§ 65. Упорядкування лінійних таблиць

Вивчивши цей параграф, ми:

познайомимося із задачею впорядкування лінійних таблиць;

дізнаємося, які ідеї складають основу простих методів упорядкування таблиці;

з'ясуємо, як складаються алгоритми впорядкування таблиці за методом вибору й за методом вставки;

познайомимося з програмами, які реалізують упорядкування таблиць за методами вибору і вставки.

====65.1. Задача впорядкування лінійної таблиці==========

Ми знаємо, що в лінійній таблиці всі її елементи перенумеровані. Лінійна таблиця є *впорядкованою*, якщо із зростанням номерів її елементів зростає або спадає їх певна ознака. У першому випадку кажуть, що таблиця впорядкована за зростанням, у другому — за спаданням цієї ознаки. Якщо деякі елементи таблиці можуть мати однакове значення ознаки, то йдеться не про її зростання або спадання, а про неспадання і незростання.

Одну й ту саму таблицю можна впорядкувати за різними ознаками. Наприклад, лінійна таблиця, яка містить прізвища учнів, може бути впорядкованою за алфавітом, за довжиною прізвищ учнів тощо. Таблиця, яка містить цілі числа, може бути впорядкованою за значенням самих чисел або суми цифр, з яких складаються ці числа, і т. ін.

Будемо розглядати методи впорядкування лінійних таб-лиць на прикладі таблиці з числовими елементами.

Задачу ***впорядкування таблиці*** сформулюємо таким чином: дано таблицю `a`, яка складається з `n` цілочисельних елементів. Переставити елементи таблиці так, щоб вони були розташовані за зростанням (неспаданням) їх значень.

Розглянемо два прості методи розв'язання цієї задачі. Вони прості тому, що побудовані на цілком зрозумілих ідеях.

====65.2. Упорядкування методом вибору==========

Ідея ***методу вибору*** полягає в послідовному пошуку спочатку найменшого елемента в таблиці й перенесенні його на перше місце, потім — наступного за величиною й перенесенні його на друге місце і т. д. При перенесенні елемента в певне місце той, що там знаходився, займає його місце.

Покажемо, як відбувається впорядкування таблиці за методом вибору на прикладі таблиці з таких елементів: 4, 7, 2, 8, 17, 19, 5. Кількість елементів $n=7$. Елементи, які були перенесені на початок таблиці, будемо виділяти жирним шрифтом.

№ етапу	Діапазон таблиці, що розглядається	Найменший елемент	Місце, на яке переносимо	Результат перенесення
1	від 1 до 7: 4, 7, 2, 8, 17, 19, 5	**2**	1	**2**, 7, 4, 8, 17, 19, 5
2	від 2 до 7: 7, 4, 8, 17, 19, 5	**4**	2	**2**, **4**, 7, 8, 17, 19, 5
3	від 3 до 7: 7, 8, 17, 19, 5.	**5**	3	**2**, **4**, **5**, 8, 17, 19, 7
4	від 4 до 7: 8, 17, 19, 7.	**7**	4	**2**, **4**, **5**, **7**, 17, 19, 8
5	від 5 до 7: 17, 19, 8	**8**	5	**2**, **4**, **5**, **7**, **8**, 19, 17
6	від 6 до 7: 19, 17	**17**	6	**2**, **4**, **5**, **7**, **8**, **17**, 19

====65.3. Розробка алгоритму впорядкування таблиці методом вибору==========

Для розробки алгоритму впорядкування таблиці методом вибору звернемо увагу на наведений вище приклад. Як бачимо, крок за кроком змінюється нижня межа діапазону перегляду таблиці: від `1` на першому кроці до `n-1` на останньому кроці. Позначимо нижню межу як `ng` і подамо весь процес упорядкування таблиці у вигляді циклу:

```
для ng від 1 до n-1
 пс
  <знайти найменший серед елементів таблиці від a[ng] до a[n]>
  <поміняти місцями найменший елемент з елементом a[ng]>
 кс
```

Розглянемо по черзі складові нашого циклу.

Першою складовою є пошук найменшого серед елементів таблиці від `a[ng]` до `a[n]`. Крім найменшого елемента (позначимо його `min`), нам потрібно знайти і його місце в таблиці, тобто його індекс (позначимо його `m`). Таким чином, `min=a[m]`. Нагадаємо, що аналогічну задачу ми розв'язували при визначенні переможця спортивних змагань.

Запишемо алгоритм першої складової нашого циклу:

```
min := a[ng]
m := ng
для i від ng+1 до n
 якщо a[i]<min то
  пс
   min := a[i]
   m := i
  кс
```

Друга складова циклу — це обмін місцями двох елементів таблиці, тобто найменшого елемента (це елемент `a[m]`) й елемента `a[ng]`. Обміняти елементи місцями означає надати змінній `a[m]` значення, яке зберігає змінна `a[ng]`, а змінній `a[ng]` — значення, яке зберігає змінна `a[m]`. Спробуємо це зробити.

За командою `a[m]:=a[ng]` значення змінної `a[ng]` буде надано змінній `a[m]`, і вона втратить своє попереднє значення. На щастя, воно зберігається ще й в змінній `min`, адже `min=a[m]`. Ми скористаємося цим і надамо змінній `a[ng]` нового значення командою: `a[ng]:=min`.

Таким чином, друга складова нашого циклу містить усього дві команди:

```
a[m] := a[ng];
a[ng] := min;
```

Тепер для запису алгоритму не вистачає лише опису змінних.

====65.4. Програма «Впорядкування таблиці методом вибору» ==========

Розглянутий нами алгоритм упорядкування таблиці методом вибору перетворимо на програму на мові Паскаль. Програма матиме такі складові: введення елементів невпорядкованої таблиці (скористаємося для цього датчиком випадкових чисел); упорядкування таблиці; виведення елементів таблиці після її впорядкування. Виведення елементів таблиці здійснюється так само, як і їх введення, у циклі. Елементи таблиці при введенні й виведенні будемо нумерувати. На екрані розмістімо їх у стовпчику для зручності наочного порівняння, тому кількість елементів обмежимо: візьмемо `n=10`.

```
program sorting_by_choosing;
 uses Crt;
 const n=10;
 var i, m, min, ng: integer;
 a: array [1..n] of integer;
begin
 clrscr;
 randomize;
 writeln('Таблиця до впорядкування:');
 for i:=1 to n do                                    {Формуємо таблицю}
  begin
   a[i] := random(50);                               {Визначаємо елементи таблиці}
   writeln (i, '. ', a[i]);                          {Виводимо їх з номерами на екран}
  end;
 for ng:=1 to n-1 do                                 {Упорядковуємо таблицю за n–1 прохід}
  begin
   min := a[ng];
   m := ng;
   for i:=ng+1 to n do                               {Розпочинаємо перегляд таблиці}
    if a[i]<min then                                 {для пошуку найменшого елемента}
     begin
      min := a[i];
      m := i;
     end;                                            {Перегляд таблиці завершено,}
   a[m] := a[ng];                                    {обмінюємо місцями}
   a[ng] := min;                                     {найменший і початковий елементи}
  end;
 writeln('Упорядкована таблиця:');                   {Виводимо впорядковану таблицю}
 for i:=1 to n do
  writeln (i, '. ', a[i]);
 readln
end.
```

Ми записали програму для впорядкування цілочисельної таблиці, проте її можна застосувати й для таблиці величин типу **string**, тому що для них виконуються операції порівняння (за алфавітним порядком). Потрібно лише внести відповідні зміни в описову частину програми.

====65.5. Упорядкування таблиці методом простих вставок==========

Ідея ***методу вставки*** полягає в тому, щоб почергово вставляти один елемент за одним у відповідне його значенню місце.

Уявіть, що на плацу вишукувалася шеренга солдат. Солдати стоять один поряд з одним за зростом. Місце правофлангового на плацу є визначеним. Командир веде до взводу новачка, щоб поставити його в стрій. Командир проводить новачка вздовж строю, рухаючись з лівого флангу до правого. Кожний солдат, починаючи з лівофлангового, менший за зростом за новачка, робить крок вліво, пропускаючи новачка вперед, доки черга не дійде до солдата, який не менше

новачка за зростом. Тепер новачок стає в шеренгу на вільне місце. Якщо ж новачок виявився вищим за всіх солдат, він стає правофланговим.

Тепер уявіть, що нам потрібно розставити за зростом учнів класу. Скористаємося описаним вище способом. Спочатку поставимо на місце правофлангового першого зі списку, потім визначимо місце другого, далі – третього, і так за списком по черзі вишукуємо весь клас. Кожного разу ми вирішуватимемо одну й ту саму задачу: знайти місце чергового учня в упорядкованому строю попередніх. Саме така ідея є основою методу простих вставок.

Покажемо, як працює метод простих вставок, на прикладі впорядкування числової таблиці з елементами: 7, 9, 4, 6, 8 (*n=5*). Уведемо допоміжну змінну `x`, якій будемо надавати значення елемента, що вставляється в таблицю. Як тільки значення елемента передане змінній `x`, місце елемента можна вважати вільним. Будемо позначати перенесення значення елемента у змінну *x* знаком "↑" (наприклад, 9 ↑ *x*); переміщення елемента на одну позицію вправо – знаком "→" (наприклад, 7→); вставляння значення *x* у вільне місце таблиці – знаком "*x* ↓". Розглянуті елементи будемо виділяти жирним шрифтом.

Оскільки ми маємо впорядкувати 5 елементів, за методом вставок це буде здійснено в 4 етапи.

Етап	Виконувані дії	Значення *x*	Таблиця 7, 9, 4, 6, 8
1	9 ↑	**9**	**7**, _, 4, 6, 8
	7>9 "ні", *x* ↓		**7**, **9**, 4, 6, 8
2	4 ↑	**4**	**7**, **9**, _, 6, 8
	9>4 "так", 9→		**7**,_, **9**, 6, 8
	7>4 "так", 7→		_, **7**, **9**, 6, 8
	край таблиці, *x* ↓		**4**, **7**, **9**, 6, 8
3	6 ↑	**6**	**4**, **7**, **9**, _, 8
	9>6 "так", 9→		**4**, **7**, _, **9**, 8
	7>6 "так", 7→		**4**, _, **7**, **9**, 8
	4>6 "ні", *x* ↓		**4**, **6**, **7**, **9**, 8
4	8 ↑	**8**	**4**, **6**, **7**, **9**, _
	9>8 "так", 9→		**4**, **6**, **7**, _, **9**
	7>8 "ні", *x* ↓		**4**, **6**, **7**, **8**, **9**

Зверніть увагу на те, що на завершення кожного етапу проглянута частина таблиці (її виділено підсвіткою) є впорядкованою.

====65.6. Розробка алгоритму впорядкування таблиці методом простих вставок========

З наведеного вище прикладу можна побачити, що процес упорядкування таблиці здійснюється почергово для 2-го, 3-го, …, останнього (`n`-го) елементів таблиці. Отже, маємо цикл для `i` від `2` до `n`, де через `i` позначено номер елемента в таблиці:

```
для i від 2 до n
  пс
```

надати змінній *x* значення *a*[*i*]

пересунути елементи зліва від *a*[*i*], більші за *x*, на одну позицію вправо і запам'ятати номер елемента, що виявився не більшим за *x*

вставити *x* у таблицю

```
кс
```

Розглянемо по черзі складові нашого циклу.

Перша складова є зовсім простою: вона здійснюється командою `x:=a[i]`.

Друга складова передбачає перевірку й пересування елементів ліворуч `a[i]`, тобто елементів, індекс яких `k` менший за `i` і набуває почергово таких значень:

```
k = i-1, i-2, …, 1
```

Для кожного значення `k` перевіряємо умову: `a[k]>x`. Якщо результатом перевірки є «так», то `a[k]` зміщуємо на одну позицію командою `a[k+1]:=a[k]` і переходимо до перегляду наступного елемента: `k:=k-1`. Якщо для деякого елемента `a[k]` результатом перевірки умви є «ні», то номер цього елемента запам'ятовуємо у змінній `j:` `(j:=k)`, і подальша перевірка має припинитися.

Може статися, що всі перевірки дали результат «так», тоді вихід із циклу має завершитися через вичерпання всіх значень `k>0`.

Реалізуємо перевірку й пересування елементів таблиці у вигляді циклу з передумовою. До початку циклу змінним `j` і `k` потрібно надати початкових значень: `k:=i-1, j:=0`. Цикл має виконуватися, доки *не знайдено* елемента, який пересувати не потрібно (про це свідчитиме нульове значення змінної `j`), і *не завершено* перегляд таблиці (про це свідчитиме значення змінної `k>0`). Отже, умовою циклу є: `(j=0) і (k>0)`. Якщо змінна `j` набуде ненульового значення або змінна `k` зменшиться до нуля, відбудеться вихід із циклу.

Запишемо алгоритм другої складової нашого алгоритму:

```
j := 0; k := i-1
доки (j=0) і (k>0)
 якщо a[k]>x
   то пс
   a[k+1] := a[k]
   k := k-1
   кс
 інакше
   j := k
```

Третя складова має забезпечувати вставляння елемента `x` на визначене для нього місце. Змінна `j` або зберігає значення місця першого елемента таблиці, не більшого за `x`, або дорівнює нулю, якщо такого не знайшлося. Отже, у будь-якому випадку місце елемента `x` в таблиці визначається значенням суми `j+1`, і його вставляння в таблицю здійснюється командою: `a[j+1]:=x`.

Тепер наш алгоритм розроблено повністю. Для його подання в остаточному вигляді потрібно поєднати відповідним чином усі його складові й доповнити алгоритм описом змінних.

====65.7. Програма «Впорядкування таблиці методом простих вставок» ==========

За розробленим алгоритмом упорядкування таблиці методом простих вставок можна скласти програму впорядкування заданої таблиці елементів. Ми можемо реалізувати розроблений алгоритм мовою Паскаль, діючи за такою схемою, як і для впорядкування таблиці методом вибору. Проте цікавіше зробити по-іншому.

Застосуємо розроблений алгоритм для упорядковування списку слів (наприклад, англійських слів), які вводяться з клавіатури по черзі, одне за одним. Кожного разу після вставляння слова в таблицю будемо виводити її на екран. Це надасть нам можливість спостерігати в динаміці, як працює метод.

Наведемо програму.

```
program sorting_by_ inserting;
 uses Crt;
 const n=10;
 var i, j, k: integer; x: string;
 a: array[1..n] of string;
begin
 clrscr;
 writeln('Уведіть чергове слово');
 for i:=1 to n do                                  {Цикл для введення слів}
  begin
   readln(x);                                      {Уведення чергового слова}
   j := 0;                                         {Реалізація методу вставки}
   k := i-1;
   while (j=0) and (k>0) do                        {Цикл для пошуку місця чергового
    if a[k]>x                                      {слова серед раніше введених слів}
     then begin
      a[k+1] := a[k];
      k :=  k-1;
      end
     else j :=  k;
    a[j+1] :=  x;                                  {Уставляння чергового слова в список}
    writeln;                                       {Пропуск рядка}
   writeln('Список: ');
   for k := 1 to i do                              {Виведення поточного списку слів}
  writeln(k, '. ', a[k]);                          {з номерами у стовпчик}
  end;
 readln
end.
```

Цією програмою можна скористатися і для впорядкування послідовності чисел, якщо внести відповідні зміни в опис змінних.

ВИСНОВКИ

Упорядкування масивів даних є надзвичайно важливою задачею, тому що при безсистемному нагромадженні вони стають недоступними для використання людиною і втрачають свою цінність. Упорядкування означає розміщення даних у певному порядку — за зростанням або спаданням певної ознаки. Один і той самий масив даних може бути впорядкованим за різними ознаками. Для впорядкування даних існують різні методи. Метод вибору передбачає багатократний перегляд таблиці для пошуку спочатку найменшого серед усіх елементів і його перенесення в початок таблиці, потім — наступного за величиною і т. д. Основу методу вставки складає послідовний перегляд елементів з метою винайдення місця для вставляння нового елемента в упорядковану таблицю. Ці методи не є швидкодіючими. На практиці для впорядкування великих масивів даних застосовують більш досконалі, проте й більш складні методи.

Контрольні питання та вправи

1. Таблиця об'єднує 10 елементів. Для її впорядкування застосовано метод вибору. Скільки разів буде виконано операцію порівняння елементів у процесі впорядкування таблиці?

а) 9;
б) 10;
в) 45;
г) 10^2.

2. Таблиця об'єднує 10 елементів: 1, 2, 3, 4, 5, 6, 7, 8, 10, 9. За яким методом її можна швидше впорядкувати за зростанням?

а) за методом вибору;
б) за методом вставки.

3. Перша таблиця містить елементи: 10, 2, 3, 4, 5, 6, 7, 8, 9, 1. Друга — елементи: 1, 2, 3, 4, 5, 6, 7, 8, 10, 9. Для впорядкування таблиць за зростанням застосували метод вибору. Яку з таблиць було впорядковано швидше?

а) першу;
б) другу;
в) швидкість упорядкування таблиць є однаковою.

4. Для якої з таблиць метод вставки швидше завершить її впорядкування за зростанням?

а) 1, 2, 3, 4, 5, 7, 6, 8, 10, 9;
б) 10, 2, 3, 4, 5, 6, 7, 8, 9, 1;
в) 1, 2, 3, 4, 5, 6, 10, 9, 8, 7;
г) 5, 2, 3, 4, 1, 6, 7, 8, 9, 10;
д) для всіх таблиць кількість дій однакова.

5. Модифікуйте програму впорядкування таблиці методом вибору так, щоб вона здійснювала впорядкування заданої числової таблиці за спаданням.

6. Модифікуйте програму впорядкування таблиці методом вибору так, щоб вона розташовувала слова у списку за алфавітом.

7. Модифікуйте програму впорядкування таблиці методом вставки так, щоб вона із чисел, що вводяться з клавіатури, утворила їх список, упорядкований за зростанням.

8. Модифікуйте програму впорядкування таблиці методом вставки так, щоб вона впорядковувала задану таблицю чисел за спаданням.

9. Модифікуйте програму впорядкування таблиці методом вставки так, щоб вона впорядковувала задану таблицю слів за алфавітом.

10. Таблиця, утворена за допомогою датчика випадкових чисел, містить `n` цілих чисел. Складіть програму визначення кількості чисел, які стоять на своєму місці, тобто займають те місце в таблиці, на якому вони опинилися б після її впорядкування за неспаданням.

11. Складіть програму вилучення із заданого списку тих прізвищ, що повторюються.

12. У супермаркеті здійснюється реєстрація кожної покупки, зробленої покупцем. Складіть програму, яка за списком найменувань товарів, придбаних упродовж дня, визначає, який товар користувався найбільшим попитом.