§ 64. Пошук елемента в таблиці

Вивчивши цей параграф, ми:

познайомимося з типовими задачами опрацювання таблиць;

дізнаємося, як здійснюється пошук елемента в невпорядкованій лінійній таблиці;

довідаємося про алгоритм бінарного пошуку і його особливості;

з'ясуємо, як порядок зберігання даних у таблицях впливає на можливість швидкого доступу до потрібної інформації.

==== 64.1. Типові задачі при роботі з таблицями==========

У таблицях розміщують і зберігають різні дані, які потрібні людині. Наприклад, дані довідкового характеру (розклад руху транспорту, адреси й номери телефонів мешканців міста, наявність лікарських препаратів в аптеках і т. ін.); дані, які характеризують економічні показники роботи підприємства (випуск продукції, рух матеріальних цінностей, кадровий склад тощо); дані спостережень (астрономічних, екологічних, медичних) тощо. Таблиці можуть об'єднувати різну кількість елементів — і невелику, якщо йдеться, скажімо, про список телефонів наших друзів, і величезну, якщо це телефонна книга мільйонного міста.

Задачі опрацювання таблиць надзвичайно різноманітні. За даними таблиці можна досліджувати тенденції, наприклад, економічних чи політичних зрушень і прогнозувати подальші події; встановлювати зв'язки між різними процесами, виявляти їх взаємовпливи тощо. При роботі з таблицями інформаційно-довідкового характеру виділяються дві основні задачі: упорядкування елементів таблиці за певною ознакою й пошук елемента із заданою властивістю.

Методи пошуку розрізняються за тим, упорядковані чи ні дані в таблиці. Великі таблиці звичайно впорядковують, тобто елементи в них розміщують за зростанням чи за спаданням певної ознаки (строго кажучи, за неспаданням або незростанням певної ознаки). Так, у міському телефонному довіднику номери телефонів розташовані за алфавітним списком абонентів, інакше ми не змогли б користуватися таким довідником. Таблиці з невеликою кількістю даних можуть містити невпорядковані дані.

Ми розглянемо два методи пошуку елемента в лінійній таблиці. Перший метод призначається для пошуку в невпорядкованих лінійних таблицях, другий — в упорядкованих.

Спочатку з'ясуємо, що є метою пошуку. При наявності елемента в таблиці недостатньо тільки зафіксувати цей факт. Важливо знайти номер, за яким потрібний елемент розміщено в таблиці.

Наприклад, якщо в таблиці, що містить перелік товарів на складі, ми знайшли потрібний нам товар за номером 378, то за цим номером ми можемо встановити характеристики цього товару, які зберігаються в інших розділах таблиці, — його вартість, кількість наявних екземплярів, дату надходження і т. ін. Отже, *задачу* ***пошуку елемента в таблиці*** будемо розглядати в такій постановці: дано таблицю a, яка складається з різних між собою цілочисельних елементів, занумерованих числами натурального ряду від 1 до n. Визначити, чи містить ця таблиця елемент x. Якщо так, вказати номер елемента x в таблиці; якщо ні — вивести відповідне повідомлення.

====64.2. Пошук у невпорядкованій таблиці. Програма «Послідовний перебір» =========

Якщо таблиця невпорядкована, то для пошуку потрібного елемента нічого іншого не залишається, як послідовно переглядати елементи таблиці, доки не знайдеться потрібний елемент або таблицю буде вичерпано (у такому разі зазначаємо, що таблиця не містить шуканого елемента). Такий метод пошуку називається ***методом послідовного перебору***.

Алгоритм послідовного перебору передбачає перегляд елементів одного за одним у циклі.

Розпочинаємо перегляд з першого елемента, його індекс i=1. Раз за разом збільшуємо значення i на 1. Перегляд таблиці припиняється, якщо для чергового значення i елемент a[i]=x або якщо таблицю вичерпано, тобто переглянутий елемент є останнім: i=n. Для організації такого циклу доречно скористатися циклом із післяумовою. Умова виходу з циклу має вигляд:

(a[i]=x) або (i=n)

Після виходу з циклу необхідно перевірити, яка з простих умов — перша чи друга — справдилася. Якщо перша, то елемент знайдено й значення змінної i є номером елемента x в таблиці.

Таблицю a сформуємо з цілих чисел від 10 до 50 за допомогою датчика випадкових чисел. Кількість елементів табли-ці n задамо як константу. Виберемо значення n=10, щоб і таблиця, і діалог із користувачем розміщувалися на одному екрані. Сформовані елементи таблиці виведемо на екран до початку пошуку, щоб зручно було задавати значення x, яке потрібно відшукати в таблиці.

Усі змінні, які використовуються в програмі, за їх смисловим значенням відносимо до цілого типу.

Наведемо текст програми пошуку елемента методом послідовного перебору.

```
program linear_search;
 uses Crt;
 const n=10;
 ar i, x: integer;
 a: array[1..n] of integer;
begin
 randomize; clrscr;
 for i:=1 to n do
  begin
   a[i] := 10+random(41);                    {Визначаємо елементи таблиці}
   writeln(i, '. ', a[i]);                   {Виводимо їх з номерами на екран}
  end;
 write('Яке значення потрібно знайти? ');
 readln(x);                                  {Уводимо шукане значення}
 i := 0;
 repeat
  i := i+1;                                  {Перебираємо по черзі індекси елементів}
 until (a[i]=x) or (i=100);                  {Перевіряємо умову завершення перегляду}
 if a[i]=x                                   {Перевіряємо, чи перша умова справдилася}
  then writeln('Номер елемента — ', i)       {Виводимо результат пошуку}
  else writeln('Такого в таблиці немає.');
 readln
end.
```

Для перевірки правильності програми слід запустити її на виконання щонайменше 4 рази з такими значеннями x: значенням, яке співпадає з першим елементом таблиці; з її останнім елементом; з одним із внутрішніх елементів; не співпадає з жодним елементом таблиці.

Зауважимо, що ми розглянули метод послідовного перебору для таблиці з цілочисельними елементами, але алгоритм розв'язання задачі зберігає свою значущість для будь-якої таб-лиці.

====64.3. Пошук у впорядкованій таблиці.Програма «Бінарний пошук» ==========

Метод послідовного перебору не є ефективним, тому його застосовують тільки в таких випадках, коли інших методів запропонувати не можна. Якщо таблиця містить елементи, впорядковані певним чином, то цією властивістю таблиці можна скористатися для суттєвого прискорення процесу пошуку елемента.

Будемо вважати, що цілочисельні елементи нашої таблиці впорядковані за зростанням.

Ідея ***методу бінарного пошуку*** полягає в тому, що на кожному кроці діапазон невизначеності (тобто діапазон таблиці, де здійснюється пошук) скорочується вдвічі. Саме це й підкреслюється назвою методу: слово «бінарний» походить від англійського binary — двійковий.

Розглянемо алгоритм бінарного пошуку.

Спочатку діапазон пошуку охоплює елементи з індексами від `1` до `n`. Позначимо нижню межу пошуку через `ng`, верхню — через `vg`: `ng=1`, `vg=n`. Знайдемо середину цього діапазону `s` як півсуму меж. Порівняємо серединний елемент `a[s]` з `x`. Маємо один із трьох випадків:

`a[s]=x`, тоді пошук завершено;

`a[s]>x`, тоді `x` слід шукати в нижній половині діапазону, тобто між `ng` і `s-1`. Отже, змінюємо верхню межу пошуку: `vg=s-1`;

`a[s]<x`, тоді `x` слід шукати у верхній половині діапазону, тобто між `s+1` і `vg`. Отже, змінюємо нижню межу пошуку: `ng=s+1`.

Продовжуємо, якщо потрібно, пошук у скороченому вдвічі діапазоні. Урешті-решт ми або знайдемо елемент `x`, або в результаті пересувань меж (нижньої вгору, а верхньої вниз) виявиться, що верхня межа стала нижче нижньої, тобто `vg<ng`. Це свідчить про відсутність елемента `x` в таблиці.

Перегляд таблиці доцільно реалізувати як цикл із післяумовою. Умова виходу із циклу має вигляд: `(a[s]=x) або (vg<ng)`.

За розробленим алгоритмом складемо програму знаходження заданого елемента в упорядкованій таблиці методом бінарного пошуку. Сформуємо елементи таблиці за формулою $a[i]=i^2$ (можна взяти й іншу, але таку, щоб значення елементів зростали зі збільшенням їх номера `i`). За смислом задачі змінні `i, x, s, vg, ng` є змінними цілого типу.

Запишемо програму.

```
program binary_search;
 uses Crt;
 const n=20;
 var i, s, vg, ng, x: integer;
 a: array[1..n] of integer;
begin
 clrscr;
 for i:=1 to n do
  begin
   a[i] := sqr(i);                    {Формуємо впорядковану таблицю}
   writeln(i, '. ', a[i]);            {Виводимо її елементи на екран}
```

```
  end;
 write('Яке значення потрібно знайти? ');
 readln(x);                                    {Вводимо шукане значення}
 ng := 1;                                      {Установлюємо нижню межу пошуку}
 vg := n;                                      {Установлюємо верхню межу пошуку}
 repeat
  s := (ng+vg) div 2;                          {Знаходимо індекс середнього елемента}
  if a[s]>x                                    {Якщо середній елемент більше, ніж шуканий, }
   then vg := s-1;                             {то зміщуємо верхню межу}
  if a[s]<x                                    {Якщо середній елемент менше, ніж шуканий, }
   then ng := s+1;                             {то зміщуємо нижню межу}
 until (a[s]=x) or (vg<ng);                    {Перевіряємо умову завершення циклу}
 if a[s]=x                                     {Перевіряємо, чи перша умова справдилася}
  then writeln('Номер елемента - ', s)         {Виводимо результат пошуку}
  else writeln('Такого в таблиці немає.');
 readln
end.
```

Для перевірки правильності програми слід запустити її на виконання щонайменше 6 разів з такими значеннями x: значенням, яке співпадає з першим елементом таблиці; з її останнім елементом; з одним із внутрішніх елементів; менше найменшого елемента; більше найбільшого елемента; знаходиться між елементами таблиці.

Оскільки з кожним проходженням циклу діапазон невизначеності скорочується вдвічі, можна довести, що для пошуку елемента або встановлення його відсутності в упорядкованій таблиці з n елементів знадобиться не більше $\log_2 n$ кроків. Отже, пошук серед тисячі елементів завершиться щонайбільше за 10 кроків (2^{10}=1024), серед мільйона елементів — усього за 20 кроків (2^{20}=1 048 576), серед мільярда — за 30 кроків. Для порівняння нагадаємо, що за методом послідовного перебору відповідна кількість кроків оцінюється як кількість елементів (тобто для тисячі елементів — тисяча кроків і т. д.).

Як бачимо, впорядковане зберігання даних дозволяє забезпечити швидкий доступ до них. Це є надзвичайно важливим з огляду на те, яку безліч інформації здобуло і продовжує здобувати людство і як конче потрібно мати можливість оперативно використовувати ці багатства, щоб якнайшвидше йти вперед.

Якщо захочете переконатися в ефективності методу бінарного пошуку і спробувати знайти потрібний елемент серед мільйона заданих (n=1000000), то не забудьте всі змінні програми оголосити не як **integer**, а як **longint**; вибрати іншу формулу для формування елементів таблиці з тим, щоб їх значення не виходили за межі діапазону даних типу **longint** (наприклад, a[i]:=10+5*i;); вилучити оператор виведення елементів таблиці на екран.

ВИСНОВКИ

Пошук елемента в таблиці є типовою задачею при роботі з таблицею. Метою пошуку є визначення номера, за яким шуканий елемент міститься в таблиці, або встановлення відсутності такого елемента в таблиці. Якщо таблиця не впорядкована, то пошук елемента здійснюється методом послідовного перегляду елементів таблиці. Якщо елементи таблиці впорядковані, то для пошуку елемента можна застосувати метод бінарного пошуку. Це дозволяє значно скоротити кількість кроків, потрібних для здійснення пошуку.

Контрольні питання та вправи

1. Які задачі є основними при роботі з інформаційно-довідковими таблицями?

а) перейменування таблиці;
б) пошук елемента із заданою властивістю;
в) упорядкування елементів таблиці;
г) копіювання даних з однієї таблиці в іншу.

2. Метою пошуку елемента в таблиці є:

а) обчислення значення цього елемента;
б) одержання повідомлення про те, чи є шуканий елемент у таблиці;
в) знаходження номера, за яким елемент міститься в таблиці, або встановлення його відсутності в таблиці.

3. Серед наведених таблиць виберіть упорядковані:

а) таблиця чисел: –56, –17, 0, 2, 2, 43, 100;
б) таблиця чисел: –7566, –7056, 2145, 0, 2145, 4321, 99001;
в) таблиця чисел: –512, –517, –6240, –7584, –23100;
г) таблиця слів: весна, зима, літо, осінь;
д) таблиця прізвищ: Барабашов, Веприк, Мельник, Потьомкін;
е) таблиця слів: bit, byte, kilobyte, gigabyte, megabyte.

4. Для пошуку елемента в невпорядкованій таблиці можна застосувати:

а) метод послідовного перебору;
б) метод бінарного пошуку.

5. Метод бінарного пошуку можна застосувати для пошуку елемента:

а) у таблиці чисел, упорядкованих за спаданням;
б) у таблиці чисел, упорядкованих за зростанням;
в) у будь-якій числовій таблиці;
г) у таблиці слів, упорядкованих за алфавітом.

6. Складіть програму, яка в невпорядкованій таблиці з 20 цілих чисел із діапазону [–10; 10], визначених датчиком випадкових чисел, здійснює пошук числа 0. Якщо таких декілька, знайти перше.

7. Утворіть дві таблиці. Перша містить назви кінцевих пунктів слідування поїздів, друга — час їх відправлення. Складіть програму, яка за запитом користувача виводить на екран повідомлення, коли відправляється його поїзд.

8. Утворіть дві таблиці. Перша містить виграшні номери лотереї, впорядковані за зростанням, друга — виграші, які випали на ці номери. Складіть програму, яка перевіряє номер лотерейного білета користувача й повідомляє, чи виграв він у лотерею, і що саме, якщо виграв.

9. Утворіть дві таблиці. Перша містить список прізвищ учнів, упорядкований за алфавітом, друга — дати їх народження. Складіть програму, яка знаходитиме дату народження учня із заданим прізвищем.

10. Складіть дві таблиці. Перша містить перелік товарів у магазині подарунків, друга — їх ціни. Товари впорядковані за зростанням ціни. Складіть програму, яка за запитом покупця повідомляє, чи є подарунки на заданий діапазон цін, і які вони, якщо є.

11. Складіть програму «Відгадаю число!». Програма пропонує користувачеві загадати ціле число від 1 до 250. Програма може задавати користувачеві тільки запитання типу: «Ваше число більше (вказує число)?». Користувач може відповідати лише одне з двох: «Так» або «Ні». Після декількох запитань програма виводить на екран повідомлення: «Ви загадали число …» (указує число) або «Ви неправильно відповідали на мої запитання!».

12. Складіть ігрову програму «Відгадай моє число!». Користувач задає діапазон чисел для гри. Програма за допомогою датчика випадкових чисел «загадує» число із заданого діапазону і пропонує користувачеві відгадати його. Програма виводить на екран запитання «Ви знаєте моє число?». Якщо користувач відповідає «Так», то програма пропонує йому ввести число й перевіряє правильність відгадування. Якщо користувач відповідає «Ні», то програма пропонує йому підказку: «Введіть яке-небудь число, а я скажу, чи більше воно від загаданого». Програма веде облік кількості підказок, які знадобилися користувачеві для відгадування числа.

бінарний пошук, метод послідовного перебору, пошук елемента в таблиці