Лабораторна робота № 14

Тема. Сортування одновимірних масивів

Мета. Формування вмінь і навиків сортування одновимірних масивів. Закріплення вмінь і навиків використання функцій вводу-виводу. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Контрольні запитання.

1. На які два типи поділяються алгоритми сортування по швидкодії? До яких масивів доцільно застосовувати алгоритми кожного типу?
2. На які три категорії поділяються алгоритми сортування без допоміжних структур?
3. Який принцип закладено в основу сортування алгоритмом бульбашок? Прямого включення? Прямого вибору? Швидкого сортування обміном на великих відстанях?
4. Вкажіть переваги та недоліки по швидкості виконання методу прямого включення відносно методу прямого вибору.
5. Від чого залежить швидкість виконання алгоритмів сортування?

Завдання.

1. Скласти програму формування та друку масиву з 20000 дійсних елементів проміжку [a,b]. Відсортувати сформований масив та визначити час сортування методом *бульбашок*. Вивести сформований масив та час його сортування.

Вимоги до програма:

- числові значення *a* та *b* ввести за допомогою оператора присвоєння;
- масив формувати за допомогою оператора присвоєння з використанням датчик випадкових чисел;
- результати формування вивести в одному діалоговому вікні, вказавши для кожного елемента масиву його індекс та значення.

Варіанти:

1. a=10, b=40;
2. a=4, b=18;
3. a=30, b=100;
4. a=8, b=50;
5. a=5, b=70;
6. a=15, b=300;
7. a=200, b=290;
8. a=17, b=80;
9. a=37, b=85;
10. a=7, b=56;
11. a=15, b=80;
12. a=200, b=400;
13. a=127, b=800;
14. a=3, b=1000;
15. a=16, b=160.

Наприклад, програма для розв'язання завдання варіанту № 8 може мати такий вигляд:

```
static void generateMas(double[] mas, int N, double a, double b)
{   Random rnd = new Random();
    for (int i = 0; i < N; i++)
        mas[i] = a + rnd.NextDouble() * (b - a);
}

static void printMas(double[] mas, int N, string s)
{   Console.WriteLine(s);
    for (int i = 0; i < N; i++)
        Console.WriteLine("mas[" + i.ToString() + "]=" + mas[i].ToString());
}

static void Main(string[] args)
{   const double a = 17, b = 80;
    int N, i, j, k, indexMin;
    Console.Write("Введіть кількість елементів масиву: ");
    N = Convert.ToInt32(Console.ReadLine());
    double[] mas = new double[N];
    double element, min;
```

```
    generateMas(mas, N, a, b);
    printMas(mas, N, "Початковий массив");
    Console.ReadKey();
    //метод бульбашок
    DateTime startTime = DateTime.Now;
    for (i = 0; i < N - 1; i++)
     for (j = 0; j < N - i - 1;j++)
      if(mas[j]>mas[j+1])
       {element = mas[j];
        mas[j] = mas[j + 1];
        mas[j + 1] = element;
       }
    DateTime finishTime = DateTime.Now;
    printMas(mas, N, "\nВідсортований массив методом бульбашок:");
    Console.WriteLine("Тривалість сортування методом бульбашок: " +
                      (finishTime - startTime).ToString());
    Console.ReadKey();
}
```

2. **Розв’язати попереднє завдання для масиву цілочисельних елементів.**
3. **Розв’язати завдання № 1, відсортувавши масив методом *прямого включення*.**

Фрагмент програми для розв’язання цього завдання (в продовження програми попереднього завдання) може виглядати так:

```
generateMas(mas, N, a, b);
printMas(mas, N, "Початковий массив");
Console.ReadKey();
//метод прямого включення
startTime = DateTime.Now;
for (i = 1; i < N;i++ )
{   j = 0;
    element=mas[i];
    while (j < i && mas[j] <= mas[i])
        j++;
    if(j<i)
    {for (k = i - 1; k >= j; k--)
      mas[k + 1] = mas[k];
     mas[j] = element;
    }
}
finishTime = DateTime.Now;
printMas(mas, N, "\nВідсортований массив методом прямого включення:");
Console.WriteLine("Тривалість сортування методом прямого включення: " +
                  (finishTime - startTime).ToString());
Console.ReadKey();
```

4. **Розв’язати завдання № 1, відсортувавши масив методом *прямого вибору*.**

Фрагмент програми для розв’язання цього завдання (в продовження програми попереднього завдання) може виглядати так:

```
generateMas(mas, N, a, b);
printMas(mas, N, "Початковий массив");
Console.ReadKey();
//метод прямого вибору
startTime = DateTime.Now;
```

```
for (i = 0; i < N-1; i++)
{   min = mas[i];
    indexMin = i;
    for (j = i + 1; j < N; j++)
     if (mas[j]<min)
      {min = mas[j];
       indexMin=j;
      }
    if (indexMin>i)
     {mas[indexMin] = mas[i];
      mas[i] = min;
     }
}
finishTime = DateTime.Now;
printMas(mas, N, "\nВідсортований массив методом прямого вибору:");
Console.WriteLine("Тривалість сортування методом прямого вибору: " +
                  (finishTime - startTime).ToString());
Console.ReadKey();
```

5. Розв'язати завдання № 1, відсортувавши масив методом *швидкого сортування обміном на великих відстанях*. Порівняти час сортування цим методом з часом сортування стандартним методом *Sort* класу *Array*.

Фрагмент програми для розв'язання цього завдання (в продовження програми попереднього завдання) може виглядати так:

```
generateMas(mas, N, a, b);
printMas(mas, N, "Початковий массив");
Console.ReadKey();
//метод швидкого сортування обміном на великих відстанях
startTime = DateTime.Now;
quickSort(mas, 0, N - 1);
finishTime = DateTime.Now;
printMas(mas, N, "\nВідсортований массив обміном на великих відстанях:");
Console.WriteLine("Тривалість швидкого сортування обміном на великих відстанях: " +
                  (finishTime - startTime).ToString());
Console.ReadKey();
generateMas(mas, N, a, b);
printMas(mas, N, "Початковий массив");
Console.ReadKey();
//стандартний метод сортування
startTime = DateTime.Now;
Array.Sort(mas);
finishTime = DateTime.Now;
printMas(mas, N, "\nВідсортований массив");
Console.WriteLine("Тривалість стандартного сортування: " +
                  (finishTime - startTime).ToString());
Console.ReadKey();
```

А сама рекурсивна функція швидкого сортування може бути такою:

```
static void quickSort(double[] mas,int i,int j)
{   if (i >= j)
     return;
    double element = mas[i];
    int i1 = i; int j1 = j;
    while(i1<j1)
     {while (j1 > i1 && mas[j1] >= element)
       j1--;
      if(i1<j1)
       {mas[i1] = mas[j1];
        mas[j1] = element;
        i1++;
       }
      while (i1 < j1 && mas[i1] <= element)
       i1++;
      if (i1 < j1)
       {mas[j1] = mas[i1];
        mas[i1] = element;
        j1--;
       }
    }
    quickSort(mas, i, i1 - 1);
    quickSort(mas, i1 + 1, j);
}
```