МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра

автоматизованих систем обробки інформації і управління

**КУРСОВА РОБОТА**

з дисципліни

“ Об’єктно-орієнтоване програмування ”

на тему

“ Векторний графічний редактор ”

Студента 2 курсу ІП-54 групи

напряму підготовки «Програмна інженерія»

спеціальності «Програмне забезпечення систем»

Литвинюк Д.Р.

Керівник Порєв В.М.

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ – 2016 рік

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 76 сторінки, 24 рисунки, 11 таблиць.

Об’єкт дослідження: зовнішнє упорядкування масивів цілих чисел.

Метою курсової роботи є розробка комплексу програм для розв’язання задачі зовнішнього упорядкування масивів розміром не менше 50000 цілочисельних елементів.

Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний алгоритм для кожного методу.

Виконана програмна реалізація зовнішнього сортування даних трьома методами: пряме злиття, природнє злиття, збалансоване багатошляхове злиття.

ЗОВНІШНЄ СОРТУВАННЯ, МЕТОД ПРЯМОГО ЗЛИТТЯ, ПРИРОДНЬОГО ЗЛИТТЯ, ЗБАЛАНСОВАНОГО БАГАТОШЛЯХОВОГО ЗЛИТТЯ.

зміст

[Вступ 6](#_Toc451981691)

[1 постановка задачі 7](#_Toc451981692)

[2 теоретичні відомості 8](#_Toc451981693)

[2.1 Метод прямого злиття 8](#_Toc451981694)

[2.2 Метод природнього злиття 9](#_Toc451981695)

[2.3 Метод збалансованого багатошляхового злиття 9](#_Toc451981696)

[3 опис алгоритмів 11](#_Toc451981697)

[3.1 Загальний алгоритм: 11](#_Toc451981698)

[3.2 Алгоритм сортування прямим злиттям. 13](#_Toc451981699)

[3.3 Алгоритм сортування методом природнього злиття. 16](#_Toc451981700)

[3.4 Алгоритм сортування сбалансованим багатошляховим злиттям. 18](#_Toc451981701)

[4 опис програмного забезпечення 23](#_Toc451981702)

[4.1 Функціональна структура програмного забезпечення 23](#_Toc451981703)

[4.2 Опис функцій частин програмного забезпечення. 23](#_Toc451981704)

[4.2.1 Функції користувача. 23](#_Toc451981705)

[4.2.2 Стандартні функції 30](#_Toc451981706)

[5 Тестування програмного забезпечення 32](#_Toc451981707)

[5.1 План тестування. 32](#_Toc451981708)

[5.2 Приклади 32](#_Toc451981709)

[6 Інструкція користувача 37](#_Toc451981710)

[6.1 Робота с програмою 37](#_Toc451981711)

[6.2 Формат вхідних даних. 39](#_Toc451981712)

[6.3 Системні вимоги: 39](#_Toc451981713)

[7 аналіз і узагальнення результатів 40](#_Toc451981714)

[Висновки 43](#_Toc451981715)

[Перелік Посилань 44](#_Toc451981716)

[Додаток А Технічне завдання 45](#_Toc451981717)

[Додаток Б Тексти програмного коду 48](#_Toc451981718)

Вступ

В курсовій роботі реалізується програма, призначена для створення векторних зобрражень.

Головна мета КР полягає у закріпленні, поглибленні та узагальненні базових теоретичних знань, отриманих під час вивчення дисципліни «Об’єктно-орієнтоване програмування», їх застосуванні до комплексного вирішення конкретного фахового завдання.

Основними цілями роботи є:

* надбання досвіду роботи з літературними та фондовими матеріалами, вміння узагальнювати та аналізувати наукову інформацію, виробляти власне ставлення до наукової чи практичної проблеми;
* набуття навичок використання основ алгоритмізації та програмування на алгоритмічних мовах високого рівня з використанням принципів об’єктно-орієнтованого програмування;
* набуття теоретичних знань та практичних навичок в області використання сучасних систем створення програмного забезпечення та освоєння принципів та методів сучасних технологій програмування;
* проведення ґрунтовного аналізу отриманих результатів і формування змістовних висновків стосовно їх якості.

У курсовій роботі можна створювати 7 фігур:

* лінію;
* штрих-пунктирну лінію;
* прямокутник;
* еліпс;
* куб;
* циліндр;
* ромб.

# постановка задачі

Користувач вводить зображення елеметів, які складають сцену. Елементами можуть бути : лінія, штрих-пунктирна лінія, прямокутник, еліпс, куб, циліндр, ромб. Програма повинна відповідати таким вимогам:

* + Введення кожного елементу виконується мишею після натискування відповідної кнопки на Toolbar.
  + Стрічка Toolbar розташовується у головному вікні під меню. Потрібно передбачити підказки (Tooltips) для кожної кнопки.
  + Потрібно передбачити скролінг по горизонталі та вертикалі, щоб забезпечити створення зображення розмірами 2000×2000. Передбачити коректне реагування на можливу зміну розмірів головного вікна.
  + Сцена може створюватися заново, або завантажуватися з текстового файлу.
  + Формат файлів сцен – текстовий.
  + Зміна кольору заливки та контуру.

# Обгрунтування проектного рішення

В даній програмі, я використовую такі класи:

* Клас, який є обробником основних повідомлень,  
  важливих для редагування: **ShapeObjectEditor**.
* Абстракний клас, від якого наслідуються усі класи графічні об’єкти: **Shape**.
* Класи, графічі об’єкти, які наслідуються від **Shape**: **EllipseShape, RectShape, RombShape, PuncktLineShape, CubeShape, LineShape, CilinderShape**.
* Абстракний клас, від якого наслідуються усі класи редагування об’єктів: **Editor**
* Базовий клас для забезпечення поліморфізму редагування об’єктів, який наслідується від Editor: ShapeEditor
* Класи редагування об’єктів, які наслідуються від ShapeEditor: EllipseEditor, RectEditor, RombEditor, PuncktLineEditor, CubeEditor, LineEditor, CilinderEditor.
* Клас, який відповідає за скроллінг: ScrollOperations.

Також я використовую модуль для панелі інструментів ToolBar.

На початку роботи программи ствоорюється об’єкт класу ShapeObjectEditor. Потім при виборі на панелі інструментів чи у меню певної фігури, створюється об’єкт класу ShapeEditor, усередені ShapeObjectEditor. Потім оброблюється повідомлення користовуча, які оброблює спочатку ShapeObjectEditor, а потім ShapeEditor. При натиснені лівої кнопки миші і до її відпущення малюється «гумовий слід» фігури, а потім створюється об’кт класу Shape і додається в кінець массиву графічних об’єктів. Згодом відбувається відображення оновленого масиву фігур. Також ShapeEditor та ShapeObjectEditor відповідають за запис у файл та читання з файлу ім’я яких передаються з обробника повідомлення,а також змінну кольорів, які теж передаються з обробника повідомленнь. Діалогові вікна для запису у файл, читанню з файлу та змінну кольорів, викликаються з обробника повідомленнь.

# Аналіз можливих варіантів рішення завдання.

Існує багато можливих варіантів рішення задачі створення векторного редактура на С++ з використанням Windows API. Знаходження оптимального рішення обов’зково спричинить палку суперечку між прогрмістами. Розглянемо деякі частини програми для яких очевидні декілька варіантів реалізації, які відрізняються від моєї:

* Можна було не створювати клас ShapeObjectEditor, а створювати об’єкт ShapeEditor всередині обробника повідомленнь, але це збільшило б розмір коду головного файлу програми і зробило б не зручним його читання.
* Можна було створити окрему функція для створення підказок(Tooltips).
* Можна було створити окремі класи для керування діалоговими вікнами запису у файл, читанням з файлу та змінною кольорів.
* Можна було не створювати клас, який відповідає за скроллінг, а реалізувати скроллінг усередині обробника повідомленнь.
* Можна було використати множине наслідування для створення куба, циліндра та ромба але це спричинило б проблем з їх заливкою.
* Можна було використати для збереження фігур вектор, а не масив.

# теоретичні положення

**Ве́кторна гра́фіка** — створення зображення в комп'ютерній графіці з сукупності геометричних примітивів — (точок, ліній, кривих, полігонів), тобто об'єктів, які можна описати математичними виразами.

**Лі́нія**— геометричний об'єкт, геометричне місце точок, що задовільняє певне рівняння.

**Прямоку́тник** — це чотирикутник, усі кути якого прямі[1]. Протилежні сторони прямокутника рівні. Є окремим випадком паралелограма[1].

**Еліпсом** називається множина всіх точок площини, для кожної з яких сума відстаней до двох даних точок F1 і F2 цієї площини є величина стала, більша за відстань між F1 і F2.

**Куб** або **гексаедр** — правильний многогранник, кожна грань якого є квадратом. Окремий випадок паралелепіпеда і призми.

**Цилі́ндр** — геометричне тіло, обмежене замкнутою циліндричною поверхнею і двома паралельними площинами, що перетинають її.

**Ромб** — це паралелограм, у якого всі сторони рівні.

**Windows Api** (application programming interfaces) — загальне найменування для цілого набору базових функцій інтерфейсів програмування застосунків операційних систем сімейства Windows корпорації Майкрософт. Є найпрямішим способом взаємодії застосунків з Windows. Для створення програм, що використовують Windows API, Майкрософт випускає SDK, який називається Platform SDK і містить документацію, набір бібліотек, утиліт і інших інструментальних засобів.

**Те́кстовий файл** — форма подання послідовності символів у комп'ютері, де кожен символ із задіяного набору символів кодується одним байтом чи послідовністю двох, трьох і т. д. байтів. На відміну від терміна «текстовий формат», що характеризує вміст даних, термін «текстовий файл» стосується файлу та характеризує його як контейнер, який зберігає такі дані.

**Діало́гове вікно́** — особливий тип вікна, яке задає запитання і дозволяє вибрати варіанти виконання дії, або ж інформує користувача. Діалогові вікна зазвичай відображаються тоді, коли програмі або операційній системі для подальшої роботи потрібна відповідь.

**Пане́ль інструме́нтів** (англ. toolbar) — елемент графічного інтерфейсу користувача, призначений для розміщення на ньому кількох інших елементів.

# ОПИС РЕАЛІЗАЦІЇ ПРОЕКТУ

Таблиця 5.1 – Опис класів

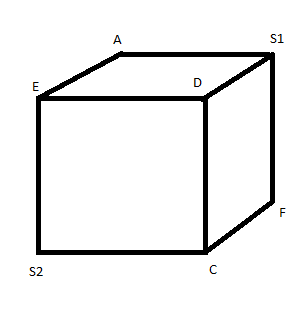
|  |  |
| --- | --- |
| Ім’я | Опис |
| **Клас ShapeObjectEditor** | |
| Поля | |
| ShapeEditor \*pse | Використовується для створення фігур |
| bool onLBdown = false | Кнопка натиснута |
| bool onLBup = false | Кнопка відпущена |
| Методи | |
| void StartRombEditor(COLORREF,COLORREF); | створює новий об'єкт RombEditor |
| void StartLineEditor(COLORREF, COLORREF); | створює новий об'єкт LineEditor |
| void StartRectEditor(COLORREF, COLORREF); | створює новий об'єкт RectEditor |
| Void StartEllipseEditor(COLORREF, COLORREF); | створює новий об'єкт EllipseEditor |
| void StartPuncktLineEditor(COLORREF,COLORREF); | створює новий об'єкт PuncktLineEditor |
| void StartCubeEditor(COLORREF, COLORREF) | створює новий об'єкт CubeEditor |
| void StartCilinderEditor(COLORREF, COLORREF) | створює новий об'єкт CilinderEditor |
| void StartWriteFile(TCHAR\*); | виконується запис у файл |
| void StartReadFile(TCHAR\*); | виконується зчитування з файлу |
| void Reset(HWND); | перевантаження |
| void OnLBdown(HWND); | функція орбробки повідомлення натиснення лівої клавіши миші |
| void OnLBup(HWND, int xk, int yk); | функція орбробки повідомлення відпусксання лівої клавіши миші |
| void OnMosuseMove(HWND) | функція орбробки повідомлення руху миші |
| void OnPaint(HWND,HDC, int xk, int yk) | функція малювання фігур |
| void OninitMenuPopup(HWND, WPARAM) | позначає вибраний елемент у меню |
| void PressButton(HWND) | позначає вибраний елемент на панелі інструментів |
| void CreateNewScene() | створення нової сцени |
| **Клас Editor** | |
| Методи | |
| Virtual void OnLBdown(HWND) = 0; | функція орбробки повідомлення натиснення лівої клавіши миші |
| Virtual void OnLBup(HWND, int xk, int yk) = 0; | функція орбробки повідомлення відпусксання лівої клавіши миші |
| Virtual void OnMosuseMove(HWND) = 0 | функція орбробки повідомлення руху миші |
| Virtual void OnPaint(HWND,HDC, int xk, int yk) = 0 | функція малювання фігур |
| **Клас ShapeEditor** | |
| Поля | |
| static int curr\_length | поточна довжина масиву pcshape |
| static Shape \*pcshape[MY\_SHAPE\_ARRAY\_SIZE] | масив фігур |
| POINT pt\_old; | початкові координати |
| POINT pt\_start | кінцеві координати |
| COLORREF brColor | колір заливки |
| COLORREF penColor | колір контуру |
| Методи | |
| Void OverMeesage() | повідомлення про закінчення місця у масиві pcshape |
| Void CreateNewScene() | створення нової сцени |
| void StartWriteFile(TCHAR\*); | виконується запис у файл |
| void StartReadFile(TCHAR\*); | виконується зчитування з файлу |
| void OnLBdown(HWND); | функція орбробки повідомлення натиснення лівої клавіши миші |
| void OnLBup(HWND, int xk, int yk); | функція орбробки повідомлення відпусксання лівої клавіши миші |
| void OnMosuseMove(HWND) | функція орбробки повідомлення руху миші |
| void OnPaint(HWND,HDC, int xk, int yk) | функція малювання фігур |
| Virtual void OninitMenuPopup(HWND, WPARAM) | позначає вибраний елемент у меню |
| Virtual void PressButton(HWND) | позначає вибраний елемент на панелі інструментів |
| **Клас Shape** | |
| Поля | |
| long xs1, ys1, xs2, ys2; |  |
| std::string type; |  |
| COLORREF brColor | колір заливки |
| COLORREF penColor | колір контуру |
| Методи | |
| void Set(long x1,long y1, long x2, long y2,COLORREF,COLORREF,int xk,int yk); | ініціалізу об'єкт shape |
| virtual void Show(HDC,int xk, int yk) = 0; | малює об'єкт shape |
| std::string RetData(); | повертає дані про об'єкт shape |

Розглянемо, як створюються різні графічні об’єкти методом Show().

Для лінії та штрих-пунктирної лінії, використовуються стандартні фукції MoveToEx(початкова точка) LineTo(кінцева точка),

Для прямокутника та еліпса використовуються стандартні фукції Rectangle та Ellipse, які потребують лише координатии двох протилежних кутів.

Для малювання куба я використовую 3 полігони, для яких потрібно 7 точок. Маючи координати 2 точок S1 та S2 які задає користувач, я вираховую координати 5 інших.



xs\_D = xs\_1 - (xs\_1 - xs\_2) / 3,

ys\_D = ys\_1 + (ys\_2 - ys\_1) / 3,

xs\_A = xs\_2 + (xs\_1 - xs\_2) / 3,

ys\_A = ys\_1,

xs\_E = xs\_2,

ys\_E = ys\_D,

xs\_C = xs\_D,

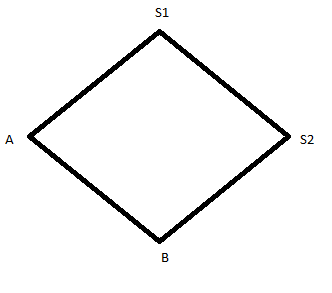
ys\_C = ys\_2,

xs\_F = xs\_1,

ys\_F = ys\_2 - (ys\_2 - ys\_1) / 3;

Відмальовуються 3 полігони: EA(S1)D, ED(S2)C, SC(S1)F.

Для малювання ромба я використовую 1 полігон, для цьгог маючи точки S1 та S2 прораховую координати 2 інших.



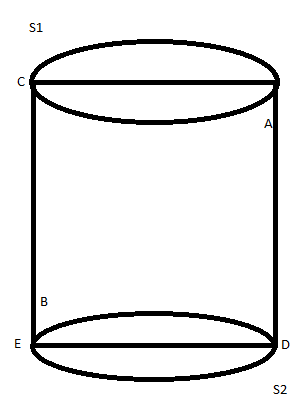
xs\_A = xs\_1 - (xs\_2 - xs\_1),

ys\_A = ys\_2,

xs\_B = xs\_1,

ys\_B = ys\_1 + ((ys\_2 - ys\_1)\* 2);

Для малювання циліндра я використовую 2 еліпси, прямокутник та лінію.



Спочатку малюється еліпс за кутами B(S2), потім прямокутник CD, на нього накладається еліпс AS1, а нижня сторона прямокутника замальовується прямою ED.

# опис алгоритмів

## Загальний алгоритм:

tabStictic- вклюячає вкладки по трьом методам;

Merge- вкладка для метода прямого злиття;

Nature- вкладка для метода природнього злиття;

Multi-вкладка для метода збалансованого багатошляхового злиття;

textBoxN- поле для вводу кількості елементів.

textBoxMin-поле для вводу мінімального елемента.

timeMerge, porMerge, pereMerge, timeNature, porNature, pereNature, timeMulti, porMulti, pereMulti-текстові поля у які виводиться статистика по кожному елементу.

buttonStart, buttonStartShow, button\_resNature, button\_showNature, button\_resMerge, button\_showMerge, button\_resMulti, button\_showMulti- кнопки керування.

1 Початок.

2 Ініціалізація об’єктів: TabControl: tabStatistic; TabItem: Merge, Nature, Multi; TextBox: textBoxN, textBoxMax, textBoxMin, timeMerge, porMerge, pereMerge, timeNature, porNature, pereNature, timeMulti, porMulti, pereMulti; Button: buttonStart, buttonStartShow, button\_resNature, button\_showNature, button\_resMerge, button\_showMerge, button\_resMulti, button\_showMulti.

3 Генерування файлу

3.1 Введеня користувачем кількості елементів у вхідний файл.

3.2 Введення користувачем максимального значення для генерації файлу.

3.3 Введення користувачем мінімального значення для генерації файлу.

4 ЯКЩО натиснути на будь-яку кнопку крім кнопки buttonStart ТО буде виведене повідомлення про помилку.

5 Натиснення на кнопку buttonStart.

6 Перевірка на коректність вхідних значень

6.1 ЯКЩО введений користувачем рядок кількість елементів не відповідає типу цілочисельне число більше за 0 та менше за 20 000 000 ТО буде виведене повідомлення про помилку.

6.2 ЯКЩО введений користувачем рядок максимального значення для генерації файлу не відповідає типу цілочисельне число більше за -2 147 483 648 та менше за 2 147 483 648 ТО буде виведене повідомлення про помилку.

6.3 ЯКЩО введений користувачем рядок мінімального значення для генерації файлу.не відповідає типу цілочисельне число більше за -2 147 483 648 та менше за 2 147 483 648 ТО буде виведене повідомлення про помилку.

6.4 ЯКЩО мінімальне значення для генерації файлу більше за максимальне значення для генерації файлу ТО буде виведене повідомлення про помилку.

7 ЯКЩО вхідні данні коректні ТО Генерація вхідного файлу зі вказаними вхідними даними.

8 ЯКЩО натиснути на buttonStartShow ТО буде відображений вхідний файл.

9 ЯКЩО натиснути на button\_showNature ТО буде виведене повідомлення про помилку.

10 ЯКЩО натиснути на button\_showMulti ТО буде виведене повідомлення про помилку.

11 ЯКЩО вибрати вкладку Merge на tabStictic ТО

11.1 ЯКЩО натиснути на button\_showMerge, а сортування природнім злиттям ще не було виконано ТО буде виведене повідомлення про помилку, ІНАКШЕ буде виведений відсортований файл прямим злиттям.

11.2 ЯКЩО натиснути на button\_resMerge ТО

11.2.1 Відбудеться сортування методом прямого злиття

11.2.2 У timeMerge буде виведено час роботи сортування.

11.2.3 У porMerge буде виведено кількість порівнянь у прямому злитті.

11.2.4 У pereMerge кількість перестановок, які відбулися під час сортування прямим злиттям.

12 ЯКЩО вибрати вкладку Nature на tabStictic ТО

12.1 ЯКЩО натиснути на button\_showNature, а сортування природнім злиттям ще не було виконано ТО буде виведене повідомлення про помилку, ІНАКШЕ буде виведений відсортований файл природнім злиттям.

12.2 ЯКЩО натиснути на button\_resNature ТО

12.2.1 Відбудеться сортування методом природнє злиття.

12.2.2 У timeNature буде виведено час роботи сортування.

12.2.3 У porNature буде виведено кількість порівнянь у природньому злитті.

12.2.4 У pereNature кількість перестановок, які відбулися під час сортування природнім злиттям.

13 ЯКЩО вибрати вкладку Multi на tabStictic ТО

13.1 ЯКЩО натиснути на button\_showMulti, а сортування збалансованим багатошляховим злиттям ще не було виконано ТО буде виведене повідомлення про помилку, ІНАКШЕ буде виведений відсортований файл збалансованим багатошляховим злиттям.

13.2 ЯКЩО натиснути на button\_resMulti ТО

13.2.1 Відбудеться сортування методом збалансоване багатошляхове злиття.

13.2.2 У timeMulti буде виведено час роботи сортування.

13.2.3 У porMulti буде виведено кількість порівнянь у збалансованому багатошляховому злитті.

13.2.4 У pereNature кількість перестановок, які відбулися під час сортування збалансованим багатошляховим злиттям.

14 ЯКЩО натиснути на кнопку «Зачинити» ТО програма закінчить свою роботу.

15 Кінець.

## Алгоритм сортування прямим злиттям.

a1,a2- використовуються для зчитування даних с файлу і запису даних у файли.

k- довжина серій.

count- кількість елементів у файлі.

i1,j1- лічильники.

f- файл, дані якого впорядковують.

f1,f2-допоміжні файли.

1 Початок.

2 Створення та ініціалізація змінних: цілочисельні a1=0, a2=0, k=1, count=0, i1=0, j1=0; файли f,f1,f2.

3 Перевірка можливості зчитувати файл.

3.1 ЯКЩО файл не можливо зчитати вивести повідомлення про помилку, кінець, ІНАКШЕ перейти на Крок 4.

4 Підрахунок кількості елементів у файлі.

4.1 Відкриття вхідного файлу f для читання

4.2 Цикл с передумовою ( не кінець файлу f)

4.2.1 Зчитування елемента а1 с файлу f

4.2.2 Збільшення лічильника елементів count на 1.

4.3 Закриття файла f.

5 Головний результуючий цикл с передумовою (k<count)

5.1 Відкриття файлу f для читання, відкрити допоміжних файлів f1,f2 для запису.

5.2 Зчитування елемента а1 с файлу f

5.3 Цикл с передумовою (не кінець файлу f)

5.3.1 Цикл заповнення файлу f1(i=0…k) і ( не кінець файлу f)

5.3.1.1 Запис у файл f1 число a1

5.3.1.2 Зчитування елемента а1 с файлу f

5.3.2 Цикл заповнення файлу f2(i=0…k) і (не кінець файлу f)

5.3.2.1 Запис у файл f2 число a1

5.3.2.2 Зчитування елемента а1 с файлу f

5.4 Закриття файлів f, f1, f2.

5.5 Відкриття файлів f1,f2 для читання, відкрити файл f для запису.

5.6 ЯКЩО не кінець файлу f1 ТО Зчитування елемента а1 с файлу f1.

5.7 ЯКЩО не кінець файлу f2 ТО Зчитування елемента а2 с файлу f1.

5.8 Цикл с передумовою((не кінець файлу f1) і (не кінець файлу f2)) для запису послідовностей з файлів f1 та f2 у файл f

5.8.1 Присвойти зміним i1=0,j1=0.

5.8.2 Цикл с передумовою (поки не кінець файлу f1) і (поки не кінець файлу f2) і (i1<k) і (j1<k)) для порівняння елементів у послідовностях

5.8.2.1 ЯКЩО(a1<a2), ТО

5.8.2.1.1 Запис у файл f число a1

5.8.2.1.2 Зчитування елемента а1 с файлу f1

5.8.2.1.3 Збільшення лічильника i1 на 1

5.8.2.2 ІНАКШЕ

5.8.2.2.1 Запис у файл f число a2

5.8.2.2.2 Зчитування елемента а2 с файлу f2

5.8.2.2.3 Збільшення лічильника j1 на 1

5.8.3 Цикл с передумовою((i1<k) і (не кінець файлу f1)) для дозапису у файл f кінці послідовностей з f1

5.8.3.1 Запис у файл f число a1

5.8.3.2 Зчитування елемента а1 с файлу f1

5.8.3.3 Збільшення лічильника i1 на 1

5.8.4 Цикл с передумовою((j1<k) і (не кінець файлу f2)) для дозапису у файл f кінці послідовностей з f2

5.8.4.1 Запис у файл f число a2

5.8.4.2 Зчитування елемента а2 с файлу f2

5.8.4.3 Збільшення лічильника j1 на 1

5.9 Цикл с передумовою(не кінець файлу f1) для дозапису у файл f неповних послідовностей з f1

5.9.1 Запис у файл f число a1

5.9.2 Зчитування елемента а1 с файлу f1

5.10 Цикл с передумовою(не кінець файлу f2) для дозапису у файл f неповних послідовностей з f2

5.10.1 Запис у файл f число a2

5.10.2 Зчитування елемента а2 с файлу f2

5.11 Збільшити лічильник k у два рази.

5.12 Закрити файли f, f1, f2.

6 Кінець.

## Алгоритм сортування методом природнього злиття.

a1,a2- використовуються для зчитування даних с файлу і запису даних у файли.

mark- для визначення файлу, у який потрібно ввести запис.

s1, s2- для перевірки чи виконувався запис у 1 чи другий допомоміжний файл відповідно.

marker- позначає кінець серії.

f- файл, дані якого впорядковують.

f1,f2-допоміжні файли.

1 Початок

2 Створення та ініціалізація змінних: цілочисельних a1=0,a2=0,mark,s1=1,s2=1; marker=Int32.MaxValue; файлів f, f1, f2.

3 Перевірка можливості зчитувати файл.

3.1 ЯКЩО файл не можливо зчитати вивести повідомлення про помилку, кінець, ИНАКШЕ перейти на Крок 4.

4 Цикл с передумовою(s1>0 і s2>0) у якому виконується сортування .

4.1 Присвоювання змінним значення mark=1,s1=0,s2=0,fl=true,fl1=true.

4.2 Відкриття файлу f для читання, відкрити допоміжних файлів f1,f2 для запису.

4.3 ЯКЩО(не кінець файлу f) ТО

4.3.1 Зчитування елемента а1 с файлу f.

4.3.2 Запис у файл f1 число a1.

4.3.3 Збільшення лічильника s1 на 1.

4.3.4 ЯКЩО(не кінець файлу f) ТО зчитування елемента а1 с файлу f.

4.4 Цикл с передумовою (поки не кінець файлу f) у якому формуються серії і записуються у файл f1 чи f2.

4.4.1 ЯКЩО(а2<a1) ТО

4.4.1.1 Якщо (mark==1) ТО

4.4.1.1.1 Запис у файл f1 marker.

4.4.1.1.2 Присвоювання змінній значення mark=2.

4.4.1.2 ІНАКШЕ

4.4.1.2.1 Запис у файл f2 marker.

4.4.1.2.2 Присвоювання змінній значення mark=1.

4.4.2 ЯКЩО (mark==1) ТО

4.4.2.1 Запис у файл f1 число a2.

4.4.2.2 Збільшення лічильника s1 на 1.

4.4.3 ІНАКШЕ

4.4.3.1 Запис у файл f2 число a2.

4.4.3.2 Збільшення лічильника s2 на 1.

4.4.4 Привласнити a1=a2.

4.4.5 Зчитування елемента а2 с файлу f.

4.5 ЯКЩО(s2>0 і mark==2), ТО Запис у файл f2 marker.

4.6 ЯКЩО(s1>0 і mark==1), ТО Запис у файл f1 marker.

4.7 Закрити файли f,f1,f2.

4.8 Відкриття файлів f1,f2 для читання, відкрити файл f для запису.

4.9 ЯКЩО(не кінець файлу f1) ТО зчитування елемента а1 с файлу f1.

4.10 ЯКЩО(не кінець файлу f2) ТО зчитування елемента а2 с файлу f2.

4.11 Цикл с передумовою((поки не кінець файлу f1) і (поки не кінець файлу f2)) у якому порівнюються елементи серій з файлів f1 i f2 і записуються у файл f.

4.11.1 Цикл с передумовою ((a1 != marker) i (a2 !=marker)) у якому порівнюються елементи серій з файлів

4.11.1.1 ЯКЩО (a1<=a2), ТО

4.11.1.1.1 Запис у файл f число a1

4.11.1.1.2 Зчитування елемента а1 с файлу f1.

4.11.1.2 ІНАКШЕ

4.11.1.2.1 Запис у файл f число a2.

4.11.1.2.2 Зчитування елемента а2 с файлу f2.

4.11.2 Цикл с передумовою (a1!=marker) для дозапису серії з файлу f1 у файл f

4.11.2.1 Запис у файл f число a1.

4.11.2.2 Зчитування елемента а1 с файлу f1.

4.11.3 Цикл с передумовою (a2!=marker) для дозапису серії з файлу f2 у файл f

4.11.3.1 Запис у файл f число a2.

4.11.3.2 Зчитування елемента а2 с файлу f2.

4.11.4 ЯКЩО(не кінець файлу f1) ТО зчитування елемента а1 с файлу f1.

4.11.5 ЯКЩО(не кінець файлу f2) ТО зчитування елемента а2 с файлу f2.

4.12 Цикл с передумовою ((a1!=marker) і (поки не кінець файлу f1)) для дозапису останьої серії з файлу f1 у файл f.

4.12.1 Запис у файл f число a1

4.12.2 Зчитування елемента а1 с файлу f1.

4.13 Цикл с передумовою ((a2!=marker) і (поки не кінець файлу f2)) для дозапису останьої серії з файлу f2 у файл f.

4.13.1 Запис у файл f число a2

4.13.2 Зчитування елемента а2 с файлу f2.

4.14 Закрити файли f, f1, f2.

5 Кінець.

## Алгоритм сортування сбалансованим багатошляховим злиттям.

n-кількість елементів у вхідному файлі.

m-кількість файлів на які розбивається вхідний файл.

a1,a2- використовуються для зчитування даних с файлу і запису даних у файли.

l,count, tmp- допоміжні змінні.

marker- позначає кінець серії.

namesA[],namesB[]- містять імена файлів, які використовуються у программі.

mark- вказує, які файли треба відкрити для запису, а які для читання.

fl1[]- вказує чи відкриті файли.

flOpen[]- вказує чи закінчилась у файлі поточна серія.

opening[]- вказує чи відкривався файл у поточній фазі.

f- вхідний файл, дані якого впорядковують.

currentFile- допоміжний файл.

writers[]- файли, які відкриті для запису.

readers[]- файли, які відкриті для читання.

1 Початок

2 Створення та ініціалізація змінних: цілочисельних n-вхідна зміна,m=n/19000 + 1, a1=0,a2=0, l=0 ,marker=Int32.MaxValue, count,tmp; цілочисельний масив last розмірністю m; рядок res, nameMain=”ім’я вхідного файла”; масиви рядків namesA, namesB, розмірністю m; файли f, currentFile; масиви файлів writers,readers розмірністю m; логічні mark; масиви логічних типів fl, fl1,flOpen,opening розмірністю m.

3 Розбиття вхідного файла на m.

3.1. Відкриття файлу f для читання.

3.2. Відкриття файлів для запису writers з іменами namesA.

3.3. Встановити для всіх значень fl1[] значення true.

3.4. Зчитування елемента а1 с файлу f.

3.5. Переприсвоїти currentFile=writers[l].

3.6. Запис у файл currentFile число a1

3.7. ЯКЩО(не кінець файлу f) ТО зчитування елемента а2 с файлу f.

3.8. Цикл с передумовою (не кінець файлу f) у якому формуються файли.

3.8.1. ЯКЩО(а1<а2) ТО

3.8.1.1. Запис у файл currentFile число a2

3.8.2. ІНАКШЕ

3.8.2.1. Запис у файл currentFile marker.

3.8.2.2. Збільшення l на 1.

3.8.2.3. ЯКЩО (l>=m) ТО переприсвоїти l=0.

3.8.2.4. Переприсвоїти currentFile=writers[l].

3.8.2.5. Запис у файл currentFile число a2

3.8.3. Присвоїти a1=a2.

3.8.4. Зчитування елемента а2 с файлу f.

3.9. Запис у файл currentFile marker.

3.10. Закрити файл f.

3.11. Закрити файли writers.

4 Виконання сортування

4.1. Присвоїти mark=false,k=0,min=0,count=m.

4.2. Цикл с передумовою(k!=1) у якому виконується сортування .

4.2.1. Присвоїти l=0,tmp=-1,k=0.

4.2.2. Цикл відкриття файлів (i=0…count)

4.2.2.1. Присвоїти fl[i]=true, flOpen[i]=true, opening[i]=false.

4.2.2.2. ЯКЩО(mark==false) ТО

4.2.2.2.1. Відкрити файл writers[i] з іменем namesB[i] для запису

4.2.2.2.2. Відкрити файл readers[i] з іменем namesA[i] для читання

4.2.2.3. ІНАКШЕ

4.2.2.3.1. Відкрити файл writers[i] з іменем namesA[i] для запису

4.2.2.3.2. Відкрити файл readers[i] з іменем namesB[i] для читання

4.2.3. ЯКЩО(mark==true) ТО mark=false, ІНАКШЕ mark=true.

4.2.4. Цикл зчитування елементів з файлів (i=0…count)

4.2.4.1. ЯКЩО(не кінець файлу readers[i]) ТО зчитування елемента last[i] с файлу readers[i], ІНАКШЕ fl[i]=false

4.2.4.2. ЯКЩО (fl[i]==false) ТО flOpen[i]=false.

4.2.5. Цикл с передумовою(Хочаб одне з значень fl[]=true)

4.2.5.1. Збільшення tmp на 1

4.2.5.2. ЯКЩО (tmp==m) ТО tmp=0.

4.2.5.3. Присвоїти currentFile=writers[tmp].

4.2.5.4. ЯКЩО(opening[tmp]==false) ТО

4.2.5.4.1. Збільшення k на 1.

4.2.5.4.2. Присвоїти opening[tmp]=true.

4.2.5.5. Цикл с передумовою(Хочаб одне з значень flOpen[]=true)

4.2.5.5.1. Цикл (i=0…count)

4.2.5.5.1.1. ЯКЩО ((fl[i]==true) i (flOpen[i]==true)) ТО

4.2.5.5.1.1.1. Присвоїти min=last[i].

4.2.5.5.1.1.2. Присвоїти l=i.

4.2.5.5.1.1.3. Вихід з циклу.

4.2.5.5.2. Цикл (i=l…count) пошук мінімального з масиву last[i].

4.2.5.5.2.1. ЯКЩО ((fl[i]==true) i (flOpen[i]==true)) ТО

4.2.5.5.2.1.1. ЯКЩО(min>last[i]) ТО min=last[i]; l=і;

4.2.5.5.3. Запис у файл currentFile число last[l].

4.2.5.5.4. ЯКЩО(не кінець файлу readers[l]) ТО зчитування елемента last[l] с файлу readers[l], ІНАКШЕ flOpen[l]=false.

4.2.5.5.5. ЯКЩО(last[l]==marker) ТО flOpen[l]=false.

4.2.5.6. Запис у файл currentFile marker.

4.2.5.7. Цикл (i=0…count)

4.2.5.7.1. ЯКЩО(не кінець файлу readers[i]) ТО зчитування елемента last[i] с файлу readers[i], ІНАКШЕ fl[i]=false.

4.2.5.7.2. ЯКЩО(fl[i]==true) ТО flOpen[i]=true, ІНАКШЕ flOpen[i]=false.

4.2.6. Закриття усіх файлів readers i writers.

4.2.7. Присвоїти count=k.

4.2.8. ЯКЩО (mark==true) ТО res=namesB[0], ІНАКШЕ res=namesA[0].

5 Видалення усіх файлів з іменами namesA, namesB, крім res.

6 Кінець.

# опис програмного забезпечення

## Діаграма класів

Рисунок 4.1- діаграма класів

## Опис функцій частин програмного забезпечення.

### Функції користувача.

Таблиця 4.1- Специфікація функцій користувача

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | Multi\_Sorting\_2 | Bool AnyTrue(bool[]) | Перевіряти чи є в масиві  елемент зі значенням true | Bool[] grad-Вхідний масив | Якщо такий елемент є то true, інакше false |
| 2 | Multi\_Sorting\_2 | CreateNames | Створює ім’я для допоміжних файлів | - | - |
| 3 | Multi\_Sorting\_2 | DeleteFiles | Видаляє допоміжні файли | - | - |
| 4 | Multi\_Sorting\_2 | DivideFiles | Розбиває вхідний файл на n допоміжних | - | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продовження таблиці 4.1 | | | | | |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 5 | Multi\_Sorting\_2 | MakeSort | Виконує функції: CreateNam-es, DivideFiles, SortFiles,  DeleteFiles. | - | - |
| 6 | Multi\_Sorting\_2 | SortFiles | Виконує сортування збалансова-ним багатошляховим злиттям | - | - |
| 7 | Multi\_Sorting\_2 | Multi\_Sorting\_2  (string name, int count,string nameOut) | Конструкт-ор класу Multi\_Sorting\_2 | name- ім’я вхідного файлу,  count- кількість елементів у файлу,  nameOut- ім’я вихідного файлу | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продовження таблиці 4.1 | | | | | |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 8 | Nature\_Sorting | MakeSort | Виконує сортування природнім злиттям | - | - |
| 9 | Nature\_Sorting | Nature\_Sorting  (string name,string nameOut) | Конструкт-ор класу Nature\_Sorting | name- ім’я вхідного файлу,  nameOut- ім’я вихідного файлу | - |
| 10 | Merge\_Sort | MakeSort | Виконує сортування прямим злиттям | - | - |
| 11 | Merge\_Sort | Merge\_Sort(  string name,int n,string nameOut | Конструкт-ор класу Merge\_Sort | name- ім’я вхідного файлу,  n- кількість елементів у файлу,  nameOut- ім’я вихідного файлу | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продовження таблиці 4.1 | | | | | |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 12 | Generate\_files | Generate | Заповнює файл випадковими числами | - | - |
| 13 | Generate\_files | Generate\_files (string nameF1, string nameF2, string nameF3,int n,int maxi,int min) | Конструкт-ор класу Generate\_fi-les | nameF1- ім’я вхідного файлу для прямого злиття,  nameF2- ім’я вхідного файлу для природнього злиття,  nameF3- ім’я вхідного файлу для ЗБШЗ,  n-кількість елементів у файлах  max- максимальне для генерації,  min- мінімальне | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продовження таблиці 4.1 | | | | | |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 14 | DeBinaireFile | CreateTXT() | Переписує дані з бінарного файлу в текстовий | - | - |
| 15 | DeBinaireFile | CreateTXT(int n) | Переписує n перших елементів з бінарного файлу в текстовий | - | - |
| 16 | DeBinaireFile | DeBinaireFile  (string inName,string outputName) | Конструк-тор класу | inName- ім’я вхідного файлу,  outputName- ім’я вихідного файлу | - |
| 17 | Main\_Sort | Create\_Merge | Створює об’єкт класу MergeSort, та викликає його метод MakeSort() | - | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продовження таблиці 4.1 | | | | | |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 18 | Main\_Sort | CreateMulti | Створює об’єкт класу Multi\_Sort-ing\_2, та викликає його метод MakeSort() | - | - |
| 19 | Main\_Sort | CreateNature | Створює об’єкт класу Nature\_Sorting, та викликає його метод MakeSort() | - | - |
| 20 | Main\_Sort | GenerateFiles | Створює об’єкт класу Generate\_fi-les, та викликає його метод Generate() | - | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продовження таблиці 4.1 | | | | | |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 21 | Main\_Sort | Main\_Sort  (int n,int max,int min) | Конструк-тор класу Main\_Sort | n-кількість елементів у файлах  max- максимальне для генерації,  min- мінімальне для генерації | - |

### Стандартні функції

Таблиця 4.2-Специфікаця стандартних функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 11 | BinaryReader | int32 ReadInt32() | Зчитування елемента файлу типу int32 | - | Число типу int32() |
| 22 | StreamWriter | Void Write(string value) | Запис у файл рядка | Value- рядок, який треба додати до файлу | - |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Продовження таблиці 4.2 | | | | | |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 33 | Random | Int Next(int max) | Генерація випадкового числа | max- найбільше число, яке може бути згенероване | Число типу int32() |
| 54 | BinaryWriter | Void Write(int value) | Запис у файл числа | Value- число, яке треба додати до файлу | - |
| 65 | File | void OpenRead (string path) | Відкриття файла для читання | Path – ім’я файлу | - |
| 76 | File | Create(string path) | Створення файлу с вказаним ім’ям | Path- ім’я файлу | - |
| 77 | Math | Int Abs(int val) | Знаходить модуль числа. | Val- число модуль якого шукає функція | Int -модуль числа |
| 98 | Stopwatch | Elapsed | - | Повертає загальну кількість часу визначену, цим екзкемпляром | Повертає загальну кількість часу визначену, цим екзкемпляром |

# Тестування програмного забезпечення

## План тестування.

Для перевірки коректності роботи програми, виконаємо тестування програми.

1. Тестування правильності введених значень.
   1. Тестування при введені некоректних значень для кількості елементів.
   2. Тестування при введені некоректних значень для максимального заначення.
   3. Тестування при введені некоректних значень для мінімального значеня.
   4. Тестування при спробі розпачати сортування, до того як було згенеровано файл.
   5. Тестування при спробі відкрити вхідний файл до того, як він був згенерований.
   6. Тестування при введенні мінімального значення більшого за максимальне.
2. Тестування коректності роботи методів сортування.
   1. Перевірка коректності роботи методу Прямого злиття.
   2. Перевірка коректності роботи методу Природнього злиття.
   3. Перевірка коректності роботи методу Збалансованого багатошляхового злиття.

## Приклади

У Таблиця 5.1 та на Рисунок 5.1 наведено приклад роботи програми при введені некоректних даних, де після введення некоректних даних, виводиться повідомлення про помилку.

Таблиця 5.1 – Приклад роботи програми при введені некоректної кількості елементів

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення некоректної кількості елементів |
| Початковий стан програми | Відкрите вікно програми |

|  |  |
| --- | --- |
| Продовження таблиці 5.1 | |
| Вхідні дані | Кількість елементів:-100  Максимальне значення:1000  Мінімальне значення: -100 |
| Схема проведення тесту | Заповнення вхідних значень |
| Очікуваний результат | Повідомлення про помилку не коректності даних. |
| Стан програми після проведення випробувань | Видано помилку «Ви ввели не коректні дані, повторіть запис» |

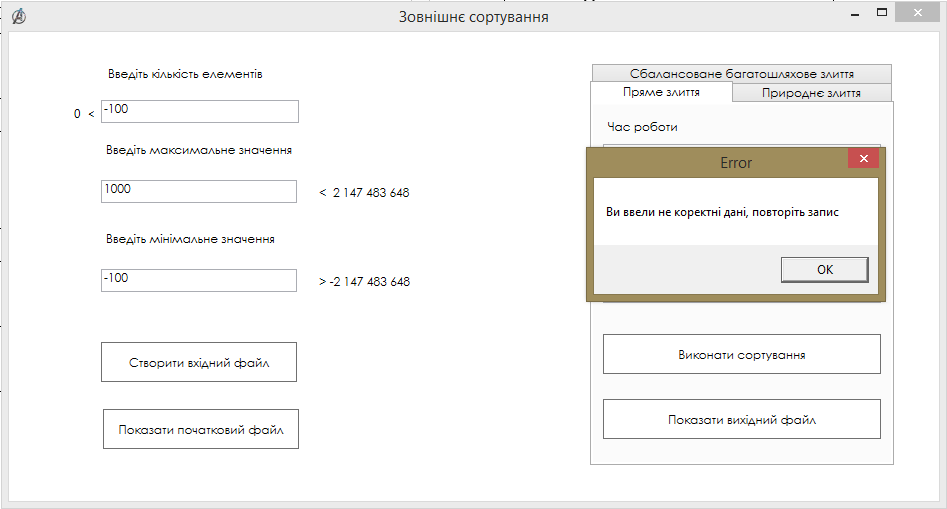
У Таблиця 5.2 та на Рисунок 5.2 наведено приклад роботи програми при введенні некоректгного максимального заначення., де після введення некоректних даних, виводиться повідомлення про помилку.

Рисунок 5.1

Таблиця 5.2-Приклад роботи програми при введені некоректгного максимального заначення

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення некоректгного максимального заначення. |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Кількість елементів: 80000  Максимальне значення: «ждвфіларивполі»  Мінімальне значення: -100 |
| Схема проведення тесту | Заповнення вхідних значень |
| Очікуваний результат | Повідомлення про помилку не коректності даних. |
| Стан програми після проведення випробувань | Видано помилку «Ви ввели не коректні дані, повторіть запис» |

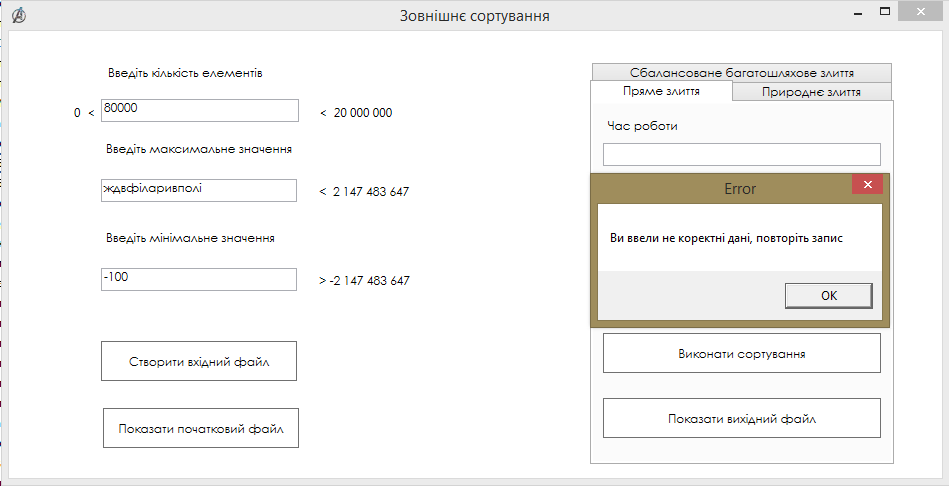


Рисунок 5.2

У Таблиця 5.3 та на Рисунок 5.3 наведено приклад роботи програми при введенні некоректгного мінімального заначення., де після введення некоректних даних, виводиться повідомлення про помилку.

Таблиця 5.3- Приклад роботи програми при введенні некоректгного мінімального заначення

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення некоректгного мінімального заначення. |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Кількість елементів: 80000  Максимальне значення: 20000  Мінімальне значення: -2147483669 |
| Схема проведення тесту | Заповнення вхідних значень |
| Очікуваний результат | Повідомлення про помилку не коректності даних. |
| Стан програми після проведення випробувань | Видано помилку «Ви ввели не коректні дані, повторіть запис» |

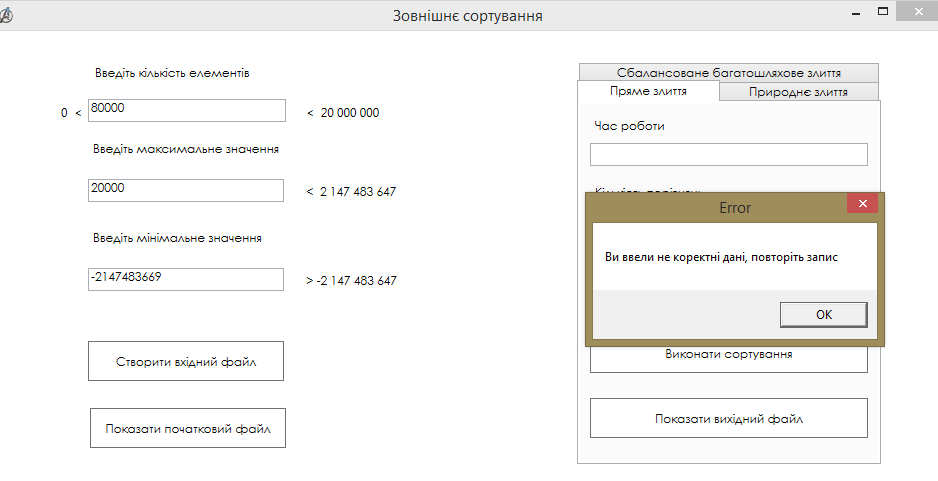


Рисунок 5.3

У Таблиця 5.4 та на Рисунок 5.4 наведено приклад роботи програми при спробі розпочати сортування до того, як було згенеровано вхідний файл.

Таблиця 5.4- Приклад роботи програми при спробі розпочати сортування до того, як було згенеровано вхідний файл.

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість розпачати сортування, до того як було згенеровано файл. |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Кількість елементів: “”  Максимальне значення: “”  Мінімальне значення: “” |
| Схема проведення тесту | Початок сортування одним с методів |
| Очікуваний результат | Повідомлення про помилку не можливості виконати сортування. |

|  |  |
| --- | --- |
| Продовження таблиці 5.4 | |
| Стан програми після проведення випробувань | Видано помилку «Ви не маєте можливості виконати сортування» |

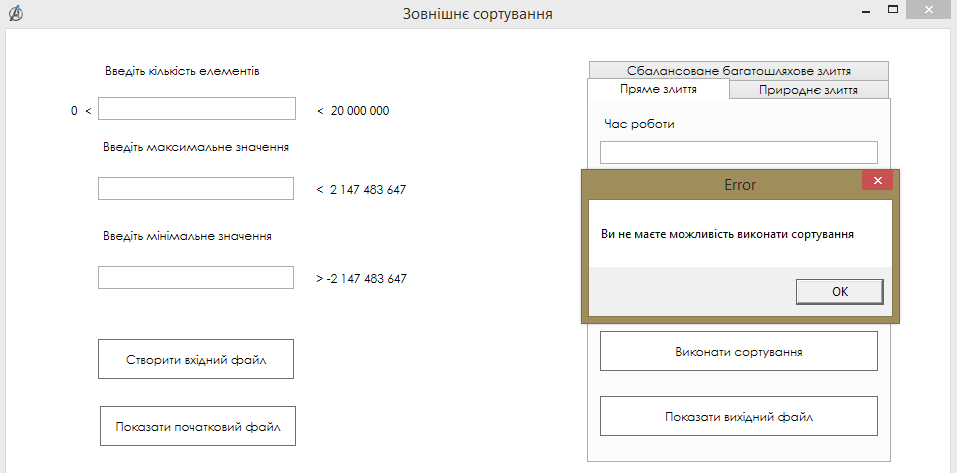


Рисунок 5.4

У Таблиця 5.5 та на Рисунок 5.5 наведено приклад роботи програми при спробі відкрити вхідний файл до того, як він був згенерований.

Таблиця 5.5- Приклад роботи програми при спробі відкрити вхідний файл до того, як він був згенерований.

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість відкрити вхідний файл до того, як він був згенерований |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Кількість елементів: “”  Максимальне значення: “”  Мінімальне значення: “” |
| Схема проведення тесту | Відкриття вхідного файлу |

|  |  |
| --- | --- |
| Продовження таблиці 5.5 | |
|  |  |
| Очікуваний результат | Повідомлення про помилку. |
| Стан програми після проведення випробувань | Видано помилку «Ви не згенерували файл» |

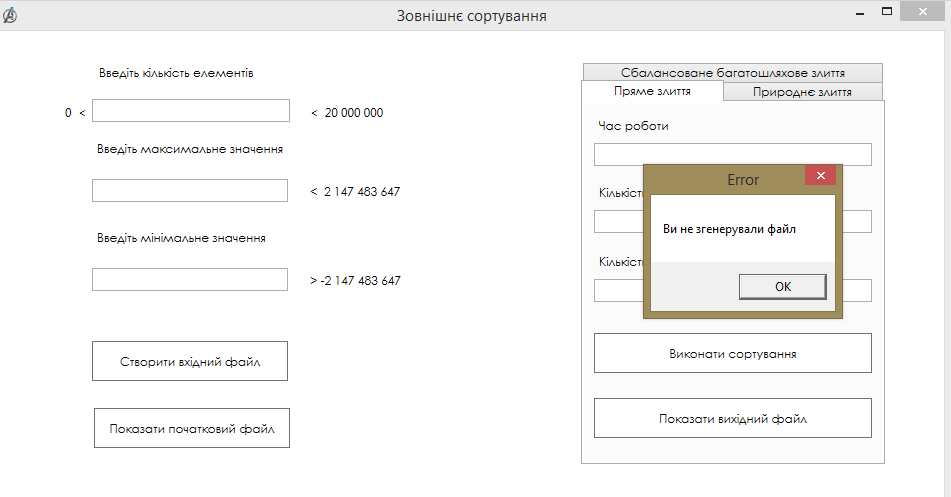


Рисунок 5.5

У Таблиця 5.6 та на Рисунок 5.6 наведено приклад роботи програми при введенні мінімального значення більшого за максимальне, де після введення некоректних даних, виводиться повідомлення про помилку.

Таблиця 5.6- Приклад роботи програми при введенні мінімального значення більшого за максимальне.

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення мінімального значення більшого за максимальне. |
| Початковий стан програми | Відкрите вікно програми |

|  |  |
| --- | --- |
| Продовження таблиці 5.6 | |
| Вхідні дані | Кількість елементів: “80000”  Максимальне значення: “100000”  Мінімальне значення: “200000” |
| Схема проведення тесту | Заповнення вхідних значень |
| Очікуваний результат | Повідомлення про помилку не коректності даних.. |
| Стан програми після проведення випробувань | Видано помилку «Ви ввели не коректні дані, повторіть запис» |

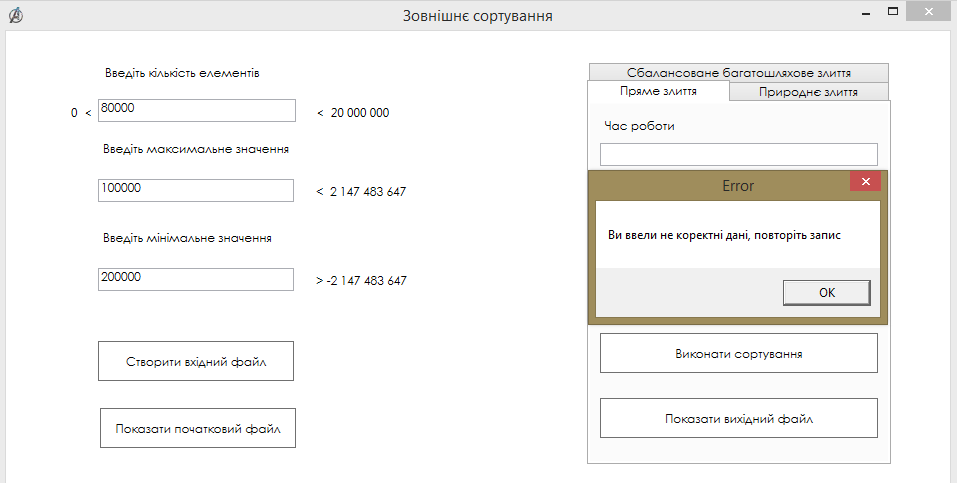


Рисунок 5.6

У таблиці Таблиця 5.7 та на Рисунок 5.7 наведено приклад роботи програми, де перевіряється коректність роботи методу Прямого злиття.

Таблиця 5.7- Приклад роботи програми для перевірки коректності роботи методу Прямого злиття.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи методу прямого злиття. |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Кількість елементів:20  Максимальне значення:100000  Мінімальне значення: -200000 |
| Схема проведення тесту | Заповнення вхідних значень  Генерація вхідного файлу  Виконання методу прямого злиття |
| Очікуваний результат | Створення нового файлу з відсортованими даними вхідного файлу. |
| Стан програми після проведення випробувань | Створився вихідний файл. |

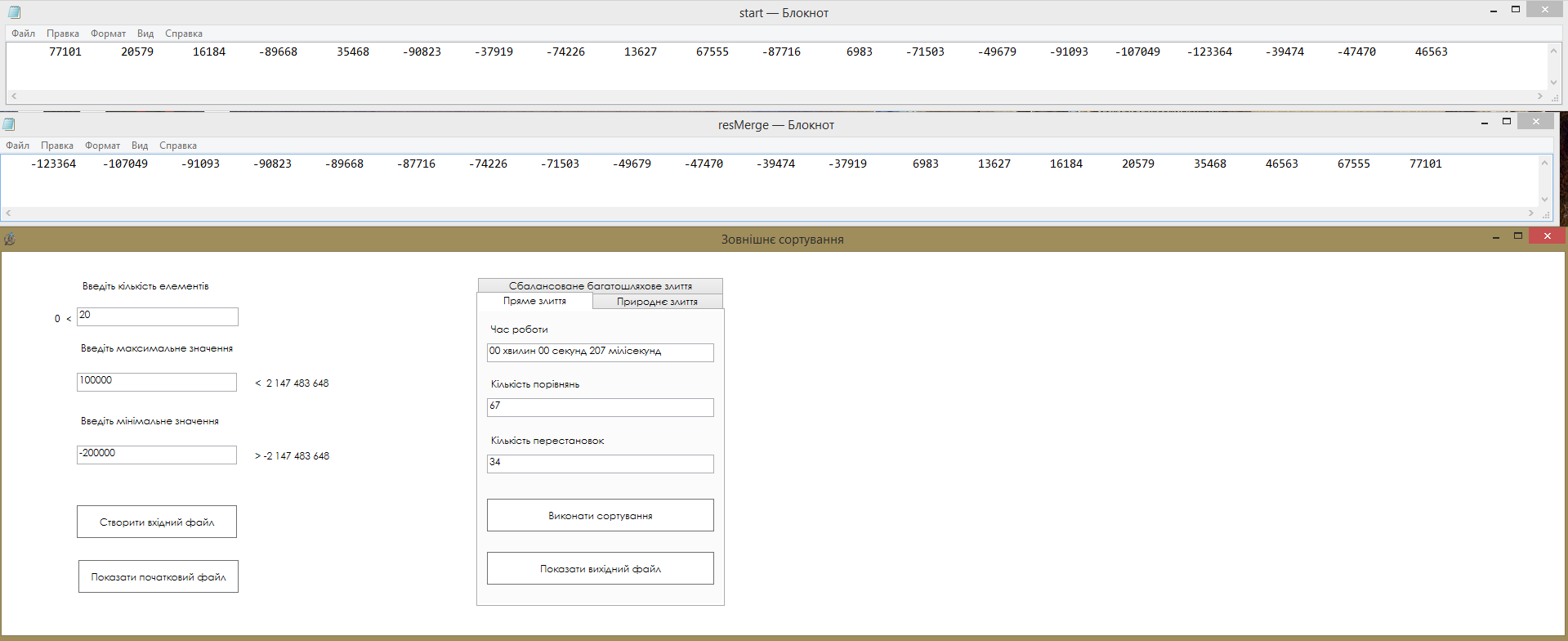


Рисунок 5.7

У Таблиця 5.8 та на Рисунок 5.8 наведено приклад роботи програми, де перевіряється коректність роботи методу Природнього злиття.

Таблиця 5.8- Приклад роботи програми для перевірки коректності роботи методу Природнього злиття.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи методу Природнього злиття. |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Кількість елементів:20  Максимальне значення:100000  Мінімальне значення: -200000 |
| Схема проведення тесту | Заповнення вхідних значень  Генерація вхідного файлу  Виконання методу прямого злиття |
| Очікуваний результат | Створення нового файлу з відсортованими даними вхідного файлу. |
| Стан програми після проведення випробувань | Створився вихідний файл. |

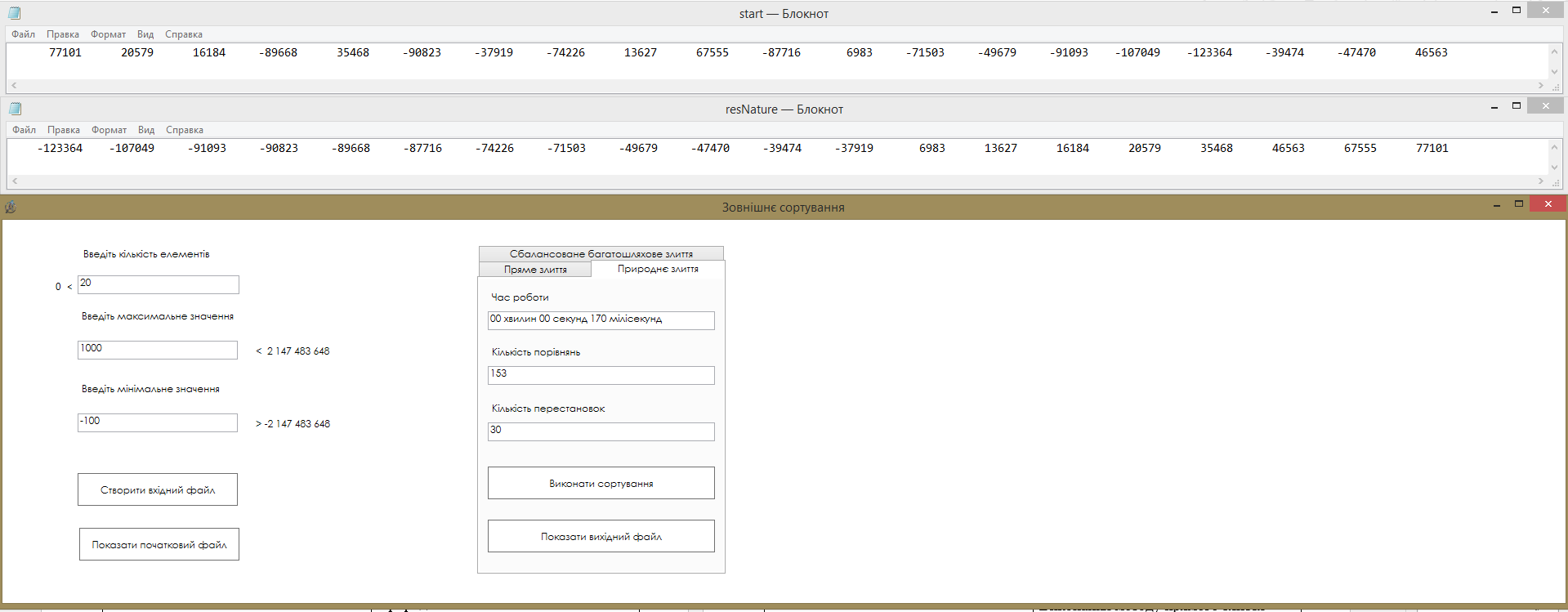


Рисунок 5.8

У Таблиця 5.9 та на Рисунок 5.9 наведено приклад роботи програми, де перевіряється коректність роботи методу Збалансованого багатошляхового злиття.

Таблиця 5.9- Приклад роботи програми для перевірки коректності роботи методу Збалансованого багатошляхового злиття.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи методу Збалансованого багатошляхового злиття. |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | Кількість елементів:20  Максимальне значення:100000  Мінімальне значення: -200000 |
| Схема проведення тесту | Заповнення вхідних значень  Генерація вхідного файлу  Виконання методу прямого злиття |
| Очікуваний результат | Створення нового файлу з відсортованими даними вхідного файлу. |
| Стан програми після проведення випробувань | Створився вихідний файл. |

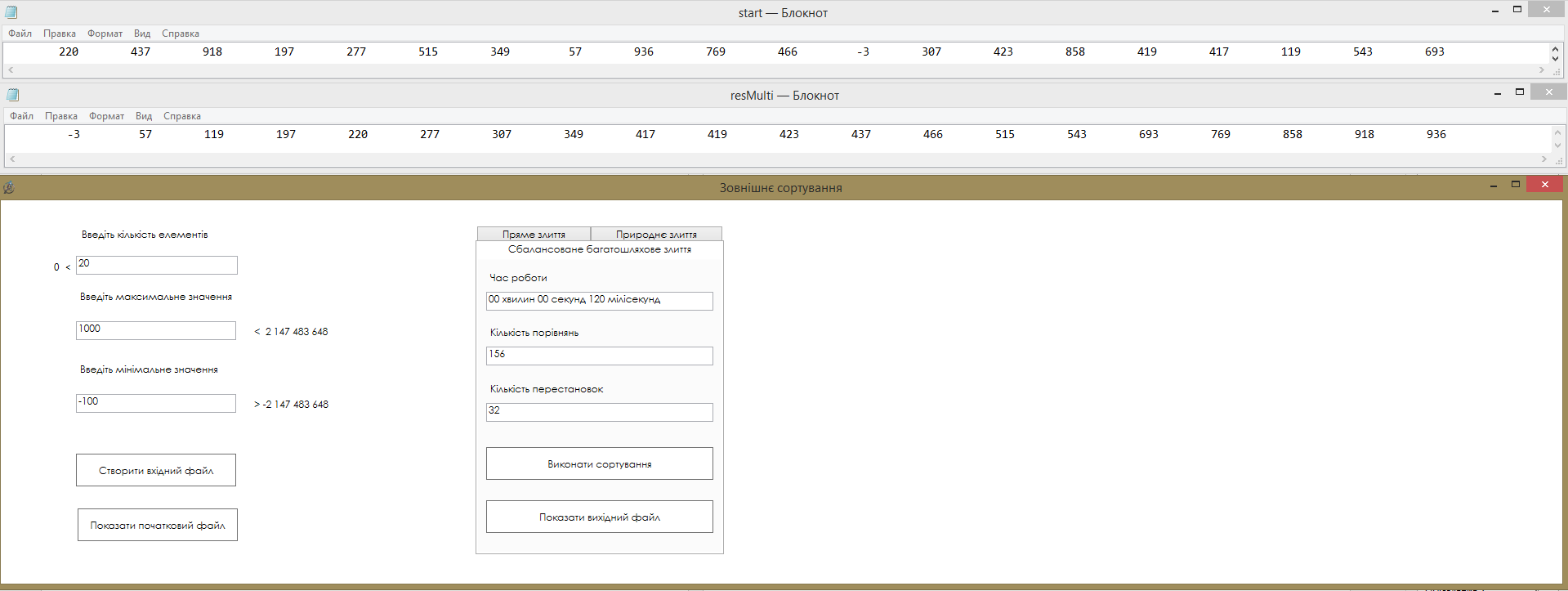


Рисунок 5.9

# Інструкція користувача

## Робота с програмою

Для того, щоб почати роботу, треба створити вхідний файл. Щоб,створити файл потрібно ввести такі данні:

* Кількість елементів файлі(Рисунок 6.1).
* Максимальне число, яке може бути згенеровано(Рисунок 6.2).
* Мінімальне число, яке може бути згенеровано(Рисунок 6.3).

Після того,як вхідні данні буде введено, потрібно натиснути на кнопку «Створити вхідний файл»(Рисунок 6.4 ), після цього можна натиснути на кнопку «Показати початковий файл»(Рисунок 6.5), щоб переглянути його у блокноті, або вибрати один з трьох запропонованих методів сортування(Рисунок 6.6). Для кожного метода існує копка «Виконати сортування»(Рисунок 6.7) та «Показати вихідний файл»(Рисунок 6.8). Після натиснення кнопки «Виконати сортування» буде доступний новий відсортований файл, який можна буде переглянути натиснувши на кнопку «Показати вихідний файл». Також після сортування буде виведено інформацію про час роботи програми(Рисунок 6.9), кількість порівнянь(Рисунок 6.10), у цьому методі та кількість перестановок(Рисунок 6.11).

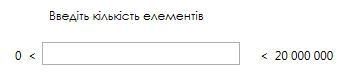


Рисунок 6.1

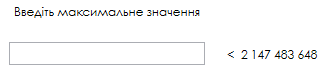


Рисунок 6.2

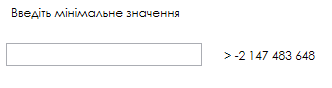


Рисунок 6.3



Рисунок 6.4



Рисунок 6.5



Рисунок 6.6



Рисунок 6.7



Рисунок 6.8



Рисунок 6.9



Рисунок 6.10

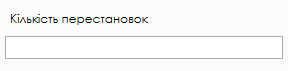


Рисунок 6.11

## Формат вхідних даних.

Кількість елементів для файлу повинно бути ціле число більше за 0.

Максимальне число, яке може бути згенеровано, повинно бути ціле і належати проміжку від -2 147 483 646 до -2 147 483 646.

Мінімальне число, яке може бути згенеровано, повинно бути ціле і належати проміжку від -2 147 483 646 до -2 147 483 646.

## Системні вимоги:

Операцыйна система: Windows 7/Windows 8/Windows 10;

ОЗП: 512 Мб;

Процесор: 1,8 Gh;

Роздільна здатність екрану: 1280x1024;

Вільний простір на вінчестері: 100 Мб;

Миша, клавіатура, монітор.

# аналіз і узагальнення результатів

У курсовій роботі було реалізовано три методи зовнішнього сортування: пряме злиття, природнє злиття та збалансоване багатошляхове злиття. Найшвидшим з наведених методів є збалансоване багатошляхове злиття, далі йде природнє злиття. Найповільніший метод- прямого злиття. Це дводить графік залежності часу від кількості елементів(Рисунок 7.1).

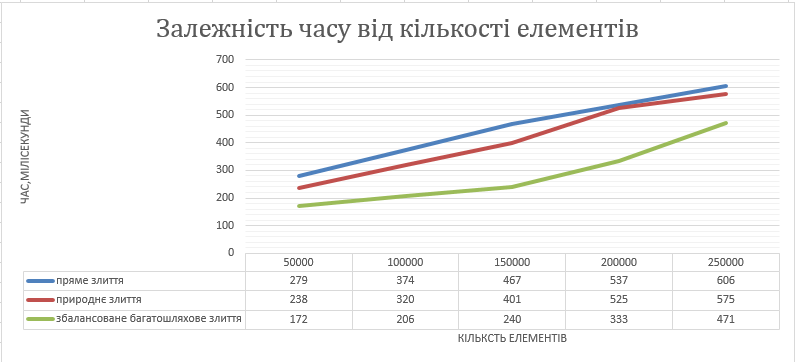


Рисунок 7.1

Кількість порівнянь у методі прямого злиття найменша, а у природнього злиття та до 150 000 елементів майже не відрізняється, а після кількість порінянь у збалансованого багатошляхового злиття збільшується порівняно с природнім. Це доводить графік залежності кількості порівнянь від кількості елементів (Рисунок 7.2).



Рисунок 7.2

Кількість перестановок найменшою буде у збалансованого багатошляхового злиття, а кількість перестановок у природньому та прямому злитті майже однакові. Це доводить графік залежності кількості перестановок від кількості елементів(Рисунок 7.3).



Рисунок 7.3

Теоретична складність алгоритму прямого злиття O(log n), де n кількість елементів у файлі. Складність алгоритму природнього злиття (2\*log2n), де n- кількість елементів у файлі. Складність алгоритму збалансованого багатошляхового злиття (m\*logmn), де n- кількість елементів у файлі, m- кількість файлів, які використовуються для сортування. [4]

Усі можливі критичні ситуації оброблені.

Висновки

В ході виконання курсової роботи була розроблена програма для зовнішнього впорядкування масивів.

На аналітичному етапі виконання роботи було досліджено 3 методи зовнішнього сортування: прямого злиття, природнього злиття, збалансованого багатошляхового злиття, визначено їх особливість та складність.

Було складено загальний алгоритм, та алгоритм до методів сортування, які потрібно було реалізувати за завданням.

Була побудована діаграма класів до програми, та здійснений опис функцій.

Було виконано тестування правильності введених значень та тестування коректності роботи методів сортування.

Була розроблена інструкція для користувача, для легшого користування програмним забезпечення.

Також був проведений аналіз методів сортування.

Під час виконання роботи були набутті теоретичні та практичні навички в області використання сучасних систем створення програмного забезпечення та освоєння принципів та методів сучасних технологій програмування. Також було вироблене вміння застосовувати методи обчислювальної математики та прикладного програмування для розв’язання прикладних задач.

З використанням мови програмування С# та графічної підсистеми WPF було розроблено програму, що реалізує відповідні методи сортування.

Перелік Посилань

1. Ваникіна, Галина. Лабораторная работа 43. Алгоритмы сортировки массивов. Внешняя сортировка [Електронний документ] . (<http://www.intuit.ru/studies/courses/648/504/lecture/11473?page=3>). Перевірено 20.05.2016.
2. Татьяна, Сундукова. 6.4. Сбалансированное многопутевое слтяние [Електронний документ] . (http://mf.gsu.by/Math/structure/l7.6.4.html). Перевірено 18.05.2016.
3. Травин, Андрей. 6.4. Общие сведения и определения [Електронний документ] . (http://studopedia.ru/2\_80093\_obshchie-svedeniya-i-opredeleniya.html). Перевірено 21.05.2016.
4. Станислав, Кузнецов. Методы сортировки и поиска [Електронний документ] . (http://citforum.ru/programming/theory/sorting/sorting1.shtml). Перевірено 12.05.2016.

Додаток А Технічне завдання

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко Максим Миколайович

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016 р.

Виконавець:

Студент Литвинюк Дмитро Романович

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: “Зовнішнє упорядкування масивів”

з дисципліни:

«Основи програмування»

Київ 201

1.1 Мета: Метою курсової роботи є розробка комплексу програм для розв’язання задачі зовнішнього упорядкування масивів розміром не менше 50000 цілочисельних елементів.

1.2 Найменування та галузь застосування об'єкта розробки:Дана робота присвячена розробці програмного забезпечення для упорядкування цілочисельних даних записаних у файл.

1.3 Підстава для проведення робіт:Підставою для розробки програмного забезпечення є навчальний план спеціальності 6.050301 «Програмна інженерія», робоча програма дисципліни „Основи програмування”, індивідуальне завдання.

1.4 Дата початку роботи: «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 201\_ р.

1.5 Дата закінчення роботи: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 201\_ р.

1.6 Призначення розробки: Функціональним призначенням програми є автоматизація наступних процесів: зовнішнє впорядкування масивів трьома методами: прямого злиття, природнього злиття, збалансованого багатошляхового злиття, збереження результату у файл, оцінка ефективності роботи методів. Програма може експлуатуватися для впорядкування цілочисельних даних у сховищах.

1.7 Вимоги до програми та програмної документації: Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ИСО 5807 - 85 ГОСТ на розробку програмних документів, схем алгоритмів програм, даних та систем.

ГОСТ 19.781 - 74 - Вимоги до розробки програмного забезпечення.

ГОСТ 19.101-77 (СТ СЭВ 1626 - 79) - Держстандарт на розробку програмної документації, видів програм та програмних документів.

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

1.8 Стадії та етапи розробки:

1. Аналіз методів вирішення поставленої задачі (до \_\_.\_\_.201\_ р.)

2. Розробка сценарію роботи програми (до \_\_.\_\_.201\_ р.)

3. Розробка алгоритмічного забезпечення (до \_\_.\_\_.201\_ р.)

4. Розробка програмного забезпечення (до \_\_.\_\_.201\_ р.)

5. Розробка інтерфейсу, планування, тестування розробленої програми (до \_\_.\_\_.201\_ р.)

6. Розробка пояснювальної записки (до \_\_.\_\_.201\_ р.).

7. Захист курсової роботи (до \_\_.\_\_.201\_ р.).

1.9 Порядок контролю та приймання. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв її оцінювання.

Додаток Б Тексти програмного коду

Prog\_Cursach.exe

*Тексти програмного коду програмного забезпечення*

*Зовнішнього упорядкування масивів*

(Вид носія даних)

*CD-RW*

(Обсяг програми (документа), арк., Кб)

*(20, 29 Кб)*

*студента групи ІП-54 І курсу*

*Литвинюка Дмитра Романовича*

namespace Prog\_Cursach

{

class Main\_Sort

{

int n, max, min;

public Int64 PereMulti { get; private set; }// кількість перестановок у Збалансованому багатошляховому злитті

public Int64 PorMulti { get; private set; }// кількість порівнянь у Збалансованому багатошляховому злитті

public Int64 PereNature { get; private set; }// кількість перестановок у природньому злитті

public Int64 PorNature { get; private set; }// кількість порівнянь у природньому злитті

public Int64 PereMerge { get; private set; }// кількість перестановок у прямому злитті

public Int64 PorMerge { get; private set; }// кількість порівнянь у прямому злитті

string nameFront = "fileFront.txt",//вхідний файл для прямого злиття

nameNature = "fileNature.txt",//вхідний файл для природнього злиття

nameMulti = "fileMulti.txt",//вхідний файл для збалансованого багатошляхового злиття

OutMerge = "resMerge.txt",//вихідний файл для прямого злиття

OutNature = "resNature.txt",//вихідний файл для природнього злиття

OutMulti = "resMulti.txt",//вихідний файл для збалансованого багатошляхового злиття

nameStart = "start.txt";//початковий файл у текстовому форматі

public Main\_Sort()

{

n = 10;

max = 20;

min = -20;

}

public Main\_Sort(int n, int max, int min)// конструктор класу n-кількість елемнтів max-максимальне min- мінімальне

{

this.n = n;

this.max = max;

this.min = min;

}

public void GenerateFiles()// метод для генерації вхідного файлу

{

Generate\_files generation = new Generate\_files(nameFront, nameNature, nameMulti, n, max, min);

generation.Generate();//генерація файлу

DeBinaireFile start = new DeBinaireFile(nameFront, nameStart);

start.CreateTXT(false);//створення с бінарного файлу текстовий

}

public void Create\_Merge()//пряме злиття

{

Merge\_Sort sortMerge = new Merge\_Sort(nameFront, n, OutMerge);

sortMerge.MakeSort();//виконання сортування

PereMerge = sortMerge.Pere;//статистика

PorMerge = sortMerge.Por;

}

public void CreateNature()//природнє злиття

{

Nature\_Sorting sortNature = new Nature\_Sorting(nameNature, OutNature);

sortNature.MakeSort();//сортування

PereNature = sortNature.Pere;//статистика

PorNature = sortNature.Por;

}

public void CreateMulti()//збалансоване багатошляхове злиття

{

Multi\_Sorting\_2 sortMulti = new Multi\_Sorting\_2(nameMulti, n, OutMulti);

sortMulti.MakeSort();//сортування

PereMulti = sortMulti.Pere;//статистика

PorMulti = sortMulti.Por;

}

public void ShowMulti()//відкриває відсортований збалансованим багатошляховим злиттям текстовий файл

{

Process.Start(OutMulti);

}

public void ShowNature()//відкриває відсортований природнім злиттям текстовий файл

{

Process.Start(OutNature);

}

public void ShowMerge()//відкриває відсортований прямим злиттям текстовий файл

{

Process.Start(OutMerge);

}

public void ShowStart()//відкриває відсортований збалансованим багатошляховим злиттям текстовий файл

{

Process.Start(nameStart);

}

}

}

сlass Nature\_Sorting//природнє злиття

{

string nameMain = "",//вхідний файл

nameF1 = @"fNature1.txt",//проміжний файл №1

nameF2 = @"fNature2.txt",//проміжний файл №2

nameOut = "";// вихідний файл

const int marker = Int32.MaxValue;//маркер кінця серії

private int por = 0,//кількість порівннянь

pere = 0;//кількіість перестановок

public Int64 Por { get { return por; } }

public Int64 Pere { get { return pere; } }

public Nature\_Sorting(string name,string nameOut)//конструктор класу

{

nameMain = name;

this.nameOut = nameOut;

}

public void MakeSort()//виконання сортування

{

int a1 = 0, a2 = 0, mark;

bool fl = true,

fl1 = true,

emp = true,

emp1 = true;

while (emp == true && emp1 == true)// цикл у якому відбувається сортування

{

mark = 1;

fl = true;

fl1 = true;

emp = false;

emp1 = false;

using (FileStream file = File.OpenRead(nameMain))

using (FileStream file1 = File.Create(nameF1))

using (FileStream file2 = File.Create(nameF2))

using (BinaryReader f = new BinaryReader(file))

using (BinaryWriter f1 = new BinaryWriter(file1))

using (BinaryWriter f2 = new BinaryWriter(file2))

{

try

{

a1 = f.ReadInt32();//зчитування з початкового

f1.Write(a1);//запис у проміжний файл №1

emp = true;

}

catch (EndOfStreamException e)

{

fl = false;

}

try

{

a2 = f.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

while (fl == true)//цикл, де визначаються серії і розподіляються по файлам

{

por++;

if (a2 < a1)//зміна файлу для запису

{

if (mark == 1)

{

f1.Write(marker);

mark = 2;

}

else

{

f2.Write(marker);

mark = 1;

}

}

//запис у поточний файл

if (mark == 1)

{

f1.Write(a2);

}

else

{

f2.Write(a2);

emp1 = true;

}

a1 = a2;

try

{

a2 = f.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

//додавання до поточного файлу маркеру кінця серії, після закінчення зчитування с вхідного файлу

if (emp == true && mark == 1)

{

f1.Write(marker);

}

if (emp1 == true && mark == 2)

{

f2.Write(marker);

}

}

fl = true;

fl1 = true;

using (FileStream file = File.Create(nameMain))

using (FileStream file1 = File.OpenRead(nameF1))

using (FileStream file2 = File.OpenRead(nameF2))

using (BinaryWriter f = new BinaryWriter(file))

using (BinaryReader f1 = new BinaryReader(file1))

using (BinaryReader f2 = new BinaryReader(file2))

{

try

{

a1 = f1.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

try

{

a2 = f2.ReadInt32();

}

catch (EndOfStreamException e)

{

fl1 = false;

}

while (fl == true && fl1 == true)//упорядкування серій

{

while (a1 != marker && a2 != marker)

{

por++;

if (a1 <= a2)

{

f.Write(a1);

try

{

a1 = f1.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

else

{

f.Write(a2);

pere++;

try

{

a2 = f2.ReadInt32();

}

catch (EndOfStreamException e)

{

fl1 = false;

}

}

}

while (a1 != Int32.MaxValue)//дозапис серії з першого файлу

{

f.Write(a1);

try

{

a1 = f1.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

while (a2 != marker)//дозапис серії с другого файлу

{

f.Write(a2);

try

{

a2 = f2.ReadInt32();

}

catch (EndOfStreamException e)

{

fl1 = false;

}

}

try

{

a1 = f1.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

try

{

a2 = f2.ReadInt32();

}

catch (EndOfStreamException e)

{

fl1 = false;

}

}

while (fl == true && a1 != marker)//дозапис з першого файлу

{

f.Write(a1);

try

{

a1 = f1.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

while (fl1 == true && a2 != marker)//дозапис з другого файлу

{

f.Write(a2);

try

{

a2 = f2.ReadInt32();

}

catch (EndOfStreamException e)

{

fl1 = false;

}

}

}

}

//видалення проміжних файлів

File.Delete(nameF1);

File.Delete(nameF2);

DeBinaireFile outFile = new DeBinaireFile(nameMain, nameOut);

outFile.CreateTXT(true);//запис відсортованого файлу з бінарного файла у текстовий

}

}

}

class Multi\_Sorting\_2//сортування збалансованим багатошляховим злиттям

{

const int marker = Int32.MaxValue;//маркер кінця серії

private int por = 0,

pere = 0;

public Int64 Por { get { return por; } }

public Int64 Pere { get { return pere; } }

int n = 0,//к-ст елементів

m = 0;//к-ст файлів на яку розбивається початковий

string nameMain = "",//початковий файл

nameOut = "";//вихідний файл

string[] namesA,//массиви допоміжних файлів

namesB;

public Multi\_Sorting\_2(string name, int count,string nameOut)//конструктор класу

{

nameMain = name;

this.nameOut = nameOut;

n = count;

//задання кількості допоміжних файлів

if (n < 50000)

m = 2;

else

{

if (n < 2500000)

m = n / 25000;

else

m = 100;

}

if (m % 2 == 1)

m--;

namesA = new string[m];

namesB = new string[m];

}

Random random = new Random();

private bool AnyTrue(bool[] gap,int c)//чи є одне зі значень вхідного масиву істиними

{

bool res = false;

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

{

if (gap[i] == true)

{

res = true;

break;

}

}

return res;

}

private void CreateNames()//створює ім'я для допоміжних файлів

{

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

{

namesA[i] = "promA#" + i + ".txt";

namesB[i]= "promB#" + i + ".txt";

}

}

private void DivideFiles()//розбиття вхідного файлу на допоміжні

{

FileStream[] files = new FileStream[m];

BinaryWriter[] writers = new BinaryWriter[m];

BinaryWriter currentFile;

int a1 = 0,

a2 = 0,

l = 0;

bool fl = true;

bool[] fl1 = new bool[m];

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

{

fl1[i] = true;

files[i] = File.OpenWrite(namesA[i]);

writers[i] = new BinaryWriter(files[i]);

}

using (FileStream file = File.OpenRead(nameMain))

using (BinaryReader f = new BinaryReader(file))

{

try

{

a1 = f.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

currentFile = writers[l];

currentFile.Write(a1);

try

{

a2 = f.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

while (fl == true)//цикл розбиття вхідного файлу

{

por++;

if (a1 < a2)//зписується в поточний файл

{

currentFile.Write(a2);

}

else

{

currentFile.Write(marker);//змінюється файл

l++;

if (l >= m)//то знову записується у 1

l = 0;

currentFile = writers[l];

currentFile.Write(a2);

}

a1 = a2;

try

{

a2 = f.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

currentFile.Write(marker);

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

{

writers[i].Close();

}

}

File.Delete(nameMain);

}

public void SortFiles()//виконання сортування

{

bool mark = false;

int k = 0, min = 0, l = 0, count = m, tmp = 0;

FileStream[] filesRead = new FileStream[m],

files = new FileStream[m];

BinaryReader[] readers = new BinaryReader[m];

BinaryWriter[] writers = new BinaryWriter[m];

BinaryWriter currentFile;

bool[] fl = new bool[m],

flOpen = new bool[m],

opening = new bool[m];

int[] last = new int[m];

while(k!=1)//виконувати доти доки не буде все записано в один файл

{

l = 0;

tmp = -1;

k = 0;

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

{

fl[i] = true;

flOpen[i] = true;

if (!mark)

{

filesRead[i] = File.OpenRead(namesA[i]);

files[i] = File.Create(namesB[i]);

}

else

{

filesRead[i] = File.OpenRead(namesB[i]);

files[i] = File.Create(namesA[i]);

}

writers[i] = new BinaryWriter(files[i]);

readers[i] = new BinaryReader(filesRead[i]);

opening[i] = false;

}

if (mark)

mark = false;

else

mark = true;

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

{

try

{

last[i] = readers[i].ReadInt32();

}

catch (EndOfStreamException e)

{

fl[i] = false;

}

if (fl[i])

flOpen[i] = true;

else

flOpen[i] = false;

}

while (AnyTrue(fl,count))//поки хоча б один с проміжних файлів має дані

{

tmp++;

if (tmp == m)

tmp = 0;

currentFile = writers[tmp];

if (!opening[tmp])

{

k++;

opening[tmp] = true;

}

while (AnyTrue(flOpen,count))//злиття серій

{

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

{

if (fl[i] && flOpen[i])

{

min = last[i];

l = i;

break;

}

}

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

{

if (fl[i] && flOpen[i])

{

por++;

if (min > last[i])

{

min = last[i];

l = i;

}

}

}

if (tmp !=l)

pere++;

currentFile.Write(last[l]);

try

{

last[l] = readers[l].ReadInt32();

}

catch (EndOfStreamException e)

{

fl[l] = false;

}

if(last[l]==marker)

{

flOpen[l] = false;

}

}

currentFile.Write(marker);

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

{

try

{

last[i] = readers[i].ReadInt32();

}

catch (EndOfStreamException e)

{

fl[i] = false;

}

if (fl[i])

flOpen[i] = true;

else

flOpen[i] = false;

}

}

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

{

readers[i].Close();

writers[i].Close();

}

count = k;

}

string s = "";

if (mark)

s = namesB[0];

else

s = namesA[0];

DeBinaireFile outFile = new DeBinaireFile(s, nameOut);

outFile.CreateTXT(n);

}

private void DeleteFiles()//видалення проміжних файлів

{

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

{

File.Delete(namesA[i]);

File.Delete(namesB[i]);

}

}

public void MakeSort()//збір усіх методів

{

CreateNames();

DivideFiles();

SortFiles();

DeleteFiles();

}

}

}

public partial class MainWindow : Window

{

int n = 0,//кількість елементів

max = 0,//максимальне значення

min = 0;//мінімальне значення

//мітки для надання можливості використовувати кнопки

bool canMake = false, canMergeShow = false,

canNatureShow = false,

canMultiShow = false,

canMerge = false,

canNature = false,

canMulti = false;

Main\_Sort sort;

private void buttonStart\_Click(object sender, RoutedEventArgs e)

{

buttonStart.Cursor = Cursors.Wait;

try {

n = Convert.ToInt32(textBoxN.Text);//зчитування кількості елементів

max = Convert.ToInt32(textBoxMax.Text);//зчитування максивмального елемента

min = Convert.ToInt32(textBoxMin.Text);//зчитування мінімального елементу

if (min > max)//якщо мінімальний буде більший за максимальний

{

MessageBox.Show("Ви ввели не коректні дані, повторіть запис", "Error");

}

else

{

if (n > 0 && n < 20000000)

{

sort = new Main\_Sort(n, max, min);

canMake = true;

canNature = true;

canMerge = true;

canMulti = true;

sort.GenerateFiles();//генерація файлу с використанням вхідних даних

}

else

MessageBox.Show("Ви ввели не коректні дані, повторіть запис", "Error");

}

}

catch(Exception e1)

{

MessageBox.Show("Ви ввели не коректні дані, повторіть запис", "Error");

}

buttonStart.Cursor = Cursors.Arrow;

}

private void button\_showNature\_Click(object sender, RoutedEventArgs e)//демонстрація відсортованого файлу методом природнього злиття

{

if (canNatureShow)

sort.ShowNature();

else

MessageBox.Show("Ви не виконали сортування природнім злиттям", "Error");

}

private void button\_resMerge\_Click\_1(object sender, RoutedEventArgs e)//сортування методом прямого злиття

{

if (canMerge)

{

Stopwatch mergeWatch = new Stopwatch();

button\_resMerge.Cursor = Cursors.Wait;//переведення курсора в режим очікування

mergeWatch.Start();//початок відліку часу

sort.Create\_Merge();//сортування

mergeWatch.Stop();//зупинка відліку часу

TimeSpan ts = mergeWatch.Elapsed;

timeMerge.Text = String.Format("{0:00}", ts.Minutes) + " хвилин " + String.Format("{0:00}", ts.Seconds) + " секунд " + String.Format("{0:00}", ts.Milliseconds) + " мілісекунд";//час роботи

pereMerge.Text = sort.PereMerge.ToString();//кількість порівнянь

porMerge.Text = sort.PorMerge.ToString();//кількість перестановок

button\_resMerge.Cursor = Cursors.Arrow;

canMergeShow = true;

canMerge = false;

MessageBox.Show("Сортування прямим злиттям виконано","Message");

}

else

MessageBox.Show("Ви не маєте можливість виконати сортування", "Error");

}

private void button\_showMerge\_Click(object sender, RoutedEventArgs e)//демонстрація відсортованого файлу прямим злиттям

{

if (canMergeShow)

sort.ShowMerge();

else

MessageBox.Show("Ви не виконали сортування прямим злиттям", "Error");

}

private void buttonStartShow\_Click(object sender, RoutedEventArgs e)//демонстрація початкового файлу

{

if (canMake)

sort.ShowStart();

else

MessageBox.Show("Ви не згенерували файл", "Error");

}

private void button\_resMulti\_Click(object sender, RoutedEventArgs e)//сортування методом збалансованого багатошляхового злиття

{

if (canMulti)

{

Stopwatch multiWatch = new Stopwatch();

button\_resMulti.Cursor = Cursors.Wait;

multiWatch.Start();

sort.CreateMulti();//сортування

multiWatch.Stop();

TimeSpan ts = multiWatch.Elapsed;

timeMulti.Text = String.Format("{0:00}", ts.Minutes) + " хвилин " + String.Format("{0:00}", ts.Seconds) + " секунд " + String.Format("{0:00}", ts.Milliseconds) + " мілісекунд";

porMulti.Text = sort.PorMulti.ToString();

pereMulti.Text = sort.PereMulti.ToString();

button\_resMulti.Cursor = Cursors.Arrow;

canMultiShow = true;

canMulti = false;

MessageBox.Show("Сортування збалансованим багатошляховим злиттям виконано", "Message");

}

else

MessageBox.Show("Ви не маєте можливість виконати сортування");

}

private void button\_showMulti\_Click(object sender, RoutedEventArgs e)//демонстрація файлу відсортованого багатошляховим злиттям

{

if (canMultiShow)

sort.ShowMulti();

else

MessageBox.Show("Ви не виконали сортування збалансованим багатошляховим злиттям", "Error");

}

private void button\_resNature\_Click(object sender, RoutedEventArgs e)//виконання сортування природнім злиттям

{

if (canNature)

{

Stopwatch natureWatch = new Stopwatch();

button\_resNature.Cursor = Cursors.Wait;

natureWatch.Start();

sort.CreateNature();//сортування

natureWatch.Stop();

TimeSpan ts = natureWatch.Elapsed;

timeNature.Text = String.Format("{0:00}", ts.Minutes) + " хвилин " + String.Format("{0:00}", ts.Seconds) + " секунд " + String.Format("{0:00}", ts.Milliseconds) + " мілісекунд";

porNature.Text = sort.PorNature.ToString();

pereNature.Text = sort.PereNature.ToString();

button\_resNature.Cursor = Cursors.Arrow;

canNatureShow = true;

canNature = false;

MessageBox.Show("Сортування природнім злиттям виконано", "Message");

}

else

MessageBox.Show("Ви не маєте можливість виконати сортування", "Error");

}

private void button\_ShowRes\_Click(object sender, RoutedEventArgs e)

{

}

public MainWindow()

{

InitializeComponent();

labelMax.Content ="< 2 147 483 647";

labelMin.Content = "> -2 147 483 647";

labelN.Content = "0 <";

labelNmax.Content = "< 20 000 000";

}

}

class DeBinaireFile//створення с бінарного файлу текстового

{

private string nameMain="",//ім'я вхідного файлу

outName = "";//ім'я вихідного файлу

public DeBinaireFile(string inName,string outputName)//конструктор класу

{

nameMain = inName;

outName = outputName;

}

public void CreateTXT(bool top)//створення .txt , якщо параметр bool

{

using (StreamWriter writer = new StreamWriter(outName,false))

using (FileStream file = File.OpenRead(nameMain))

using (BinaryReader reader = new BinaryReader(file))

{

int y = 0;

bool fl = true;

while (fl)

{

for (int i = 0; i < 92 && fl; i++)

{

try

{

y = reader.ReadInt32();

writer.Write(String.Format("{0,11}",y));

}

catch (EndOfStreamException e)

{

fl = false;

}

}

writer.WriteLine();

}

}

if(top)

File.Delete(nameMain);//видалення бінарного файлу

}

public void CreateTXT(int m)//створення .txt , якщо параметр int

{

using (StreamWriter writer = new StreamWriter(outName, false))

using (FileStream file = File.OpenRead(nameMain))

using (BinaryReader reader = new BinaryReader(file))

{

int y = 0, i = 0;

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

{

i++;

try

{

y = reader.ReadInt32();

writer.Write(String.Format("{0,11}", y));

}

catch (EndOfStreamException e)

{

}

if (i == 91)

{

i = 0;

writer.WriteLine();

}

}

}

File.Delete(nameMain);//видалення вхідного файлу

}

}

class Generate\_files

{

private string nameFront = "",//вхідний файл для прямого злиття

nameNature = "",//вхідний файл для природнього злиття

nameMulti = "";//вхідний файл для збалансованого багатошляхового злиття

private int count = 0,//к-ст елементів

//діапазон

max = 0,

min = 0;

bool fl = true,

fl1 = true;

Random random = new Random();

public Generate\_files(string nameF1, string nameF2, string nameF3,int n,int maxi,int min)//конструктор класу

{

nameFront = nameF1;

nameNature = nameF2;

nameMulti = nameF3;

count = n;

if (maxi > 0)

max = maxi;

else

{

fl = false;

max = Math.Abs(maxi);

}

if (min < 0)

{

this.min = Math.Abs(min);

fl1 = false;

}

else

this.min = min;

}

public void Generate()//генерація файлів

{

using (FileStream writerFront = File.Create(nameFront))

using (BinaryWriter wrotrFront = new BinaryWriter(writerFront))

using (FileStream writerNature = File.Create(nameNature))

using (BinaryWriter wrotrNature = new BinaryWriter(writerNature))

using (FileStream writerMulti = File.Create(nameMulti))

using (BinaryWriter wrotrMulti = new BinaryWriter(writerMulti))

{

int c = 0;

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

{

//генерація залежить від знаків max і min

if (fl && !fl1)

c = random.Next(max + 1) - random.Next(min + 1);

else

{

if(!fl && !fl1)

c = -(random.Next(max+1, min+1));

else

{

c = random.Next(min, max + 1);

}

}

wrotrFront.Write(c);

wrotrNature.Write(c);

wrotrMulti.Write(c);

}

}

}

}

class Merge\_Sort//сортування прямим злиттям

{

string nameMain = "",//вхідний файл

nameF1 = "f1.txt",//проміжний файл №1

nameF2 = "f2.txt",//проміжний файл №2

nameOut = "";//вихідний файл

int count = 0;//к-сть елементів

private int por = 0,

pere = 0;

public Int64 Por { get { return por; } }

public Int64 Pere { get { return pere; } }

internal DeBinaireFile DeBinaireFile

{

get

{

throw new System.NotImplementedException();

}

set

{

}

}

public Merge\_Sort(string name,int n,string nameOut)//конструктор класу

{

nameMain = name;

count = n;

this.nameOut = nameOut;

}

public void MakeSort()//виконання сортування

{

int a1 = 0, a2 = 0, k = 1;

bool fl = true,

fl1 = true;

while (k < count)//виконання сортування

{

using (FileStream file = File.OpenRead(nameMain))

using (BinaryReader f = new BinaryReader(file))

using (FileStream file1 = File.Create(nameF1))

using (FileStream file2 = File.Create(nameF2))

using (BinaryWriter f1 = new BinaryWriter(file1))

using (BinaryWriter f2 = new BinaryWriter(file2))

{

a1 = f.ReadInt32();

fl = true;

fl = true;

while (fl == true)//розбиття вхідного файлу на проміжні

{

for (int i = 0; i < k && fl == true; i++)//запис у №1

{

try

{

f1.Write(a1);

a1 = f.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

for (int j = 0; j < k && (fl == true); j++)//запис у №2

{

try

{

f2.Write(a1);

a1 = f.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

}

}

using (FileStream file1 = File.OpenRead(nameF1))

using (FileStream file = File.Create(nameMain))

using (BinaryWriter f = new BinaryWriter(file))

using (FileStream file2 = File.OpenRead(nameF2))

using (BinaryReader f1 = new BinaryReader(file1))

using (BinaryReader f2 = new BinaryReader(file2))

{

fl = true;

fl1 = true;

try

{

a1 = f1.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

try

{

a2 = f2.ReadInt32();

}

catch (EndOfStreamException e)

{

fl1 = false;

}

while (fl == true && fl1 == true)//виконання злиття поки один с файлів не закінчиться

{

int i = 0;

int j = 0;

while (i < k && j < k && (fl == true) && (fl1 == true))//упорядкування серій

{

por++;

if (a1 < a2)

{

f.Write(a1);

try

{

a1 = f1.ReadInt32();

i++;

}

catch (EndOfStreamException e)

{

fl = false;

}

}

else

{

f.Write(a2);

pere++;

try

{

a2 = f2.ReadInt32();

j++;

}

catch (EndOfStreamException e)

{

fl1 = false;

}

}

}

while (i < k && (fl))//дозапис серій с допоміжного файлу №1

{

f.Write(a1);

try

{

a1 = f1.ReadInt32();

i++;

}

catch (EndOfStreamException e)

{

fl = false;

}

}

while (j < k && (fl1))//дозапис серій с допоміжного файлу №2

{

f.Write(a2);

try

{

a2 = f2.ReadInt32();

j++;

}

catch (EndOfStreamException e)

{

fl1 = false;

}

}

}

while (fl)//дозапис с допоміжного файлу №1

{

f.Write(a1);

try

{

a1 = f1.ReadInt32();

}

catch (EndOfStreamException e)

{

fl = false;

}

}

while (fl1)//дозапис с допоміжного файлу №2

{

f.Write(a2);

try

{

a2 = f2.ReadInt32();

}

catch (EndOfStreamException e)

{

fl1 = false;

}

}

}

k \*= 2;

}

File.Delete(nameF1);

File.Delete(nameF2);

DeBinaireFile outFile = new DeBinaireFile(nameMain, nameOut);

outFile.CreateTXT(true);

}

}