у вигляді довільного графу. В ієрархічних - дані представляються у вигляді ієрархічної (деревоподібної) структури. Найпоширеніші моделі баз даних - реляційні. В них дані представляються у вигляді таблиць. Таке подання даних, що складається зі стовпців, які розташовуються у певному порядку зліва направо, називається відношенням. Тому ці таблиці називаються реляційними (від англ. relation - відношення).

Під час проектування інформаційних систем та розробленні алгоритмів оброблення даних в них необхідно враховувати такі властивості економічної інформації:

* вхідна інформація, в основному, фіксується в первинних документах, які не завжди придатні для автоматичного введення в комп'ютерну пам'ять;
* ті самі вхідні дані використовуються багаторазово для здобуття показників у різних економічних розрізах для всіх служб і видів господарської діяльності;
* основна частина економічної інформації підлягає періодичному, регулярному оновленню;
* здобута вихідна інформація часто використовується як вхідна при подальших розрахунках;
* економічна інформація характеризується тривалістю збереження.

Поняття структури даних можна визначити як особливу систему організації взаємозв’язаних елементів даних. Структура самих елементів залежить від конкретної задачі. Вони можуть бути як дуже простими (наприклад, цілі числа або строки символів) так і дуже складними (наприклад, для зображення матриці часто використовують одномірний масив, кожний елемент якого в свою чергу також є одномірним масивом). Розглянемо деякі структури даних, що найчастіше зустрічаються.

**Лінійні структури даних**

Серед простих структур даних можна виділити два найважливіших – одномірний масив і зв’язаний список.

**Одномірний масив** – це послідовність із n однотипних елементів, що розташовані підрід в оперативній пам’яті комп’ютера, доступ до яких виконується по значенню індексу масиву:

|  |  |  |  |
| --- | --- | --- | --- |
| Елемент [0] | Елемент [1] | … | Елемент [n-1] |

В більшості випадків індекс масиву, що складається із n елементів, є цілим числом, що має значення від 0 до n-1 або від 1 до n В деяких мовах програмування допускається, щоб індекс набував любих цілочисельних значень, включаючи від’ємні. Головне, щоб він не виходив за межі раніше встановленого діапазону, що встановлює верхню і нижню межу зміни індексу.

Іноді в мовах програмування для доступу до елементів масиву можуть використовуватись нечислові індекси. Такі масиви називаються асоціативними. Наприклад, можна створити масив із 12 елементів, що відповідає місяцям року, доступ до яких здійснюється по назві місяців.

Час звернення до любого елементу масиву однаковий і не залежить від його положення в масиві. Ця особливість відрізняє масиви від зв’язаних списків, мова про які піде пізніше. Проте кожен елемент масиву займає однакову кількість чарунок пам’яті в комп’ютері.

На основі одномірних масивів створюються різні структури даних, серед яких особливу увагу заслуговують рядки.

*Рядки* – послідовність символів раніше обумовленого алфавіту, що закінчується особливим символом, що служить ознакою кінця рядка. Рядки, елементами яких є лише 0 та 1, називаються двійковими (бінарними) або бітовими.

Рядки є основним елементом, що використовується при обробці текстових даних, визначення синтаксису мови програмування, компілювання створених на ній програм, а також при вивченні абстрактних обчислювальних моделей.

Операції, що виконуються над масивами, залежать від його типу – операції, що виконуються над рядками, відрізняються від операцій, що виконуються над масивом чисел (визначення довжини рядків, об’єднання двох рядків для створення однієї, порівняння двох рядків).

**Зв’язаний список** – послідовність (в т.ч. і порожня) кількох елементів даних, що називаються вузлами. В кожному вузлі зберігається інформація двох видів: самі дані та одна або більше посилань на інші вузли списку, що називаються вказівниками. Якщо поточний елемент є останнім, то в нього поміщується нульовий вказівник зі спеціальним значенням NULL – це вказує на те, що вказівник ні на що не вказує.

В *однонапрямленому* зв’язаному спискові кожний вузол вміщує лише один вказівник, в якому міститься посилання на наступний елемент списку, або NULL, якщо поточний елемент є останнім.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Елемент [0] |  |  | Елемент [1] |  | … | Елемент [n-1] | Null |

Для досягнення заданого елемента зв’язаного списку необхідно обрати перший вузол списку, потім пересуватися по ланцюгу елементів до необхідного вузла. Тому час доступу до конкретного елемента списку залежить від місця його положення в списку і може дуже суттєво відрізнятися від того, що затрачується в масиві для подібної операції. Це є недоліком зв’язаного списку. Перевага зв’язаного списку полягає в тому, що для списку немає необхідності резервувати оперативну пам’ять комп’ютера. Крім того, вставка та видалення елементів списку не завдають труднощів і виконуються достатньо швидко шляхом зміни вказівників.

Існує модифікація структури однонапрямленого зв’язаного - двонапрямлений зв’язаний список. Відміна між ними в тому, що кожен вузол двонапрямленого списку крім першого та останнього включають в себе вказівники на попередній та наступний вузли:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Null | Елемент [0] |  |  |  | Елемент [1] |  | … |  | Елемент [n-1] | Null |

Масив та зв'язаний список частіше всього використовують для зображення ще однієї абстрактної структури даних – лінійний список або просто список. Список являє собою скінчену послідовність елементів даних, що організовані в заданому лінійному порядку. До цієї структури даних можуть бути застосовані наступні операції: пошук, видалення та додавання елемента.

Серед всієї множини можна виділити два типи списків: стеки та черги.

***Стек*** – це такий список, в якому можна виділити тільки останній елемент, а новий елемент можна додати тільки в кінець. Цей кінець називають вершиною стеку, оскільки стеки зазвичай зображують на малюнках у вигляді вертикального прямокутника. Принцип роботи стека – LIFO (last-in-first-out), оскільки першим видаляється елемент, котрий був останнім у стеці. Цей стек нагадує купку тарілок, оскільки ми можемо взяти з неї тільки верхню тарілку, а нову покласти наверх купки. Стеки широко використовуються на практиці – без них не обійтися при реалізації рекурсивних алгоритмів.

***Черга*** – це такий список, елементи якого видаляються з однієї сторони структури (вона називається головою), а додаються з іншої (вона називається хвостом). Перша операція називається видаленням елемента з черги, а друга – постановкою в чергу. Принцип роботи черги – FIFO (first-in-first-out). Аналогія – черга в магазині. Область використання – алгоритми розв’язку задач теорії графів.

В багатьох випадках необхідно вибрати серед множини елементів, що динамічно змінюється, елемент з найвищим пріоритетом. Для розв’язку таких задач використовують структуру даних, що називається чергою з пріоритетами або просто пріоритетною чергою. До основних операцій над чергою з пріоритетами відносяться: пошук найбільшого елемента, видобування найбільшого елемента та додавання нового елемента. Дві останні операції повинні бути реалізовані так, щоб в результаті їх виконання утворилась нова черга с пріоритетами або перевпорядкована стара. Найпростіше реалізувати таку структуру на основі масиву або впорядкованого масиву, хоча так не досягається максимальна ефективність. Кращою є реалізація на основі структури даних, що називається пірамідою.

**Перелік спискових типів даних, до яких належать: лінійні списки, деревоподібні структури та графи**

**Складені типи даних.**

**Складеним типом даних** назвемо тип даних, що складається із скінченної та наперед заданої множини елементів певного типу, які не обов'язково є атомарними.

Перерахуємо складені типи даних та дамо їм коротку характеристику.

**Множина** – скінчена сукупність елементів, в якої R = Ø.

**Послідовність** — абстрактна структура, у якої множина R складається з одного відношення лінійного порядку (тобто для кожного елемента, крім першого і останнього, є попередній і наступний елементи).

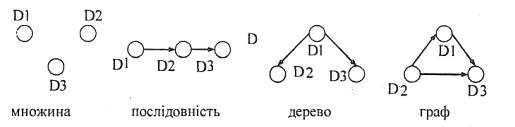
**Матриця** – структура, в якої множина R складається із двох відношень лінійного порядку.

**Дерево** – множина R складається з одного відношення ієрархічного порядку.

**Граф** – множина R складається з одного відношення бінарного порядку.

**Гіперграф** – множина R складається із двох і більше відношень різного порядку.

Приклади СД подано на рис. 5.



**Рисунок 5**. Приклади подання структур даних

***Для самостійного вивчення*:** Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

***Література***

1. Ковалюк Т. В. Алгоритмізація та програмування: Підручник. — Львів: «Магнолія 2006», 2013. — 400 с., ил.
2. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..
3. Куприянова Л.М. Программирование, алгоритмические языки и вычислительная математика: Учеб. пособие. — М.: Финансы и статистика, 1985. —223 с.

**Контрольні питання**

1. Що таке тип даних?
2. В чому полягає концепція типу даних?
3. За якими типами можна поділити структури даних?
4. Визначте одиниці фізичної структури даних.
5. Визначте основні моделі даних.
6. Наведіть приклади лінійних та нелінійних структур даних.
7. Чим відрізняється стек від черги?
8. Назвіть складені типи даних.

1. **Америка́нський інститут націона́льних станда́ртів** (англ. ***A****merican* ***N****ational* ***S****tandards* ***I****nstitute*, ANSI) — об'єднання американських промислових і ділових груп, що розробляє торгові і комунікаційні стандарти [↑](#footnote-ref-1)