# КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



## ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №2**

# з курсу

**«Об’єктно-орієнтоване програмування»**

*Студента 2 курсу*

*групи ПП-11 спеціальності 122 «Комп'ютерні науки» ОП «Прикладне програмування»*

%username%

*Викладач:*

к.ф.-м.н., доц. Шолохов О.В.

## Київ – 202

**1.Назва роботи**

Масиви

1. **Тема роботи**

Масиви

1. **Мета роботи**

Створити консольну програму, що реалізує основні операції з одно-, дво- та *n*-вимірними

масивами: пошук, вставку, видалення, сортування заданим алгоритмом, заповнення за заданим

алгоритмом, трансформацію за заданим алгоритмом, копіювання в інший масив елементів, що

задовольняють певному критерію.

*Зауваження*: алгоритми заповнення і трансформації масиву, критерії елементів щодо

копіювання задаються викладачем при захисті роботи

1. **Умова завдання**

(19 варіант)

Створити консольну програму, що реалізує основні операції з одно-, дво- та *n*-вимірними

масивами: пошук, вставку, видалення, сортування заданим алгоритмом, заповнення за заданим

алгоритмом, трансформацію за заданим алгоритмом, копіювання в інший масив елементів, що

задовольняють певному критерію.

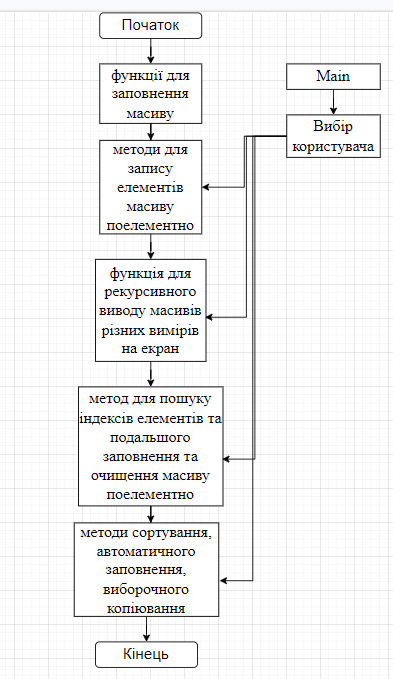
*Зауваження*: алгоритми заповнення і трансформації масиву, критерії елементів щодо

копіювання задаються викладачем при захисті роботи

1. **Рішення**

Описуємо функції для заповнення масиву випадковими числами. Описуємо методи для запису елементів масиву поелементно. Оголошуємо функцію для рекурсивного виводу масивів різних вимірів на екран. Описуємо метод для пошуку індексів елементів та подальшого заповнення та очищення масиву поелементно, методи сортування, автоматичного заповнення, виборочного копіювання. Після цього в функції Main створюємо потрібні змінні та описуємо функцію меню, що буде викликати відповідні функції.

Блок-схема:



Код С#:

using System;

namespace Pa\_231

{

class Program

{

private static System.Random rand = new System.Random();

public static void ChosenSortingFunction<T>(T[] lowLevelArr)

{

System.Array.Sort(lowLevelArr);

}

public static void ChosenFillingFunction<T>(T[] lowLevelArr, int input)

{

for (int i = 0; i < lowLevelArr.Length; i++)

{

lowLevelArr[i] = (T)System.Convert.ChangeType(rand.Next(-10, 10) \* input, typeof(T));

}

}

public static bool ChosenCopyFilter<T>(T input)

{

return (bool)System.Convert.ChangeType(input, typeof(bool));

}

public static int InputIntMinMax(string text, int min, int max)

{

int output = -1;

string input;

while (true)

{

System.Console.WriteLine("\n" + text);

input = System.Console.ReadLine();

if (!System.Int32.TryParse(input, out output)) System.Console.WriteLine(" ERROR 101: Entered number is not an int");

else if (output < min) System.Console.WriteLine(" Error 202: Inclusive minimum permitted value is " + min);

else if (output > max) System.Console.WriteLine(" Error 303: Inclusive maximum permitted value is " + max);

else break;

}

return output;

}

public static int InputIntMin(string text, int min)

{

return InputIntMinMax(text, min, System.Int32.MaxValue);

}

public static int InputIntMax(string text, int max)

{

return InputIntMinMax(text, System.Int32.MinValue, max);

}

public static int InputInt(string text)

{

return InputIntMinMax(text, System.Int32.MinValue, System.Int32.MaxValue);

}

public static int InputChar()

{

string input;

while (true)

{

System.Console.WriteLine("\nEnter the desired char");

input = System.Console.ReadLine();

if (input.Length != 1) System.Console.WriteLine(" Error 505: Invalid length");

else break;

}

return System.Convert.ToChar(input);

}

private static string PrintArrayRecursion<T>(T[] arr, int[] dimensionSizes, int depth, int step, int index)

{//0,len,0

string output = "{";

int lowLevelLen = dimensionSizes[dimensionSizes.Length - 1];

if (depth == dimensionSizes.Length - 1)

{

output += arr[index];

for (int i = 1; i < lowLevelLen; i++)

{

output += ", " + arr[index + i];

}

}

else

{

step /= dimensionSizes[depth];

output += PrintArrayRecursion(arr, dimensionSizes, depth + 1, step, index);

for (int i = 1; i < dimensionSizes[depth]; i++)

{

output += ", " + PrintArrayRecursion(arr, dimensionSizes, depth + 1, step, index + step \* i);

}

}

return output + "}";

}

public static void PrintArray<T>(T[] arr, int[] dimensionSizes)

{

string text = "";

if (dimensionSizes.Length == 1)

{

text += arr[0];

for (int i = 1; i < dimensionSizes[0]; i++)

{

text += ", " + arr[i];

}

}

else if (dimensionSizes.Length == 2)

{

for (int i = 0; i < dimensionSizes[0]; i++)

{

text += arr[i \* dimensionSizes[1]];

for (int j = 1; j < dimensionSizes[1]; j++)

{

text += "\t" + arr[i \* dimensionSizes[1] + j];

}

text += "\n\n\n\n";

}

}

else text = PrintArrayRecursion(arr, dimensionSizes, 0, arr.Length, 0);

System.Console.WriteLine(text);

}

public static int[] FindIndex<T>(T[] arr, int[] dimensionSizes, T val)

{

int[] output = new int[dimensionSizes.Length];

int index = System.Array.IndexOf(arr, val);

for (int i = output.Length - 1; i >= 0; i--)

{

output[i] = index % dimensionSizes[i];

index /= dimensionSizes[i];

}

return output;

}

public static int indexFromFullIndex<T>(T[] arr, int[] dimensionSizes, int[] fullIndex)

{

int index = 0, step = arr.Length;

for (int i = 0; i < fullIndex.Length; i++)

{

step /= dimensionSizes[i];

index += fullIndex[i] \* step;

}

return index;

}

public static void SetValue<T>(T[] arr, int[] dimensionSizes, int[] fullIndex, T val)

{

arr[indexFromFullIndex(arr, dimensionSizes, fullIndex)] = val;

}

public static void ClearValue<T>(T[] arr, int[] dimensionSizes, int[] fullIndex)

{

System.Array.Clear(arr, indexFromFullIndex(arr, dimensionSizes, fullIndex), 1);

}

public static void SortArray<T>(T[] arr, int[] dimensionSizes)

{

int lowLevelLen = dimensionSizes[dimensionSizes.Length - 1];

T[] lowLevelArr = new T[lowLevelLen];

for (int i = 0; i < arr.Length; i += lowLevelLen)

{

for (int j = 0; j < lowLevelLen; j++)

{

lowLevelArr[j] = arr[i + j];

}

ChosenSortingFunction(lowLevelArr);

lowLevelArr.CopyTo(arr, i);

}

}

public static void FillArray<T>(T[] arr, int[] dimensionSizes)

{

int lowLevelLen = dimensionSizes[dimensionSizes.Length - 1];

T[] lowLevelArr = new T[lowLevelLen];

for (int i = 0; i < arr.Length; i += lowLevelLen)

{

for (int j = 0; j < lowLevelLen; j++)

{

lowLevelArr[j] = arr[i + j];

}

ChosenFillingFunction(lowLevelArr, i / lowLevelLen + 1);

lowLevelArr.CopyTo(arr, i);

}

}

public static T[] SelectiveCopy<T>(T[] arr)

{

bool[] successLog = new bool[arr.Length];

int successes = 0;

for (int i = 0; i < arr.Length; i++)

{

if (ChosenCopyFilter(arr[i]))

{

successLog[i] = true;

successes++;

}

else successLog[i] = false;

}

T[] secondArr = new T[successes];

for (int i = 0, j = 0; i < arr.Length; i++)

{

if (successLog[i])

{

secondArr[j] = arr[i];

j++;

}

}

return secondArr;

}

public static void Main()

{

int[] dimensionSizes, newDimensionSizes;

int menu = 1, dimensionNumber, newDimensionNumber, arrSize = 1, newArrSize;

int[] arr, secondArr;

dimensionNumber = InputIntMin("Enter the desired number of dimensions:", 1);

dimensionSizes = new int[dimensionNumber];

for (int i = 0; i < dimensionNumber; i++)

{

dimensionSizes[i] = InputIntMin("Enter the " + (i + 1) + " dimension:", 1);

arrSize \*= dimensionSizes[i];

}

arr = new int[arrSize];

secondArr = new int[0];

while (menu != 0)

{

menu = InputIntMinMax("Enter:\n1 - to print the array,\n2 - to find a value,\n3 - to set a value,\n4 - to clear a value,\n5 - to sort the array,\n6 - to fill the array,\n7 - to resize the array,\n8 - to reverse the array,\n9 - to selectively copy values to a secondary array,\n10 - to print the secondary array,\n0 - to exit", 0, 10);

switch (menu)

{

case 1:

PrintArray(arr, dimensionSizes);

break;

case 2:

int[] indexF = FindIndex(arr, dimensionSizes, InputInt("Enter value:"));

System.Console.Write("[" + indexF[0]);

for (int i = 1; i < dimensionNumber; i++)

{

System.Console.Write("," + indexF[i]);

}

System.Console.WriteLine("]");

break;

case 3:

int[] indexS = new int[dimensionNumber];

for (int i = 0; i < dimensionNumber; i++)

{

indexS[i] = InputIntMinMax("Enter the index in " + (i + 1) + " dimension:", 0, dimensionSizes[i] - 1);

}

SetValue(arr, dimensionSizes, indexS, InputInt("Enter value:"));

break;

case 4:

int[] indexC = new int[dimensionNumber];

for (int i = 0; i < dimensionNumber; i++)

{

indexC[i] = InputIntMinMax("Enter the index in " + (i + 1) + " dimension:", 0, dimensionSizes[i] - 1);

}

ClearValue(arr, dimensionSizes, indexC);

break;

case 5:

SortArray(arr, dimensionSizes);

break;

case 6:

FillArray(arr, dimensionSizes);

break;

case 7:

newDimensionNumber = InputIntMin("Enter the new number of dimensions:", 1);

newDimensionSizes = new int[newDimensionNumber];

newArrSize = 1;

for (int i = 0; i < newDimensionNumber; i++)

{

newDimensionSizes[i] = InputIntMin("Enter the " + (i + 1) + " dimension:", 1);

newArrSize \*= newDimensionSizes[i];

}

if (newArrSize != arrSize) System.Console.WriteLine(" Error 606: New array size doesn't match");

else dimensionSizes = newDimensionSizes;

break;

case 8:

System.Array.Reverse(arr);

break;

case 9:

secondArr = SelectiveCopy(arr);

break;

case 10:

PrintArray(secondArr, new int[] { secondArr.Length });

break;

}

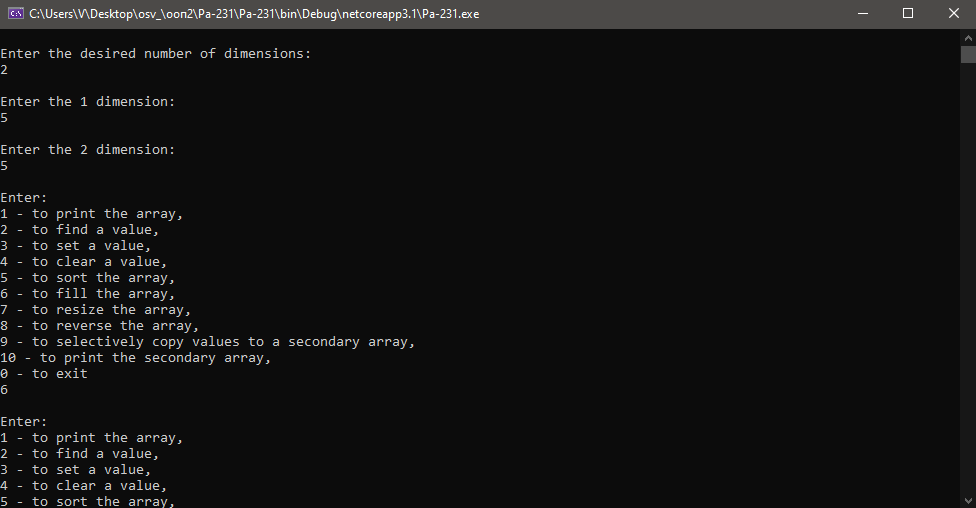
}

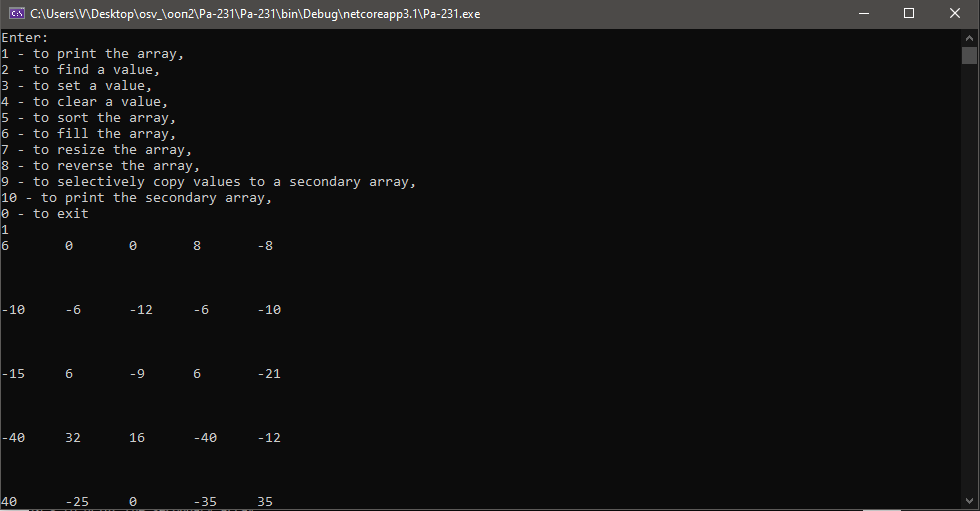
}

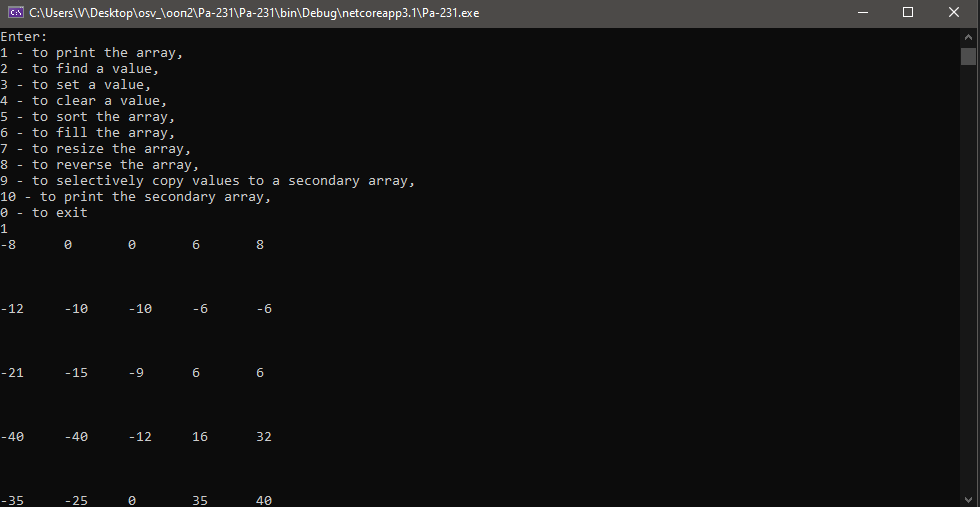
}

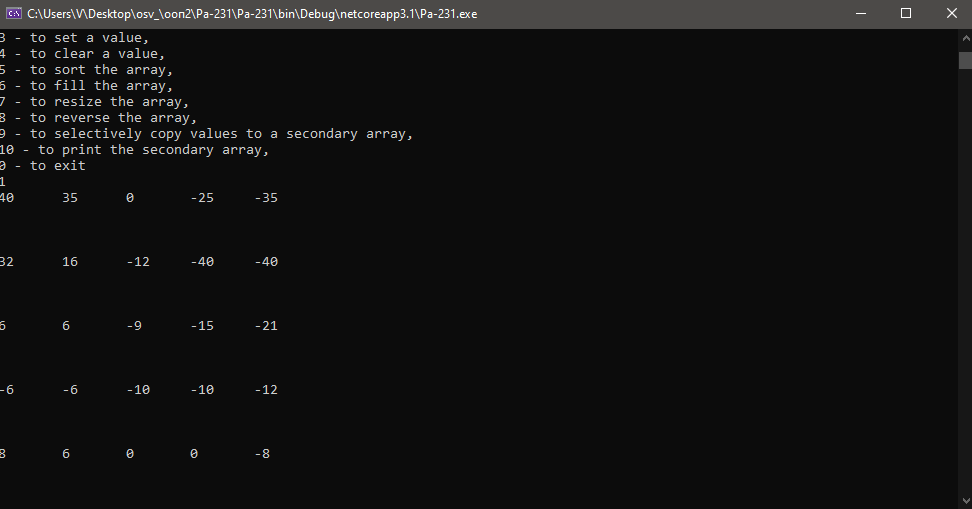
}

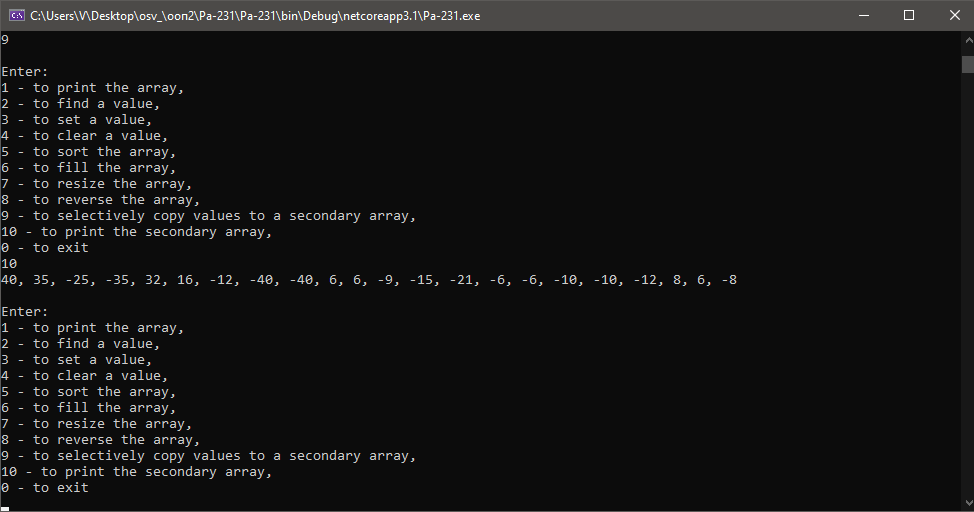
Результат роботи програми C#:











**Контрольні питання**

1. Які види масивів можна створювати в C#?

Cтатичні та динамічні

1. Як оголошуються різні види масивів?

По-різному

1. Оператор *foreach* – його синтаксис та приклади роботи з масивами.

foreach (int element in array) {

Console.WriteLine(element);

}

4. Елементи класу : *Rank -* отримує кількість вимірів масиву*, Length -* зберігає довжину масиву*, GetLength -* повертає довжину масиву*, Clear -* видаляє всі елементи масиву*, Sort -* сортує масив*, Clone -* створює копію масиву з посиланням на оригінал*, IndexOf -* повертає індекс вказаного елементу*, Reverse -* перевертає масив*,*

*GetLowerBound -* отримує індекс першого елементу вказаного виміру масиву*, GetUpperBound -* отримує індекс першого елементу вказаного виміру масиву*, GetType -* повертає тип вказаного об’єкту (масиву)*.*

1. **Висновки**

В результаті виконання даної лабораторної роботи я оволодів практичними навичками зі створення консольних програм, для реалізації основних операцій з одно-, дво- та *n*-вимірними масивами: пошук, вставка, видалення, сортування заданим алгоритмом, заповнення за заданим алгоритмом, трансформація за заданим алгоритмом, копіювання в інший масив елементів, що задовольняють певному критерію. Методи та змінні викликаються з меню, реалізованого за допомогою switch. Вважаю дану лабораторну роботу виконаною в повному обсязі.