Лабораторна робота №5. (4 год.)

Тема: Обробка масивів.
Мета: Навчитись розробляти алгоритми та програми, які виконують дії над одновимірними масивами.

Теоретичні відомості

Інколи при складанні програм зручно оперувати даними, представленими у вигляді таблиць, дані в такому виді називають *масивами*. В такому випадку, можна оперувати групою даних за допомогою одного ім'я і різноманітних індексів, це дає можливість спростити програму. Є масиви *одновимірні* та *багатовимірні*.

Одновимірний масив – це просто список елементів даних. Якщо представити такий масив у вигляді таблиці, то це буде її одна стрічка. Кожний елемент даних, що зберігається в масиві називається *елементом* масиву. Порядковий номер елементу в масиві називається *індексом*. Індекс елемента записується поряд з назвою масиву у круглих дужках без пробілів і може бути лише цілим числом. Найменший порядковий номер масиву називається *нижньою* границею, найбільший – *верхньою*.

Перед використанням масивів їх, так само як і змінні, потрібно описати. Елементи масивів можуть мати такі самі типи, як і змінні. Стрічками

```
Dim a(1 To 100) As Integer
Dim b(0 To 2) As Single
Dim c_array(1 To 5) As String
```

відповідно описуються масив *a* зі 100 елементів цілих чисел, масив *b* із трьох елементів дробових чисел, масив *c_array* із шістьма елементами стрічкових даних.

Якщо у програмі потрібно використати якийсь елемент масиву, то до нього потрібно звертатися таким чином: вказується ім'я масиву та індекс, наприклад:

a(1) – дія повинна виконатися з першим елементом масиву a;

```
For i=1 To 10
      a(i+1)=a(i)+10
Next i
```

– у цьому фрагменті кожний наступний елемент масиву *a* дорівнюватиме попередньому елементу збільшеному на 10.

Як показано в останньому прикладі для того, щоб одержати доступ до кожного елементу масиву, необхідно використовувати цикли.

Існує ряд алгоритмів для обробки даних масивів. Розглянемо деякі з них.

Сортування

Для виконання сортування масив зручно представити у вигляді лінійки елементів, в яких зберігаються числа. Наприклад:

Масив А

-4	16	12	0	-77	11	32	77	-41	2	12	-98	91	значення елементів
1	2	3	4	5	6	7	8	9	10	11	12	13	індекси

При сортуванні, наприклад, за зростанням можна обрати два шляхи:

Перший метод *послідовного пошуку екстремальних елементів*:

Знайти в масиві А найменший елемент та його номер і встановити його на 1-ше місце, а 1-й елемент - на місце, де був знайдений найменший елемент. У даному прикладі найменшим є 12-й елемент, а його значення дорівнює 98.

Після перестановки масив буде мати такий вигляд:

Масив А

-98	16	12	0	-77	11	32	77	-41	2	12	-4	91	значення елементів
1	2	3	4	5	6	7	8	9	10	11	12	13	індекси

Тепер аналогічні дії можна повторити, але пошук мінімального і перестановку треба почати з другого елементу. В даному випадку мінімальним буде п`ятий елемент із значенням -77. Після перестановки масив буде мати такий вигляд :

Масив А

-98	-77	12	0	16	11	32	77	-41	2	12	-4	91	значення елементів
1	2	3	4	5	6	7	8	9	10	11	12	13	індекси

Таким чином, описані дії треба повторювати доти, доки не буде досягнуто 12-го елементу. Після цього масив А буде відсортовано за зростанням.

Отже, якщо розмірність масиву дорівнює N, то треба створити два вкладених цикли, де зовнішній має N-1 разів повторити внутрішній. У внутрішньому циклі треба виконувати пошук найменшого елементу і його індексу. При цьому, після закінчення внутрішнього циклу треба виконувати дві дії:

- робити перестановку знайденого мінімального елементу з елементом, що знаходиться на початку чергового пошуку.
- збільшувати на 1 (одиницю) спеціальну змінну **b**, яка має вказувати на початковий номер елементу, з якого наступного разу буде відбуватись пошук мінімального елементу.

Наприклад:

```
b=1 'починаємо шукати з 1 елемента
for i = b to 12
      Num = b 'припустимо, що номер мін. елемента = b
      Min = A(Num) 'а це його значення
      for k = b to 12
            if {.......} then' тут відшукаємо номер найменшого
`елементу в масиві в  діапазоні індексів від b до 12
            end if
      next k
  'переставимо значення елемента B та Num_Min
  z = A(b)
  'запам`ятаємо значення початкового елемента в Z
  A(b) = A(Num)
  'перепишемо значення MIN в ел. B
  A(Num) = z 'а значення ел.B - в ел. Num_Min
  b = b + 1 'зсунемо початок наступного пошуку на 1
Next i
```

Другий метод *„бульбшковий"*:

Полягає в тому, що елементи масиву порівнюються *послідовно* та *попарно*, і якщо наступний буде меншим за попередні, то вони міняються місцями. Тобто більший з двох елементів „спливає" - просувається в бік зростання значень масиву. Тому цей метод називається „бульбашковим". Робота починається з першого та другого елементів. У даному випадку перший елемент менший ніж другий, отже, перестановку не виконуємо. Після цього порівнюємо другий і третій елементи. Треба робити перестановку. Тепер масив має такий вигляд:

Масив А

-4	12	16	0	-77	11	32	77	-41	2	12	-98	91	значення

													елементів
1	2	3	4	5	6	7	8	9	10	11	12	13	індекси

Для третього і четвертого елементів дії аналогічні.

Масив А

-4	12	0	16	-77	11	32	77	-41	2	12	-98	91	значення елементів
1	2	3	4	5	6	7	8	9	10	11	12	13	індекси

Ці дії треба повторювати до порівняння 12 та 13 елементів, а потім цю всю процедуру (спочатку) треба повторювати ще 11 разів.

У випадку якщо масив має розмірність *n*, треба створити 2 вкладених цикли з кількістю проходів *n*-1 кожний. У внутрішньому циклі будуть порівнюватись та переставлятись за необхідністю сусідні елементи, а зовнішній має його повторити *n*-1 разів.

Розглянемо приклад сортування масиву за „бульбашковим" методом

1) Розробити алгоритм і програму сортування масиву з 10 елементів, які можуть набувати випадкових цілих значень від 1 до 100.

Розміщуємо на формі об'єкти та відповідно до малюнку змінюємо їх властивості (рис.1)

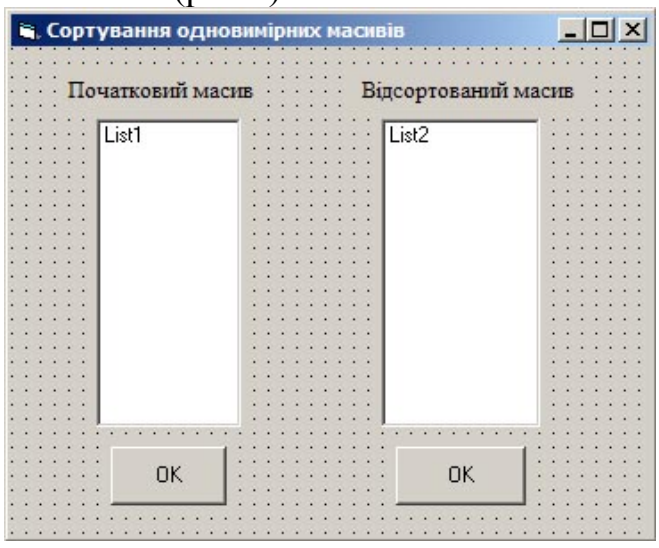

Рис.1

За умови подвійного натиснення мишкою кнопки „ОК" переходимо у вікно програми, яке матиме наступний вид (рис.2):

```
Dim X(1 To 10) As Single ' об`являємо масив
Dim i, j, p As Integer 'об`являємо змінні цілого типу

' ---------------------------------------------
Sub_Command1_Click()' підпрограма обробки події натискання
'кнопки Command1
List1.Clear ' чистимо список
Randomize ' вмикаємо генератор випадкових чисел

For i = 1 To 10
      X(i) = Int(100 * Rnd) ' одержуємо випадкове число в діапазоні
'від 0 до 1, множимо його на 100 і беремо цілу частину
Next i

For i = 1 To 10
      List1.AddItem X(i) ' виводимо значення елементів масиву в
' список List1
Next i
End Sub

' ---------------------------------------------
Private Sub_Command2_Click()' підпрограма обробки події натискання
'кнопки Command2

List2.Clear ' чистимо список

For j = 1 to 9
      For i = 1 to 9
            If X(i) > X(i+1) Then ' сусідні елементи розташовані не
      ' за правилом впорядкування. Тому робимо перестановку елементів
                  P = X(i)
                  X(i) = X(i + 1)
                  X(i + 1 ) = p
            End If
      Next i
Next j

For i = 1 To 10
      List2.AddItem X(i) ' виводимо значення елементів масиву в
' список List2
Next i

End Sub
```

Рис.2

У процедурі *Command*1_*Click*() показано як заповнити у циклі масив, що називається *x_array* цілими випадковими числами. Функція *Rnd* без аргументів повертає дробове число на проміжку від 0 до 1, помноживши це число на 100 і відкинувши дробову частину (функція *Int*()), одержуємо ціле число менше 100. Процедура *Randomize* дозволяє кожний раз після очищення вінка *List*1 одержувати нові значення елементів масиву.

У процедурі *Command*2_*Click*() здійснюється сортування одержаного масиву.

Результат роботи програми показано на рис.3, слід зауважити, що при кожному новому запуску програми початковий масив буде іншим, тому при перевірці правильності роботи програми слід звернути увагу лише на факт сортування даних, а не на те, які дані відсортовуються.

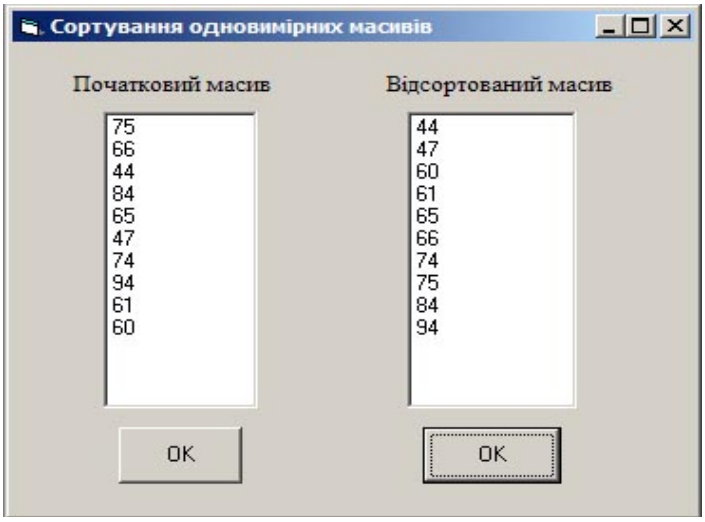

Рис.3

Порядок виконання роботи

1. Відтворити проект, поданий у прикладі 1.
2. Виконати сортування елементів масиву за зростанням та за зменшенням, використовуючи метод послідовного пошуку екстремальних елементів.

Звіт з лабораторної повинен містити:

- *тему та мету;*
- *блок-схему алгоритмів для обох методів сортування масивів;*
- *схематичний вид розташування об'єктів на формі до завдання 2;*
- *програму;*
- *результат;*
- *висновки.*

Контрольні запитання

1. Що таке масив даних?
2. Що таке одновимірний масив?
3. Які дані можуть бути записані у масив?
4. Які оператори використовуються для опису масивів?
5. Що таке індекс і для чого він використовується?

6. Які типи змінних можна використовувати в якості індексів масивів?
7. Яким чином здійснюється доступ до елементів масиву?
8. Чому при роботі з масивами зручно використовувати цикли?
9. Наведіть приклади з життя, в яких можна було б використовувати масиви.
10. Яким чином можна поміняти значення пари змінних у масивах місцями?
11. В чому суть методу послідовного пошуку екстремальних елементів при сортуванні масивів?
12. В чому суть "бульбашкового методу" при сортуванні масивів?
13. Чому в "бульбашковому" методі треба повторювати (скільки разів?) процедуру перестановки елементів?