Міністерство освіти і науки України

Галицький фаховий коледж імені В’ячеслава Чорновола

циклова комісія інформатики та

комп’ютерних дисциплін

КУРСОВИЙ ПРОЕКТ

з навчальної дисципліни

“Об’єктно-орієнтоване програмування”

на тему:

**Програма для дослідження алгоритмів сортування**

Студент гр. КН-31

Нищота Юрій Тарасович

Керівник Кузик Василь Миколайович

Курсовий проект захищено з оцінкою

“\_\_\_\_\_\_\_\_\_\_\_\_”

“\_\_\_” \_\_\_\_\_\_\_ 20\_\_ р.

Члени комісії: Сиротюк Наталія Станіславівна

Кульчинська Наталія Зіновіївна

Тернопіль – 2022

ЗМІСТ

[ВСТУП 3](#_Toc120736498)

[1 ПОСТАНОВКА ЗАВДАННЯ 4](#_Toc120736499)

[1.1 Опис завдання 4](#_Toc120736500)

[1.2 Огляд існуючих ПС 4](#_Toc120736501)

[1.3 Аналіз вимог до ПС 7](#_Toc120736502)

[2 ПОБУДОВА МОДЕЛІ ПС 8](#_Toc120736503)

[2.1 Визначення основних функцій системи та структури системи 8](#_Toc120736504)

[2.2 Проектування інтерфейсу системи 8](#_Toc120736505)

[2.3 Вибір технології і засобів реалізації створення програмної системи 11](#_Toc120736506)

[2.3.1 Опис технології реалізації 11](#_Toc120736507)

[2.3.2 Вибір засобів створення інтерфейсної частини 11](#_Toc120736508)

[3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ 12](#_Toc120736509)

[3.1 Розробка основних форм інтерфейсу користувача та їх опис 12](#_Toc120736510)

[3.2 Програмування графічного користувацького інтерфейсу 12](#_Toc120736511)

[3.3 Опис програмних рішень 13](#_Toc120736512)

[4 ТЕСТУВАННЯ ТА ОЦІНКА ЯКОСТІ ПС 25](#_Toc120736513)

[4.1 План тестування. 25](#_Toc120736514)

[4.2 Аналіз можливих проблем 27](#_Toc120736515)

[ВИСНОВКИ 28](#_Toc120736516)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 29](#_Toc120736517)

# ВСТУП

В деяких сферах діяльності люди працюють з масивами. Набором чисел різного діапазону, роблять з ними математичні дії, множать одне на одне, віднімають, тощо.

Метою цієї програми буде сортування цих масивів за допомогою методів сортування, які на даний час є доволі багато. Кожний метод має свою ідеологію і відрізняється від інших.

Методи сортування масивів:

* метод вибору;
* метод вставками;
* сортування “Бульбашкою”;
* сортування злиттям;
* швидке сортування;
* пірамідальне сортування;
* плавне сортування;
* сортування за підрахунками.

Перших п’ять сортувань будуть застосовані в цій програмі.

У вигляді форми за допомогою засобів C# та його графічного інтерфейсу програмування додатків Windows Forms, створимо програму яка буде сортувати випадкові масиви. Для реалізації поставленої задачі було використано велику кількість різноманітних способів і стилів та застосовано мову C#.

Результатом розробки стала завершена робота, яка готова до використання.

# 1 ПОСТАНОВКА ЗАВДАННЯ

1.1 Опис завдання

Метою даного курсового проекту є написання програми сортування масивів.

Основними завданнями КП визначено:

* розробка програми, зрозумілої для користувача у використанні;
* пояснення кожного методу сортування;
* написання зручного та зрозумілого інтерфейсу;
* знаходження швидкого та ефективного методу сортування.

Результати даного курсового проекту можуть бути корисними для людей які працюють з масивами. Цю програму можна використати як на різних конкурсах, змаганнях з знаходженням бистрого методу сортування масиву.

1.2 Огляд існуючих ПС

Для перевірки наявної ПС, було обрано онлайн програма, що за типом роботи має схожу характеристику і сферу застосування.

Прикладом є онлайн калькулятор сортування чисел на сайті calculators.vip. Ця система включає в собі багато функцій. Користувач може ввести числа, вибрати роздільник між ними та кнопкою перейти на результат.

В результативному блоці користувач може змінити роздільник, видалити дублікати чисел, включити зворотний порядок сортування. Нажаль, щоб очистити всі вікна потрібно оновити сайт.

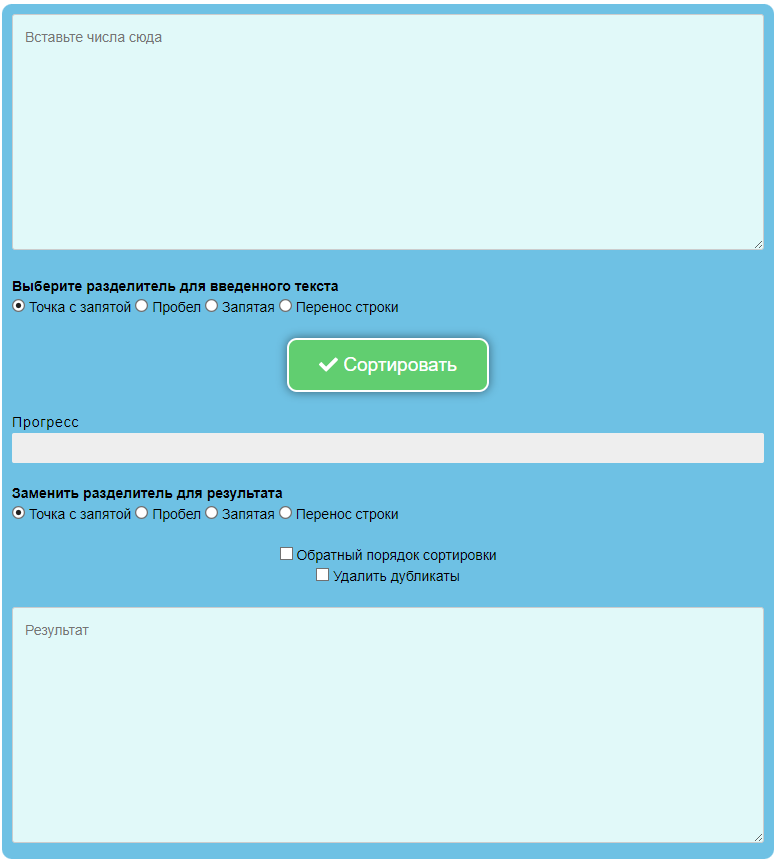
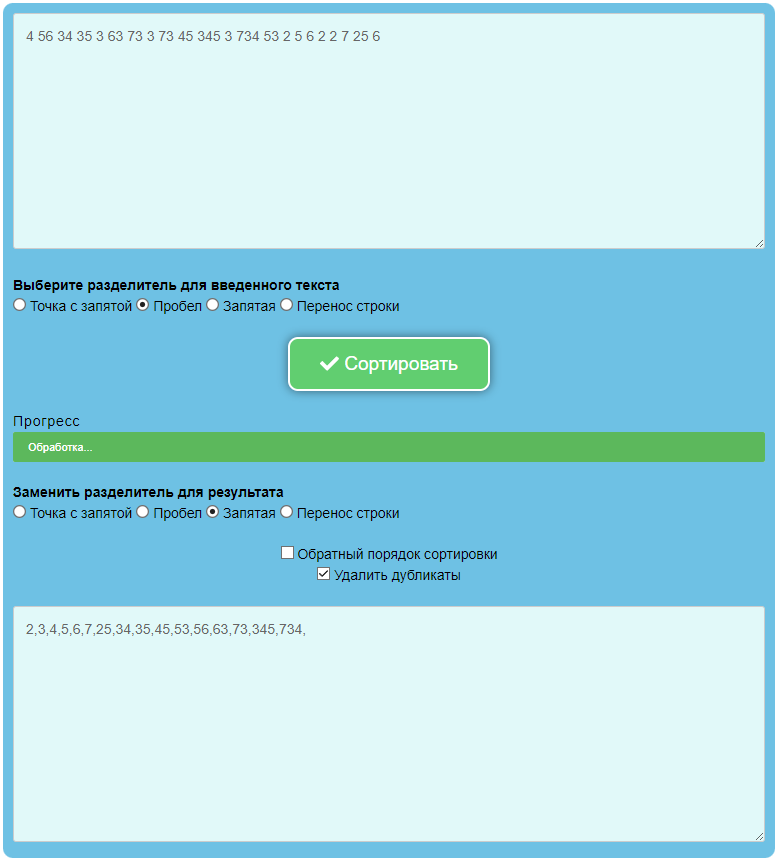
Ще один приклад онлайн тестування зображено на рисунку 1.2. 

Рисунок 1.1 – Введення чисел

Рисунок 1.2 – Результат

Виходячи з переглянутого та враховуючи функціонал можна заявити, що програма не є дуже складною для її реалізації. Все залежить від власного бажання та наполегливості для її створення.

1.3 Аналіз вимог до ПС

Дана програмна система включає в себе невелику кількість вимог для повноцінного створення зв’язку з користувачем. В основному вони пов'язані з графічним інтерфейсом програмного продукту. Користувацький зовнішній інтерфейс повинен мати приємний вигляд і нескладну конструкцію побудови для кращого розуміння у перспективі використання програмної системи.

Візуальна графічна частина є невід’ємною складовою програми, але щоб ПС активно і безпомилково працювала, потрібно забезпечувати правильність застосування алгоритмів. Усі текстові поля для запису числових значень повинні мати спеціальну обробку для заборони занесення даних текстового типу. Необхідна перевірка на введення мінімально потрібної кількості даних користувачем, для подальшої роботи ПС.

Якщо детальніше розглядати наявну ПС, то основними вимогами будуть: налагодження правильного занесення даних і їх виведення, безпомилковість у написаних алгоритмах, створення алгоритмів для розв’язання різних типів завдань з різним рівнем складності, забезпечення створення спеціальних засобів для полегшення роботи користувачеві під час роботи з ПС.

2 ПОБУДОВА МОДЕЛІ ПС

2.1 Визначення основних функцій системи та структури системи

Дане ПС характеризується невеликою кількістю функцій, адже основною метою завдання є правильність алгоритму програми.

До основного функціоналу даної програмної системи входить: введення та виведення даних, можливість сортування масивів, результуюче поле.

Користувач вписує у комірки розмір масиву, можливе випадкове максимальне і мінімальне число та продовжує роботу програми. Без введення даних, користувач не може продовжувати роботу, оскільки цей алгоритм є важливим і без неї програма не спрацює.

Методом обробки інформації в даній програмі є:

* введення даних з клавіатури текстові поля форми, їх зчитування;
* реалізація користувацьких функцій які будуть сортувати масиви;

- виведення результату після виконання операції над масивом, події натискання на кнопки, подія відкривання форми.

2.2 Проектування інтерфейсу системи

Першим етапом проектування даної програмної буде створення Windows Forms за допомогою елементів керування. Система буде поділятись на декілька форм, а саме:

* головне меню;
* довідка для головного меню;
* форми сортувань масивів;
* форми для довідок сортувань.

Все це зображено на рисунках 2.1, 2.2.

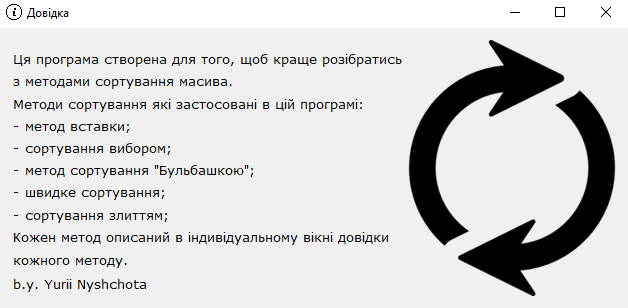
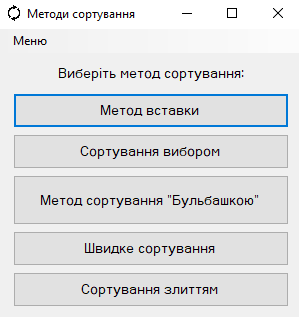


Рисунок 2.2 – Приклад довідки програми

Рисунок 2.1 – Приклад форми головного меню

Форма сортування масиву та довідки на рисунках 2.3, 2.4.

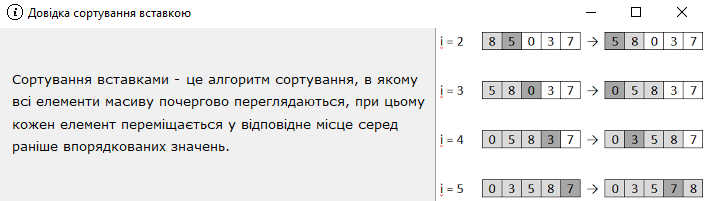
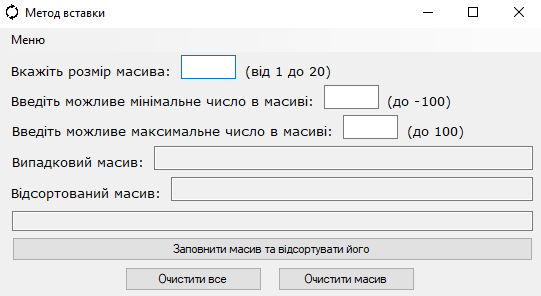
 

Рисунок 2.4 – Приклад довідки до сортування вставкою

Рисунок 2.3 – Приклад одної з форм сортувань

Головна форма буде вмістити декілька кнопок, які будуть пересилати користувача на нову форму з сортуванням масиву. Також на кожній формі будуть створені кнопки типу “меню”, в якому користувач може відкрити довідку про програму або ж сортування, повернутись назад якщо він перейшов на нову форму та завершити програму, тобто закрити. В формі сортувань користувач може ввести розмір масиву, можливе випадкове мінімальне та максимальне число або коротше кажучи діапазон масиву. І три кнопки, заповнення та сортування, очищення створеного масиву, очищення всіх полів.

2.3 Вибір технології і засобів реалізації створення програмної системи

2.3.1 Опис технології реалізації

Проаналізувавши дане завдання, було вирішено вибрати для реалізації програмного забезпечення мову програмування С# - об’єктно-орієнтована мова програмування з безпечною системою типізацїї для платформи .NET. Також вона швидко обробляє помилки, і, будучи мовою дуже високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на інших мовах потребує значних витрат часу.

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

2.3.2 Вибір засобів створення інтерфейсної частини

Для виконання ПС використаємо інтегроване середовище програмування Visual Studio 2022.

Microsoft Visual Studio - лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-програми, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework та Silverlight.

Аби виконати поставлену задачу, будемо використовувати технологію Windows Forms, яка володіє великим набором елементів керування, зручним візуальним конструктором та безліччю інформаційних ресурсів, що стануть в пригоді при освоєнні технології та її подальшому користуванні.

3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

3.1 Розробка основних форм інтерфейсу користувача та їх опис

Програма починається з форми, яка викликається клавішею F5 в програмному забезпеченні. Вона буде працювати до поки користувач не завершить, допоки користувач не закриє її. Результуючі дані відображаються у тих самих текстових полях, де вводилися первинні значення. Для користувача форма має зручний та зрозумілий інтерфейс.

Для виконання завдання було використано 3 файли - alg.py, py\_form, temps.txt. Alg.py містить програмний код для розв’язування судоку. Його алгоритм складається із однієї загальної функції, яка приймає вхідний масив (незаповнене поле гри) та повертає його уже заповненим. Вона ж у свою чергу складається із 5 функцій та декількох процедур. Py\_form.py – файл з алгоритмом для побудови форми, яка складається із 81 текстового поля та трьох кнопок – для розв’язання судоку, очищення форми та для взяття макету з текстового документа. Цей текстовий документ має назву temps.txt. В ньому прописані декілька варіантів шаблону судоку.

3.2 Програмування графічного користувацького інтерфейсу

Дана програмна система є функціональною не тільки за допомогою прописаних алгоритмів, а й з використанням графічного зовнішнього інтерфейсу.

Для реалізації графічного інтерфейсу на С# було використано Windows Forms на основі системи .NET. .NET Framework — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з ідей .NET є сумісність служб, написаних різними мовами.

3.3 Опис програмних рішень

Для проектування даної програмної системи, використовувались певні алгоритми дій, що забезпечували стабільну, багаторазову роботу програми та її прямий зв’язок з користувачем.

При запуску програми відкривається головна форма із відповідним форматування. Кожна кнопка запрограмована на перехід нової форми. Приклад коду:

Form2 form2 = new Form2();

form2.Show();

this.Hide();

Створюється об’єкт типу даних Form2 за допомогою якого надалі можна викликати форму вбудованою функцією Show(), та заховати поточну вказівником та ще одною функцією Hide().

Так налаштовуємо п’ять кнопок які будуть пересилати користувача з форми в форму, змінюючи назви форм.

Також потрібно створити елемент керування MenuStrip який буде відігравати роль меню програми. Створюємо в ньому кнопку “Довідка” та “Вихід”, наразі це всі потрібні елементи які потрібні на головній формі. Кнопку довідка запрограмовуємо як раніше на перехід нової форми яка вже буде налаштована. А в кнопку вихід вписуємо код який буде повністю завершувати програму:

Application.Exit();

З головною формою покінчено перейдемо до першої форми сортування масиву.

Створюємо елементи керування такі як:

* Label – для керуванням користувача по формі;
* textBox – для введення та виводу даних;
* Button – створення кнопки;
* menuStrip – створення меню програми.

Необхідно створити поля для введення користувачем розміра масиву та діапазону. Заборонимо користувачу вводити лишні символи за допомогою події KeyPress. Для цього виділяємо поля які потрібні і шукаємо в списку подій KeyPress клацаємо двічі і вписуємо код.

char number = e.KeyChar;

if ((e.KeyChar <= 46 || e.KeyChar >= 58) && number != 8 && number != 45)

{

e.Handled = true;

}

В цьому коді за допомогою кода ASCII забороняємо користувачу ввести лишні символи крім клавіш цифр, мінуса та BackSpace.

Створимо три поля для виведення результату, перше поле призначаємо для виведення створення нового випадкового масиву, друге поле для виведення посортованого масиву і третє для вказівок в разі неправильних введень користувачем.

Також створюємо три кнопки, першу кнопку запрограмуємо на основні функції та дії а саме: перевірка введення даних, створення випадкового масиву, сортування. Створюємо змінні та зчитуємо дані записуючи в щойно створені змінні.

int N, a, b, rand;

int.TryParse(textBox1.Text, out N);

int.TryParse(textBox2.Text, out a);

int.TryParse(textBox3.Text, out b);

Тут перевірюємо чи користувач ввів дані в всі поля, якщо поле пусте виводимо текст в комірку для керуванням користувачем та змінюємо колір комірки.

if (textBox1.Text == string.Empty || textBox2.Text == string.Empty || textBox3.Text == string.Empty)

{

textBox6.Text = "Введіть всі значення!";

textBox6.BackColor = Color.Red;

}

Перевірюємо чи користувач ввів правильно розмір масиву.

else if (N <= 0 || N >= 21)

{

textBox6.Text = "Введіть розмір від 1 до 20!";

textBox6.BackColor = Color.Red;

}

Перевірюємо чи користувач ввів правильно мінімальне можливе число масиву.

else if (a < -100 || a > 0)

{

textBox6.Text = "Введіть мінімальне число від -100 до 0!";

textBox6.BackColor = Color.Red;

}

Перевірюємо чи користувач ввів правильно максимальне можливе число масиву.

else if (b > 100 || b < 0)

{

textBox6.Text = "Введіть максимальне число від 0 до 100!";

textBox6.BackColor = Color.Red;

}

Також перевірюємо чи користувач ввів в полях діапазону нуль або ж одиницю, при яких неможливо створити випадковий масив.

else if (a >= 0 && b <= 1 || a >= -1 && b <= 0 || a >= 0 && b <= 0)

{

textBox6.Text = "Введіть більший діапазон мінімальних та максимальних чисел!";

textBox6.BackColor = Color.Red;

}

Якщо користувач ввів все правильно та за вимогами продовжуємо програму. Робимо кнопку недоступною, щоб користувач не міг безліч раз створювати масиви. Змінюємо колір результативної комірки та змінюємо текст.

else

{

button1.Enabled = false;

textBox6.BackColor = Color.Green;

textBox6.Text = "Програма виконана!";

Створюємо одновимірний масив розміром N та заповнюємо його випадковими числами, для цього запускаємо цикл, в функції random.Next, вказуємо діапазон від а до b. Виводимо масив в поле.

int[] A = new int[N];

Random random = new Random();

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

{

rand = random.Next(a, b);

A[i] = rand;

textBox4.Text += A[i].ToString() + "\r ";

}

В цій формі застосуємо сортування методом вставками. Метод вставками – це простий алгоритм сортування. Суть його полягає в тому, що на кожному кроці алгоритму ми беремо один з елементів масиву, знаходимо позицію для вставки та вставляємо. Реалізовуємо це в вигляді коду.

int index, currentNumber;

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

{

index = i;

currentNumber = A[i];

while (index > 0 && currentNumber < A[index - 1])

{

A[index] = A[index - 1];

index--;

}

A[index] = currentNumber;

Виводимо в віддільне поле результат сортування.

}

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

{

textBox5.Text = textBox5.Text + " " + A[i].ToString();

}

}

Наступну кнопку запрограмуємо на очистку всіх полів за допомогою функції Empty, що дослівно означає пусто. Дозволяємо користувачу клацнути ще раз створити масив та посортувати його. Ще змінюємо колір результативного вікна на білий як і було на початку.

button1.Enabled = true;

textBox1.Text = string.Empty;

textBox2.Text = string.Empty;

textBox3.Text = string.Empty;

textBox4.Text = string.Empty;

textBox5.Text = string.Empty;

textBox6.Text = string.Empty;

textBox6.BackColor=Color.White;

Також налаштовуємо останню кнопку в формі яка буде очищувати тільки ті поля в яких були виведений масив. Ця кнопка створена створена тільки для комфорту користувача, щоб він міг швидко очищувати поля та знову їх заповнювати для перевірки.

Меню запрограмуємо як і в першому випадку. Викликаємо форму за допомогою об’єктів типів даних. Але тут появиться ще одна кнопка в меню, яка буде повертати користувача на основну форму програми.

this.Hide();

Form1 form1 = new Form1();

form1.Show();

Отже, основна та перша форми налаштовані і запрограмовані. В наступних формах нічого нового створюватись не буде, крім того, що буде змінюватись метод сортування які зараз будуть розглянуті.

Сортування вибором - полягає у пошуку мінімального значення елемента в масиві, і переміщення цього значення в початок масиву. Потрібно відразу обмовитися, що в даному випадку можна назвати “початком” массива (куди переміщається знайдене мінімальне значення). “Старт” в алгоритмі Сортування вибором з кожним кроком циклу зміщується в бік хвоста масиву. Тому на першій ітерації циклу, знайдене мінімальне значення міняється місцями зі значенням в нульовий осередку масиву. На другий ітерації “старт” вже буде вказувати на наступну (першу) осередок і так далі.

int temp = 0, min = 0;

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

{

min = i;

for (int j = i + 1; j < N; j++)

{

min = (A[j] < A[min]) ? j : min;

}

if (i != min)

{

temp = A[i];

A[i] = A[min];

A[min] = temp;

}

textBox5.Text = textBox5.Text + " " + A[i].ToString();

}

Сортування вибором - полягає у пошуку мінімального значення елемента в масиві, і переміщення цього значення в початок масиву. Потрібно відразу обмовитися, що в даному випадку можна назвати “початком” масиву (куди переміщається знайдене мінімальне значення). “Старт” в алгоритмі Сортування вибором з кожним кроком циклу зміщується в бік хвоста масиву. Тому на першій ітерації циклу, знайдене мінімальне значення міняється місцями зі значенням в нульовий осередку масиву. На другий ітерації “старт” вже буде вказувати на наступну (першу) осередок і так далі.

int temp = 0, min = 0;

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

{

min = i;

for (int j = i + 1; j < N; j++)

{

min = (A[j] < A[min]) ? j : min;

}

if (i != min)

{

temp = A[i];

A[i] = A[min];

A[min] = temp;

}

textBox5.Text = textBox5.Text + " " + A[i].ToString();

}

Сортування “Бульбашкою”. Алгоритм полягає в циклічних проходах по масиву, за кожен прохід елементи масиву попарно порівнюються і, якщо їх порядок неправильний, то здійснюється обмін. Обхід масиву повторюється до тих пір, поки масив не буде впорядкований.

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

{

for (int j = 0; j < N - 1; j++)

{

if (A[j + 1] < A[j])

{

int temp = A[j];

A[j] = A[j + 1];

A[j + 1] = temp;

}

}

}

Ідея алгоритму полягає в переставлянні елементів масиву таким чином, щоб його можна було розділити на дві частини і кожний елемент з першої частини був не більший за будь-який елемент з другої. Впорядкування кожної з частин відбувається рекурсивно.

Алгоритм швидкого сортування може бути реалізований як у масиві, так і в двозв'язному списку. Швидке сортування є алгоритмом на основі порівнянь і не є стабільним.

В основній кнопці застосуємо лиш новий створений масив який буде дорівнювати функцію сортування першого випадкового масиву.

int[] sortedArray = Quicksort(A, 0, N - 1);

В основній кнопці застосуємо лиш новий створений масив який буде дорівнювати функцію сортування першого випадкового масиву. Цю функцію сортування потрібно створити таким шляхом, спершу масив буде розбитий на два під масиви перший та другий. В них будуть знаходитись мінімальний та максимальний елементи.

private static int[] Quicksort(int[] array, int minIndex, int maxIndex)

{

if (minIndex>=maxIndex)

{

return array;

}

int v = getV(array, minIndex, maxIndex);

Quicksort(array, minIndex, v - 1);

Quicksort(array, v + 1, maxIndex);

return array;

}

Наступна функція буде міняти місцями числа двох підмасивів за допомогою мінімальних та максимальних чисел.

private static int getV(int[] array, int minIndex, int maxIndex)

{

int v = minIndex -1 ;

for (int i = minIndex; i <= maxIndex; i++)

{

if (array[i] < array[maxIndex])

{

v++;

Swap(ref array[v], ref array[i]);

}

}

v++;

Swap(ref array[v], ref array[maxIndex]);

return v;

}

Та остання потрібна функція для застосування цього сортування Swap, що дослівно означає своп або обмін.

private static void Swap(ref int leftI, ref int rightI)

{

int temp = leftI;

leftI = rightI;

rightI = temp;

}

Як і в інших методах виводимо вже новий відсортований масив.

Останнє сортування називається методом злиття. В основі цього способу сортування лежить злиття двох упорядкованих ділянок масиву в одну впорядковану ділянку іншого масиву. Злиття двох упорядкованих послідовностей можна порівняти з перебудовою двох колон солдатів, вишикуваних за зростом, в одну, де вони також розташовуються за зростом. Якщо цим процесом керує офіцер, то він порівнює зріст солдатів, перших у своїх колонах і вказує, якому з них треба ставати останнім у нову колону, а кому залишатися першим у своїй. Так він вчиняє, поки одна з колон не вичерпається — тоді решта іншої колони додається до нової. Метод буде створюватись по схожому принципі як і минуле сортування. Отже, спочатку масив буде розбиватись до тих пір, поки він не закінчиться.

static void Merge(int[] targetArray, int[] array1, int[] array2)

{

int array1minindex = 0;

int array2minindex = 0;

int targetarrayminindex = 0;

while (array1minindex < array1.Length && array2minindex < array2.Length)

{

if (array1[array1minindex] <= array2[array2minindex])

{

targetArray[targetarrayminindex] = array1[array1minindex];

array1minindex++;

}

else

{

targetArray[targetarrayminindex] = array2[array2minindex];

array2minindex++;

}

targetarrayminindex++;

}

while (array1minindex < array1.Length)

{

targetArray[targetarrayminindex] = array1[array1minindex];

array1minindex++;

targetarrayminindex++;

}

while (array2minindex < array2.Length)

{

targetArray[targetarrayminindex] = array2[array2minindex];

array2minindex++;

targetarrayminindex++;

}

}

Після, цього, він буде сортуватись за зростанням.

static void MergeSort(int[] array)

{

if (array.Length < 2)

{

return;

}

int mid = array.Length / 2;

int[] left = new int[mid];

int[] right = new int[array.Length-mid];

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

{

left[i] = array[i];

}

for (int i = mid; i < array.Length; i++)

{

right[i-mid] = array[i];

}

MergeSort(left);

MergeSort(right);

Merge(array, left, right);

}

В результаті готові всі форми програми, тому можна спробувати їх протестувати.

4 ТЕСТУВАННЯ ТА ОЦІНКА ЯКОСТІ ПС

4.1 План тестування.

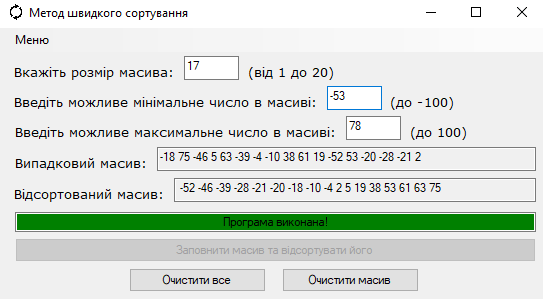
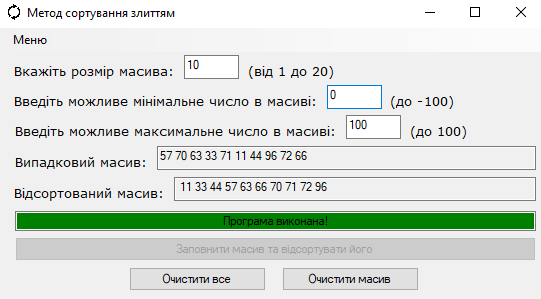
Спробуємо запустити програму та вибрати застосувати декілька сортуваннь. (рис.4.1, 4.2, 4.3)

Рисунок 4.2 – Тестування сортування злиттям

Рисунок 4.1 – Тестування методу швидкого сортування

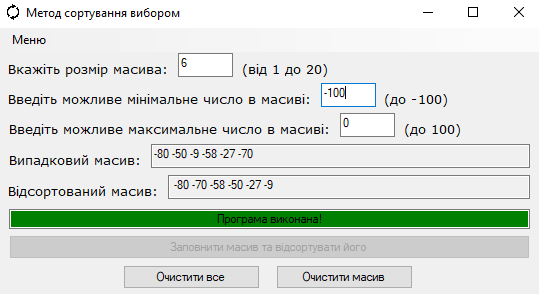
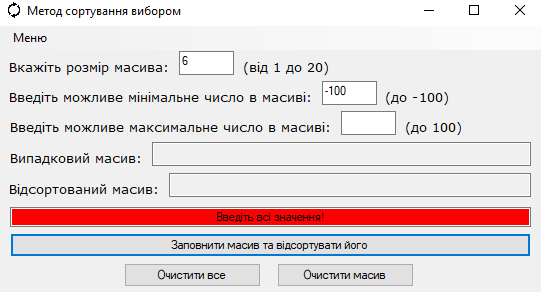
Спробуємо не ввести останній пункт. (рис.4.4)

Рисунок 4.4 – Спроба створити помилку

Рисунок 4.3 – Тестування сортування вибором

Також спробуємо, ввести більше число ніж вказане програмою. (рис.4.5)

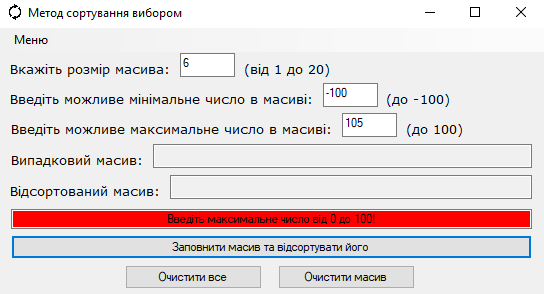
Отже, після декількох спроб сортувань та помилок, можемо сказати, що програма працює правильно та безпомилково.

Рисунок 4.5 – Спроба створити помилку

4.2 Аналіз можливих проблем

Даний програмний продукт доволі оптимізований та правильно сконструйований для генерального використання. Є чимало алгоритмів, що перевіряють правильність роботи програмної системи під час процесу використання користувачем.

Але помилки все таки можуть бути, наприклад в неправильності написанні функцій сортування масиву. Також є нюанс з сортуванням злиттям, якщо вказати не парне число програма може не зрозуміти, що робити, тому що функція розбиває масив по парно, якщо не вказати, що робити в цій ситуації процес може завершитись.

Отже, яка б програмна система не була оптимізована чи наповнена функціоналом, у ній завжди можна найти неточності та недоліки, хоч і можливо не значні.

ВИСНОВКИ

Даний курсовий проект розроблявся з ціллю збільшення знань про масиви їх побудову, створення, тощо. Також цей проект може бути корисним для людей які тільки почали вивчати вищу математику або яким цікава ця тема.

Сортуванням є також великий плюс до цього курсового проекту. Можна вивчити, що таке взагалі сортування, їхні види, класифікації, важкості, як вони відрізняються та як їх застосовувати.

Проходячи цей КП вивчено нові знання про такі сортування масиву як:

* сортування вибором;
* сортування вставкою;
* сортуванням “Бульбашкою”;
* швидке сортування;
* сортування злиттям.

А саме вивчено те, що це за сортування, як їх використовувати або створювати.

В майбутньому планується удосконалення цієї програми, що буде супроводжуватись залученням нового функціоналу та виправлення недоліків, які є в теперішній версії. Наприклад буде змінено інтерфейс, нова кольорова гамма добавить програмі хорошого і комфортного для очей вигляду, що буде зацікавлювати користувача використовувати її.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Windows Forms. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Windows\_Forms. (дата звернення 30.11.2022)
2. VisualStudio. *Wikipedia*.

URL: https://uk.wikipedia.org/wiki/Microsoft\_Visual\_Studio. (дата звернення 30.11.2022)

1. Алгоритми сортування. *Wikipedia*.

URL: https://uk.wikipedia.org/wiki/Алгоритм\_сортування. (дата звернення 30.11.2022)

1. Онлайн калькулятор сортування. *Calculators.Vip*. URL: https://calculators.vip/ua/sortirovka-chisel-po-ubyvaniyu-ili-vozrastaniyu/.

(дата звернення 30.11.2022)

1. .NET. *Wikipedia*.

URL: https://uk.wikipedia.org/wiki/.NET\_Framework. (дата звернення 30.11.2022)

1. Сортування вибором. *Purecodecpp*.

URL: https://habr.com/ua/post/181271/. (дата звернення 30.11.2022)

1. Швидке сортування. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Швидке\_сортування. (дата звернення 30.11.2022)
2. Сортування злиттям. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Сортування\_злиттям. (дата звернення 30.11.2022)
3. Сортування бульбашкою. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Сортування\_бульбашкою. (дата звернення 30.11.2022)
4. Сортування включенням. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Сортування\_включенням. (дата звернення 30.11.2022)