Лабораторна робота №2

Тема: Умовні оператори, цикли та масиви у C#.

Мета: Вивчити особливості обробки розгалужень, циклів та масивів засобами ООП.

Завдання: Здобути навички створення, відлагодження та тестування проектів обробки розгалужень, циклів та масивів засобами ООП мовою ***C#***.

(max: 5 балів)

Теоретичні відомості:

1. Умовні оператори і цикли

Умовний оператор if використовується для розгалуження процесу обчислень на два напрями. Формат оператора

```
if (логічна умова) оператор1
   [else оператор2]
```

Спочатку обчислюється логічна умова. Якщо вона має значення **true**, виконується перший оператор, інакше — другий. Після цього управління передається на оператор, який слідує за умовним оператором **if**. Гілка **else** може бути відсутньою.

Якщо в якій-небудь гілці потрібно виконати декілька операторів, їх необхідно укласти в блок. Блок може містити будь-які оператори, у тому числі описи інших умовних операторів, але не може складатися з одних описів.

Приклад 1.

```
if (a>b)
Console.WriteLine("a>b");
else
Console.WriteLine("a<=b");
```

Приклад 2.

```
if (a==b)  Console.WriteLine("a=b");
```

Логічна умова може бути складною, наприклад:

```
if ( a < b && ( a > d || a == 0 ) ) b++; else { b *= a; a = 0; }
```

Оператори **if** можуть бути вкладеними один в одний.

Вкладені оператори if

```
   if(умова_1) {оператор_1}
else if(умова_2) {оператор_2}
                ...
else if(умова_K){ оператор_K}
     else {оператор_N}
```

Наприклад:

```
if ( a < b ) if ( a < c ) m = a; else m = c;
else         if ( b < c ) m = b; else m = c;
```

Приклад 3. Створимо консольний проект для організації діалогу таким чином, щоб реалізувати наведений нижче алгоритм.

1. Вивести на консоль запрошення для введення імені.
2. Ввести своє ім'я і зберегти в текстовому рядку: string myName.
3. Якщо нічого не введено, то вивести повідомлення про це і завершити роботу.
4. Якщо щось введено, то вивести рядок привітання.
5. Вивести рядок із запитом віку
6. Якщо нічого не введено, то вивести повідомлення про це і завершити роботу.
7. Якщо вік введено, привести число до цілого (int myAge).

8. Якщо число myAge<15, вивести повідомлення "Ви ще не студент. "
9. Якщо число myAge>40, вивести повідомлення "Вчитися ніколи не пізно!"

Код програми може мати вигляд:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleHello1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Введіть ім'я");
            string myName = Console.ReadLine();
            if (myName.Length == 0)
                Console.WriteLine("Ви нічого не ввели, прощавайте");
            else
                Console.WriteLine("Здрастуйте, "+ myName);
            Console.WriteLine("Скільки Вам років?");
            int myAge = int.Parse(Console.ReadLine());
            if (myAge == 0)
               Console.WriteLine("Ви нічого не ввели, прощавайте");
             else
            {
              if (myAge<15)
                 Console.WriteLine("Ви ще не студент");
               else if (myAge>40)
                  Console.WriteLine("Вчитися ніколи не пізно!");
                else
                  Console.WriteLine("Ваш вік" +myAge);
            }
            Console.ReadKey();
        }
    }
}
```

В цьому прикладі використовується декілька операторів розгалуження **if** (останні три вкладені). Зверніть увагу, що у C# в операторі **if** діють ті самі правила синтаксису, що і в С (привіт з алгоритмізації!), якщо в тілі оператора є лише один оператор блоку, то дужки не потрібні, наприклад:

```
if (myName.Length == 0)
    Console.WriteLine("Ви нічого не ввели, прощавайте");
else
    Console.WriteLine("Здрастуйте, "+ myName);
```

Крім цього виконується явне перетворення типів з типу string в тип int з використанням методу Parse і неявне при виведенні віку на консоль.

Оператор switch (перемикач) - альтернатива багатократним вкладеним **if.** Формат оператора:

```
switch (вираз)
{
case константний_вираз_1:
[оператори_1 оператор_переходу_1]
...
case  константний_вираз _K:
[оператори_K оператор_переходу_K]
[default: оператори_N оператор_переходу_N]
}
```

Виконання оператора починається з обчислення виразу. Тип виразу найчастіше цілочисельний (включаючи char) або строковий (string). Потім управління передається першому оператору із списку, поміченому константним виразом, значення якого збіглося з обчисленим. Якщо збігу не сталося, виконуються оператори, розташовані після слова default, а при його відсутності управління передається наступному за switch оператору.

Кожна гілка перемикача повинна закінчуватися явним оператором переходу, а саме одним з операторів *break, goto* або *return*:

- оператор *break* виконує вихід з самого внутрішнього з охоплюючих його операторів switch, for, while і do;
- оператор *goto* виконує перехід на вказану після нього мітку, звичайно це мітка *case* однієї з гілок оператора switch, що пролягають нижче;
- оператор return виконує вихід з функції, в тілі якої він записаний.

Частіше за все використовується оператор *break.*

Розробимо простий консольний калькулятор для виконання арифметичних операцій. Вхідні дані будемо вводити з консолі, результати виводити на консоль. Приклад коду з лекції 3.

Приклад 4. Консольний калькулятор на 4 дії

```
using System;
namespace ConsoleCalculator
{   class Program
    {   static void Main()
        {
            string buf;
            double a, b, res;

            Console.WriteLine( "Введіть перший операнд:" );
            a = double.Parse( Console.ReadLine() );
             Console.WriteLine( "Введіть знак операції" );
            char op = (char)Console.Read();
            Console.ReadLine();

            Console.WriteLine( "Введіть другий операнд:" );
            b = double.Parse( Console.ReadLine() );
            bool ok = true;
            switch (op)
            {
                case '+' : res = a + b; break;
                case '-' : res = a - b; break;
                case '*' : res = a * b; break;
                case '/' : res = a / b; break;
                default  : res = double.NaN; ok = false; break;
            }
            if (ok) Console.WriteLine( "Результат: " + res );
            else    Console.WriteLine( "Неприпустима операція" );
        }
    }
}
```

Зверніть увагу, що тип виразу в операторі switch є char. Також зверніть увагу, що перетворення з типу string до типу char повинно бути явним, тому що тип char є типом-значенням, а тип string – посилковим.

```
char op = (char)Console.Read();
```

Для виходу з оператора switch при виконанні умови використовується оператор переходу break.

В C# для організації циклів використовуються чотири типи циклів: for, while, do, **foreach**. Всі цикли крім foreach успадковані від C і мають той самий синтаксис.

Цикл з передумовою while

```
while (умова)
{список операторів}
```

Семантика оператора while: Оператори в тілі циклу виконуються до тип пір, поки умова має значення **true**.
Умова в операторі while перевіряється до входження в цикл. Це гарантує, що цикл не буде виконуватися, якщо умова зразу має значення **false**.

Приклад 5. Напишемо програму, яка виводить для аргументу x, що змінюється в заданих границях із заданим кроком, таблицю значень наступної функції:

$$\mathbf{X} = \begin{pmatrix} x, x < 0 \\ tx, o \leqslant x < 10 \\ 2t, x \geqslant 10 \end{pmatrix}$$

Назвемо початкове значення аргументу Xn, кінцеве значення аргументу — Xk, крок зміни аргументу — dX і параметр t. Всі величини дійсні числа (double). Програма повинна виводити таблицю, що складається з двох стовпців: значень аргументу і відповідних ним значень функції.

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            double Xn = -2, Xk = 12, dX = 2, t = 2, y;
            Console.WriteLine("|    x     |     y    |"); // заголовок
таблиці

            double x = Xn;
            while (x <= Xk)
            {
                y = t;
                if (x >= 0 && x < 10) y = t * x;
                if (x >= 10) y = 2 * t;
                Console.WriteLine("| {0,6} | {1,6} |", x, y);
                x += dX;
            }
            Console.ReadKey();
        }
    }
}
```

Зверніть увагу, що в тілі циклу використовується два оператори if. На кожному кроці циклу обчислюється значення функції. Цикл завершиться коли умова циклу не буде виконана (тобто значення x стане більше 12).

Цикл з пост-умовою do…while

```
do
{  список операторів }
while(умова);
```

Семантика циклу do…while: оператори в тілі циклу виконуються до тип пір, поки умова має значення **true**.
Умова в операторі *do …while* перевіряється після ітерації циклу. В цьому циклі завжди виконується хоча б одна ітерація.
Цей тип циклу застосовується в тих випадках, коли тіло циклу необхідно обов'язково виконати хоч б один раз.

Приклад 6. В цьому прикладі на консоль виводиться текст "Будете вчитися?" до тих пір поки не буду введено"y".

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            char answer;
            do
            {
                Console.WriteLine("Будете вчитися?");
                answer = (char)Console.Read();
                Console.ReadLine();
            } while (answer != 'y');

        }
    }
}
```

Цикл з параметром for

Синтаксис оператора:

```
for (ініціалізатор; умова; список_виразів)
                { оператори}
```

Семантика оператора: ініціалізація використовується для оголошення змінних, які використовуються в циклі, і призначення їм початкових значень. У цій частині можна записати декілька операторів, розділених комою, наприклад:

```
for ( int i = 0, j = 20; ...
int k, m;
for (k = 1, m = 0; ...
```

Областю дії змінних, оголошених в частині ініціалізації циклу, є цикл. Ініціалізація виконується один раз на початку виконання циклу.

Вираз типу bool визначає умову виконання циклу: якщо його результат рівний **true**, цикл виконується.

Модифікації виконуються після кожної ітерації циклу і служать для зміни параметрів циклу. У частині модифікацій можна записати декілька операторів через кому, наприклад:

```
for ( int i = 0, j = 20; i < 5 && j > 10; i++, j-- ) ...
```

Простий або складений оператор є тілом циклу. Будь-яка з частин оператора for може бути опущена (але крапки з комою треба залишити на своїх місцях!).

Приклад 7. Написати метод обчислення функції sin(x), використовуючи розкладання в ряд Тейлора за формулою:

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

При реалізації задачі використати цикл **for**.

Число x – це значення кута в радіанах, n – кількість членів ряду. Числа **x, n** ввести з консолі. Обчислення факторіала виконати в окремому методі. Порівняти отримане значення із стандартним методом обчислення sin(x).

В цьому прикладі для обчислення ступеня використовується метод Math.Pow(). Статичний клас Math містить методи реалізації математичних функцій (дивіться розділ 8).

Код програми може бути таким:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace My_sin
{
    class Program
    {
        static double Calc_sin(double x, int n)
        {
            //обчислення розкладання sin в ряд
            double result =0;
            for (int i = 0; i < n; i++)
            {
             result=result+(Math.Pow((-1),i)*Math.Pow(x,(2*i+1)))/F(2*i+1);
            }
             return result;
        }
        static double F(int n)
        {
            double tmp = 1;
            for (int i = 1; i <= n; i++)
            {
                tmp = tmp * i;
            }

            return tmp;
        }
        static void Main(string[] args)
        {
            Console.WriteLine("Введіть x - кут в радіанах");
            double x = double.Parse(Console.ReadLine());
            Console.WriteLine("Введіть показник ступеня n");
            int n = int.Parse(Console.ReadLine());
          //виклик методу обчислення sin(x) через ряд
            double my_sinus = Calc_sin(x,n);
            //виклик методу з класу Math
            double sinus = Math.Sin(x);
            double delta = sinus - my_sinus;
            Console.WriteLine("my_sinus= {0},sin={1},delta={2}", my_sinus,
sinus, delta);
            Console.ReadKey();
        }
    }
}
```

2. Робота з масивами. Одновимірні масиви

Масив — це обмежена сукупність однотипних величин. Елементи масиву мають одне і те саме ім'я, а розрізняються за порядковим номером (індексом).

У C# масиви відносяться до *посилкових* типів, тобто розташовуються в динамічної області пам'яті (купі), тому створення масиву починається з виділення пам'яті під його елементи. Елементами масиву можуть бути величини як значимих, так і посилкових типів (у тому числі масиви). Масив значимих типів зберігає значення, масив посилкових типів — посилання на елементи. Всім елементам при створенні масиву призначається значення за замовчанням: нулі для значимих типів і null для посилкових.

В C# є три види масивів:

• одновимірні масиви;

• багатовимірні масиви.

• "зубчасті" (ступінчасті, jagged) масиви.

Нумерація елементів масиву починається з нуля!

Варіанти опису масиву:

```
тип[] ім'я;
тип[] ім'я = new тип [ розмірність ];
тип[] ім'я = { список_ініціалізаторів };
```

```
тип[] ім'я = new тип [] { список_ініціалізаторів };
тип[] ім'я = new тип [ розмірність ] { список_ініціалізаторів };
```

Приклади описів (один приклад на кожен варіант опису):

```
int[] a;                                          // 1   елементів немає
int[] b = new int[4];                             // 2   елементи рівні 0
int[] c = {61, 2, 5, -9 };                   // 3   new мається на увазі
int[] d = new int[] { 61, 2, 5, -9 };   // 4   розмірність обчислюється
int[] e = new int[4] { 61, 2, 5, -9 };      // 5   надлишковий опис
```

Тут описано п'ять масивів. Відмінність першого оператора від останніх полягає в тому, що в ньому фактично описано лише посилання на масив, а пам'ять під елементи масиву не виділена.

У кожному з решти масивів по чотири елементи цілого типа. Як видно з операторів 3–5, масив при описі можна ініціалізувати. Якщо при цьому не задана розмірність (оператор 3), кількість елементів обчислюється з кількості вказаних значень. Для полів об'єктів і локальних змінних можна опускати операцію new, вона буде виконана за замовчанням (оператор 2). Якщо присутня і розмірність, і список ініціалізаторів, розмірність має бути константою (оператор 4).

Як приклад розглянемо програму, яка визначає суму і кількість від'ємних елементів, а також максимальний елемент масиву, що складається з 6 цілочисельних елементів.

Приклад 8.

```
using System;
namespace ConsoleApplication1
{    class Class1
    {
        static void Main()
        {
            const int n = 6;
            int[] a = new int[n] { 3, 12, 5, -9, 8, -4 };
            Console.WriteLine( "Початковий масив:" );
            for ( int i = 0; i < n; ++i )
                Console.Write( "\t" + a[i] );
            Console.WriteLine();
            long sum = 0;              // сума від'ємних елементів
            int  num = 0;              // кількість від'ємних елементів
            for ( int i = 0; i < n; ++i )
                if ( a[i] < 0 )
                {
                    sum += a[i];
                    ++num;
                }
            Console.WriteLine( "Сума  від'ємних = " + sum );
            Console.WriteLine( "Кількість від'ємних = " + num );
            int max = a[0];            // максимальний елемент
            for ( int i = 1; i < n; ++i )
                if ( a[i] > max ) max = a[i];
            Console.WriteLine( "Максимальний елемент = " + max );
        }
    }
}
```

3. Багатовимірні масиви

Багатовимірний масив має більше одного виміру. Найчастіше в програмах використовуються двовимірні масиви (матриці). Варіанти опису двовимірного масиву:

```
тип[,] ім'я;
тип[,] ім'я = new тип [ разм_1, разм_2 ];
тип[,] ім'я = { список_ініціалізаторів };
тип[,] ім'я = new тип [,] { список_ініціалізаторів };
тип[,] ім'я = new тип [ разм_1, разм_2 ] { список_ініціалізаторів };
```

Приклади описів (один приклад на кожен варіант опису):

```
int[,] a;                                          // 1  елементів немає
int[,] b = new int[2, 3];                         // 2  елементи рівні 0
int[,] c = {{1, 2, 3}, {4, 5, 6}};             // 3  new мається на увазі
int[,] c = new int[,] {{1, 2, 3} {4, 5, 6}};   //  4 розмірність
обчислюється
int[,] d = new int[2,3] {{1, 2, 3} {4, 5, 6}}; // 5  надлишковий опис
```

До елементу двовимірного масиву звертаються, вказуючи номери рядка і стовпця, на перетині яких він розташований, наприклад:

```
a[1, 4]        b[i, j]        b[j, i]
```

Увага

Необхідно пам'ятати, що компілятор сприймає у якості номера рядка перший індекс, як би він не був позначений в програмі.

Як приклад розглянемо програму, яка для цілочисельної матриці розміром 3x4 визначає середнє арифметичне її елементів і кількість позитивних елементів в кожному рядку.

Для знаходження середнього арифметичного елементів масиву потрібно знайти їх загальну суму, після чого розділити її на кількість елементів. Порядок перебору елементів масиву (по рядках або по стовпцях) ролі не грає. Знаходження кількості позитивних елементів кожного рядка вимагає перегляду матриці по рядках.

Приклад 9

```
using System;
namespace ConsoleApplication1
{    class Class1
    {
        static void Main()
        {
            const int m = 3, n = 4;
            int[,] a = new int[m, n] {
                { 2,-2, 8, 9 },
                {-4,-5, 6,-2 },
                { 7, 0, 1, 1 }
             };
            Console.WriteLine( "Початковий масив:" );
            for ( int i = 0; i < m; ++i )
            {
                for ( int j = 0; j < n; ++j )
                    Console.Write( "\t" + a[i, j] );
                Console.WriteLine();
            }
            double sum = 0;
            int nPosEl;
            for ( int i = 0; i < m; ++i )
            {
                nPosEl = 0;
                for ( int j = 0; j < n; ++j )
                {
                    sum += a[i, j];
                    if ( a[i, j] > 0 ) ++nPosEl;
                }
                Console.WriteLine( "У рядку {0} {1} додатніх
елементів",
                    i, nPosEl  );
            }
            Console.WriteLine( "Середнє арифметичне всіх елементів: "
                + sum / m / n );
        }
```

```
        }
    }
```

4. Ступінчасті (вільні) масиви

У ступінчастих масивах кількість елементів в різних рядках може розрізнятися. У пам'яті ступінчастий масив зберігається інакше, ніж прямокутний: у вигляді декількох внутрішніх масивів, кожен з яких має свій розмір. Крім того, виділяється окрема ділянка пам'яті для зберігання посилань на кожен з внутрішніх масивів (рис. 1).

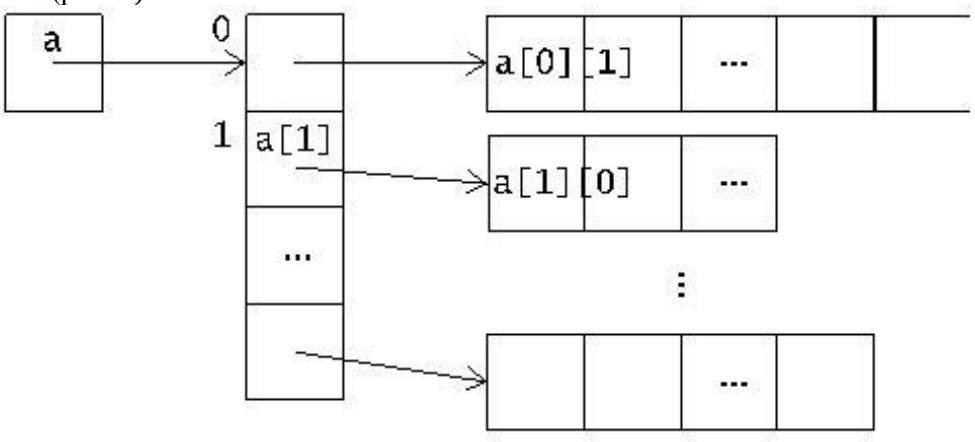

Рис. 1. Ступінчастий (вільний) масив

5. Клас System.Array

Всі масиви наслідуються від базового класу System.Array.

Таблиця 1. Основні члени (властивості і методи) класу System.Array

Елемент	Вид	Опис
Length	Властивість	Повертає кількість елементів масиву (по всій розмірності)
Rank	Властивість	Повертає кількість вимірювань масиву
BinarySearch	Статичний метод	Двійковий пошук у відсортованому масиві
Clear	Статичний метод	Призначення елементам масиву значень за замовчанням
Copy	Статичний метод	Копіювання заданого діапазону елементів одного масиву в інший
GetValue	Метод	Повертає значення елементу масиву
GetLength	Метод	Повертає розмір i-го вимірювання
IndexOf	Статичний метод	Пошук першого входження елементу в одновимірний масив
Reverse	Статичний метод	Зміна порядку слідування елементів на зворотний
Sort	Статичний метод	Сортування елементів одновимірного масиву

Приклад 10. В цьому прикладі створюється масив цілих чисел *a*. В циклі foreach на консоль виводиться кожний елемент масиву. Далі створюється ціла змінна lArray, якій призначається кількість елементів масиву, отримана за допомогою властивості масиву Length. Після цього створюється ціла змінна rArray, якій призначається розмірність масиву, отримана за допомогою властивості Rank. Обчислені значення виводяться на консоль.

```
Namespace ConsoleArray1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] a = { 10, 20, 30, 40 };
            foreach (int i in a) { System.Console.WriteLine(i); }
            int lArray = a.Length;
            int rArray = a.Rank;
            Console.WriteLine("кількість елементів ={0},розмірність = {1}",
lArray,rArray);
            Console.ReadKey();
        }
    }
}
```

Приклад 11. Сортування масиву чисел. Використовується стандартний метод *Sort* класу Array. Вхідні дані призначаються при створенні масиву.

```
static void Main(string[] args)
        {
         // визначення масиву і його ініціалізація
         int[] Data = {9, 3, 7, 5, 6, 4, 8, 1};
         // сортування значень масиву
         Array.Sort(Data);
         // друк відсортованих даних
         Console.WriteLine("Друк відсортованих даних");
         for (int i=0; i<Data.Length; i++)
         Console.WriteLine("Data["+i+"] = " + Data[i]);
         Console.ReadLine();
        }
```

6. Оператор foreach

Оператор foreach використовується для перебору елементів в спеціальним чином організованій групі даних, яка називається *колекцією*. Масив є саме такою групою (*колекцією)*. Зручність цього вигляду циклу полягає в тому, що нам не потрібно визначати кількість елементів в групі і виконувати їх перебір за індексом: ми просто вказуємо на необхідність перебрати всі елементи групи. Синтаксис оператора:

foreach (тип ім'я in вираз) тіло_циклу

Ім'я задає локальну по відношенню до циклу змінну, яка по черзі набуватиме всіх значень з масиву виразу (у якості виразу найчастіше застосовується ім'я масиву або іншої колекції даних). У простому або складеному операторі, що є тілом циклу, виконуються дії зі змінною циклу. Тип змінної повинен відповідати типу елементу масиву.

Наприклад, нехай заданий масив:

```
int[] a =  { 24, 50, 18, 3, 16, -7, 9, -1 };
```

Виведення цього масиву на екран за допомогою оператора foreach виглядає таким чином:

```
foreach ( int x in a ) Console.WriteLine( x );
```

Цей оператор виконується так: на кожному проході циклу черговий елемент масиву призначається змінній x і з нею виконуються дії, записані в тілі циклу.

У наступному прикладі 5 вирішується те ж завдання, що і в прикладі 1, але з використанням циклу foreach.

Приклад 12. Робота з одновимірним масивом з використанням циклу foreach

```
using System;
namespace ConsoleApplication1
{   class Class1
    {   static void Main()
        {
            int[] a = { 3, 12, 5, -9, 8, -4 };
            Console.WriteLine( "Початковий  масив:" );
            foreach ( int elem in a )
                Console.Write( "\t" + elem );
            Console.WriteLine();

            long sum = 0;               // сума від'ємних елементів
            int  num = 0;               // кількість від'ємних елементів
            foreach ( int elem in a )
                if ( elem < 0 )
                {
                    sum += elem;
                    ++num;
                }
            Console.WriteLine( "sum = " + sum );
            Console.WriteLine( "num = " + num );

            int max = a[0];             // максимальний елемент
            foreach ( int elem in a )
                if ( elem > max ) max = elem;

            Console.WriteLine( "max = " + max );
        }
    }
}
```

Увага

Обмеженням оператора foreach у C# є те, що з його допомогою можна лише переглядати значення в групі даних, але не змінювати їх.

Приклад 13. Робота із ступінчастим масивом з використанням циклу foreach

```
...       int[][] a = new int[3][];
          a[0]= new int [5] { 24, 50, 18, 3, 16 };
          a[1]= new int [3] { 7, 9, -1 };
          a[2]= new int [4] { 6, 15, 3, 1 };
          Console.WriteLine( "Початковий масив:" );
foreach ( int [] mas1 in a )
            {
                foreach ( int x in mas1 )
                       Console.Write( "\t" + x );
                Console.WriteLine();
            }
```

7. Клас Random і його методи

Клас *Random* призначений для генерації випадкових чисел. Це *нестатичний* клас. Він має конструктор класу: для того, щоб викликати методи класу, потрібно спочатку створити екземпляр класу.

Конструктор класу перевизначений і має дві реалізації. Одна з них дозволяє генерувати унікальні серії випадкових чисел. Початковий елемент такої серії будується на основі поточної дати і часу, що гарантує унікальність серії.

Цей конструктор викликається без параметрів. Він описаний як

```
public Random();
```

Інший конструктор з параметром –

```
public Random (int);
```

забезпечує можливість генерації серії випадкових чисел, що повторюється. Параметр конструктора використовується для побудови початкового елемента серії, тому при завданні одного і того ж значення параметра серія буде повторюватися.

Перегружений метод `public int Next();` при кожному виклику повертає позитивне ціле число, рівномірно розподілене в деякому діапазоні. Діапазон задається параметрами методу. Три реалізації методу відрізняються набором параметрів:

- public int Next () - метод без параметрів видає цілі позитивні числа у всьому позитивному діапазоні типу int;
- public int Next (int max) - видає цілі позитивні числа в діапазоні [0,max];
- public int Next (int min, int max) - видає цілі позитивні числа в діапазоні [min,max].

Метод public double NextDouble () має одну реалізацію. При кожному виклику цього методу видається нове випадкове число, рівномірно розподілене в інтервалі [0,1).

Для отримання серії випадкових чисел призначений метод `NextBytes`, описаний як:

```
public void NextBytes (byte[] buffer);
Метод має 1 параметр – масив, який заповняється випадковими числами в діапазоні [0, 255]..
```

Приклад 14

```
public void Rand()
{
   const int  initRnd = 77;
   Random realRnd = new Random();
   Random repeatRnd = new Random(initRnd);
   // випадкові числа в діапазоні [0,1)
   Console.WriteLine("випадкові числа в діапазоні[0,1)");
   for(int i =1; i <= 5; i++)
   {
      Console.WriteLine("Число " + i + "= "
         + realRnd.NextDouble() );
   }
   // випадкові числа в діапазоні[min,max]
   int min = -100, max=-10;
   Console.WriteLine("випадкові числа в діапазоні [" +
      min +"," + max + "]");
   for(int i =1; i <= 5; i++)
   {
      Console.WriteLine("Число " + i + "= "
         + realRnd.Next(min,max) );
   }
   // випадковий масив байтів
   byte[] bar = new byte[10];
   repeatRnd.NextBytes(bar);
   Console.WriteLine("Масив випадкових чисел в діапазоні [0, 255]");
   for(int i =0; i < 10; i++)
   {
      Console.WriteLine("Число " + i + "= " +bar[i]);
   }
}//Rand
```

В цьому прикладі спочатку створюються два об'єкти класу `Random`. В цих об'єктів різні конструктори. Об'єкт з ім'ям realRnd дозволяє генерувати серії випадкових чисел, що не повторюються. Об'єкт repeatRnd дає можливість повторити при необхідності серію. Метод `NextDouble` створює серію випадкових чисел в діапазоні [0, 1). Метод `Next`, що викликається в циклі з двома параметрами створює серію випадкових негативних цілих, рівномірно розподілених в діапазоні [-100, -10]. Метод `NextBytes` об'єкту `repeatRnd` дозволяє отримати при одному виклику масив випадкових чисел з діапазону [0, 255].

Приклад 15: Створення екземпляру класу і генерація числа

```
Random aRand = new Random(); //створення об'єкту класу
arr[i] = aRand.Next(min_Number1, max_Number1); //генерація випадкового числа
у заданому діапазоні.
```

Розширення проекту прикладу 4 – додамо до класу новий статичний метод ValsGenerator для генерації даних за допомогою датчика випадкових чисел і їх сортування

Приклад 16. Генерація масиву чисел і його сортування

```
class Program
{         // генератор даних
        static void ValsGenerator(int[] Vals)
        {
            // Random - клас для генерації випадкових чисел
            Random aRand = new Random();
            // заповнення масиву
            for (int i = 0; i < Vals.Length; i++)
                Vals[i] = aRand.Next(100);
        }
        static void Main(string[] args)
        {
          const int N = 10;
          int[] Data = new int[N];
          ValsGenerator(Data);
          Array.Sort(Data);
          Console.WriteLine("Друк відсортованих даних");
          for (int i = 0; i < Data.Length; i++)
          Console.WriteLine("Data[" + i + "] = " + Data[i]);
          Console.ReadLine();
        }
}
```

8. Клас Math і його методи

Клас Math містить методи для роботи з математичними функціями.

Клас Math – статичний, тобто всі його поля і методи статичні. Це означає, що для використання функцій класу не потрібно створювати об'єкт класу (оператором new).

Таблиця 2. Основні функції класу Math

Функції	Призначення
`Sin, Cos, Tan`	Тригонометричні функції
`ASin, ACos, ATan, ATan2 (sinx, cosx);`	Обернені тригонометричні функції
`Tanh, Sinh, Cosh;`	Гіперболічні
`Exp, Log, Log10`	Експоненціальні і логарифмічні
`Abs, Sqrt, Sign`	Модуль, корінь, знак
`Min, Max, Pow, IEEERemainder.`	Мінімум, максимум, ступінь, залишок

Примітка

Для того, щоб при написанні програми подивитися повний перелік методів класу, достатньо ввести в редакторі Visual Studio ім'я класу і крапку (**Math.**).

Приклад 17. Використання математичних функцій. Постановка задачі: знайти дійсні корені рівняння $6x^4-3x^3+8x^2-5=0$ за методом бісекції (ділення навпіл) на відрізках [0, 1], [-1, 0].

Алгоритм методу:

Нехай [a,b] відрізок, на якому шукаються корені. Припустимо, що функція f(x) неперервна на [a,b] і на кінцях приймає значення різних знаків $f(a)\cdot f(b)<0$.

Алгоритм методу полягає в побудові послідовності вкладених відрізків, на кінцях яких функція приймає значення різних знаків. Кожний наступний відрізок отримують діленням навпіл попереднього. Опишемо

один крок ітераційного методу. Нехай на *к*-ому кроці знайдено відрізок такий, що $f\left(a^{(k)}\right)\cdot f\left(b^{(k)}\right)<0$.

Знайдемо середину відрізку $c^{(k)}=\frac{a^{(k)}+b^{(k)}}{2}$. Якщо, $f\left(c^{(k)}\right)=0$ то $c^{(k)}$ - корінь і задача вирішена. Якщо ні, то з двох половин відрізку вибираємо той, на кінцях якого функція має протилежні знаки:

$a^{(k+1)}=a^{(k)}$, $b^{(k+1)}=c^{(k)}$, якщо $f\left(a^{(k)}\right)\cdot f\left(c^{(k)}\right)<0$

$a^{(k+1)}=c^{(k)}$, $b^{(k+1)}=b^{(k)}$, якщо $f\left(a^{(k)}\right)\cdot f\left(c^{(k)}\right)>0$

Критерій закінчення ітераційного процесу: якщо довжина відрізку знаходження кореня менше 2ε, то ітерації припиняють і за значення кореня із заданою точністю приймають середину відрізку.

Код програми може бути таким:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace bicection
{
    class Program
    {   //Знайти корені нелінійного рівняння
        //6x4-3x3+8x2-5=0
        //x1=0.74213
        //x2=-0.6365
        static double f(double x)
        {
            //рівняння, для якого шукаємо корені
            double y = 6 * Math.Pow(x, 4) - 3 * Math.Pow(x, 3) + 8 * Math.Pow(x, 2) - 5;
            return y;
        }
 // ---------------------------------------------------------------------
        static double bicect(double left, double right)
        {
            //метод бісекцій
            double eps = 0.00001;
            double center = 0;
            while (right - left > eps * 2)
            {
                center = (right - left) / 2 + left;
                if (f(center) * f(left) > 0)
                    left = center;
                else
                    right = center;
            }
            return center;
        }
//  ------------------------------------------------
          static void Main(string[] args)
          {
              //метод бісекцій
              double x1 = bicect(0, 1);
              double x2 = bicect(-1, 0);
              Console.WriteLine("Метод бісекцій");
              Console.WriteLine("x1={0},x2= {1}", x1, x2);
              Console.ReadKey();
          }
    }
}
```

В цій програмі є два методи: static double f(double x), в якому визначається рівняння, і метод static double bicect(double left, double right). Для обчислення ступеня в методі **f** використовується метод Math.Pow( ).

Другий метод static double bicect(double left, double right) реалізує ітераційний процес. В методі Main два рази викликається метод бісекцій для знаходження коренів на вказаних відрізках. Результати виводляться на консоль.

Індивідуальні завдання:

(Для виконання індивідуальних завдань № варінта є порядковим номером прізвища студента в списку групи. Усі проекти та, за наявності, тести до них завантажити у власні репозиторії на Git Hub та виконати їх збірку/build у середовищі Travis CI. Посилання на репозиторії проектів вказати у звіті та обов'язково долучати скріни успішної збірки в Travis CI.)

1. Створити програму згідно свого варіанта, використати конструкцію switch…case

(1 бал)

1. Нехай в 9-поверховому будинку по 5 квартир на поверсі. Написати програму, яка за введеним номером квартири повідомляє номер поверху, на якій вона знаходиться.
2. За введеним номером у списку групи (від 1 до 10) вивести прізвище та ім'я одногрупника.
3. Ввести ціле число з діапазону 10..20. Вивести його назву словом. Наприклад 11 – «одинадцять»
4. Для 9 маршрутів трамваю за веденим їх номером вивести початкоу та кінцеву зупинки.
5. Написати програму, яка для 8 одногрупників виводить їх ім'я за введеним прізвищем.
6. Написати програму, яка за заданим поштовим індексом виводить назву міста, за яким цей індекс закріплений. Наприклад 58000 – «Чернівці». Не менше 10 міст обрати для виконання.
7. Написати програму, яка виводить назву знака зодіака за його порядковим номером.
8. Задано ціле число n, $1 \le n \le 24$, яке вказує порядковий номер місяця в році. За введеним значенням n надрукувати назву відповідної години. Наприклад 1 – «перша година».
9. Дано порядковий номер місяця. В залежності від введеного значення вивести пору року.
10. Написати програму, яка в залежності від порядкового номера дня тижня (1,2,...7) виводить на екран його назву (понеділок,...)
11. Написати програму, яка в залежності від порядкового номера місяця (1,2,...12) виводить на екран його назву (січень,...грудень).
12. Написати програму, яка в залежності від порядкового номера кольору у спектрі (1,2,...7) виводить його назву (червоний, помаранчевий, жовтий, зелений, блакитний, синій, фіолетовий)
13. Дано ціле число n в діапазоні 20 – 69, що визначає вік (в роках). Вивести рядок-опис вказаного віку, забезпечивши правильне співставлення числа зі словом "рік", наприклад: 20 – "20 років", 32 – "32 роки", 41 – "41 рік".
14. Нехай в 9-поверховому будинку по 5 квартир на поверсі. Написати програму, яка за введеним номером квартири повідомляє номер поверху, на якій вона знаходиться.
15. За введеним номером у списку групи (від 1 до 10) вивести прізвище та ім'я одногрупника.
16. Ввести ціле число з діапазону 10..20. Вивести його назву словом. Наприклад 11 – «одинадцять»
17. Для 9 маршрутів трамваю за веденим їх номером вивести початкоу та кінцеву зупинки.
18. Дано натуральне число n (n<=100), яке вказує вік людини. Додати до цього числа відповідно слова: „рік", „роки", „років". Наприклад: 1 рік, 12 років, 94 роки.
19. Створити діалогову програму, яка запитує вік користувача і визначає, до якої вікової категорії він належить:
    1) від 1 до 10 років – дитина;
    2) від 11 до 15 років – підліток;
    3) від 16 до 20 років – юнак;
    4) від 21 до 30 років – молода людина;
    5) після 31 року – доросла людина.
20. Задано ціле число n, $1 \le n \le 12$, яке вказує порядковий номер місяця в році. За введеним значенням n надрукувати назву відповідного місяця.
21. Дано порядковий номер місяця. В залежності від введеного значення вивести пору року.
22. Написати програму, яка в залежності від порядкового номера дня тижня (1,2,...7) виводить на екран його назву (понеділок,...)
23. Написати програму, яка в залежності від порядкового номера місяця (1,2,...12) виводить на екран його назву (січень,...грудень).
24. Написати програму, яка в залежності від порядкового номера кольору у спектрі (1,2,...7) виводить його назву (червоний, помаранчевий, жовтий, зелений, блакитний, синій, фіолетовий)
25. Дано ціле число n в діапазоні 0 – 50, що визначає вік (в роках). Вивести рядок-опис вказаного віку, забезпечивши правильне співставлення числа зі словом "рік", наприклад: 20 – "20 років", 32 – "32 роки", 41 – "41 рік".

2. Використовуючи оператори циклу з передумовою та післяумовою, обчислити і вивести на екран у табличному вигляді значення функції $y = f(x)$ ***на заданому інтервалі зміни значень аргумента x від a до b з кроком*** dx***. За необхідності передбачити виникнення виключних ситуацій (невизначений результат, ділення на нуль, корінь з від'ємного числа, логарифмування числа, меншого за нуль тощо). Результати обчислень подати у вигляді наступної таблиці:***

(1 бал)

Результати обчислення функції $y = f(x)$ *на проміжку* $[a, b]$ *з кроком* dx.

x	***y = f(x)***

Варіанти:

№	Завдання	№	Завдання
1	$y = \frac{\lg(x)}{x}, \quad a = -1, \quad b = 3, \quad dx = 0.05$	14	$y = x^3, \quad a = -6, \quad b = 3, \quad dx = 0.75$
2	$y = \lg(x), \quad a = 1, \quad b = 6, \quad dx = 0.2$	15	$y = x^4 - 12x + e^x, \quad a = -6, \quad b = 3, \quad dx = 0.75$
3	$y = \sqrt[3]{x} + \ln(3x), \; a = 1.25, \; b = 6.75, \; dx = 0.25$	16	$y = \sqrt[4]{x}, \quad a = 1.25, \quad b = 6.75, \quad dx = 0.25$
4	$y = \ln(x), \quad a = 0.5, \quad b = 4.5, \quad dx = 0.4$	17	$y = \frac{\lg(x)}{4x + 13}, \quad a = 0, \quad b = 3, \quad dx = 0.025$
5	$y = e^{\sqrt{2x}} \cdot x^2, \quad a = 3, \quad b = 6, \quad dx = 0.05$	18	$y = 4^{-\cos(x)}, \quad a = 0, \quad b = 3, \quad dx = 0.05$
6	$y = \log_2(x), \quad a = 1, \quad b = 2, \quad dx = 0.025$	19	$y = \frac{x^4 - 12x}{x - 2}, \quad a = -2, \quad b = 12, \quad dx = 0.5$
7	$y = \cos^2(x) + 3\sin(x), \; a = -\frac{\pi}{2}, \; b = \frac{3\pi}{2}, \; dx = \frac{\pi}{3}$	20	$y = tg(x), \quad a = 0, \quad b = \pi, \quad dx = \frac{\pi}{20}$
8	$y = \frac{\sin(x)}{x}, \quad a = -\frac{\pi}{2}, \quad b = \frac{\pi}{2}, \quad dx = \frac{\pi}{30}$	21	$y = x^2 \cdot \cos(x), \; a = -\frac{\pi}{2}, \; b = -\frac{3\pi}{2}, \; dx = \frac{\pi}{10}$
9	$y = \cos(x) \cdot \sin(x), \quad a = -\pi, \quad b = \pi, \quad dx = \frac{\pi}{10}$	22	$y = ctg(x), \quad a = -\frac{\pi}{2}, \quad b = \frac{\pi}{2}, \quad dx = \frac{\pi}{20}$
10	$y = \frac{2 + x^3}{x + \sqrt{13\lvert x \rvert}}, \quad a = -3, \quad b = 3, \quad dx = 0.5$	23	$y = \frac{e^x}{3x - 12}, \quad a = -1, \quad b = 25, \quad dx = 1.25$
11	$y = \sqrt[4]{tg(x) + 13}, \quad a = 0, \quad b = \pi, \quad dx = \frac{\pi}{20}$	24	$y = 5x \cdot \sin(x), \quad a = -\pi, \quad b = \pi, \quad dx = \frac{\pi}{10}$
12	$y = \frac{1}{1 - \sqrt{x}}, \quad a = 0, \quad b = 6, \quad dx = 0.5$	25	$y = 2^{3x} \cdot \sin^2(x), \; a = -3.14, \; b = 3.14, \; dx = 0.314$
13	$y = ctg(x) - 2\sin(x), \; a = -\frac{\pi}{2}, \; b = \frac{\pi}{2}, \; dx = \frac{\pi}{20}$		

3. Створити програми для роз'язування задач згідно свого варіанта засобами ООП. Введення/виведення даних супроводжувати відповідними повідомленнями. Передбачити захист від введення некоректних даних з клавіатури. Виконати завдання двома способами:

А) З клавіатури вводиться одновимірний масив n цілих чисел.

Б) Заповнити масив n цілих чисел значеннями за допомогою генератора псевдовипадкових чисел з відрізка [-100; 100].

(1 бал)

1. Визначити 1) суму елементів масиву з непарними номерами; 2) добуток елементів масиву, розташованих між першим й останнім від'ємними елементами.

2. Визначити 1) максимальний елемент масиву; 2) суму елементів масиву, розташованих між першим й другим додатними елементами.

3. Визначити 1) кількість елементів масиву, менших за число сім; 2) суму елементів масиву, розташованих між першим й останнім додатними елементами.

4. Визначити 1) номер максимального елемента масиву; 2) суму модулів елементів масиву, розташованих між першим й останнім нульовими елементами.

5. Визначити 1) мінімальний елемент масиву; 2) суму елементів масиву, розташованих між першим і другим від'ємними елементами.

6. Визначити 1) кількість від'ємних елементів масиву; 2) суму елементів масиву, розташованих після мінімального за модулем елемента.

7. Визначити 1) максимальний за модулем елемент масиву; 2) суму елементів масиву, розташованих після останнього нульового елемента.

8. Визначити 1) суму модулів від'ємних елементів масиву; 2) добуток елементів масиву, розташованих до останнього від'ємного елемента.

9. Визначити 1) номер максимального за модулем елемента масиву; 2) добуток елементів масиву, розташованих після першого додатного елемента.

10. Визначити 1) добуток ненульових елементів масиву; 2) суму модулів елементів масиву, розташованих після першого від'ємного елемента.

11. Визначити 1) кількість нульових елементів масиву; 2) добуток елементів масиву, розташованих після максимального за модулем елемента.

12. Визначити 1) добуток від'ємних елементів масиву; 2) суму елементів масиву, розташованих після мінімального елемента.

13. Визначити 1) кількість додатних елементів масиву; 2) добуток елементів масиву, розташованих до мінімального за модулем елемента.

14. Визначити 1) добуток елементів масиву з парними номерами; 2) суму елементів масиву, розташованих до максимального за модулем елемента.

15. Визначити 1) номер мінімального елемента масиву; 2) добуток елементів масиву, розташованих до першого нульового елемента.

16. Визначити 1) мінімальний за модулем елемент масиву; 2) добуток елементів масиву, розташованих між першим й останнім нульовими елементами

17. Визначити 1) кількість елементів масиву, більших за число п'ять; 2) суму елементів масиву, розташованих після максимального елемента.

18. Визначити 1) номер мінімального за модулем елемента масиву; 2) суму елементів масиву, розташованих між першим й останнім додатними елементами.

19. Визначити 1) суму елементів масиву з непарними номерами; 2) суму елементів масиву, розташованих між першим й другим додатними елементами.

20. Визначити 1) максимальний елемент масиву; 2) добуток елементів масиву, розташованих між першим й останнім від'ємними елементами.

21. Визначити 1) кількість елементів масиву, менших за число сім; 2) добуток елементів масиву, розташованих між першим й другим нульовими елементами.

22. Визначити 1) кількість від'ємних елементів масиву; 2) суму елементів масиву, розташованих після мінімального елемента.

23. Визначити 1) суму від'ємних елементів масиву; 2) добуток елементів масиву, розташованих між максимальним і мінімальним елементами.

24. Визначити 1) добуток додатних елементів масиву; 2) суму елементів масиву, розташованих до останнього додатного елемента.

25. Визначити 1) номер мінімального за модулем елемента масиву; 2) добуток елементів масиву, розташованих між першим й другим нульовими елементами.

4. Створити та перевірити адекватність тесту до завдання 3(А) згідно вашого варіанту. Дані контрольного прикладу підібрати самостійно. У звіті обов'язково вказати обрані вами параметри контрольного прикладу у форматі

Вхідні дані: ….
Вихідні дані: ….

(1 бал)

5. Створити програми для роз'язування задач згідно свого варіанта. Завдання виконати двома способами. Значення елементів масиву задавати з клавіатури та за допомогою генератора псевдовипадкових чисел з відрізка [-100; 100]. Передбачити захист від введення некоректних даних з клавіатури. Супроводжувати введення/виведення початкових значень, проміжних та кінцевих результатів відповідними повідомленнями.

(1 бал)

1. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти середнє геометричне значення всіх елементів матриці.
2. Дано цілочислову матрицю *A* розмірності *n*x*m*. В ній 2 найбільші елементи замінити нулями.
3. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти середнє арифметичне максимального та мінімального елементів матриці.
4. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти суму найбільших елементів її рядків.
5. Дано цілочислову матрицю *A* розмірності *n*x*m* та координати двох її елементів. Визначити максимальне значення з вказаних чисел.
6. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти кількість входжень максимального та мінімального елементів.
7. Дано цілочислові матриці *A* та *B* розмірності *n*x*n*. Вивести на екран матрицю *A*, в якій парні стовпці замінені на непарні рядки матриці B.
8. Дано цілочислову матрицю *A* розмірності *n*x*m*, та цілі числа *k* та *p* – номери стовпців матриці. Поміняти місцями вказані стовпці.
9. Дано цілочислову матрицю *A* розмірності *n*x*m*. Вивести на екран номера рядків, в яких є хоча б один нульовий елемент та знайдені рядки.
10. Дано цілочислову матрицю *A* розмірності *n*x*n*. Вивести на екран: головну та бічну діагональ матриці, а також елементи матриці *A*, що знаходяться вище головної діагоналі, збільшені в 2 рази.
11. Дано цілочислову матрицю *A* розмірності *n*x*n*. Перетворити її наступним чином: всі елементи бічної діагоналі замінити нулями, всі інші елементи – піднести до квадрату.
12. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти суму від'ємних елементів масиву.
13. Дано цілочислову матрицю *A* розмірності *n*x*m*. В ній 2 найменші елементи замінити нулями, якщо найменшими елементами є 0 – то замінити їх одиницями.
14. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти середнє арифметичне елементів бічної діагоналі матриці.
15. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти суму найменших елементів її рядків.
16. Дано цілочислову матрицю *A* розмірності *n*x*m* та координати двох її елементів. Визначити добуток вказаних чисел.
17. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти кількість входжень кожного з елементів.
18. Дано цілочислові матриці *A* та *B* розмірності *n*x*n*. Вивести на екран матрицю *A*, в якій непарні рядки замінені на парні стовпці матриці B.
19. Дано цілочислову матрицю *A* розмірності *n*x*m*, та цілі числа *k* та *p* – номери рядків матриці. Поміняти місцями вказані рядки.
20. Дано цілочислову матрицю *A* розмірності *n*x*m*. Вивести на екран номера стовпців, в яких є хоча б один нульовий елемент та кількість знайдених нульових елементів.

21. Дано цілочислову матрицю *A* розмірності *n*x*n*. Замінити кожен елемент рядка, де знаходиться мінімальний елемент сумою елементів головної діагоналі.

22. Дано цілочислову матрицю *A* розмірності *n*x*m*. Вивести на екран вектор *b*, елементи якого є максимальними елементами кожного зі рядків.

23. Дано цілочислову матрицю *A* розмірності *n*x*m*. Вивести одновимірний масив *b*, в якому кожен елемент є добутком елементів стовпців, в яких знаходяться максимальний та мінімальний елементи відповідного рядка.

24. Дано цілочислову матрицю *A* розмірності *n*x*m*, *n*-парне. Поміняти місцями верхню та нижню половини масиву.

25. Дано цілочислову матрицю *A* розмірності *n*x*m*. Знайти добутки додатніх елементів її рядків.