Лабораторна робота № 5

ОБРОБКА МАСИВІВ У C#

Мета роботи:

- вивчити протокол оголошення та ініціалізації одновимірних масивів мовою C#;
- закріпити на практиці використання операторів розгалуження і циклу мови C# для обробки масивів.

Короткі теоретичні відомості

1. Оголошення, ініціалізація та використання одновимірних масивів

Масив – це сукупність послідовно розташованих комірок пам'яті, які можуть містити об'єкти однакового типу.

У мові C# масиви є об'єктами класу System.Array, що дозволяє програмісту використовувати широкий спектр властивостей і методів для роботи з масивами.

Для оголошення одновимірного масиву в C# використовується такий синтаксис:

тип даних[] назва масиву;

Приклади:

```
int[] b;
double[] weights;
```

У C# всі масиви є динамічними, тому, у наведених прикладах ідентифікатори `b` та `weight` фактично є *посиланнями* на майбутні масиви. Спроба використати таке посилання до його ініціалізації адресою масиву (створення самого масиву, чи задання його константою, чи присвоєння іншого раніше ініціалізованого масиву, тощо) призводить до помилки.

Масиви у C# створюються за допомогою оператора **new**:

назваМасиву = new типДаних[кількістьЕлементів];

Наприклад:

```
weights = new double[100];
```

(за умови, що раніше вже було його оголошення `double[] weight;`)

```
int[] heights = new int[120];
```

(тут оголошення, виділення й ініціалізація поєднані в один оператор)

```
int[] widths = new int[n];
```

(причому, `n` цілком може бути прочитаним з клавіатури чи обчисленим, використовуючи прочитані дані; це відрізняє масиви мови C# від масивів мови Pascal чи масивів старих версій мов C/C++, де конкретні розміри масивів потрібно було задавати на етапі написання програми).

При такому створенні масиву його елементи автоматично ініціалізуються нульовими значеннями (`0`, чи `0.0`, чи `'\0'`, чи `""`, чи `null` залежно від типу елементів).

При бажанні, можна поєднати створення масиву з ініціалізацією значень його елементів.

Наприклад:

```
float[] s1 = new float[4] { 0.1f, 0.2f, 0.4f, 0.8f };
float[] s2 = new float[] { 0.1f, 0.2f, 0.4f, 0.8f };
float[] s3 = { 0.1f, 0.2f, 0.4f, 0.8f };
```

Тобто, кількість елементів при цьому можна вказати (причому тоді її слід вказати правильно; якщо написане у квадратних дужках число і фактична кількість елементів у переліку в фігурних дужках відрізнятимуться, програма не скомпілюється), або не вказувати (її все одно можна взнати за фактичною кількістю елементів у переліку в фігурних дужках), або взагалі пропустити частину зі словом `new`, типом і кількістю елементів.

Індексація (нумерація) елементів масивів у C# починається *з нуля*, і змінити це неможливо. Тобто, `n`-елементний масив має елементи з індексами (номерами) 0, 1, …, `n`–1, але не має елемента з індексом (номером) `n`. В тих рідкісних випадках, коли справді дуже потрібно працювати з діапазоном індексів, наприклад, від 1 до `n`, можна, наприклад, створити масив розміром `n+1`, щоб усі комірки з індексами (номерами) від 1 до `n` існували; але при цьому крім них все одно існуватиме ще й комірка з індексом (номером) 0. І таке створення масиву із зайвим елементом — аварійний прийом для якихось рідкісних ситуацій, а не рекомендована практика. Хоча б тому, що в масива є чимало корисних бібліотечних методів (знаходження мінімуму/максимуму, сортування, тощо), й усі ці методи будуть враховувати елемент з індексом (номером) 0.

Враховуючи, що масиви мови C#, як правило, створюються рівно на потрібну кількість елементів, зазвичай вважається поганим стилем пам'ятати кількість елементів масиву в окремій змінній; більш правильним зазвичай вважається взнавати цю кількість безпосередньо у масива, через властивість `Length` (це властивість, а не метод, тому дужок не треба і не можна). Чому? Бо «масив сам про себе краще знає, скільки в ньому елементів, чим якась там окрема змінна». Це стає особливо важливим, коли програма працює одночасно з багатьма різними масивами, які мають різні кількості елементів, або коли під час виконання програми масив міг колись перестворитися з іншим розміром (конкретно в цій лабораторній жодної з цих ситуацій не виникає, але все ж вимагається користуватися властивістю `Length`, бо це кращий стиль, чим використання для цієї мети окремої змінної).

Для обробки масивів, як правило, використовуються цикли з лічильником-індексом, реалізовані на базі оператора `for`. Враховуючи вищесказане, найтиповішим способом поелементної обробки масиву є

```
for(int i=0; i < arr.Length; i++)
// у тілі цикла щось робимо з елементом arr[i]
```

Якщо є необхідність проходити по масиву у зворотньому напрямку, цикл перетворюється до вигляду

```
for(int i = arr.Length - 1; i >= 0; i--)
// у тілі цикла щось робимо з елементом arr[i]
```

Приклад:

```
int[] myArray = new int[100];
Random rndGen = new Random ();
for(int i = 0; i < myArray.Length; i++)
    myArray[i] = rndGen.Next(10);
```

При спробі звернутись до елемента масиву за індексом, що виходить за допустимі межі, виникає виключення **IndexOutOfRangeException** ("Індекс поза межами масиву").

Якщо необхідно послідовно отримати доступ до *значення* кожного елемента масиву (а індекси-номери при цьому не потрібні), можна використати оператор **`foreach`**.

Приклад:

```
double[] x;
x = new double[] {0.1, 0.2, 0.3};
foreach(double v in x)
{
    Console.Write(v + " ");
}
```

Однак, «можна» тут слід розуміти с*а*ме як «можна» (а не як «треба»). Також слід розуміти, що при доступі через **`foreach`** значення можна лише «дивитися», а не змінювати (доступ виходить read-only).

Для масивів, як і для рядків, є багато готових бібліотечних корисних властивостей та методів. Причому, частина цих методів пишуться через крапку після назви масиву (наприклад, `arr.Min()` знаходить мінімальне значення серед елементів конкретно того масиву, який називається `arr`), а частина є статичними класу `Array`, тобто для виклику таких методів треба писати «`Array.`» (без лапок, разом із крапкою), потім назву методу, потім у дужках, як аргумент цього методу, назву масиву (і, можливо, ще якісь аргументи). Наприклад, бібліотечний спосіб сортування масива `arr` за зростанням можна викликати як `Array.Sort(arr);`.

Однак, **у рамках цієї лабораторної роботи, ставиться вимога виконати свій варіант, не користуючись ніякими властивостями чи методами масиву, крім єдино лише властивості `.Length`.** (Само собою, квадратними дужками для звернень до окремих елементів масиву користуватися теж можна і треба.)

Водночас, хто зробить свій варіант двома способами — окремо згідно попереднього абзацу, окремо з максимальним використанням бібліотечних метолів роботи з масивами скрізь, де їхнє використання хоч щось робить простішим чи красивішим — бали будуть підвищені. Вивчити детальнішу інформацію про ці бібліотечні методи пропонується самостійно; джерелом інформації про ці бібліотечні методи можуть бути, наприклад, https://learn.microsoft.com/en-us/dotnet/api/system.array.length (шукати перелік методів слід у вертикальному меню ліворуч), або будь-які джерела на Ваш вибір.

У рамках цієї лабораторної роботи, в усіх варіантах, передбачається:

1. слід запитати в користувача, чи він бажає заповнити масив випадковим чином, чи вводити також і значення елементів; якщо користувач скаже, що бажає вводити також і значення — спитати, чи бажає вводити кожен елемент у окремому рядку, чи всі разом одним рядком через пробіли та/або табуляції.
    a. Якщо користувач скаже, що випадково, то спитати лише кількість елементів, потім створити і заповнити масив автоматично; в цьому разі, створений масив слід також вивести, бо користувач його не вводив і не знає, що в ньому;
    b. Якщо користувач скаже, що вручну і в окремих рядках, то спитати кількість елементів, створити масив, потім відповідну кількість разів читати число в окремому рядку й класти його у відповідний елемент масиву;
    c. Якщо користувач скаже, що вручну і в одному рядку, то, не питаючи кількість елементів, розбити введений користувачем рядок методом `Split()` (він сам визначить кількість елементів, див. також лабораторну про рядки) і перетворити кожен елемент окремо з рядкового подання у числове.
2. Для отримання повних балів, слід написати й використати **власні функції (методи)** — для вибору способу заповнення/введення масиву, для кожного зі способів заповнення/введення масиву, для виведення масиву, для виконання власне того завдання, яке поставлене у варіанті (якщо його за смислом доречно розбити на кілька функцій (методів), то розбити) — **а не писати все всередині `Main`**. Втім, на менші (*істотно*

менші!) бали, допускається й написати все всередині `Main`.

3. Слід обов'язково виконати завдання, не користуючись ніякими бібліотечними методами роботи з масивами (крім властивості `Length` та квадратних дужок), і, за бажанням, на додаткові бали, можна зробити також іншу версію, де, навпаки, активно використовуються бібліотечні методи роботи з масивами (але вони повинні бути доречними).
4. Малювати блок-схеми ***не*** обов'язково. При бажанні, заради збільшення балів, можна, але лише до тієї версії, де «реалізувати суть самостійно, користуючись лише властивістю `Length` та квадратними дужками». Блоксхеми слід або робити зразу або не робити взагалі, а дороблювати їх потім у цій лабораторній не можна. Щодо блоксхем не буде довгої мороки «виправте те», «виправте се»; просто бали за зроблене згідно правил будуть одними, а за зроблене всупереч правилам — значно нижчими, без права перездачі.
5. Тим студентам, хто не захищатиме усно, а лише здаватиме в гуглклас, **необхідно** викласти програму (програми) на ideone.com і перевірити, чи запуска(є/ю)ться вон(а/и) там. Якщо ні, це означає, що програма може не запуститься також і у викладача, а в цьому випадку взагалі нічого не зараховано. (Для складової «з максимальним використанням бібліотечних методів» можлива ситуація, коли на ideone.com не компілюється те, що компілюється у Вас; подолати це дозволяється **лише** шляхом усного захисту (показуйте, що у Вас воно робить що треба, й пояснюйте, чому) й ніяк інакше.)

Варіанти завдань

(У варіантах 1–5, передбачити опрацювання ситуації відсутності від'ємних (у варіанті 5, також відсутність додатних) елементів. В такому разі слід вивести відповідне повідомлення й більш нічого не робити. Водночас, варіанти 6, 7, 14 слід виконати до кінця незалежно від наявності чи відсутності від'ємних/додатних чисел.

У варіантах 6, 7, 14, «два максимальних» чи «два мінімальних» означають таке: знайти максимальне / мінімальне серед усіх елементів масиву, а потім максимальне / мінімальне серед решти, тобто не рахуючи конкретно того елемента; при цьому можлива як ситуація, що ці «два максимальних» чи «два мінімальних» будуть рівні між собою (якщо в масиві якраз є різні елементи з однаковим максимальним / мінімальним значенням), так і ситуація, що вони не рівні між собою (якщо лише один елемент масиву має максимальне / мінімальне значення). Дозволяється шукати «два максимальних» чи «два мінімальних» за два окремі проходження по масиву (перше знаходить справжній максимальний/мінімальний елемент звичайним чином, друге — шукає максимальний/мінімальний елемент лише серед решти, пропускаючи знайдений при першому проходженні); однак, це не єдиний і не найкращий спосіб, можна все це зробити і за одне проходження).

1. Знайти добуток елементів масиву, які розміщені після останнього від'ємного числа.
2. Знайти середнє арифметичне елементів масиву, які розміщені після останнього від'ємного числа.
3. Замінити всі елементи, які розміщені перед першим від'ємним числом, на число 2.
4. Замінити всі елементи, які розміщені після останнього від'ємного числа, на число 10.
5. Знайти середнє арифметичне елементів масиву, які розміщені між першим додатним та останнім від'ємним числом.
6. Замінити всі від'ємні числа на їхні квадрати, після чого знайти серед усіх елементів масиву два максимальних.
7. Замінити всі від'ємні числа на їхні модулі, після чого знайти серед усіх елементів масиву два мінімальних.
8. Знайти добуток елементів масиву, які розміщені після останнього входження мінімального числа.
9. Знайти суму елементів масиву, які розміщені перед останнім входженням мінімального числа.
10. Знайти середнє арифметичне елементів масиву, які розміщені після останнього входження максимального числа.

11. Знайти добуток елементів масиву, які розміщені перед останнім входженням максимального числа.
12. Знайти середнє арифметичне елементів масиву, які розміщені між останніми входженнями максимального та мінімального чисел.
13. Знайти суму елементів масиву, які розміщені між першими входженнями максимального та мінімального чисел.
14. Замінити всі додатні числа на їхні квадрати і після цього знайти два максимальних значення.

Бонусні задачі:

15. https://new.netoi.org.ua/index_ua.php?cid=206 (задача Table з netoi.org.ua) (Сам цей сайт мовою C# не перевіряє, тому дозволяється, на Ваш вибір, чи то не перевіряти розв'язок на сайті, чи то написати однією з мов, які сайт підтримує, й перевірити.) Цю задачу можна й бажано розв'язати ***без*** сортування; ще точніше кажучи, розв'язок без сортування оцінюватиметься вище, чим розв'язок із сортуванням, а обидва розв'язки (без сортування та із ним) — ще вище.
16. Задача A «Гра "Вгадай число"» зі змагання «53 Дорішування теми "Бінарний та тернарний пошуки" Школи Бобра (23.10.2016)» сайту far-bit-re.univer.ck.ua; цей сайт вміє перевіряти мовою C#, тому потрібно добитися, щоб було зараховано сайтом; зверніть увагу, що **інтерактивна** взаємодія Вашої програми з системою автоматичної перевірки — дуже примхлива до помилок, тому дуже важливо зробити абсолютно відповідно з умовою.
17. Задача B «Пошук елементів у масиві–1» зі змагання «53 Дорішування теми "Бінарний та тернарний пошуки" Школи Бобра (23.10.2016)» сайту far-bit-re.univer.ck.ua; цей сайт вміє перевіряти мовою C#, тому потрібно добитися, щоб було зараховано сайтом.
18. Задача C «Пошук елементів у масиві–2» зі змагання «53 Дорішування теми "Бінарний та тернарний пошуки" Школи Бобра (23.10.2016)» сайту far-bit-re.univer.ck.ua; цей сайт вміє перевіряти мовою C#, тому потрібно добитися, щоб було зараховано сайтом.
19. Задача A «Операції над множинами» зі змагання «61 День Іллі Порубльова "Школи Бобра" (15.10.2017, злиття та два вказівники)» сайту far-bit-re.univer.ck.ua; цей сайт вміє перевіряти мовою C#, тому потрібно добитися, щоб було зараховано сайтом. Цю задачу можна й бажано розв'язати ***без*** сортування; ще точніше кажучи, розв'язок без сортування оцінюватиметься вище, чим розв'язок із сортуванням, а обидва розв'язки (без сортування та із ним) — ще вище.
20. Задача B «Всюдисущі числа» зі змагання «61 День Іллі Порубльова "Школи Бобра" (15.10.2017, злиття та два вказівники)» сайту far-bit-re.univer.ck.ua; цей сайт вміє перевіряти мовою C#, тому потрібно добитися, щоб було зараховано сайтом; конкретно за цю задачу в принципі можна отримати більші бали (у предмет, а не оцінені єджаджем), якщо один раз розв'язати її із застосуванням злиття, інший раз — із застосуванням бінарного пощуку.
21. Задача C «Школярі з хмарочосів» зі змагання «61 День Іллі Порубльова "Школи Бобра" (15.10.2017, злиття та два вказівники)» сайту far-bit-re.univer.ck.ua; цей сайт вміє перевіряти мовою C#, тому потрібно добитися, щоб було зараховано сайтом; конкретно за цю задачу в принципі можна отримати більші бали (у предмет, а не оцінені єджаджем), якщо один раз розв'язати її із застосуванням злиття, інший раз — із застосуванням сортування.

Задачі на бонусні бали нікому конкретно не призначені, їх може робити будь-хто зі студентів, хто має таке бажання і ***вчасно*** **захистив (не лише здав у гуглклас, а й захистив) свій варіант**.