**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 1 з дисципліни

«Проектування алгоритмів»

„ **Проектування і аналіз алгоритмів зовнішнього сортування**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*Ткаченко Костянтин Олекчсандрович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2024

Зміст

[1 Мета лабораторної роботи 3](#_Toc114359761)

[2 Завдання 4](#_Toc114359762)

[3 Виконання 7](#_Toc114359763)

[3.1 Псевдокод алгоритмів 7](#_Toc114359764)

[3.2 Часова складність пошуку 7](#_Toc114359765)

[3.3 Програмна реалізація 7](#_Toc114359766)

[3.3.1 Вихідний код 7](#_Toc114359767)

[3.3.2 Приклади роботи 7](#_Toc114359768)

[3.4 Тестування алгоритму 8](#_Toc114359769)

[3.4.1 Часові характеристики оцінювання 8](#_Toc114359770)

[Висновок 9](#_Toc114359771)

[Критерії оцінювання 10](#_Toc114359772)

1. **Мета лабораторної роботи**

Метою лабораторної роботи є створення системи управління базами даних (СУБД) з графічним інтерфейсом користувача, що використовує файли з щільним індексом та метод Шарра для організації даних.

Основні завдання:

1. Розробка алгоритмів для основних операцій з даними:
   * пошук записів
   * додавання нових записів
   * видалення існуючих записів
   * редагування даних
2. Реалізація структури даних на основі щільного індексу з використанням методу Шарра для ефективної організації та доступу до даних.
3. Створення графічного інтерфейсу користувача для зручної взаємодії з базою даних.
4. Забезпечення постійного зберігання даних на постійному запам'ятовуючому пристрої (ПЗП).
5. Проведення тестування та аналізу ефективності реалізованої системи через:
   * заповнення бази даних випадковими значеннями (до 10000 записів)
   * вимірювання середнього числа порівнянь при пошуку записів
   * оцінку часової складності операцій в асимптотичних оцінках

Практичне значення роботи полягає у засвоєнні принципів роботи з індексованими структурами даних та отриманні навичок розробки прикладного програмного забезпечення для роботи з базами даних.

1. **Завдання**

Відповідно до варіанту (таблиця 2.1), записати алгоритми пошуку, додавання, видалення і редагування запису в структурі даних за допомогою псевдокоду (чи іншого способу по вибору).

Записати часову складність пошуку в структурі в асимптотичних оцінках.

Виконати програмну реалізацію невеликої СУБД з графічним (не консольним) інтерфейсом користувача (дані БД мають зберігатися на ПЗП), з функціями пошуку (алгоритм пошуку у вузлі чи блоці структури згідно варіанту таблиця 2.1, за необхідності), додавання, видалення та редагування записів (запис складається із ключа і даних, ключі унікальні і цілочисельні, даних може бути декілька полів для одного ключа, але достатньо одного рядка фіксованої довжини). Для зберігання даних використовувати структуру даних згідно варіанту (таблиця 2.1).

Заповнити базу випадковими значеннями до 10000 і зафіксувати середнє (із 10-15 пошуків) число порівнянь для знаходження запису по ключу.\

|  |  |
| --- | --- |
| **№** | **Структура даних** |
| 21 | Файли з щільним індексом з перебудовою індексної області, метод Шарра |

1. **Виконання**
   1. ПСЕВДОКОД АЛГОРИТМІВ

Клас DataService

Константа ConnectionString ← "рядок підключення до бази даних"

Змінна connection ← null

Метод InitializeDatabase

Спробувати:

Створити нове підключення до бази даних

Відкрити підключення

Викликати CreateTablesIfNotExist

Якщо помилка:

Викинути помилку "Ініціалізація бази даних не вдалася"

Метод CreateTablesIfNotExist

Якщо connection = null, викинути помилку "Підключення до бази даних не ініціалізоване"

Виконати SQL-запит для створення таблиць (якщо вони ще не існують)

Метод Dispose

Якщо connection не null:

Якщо стан підключення = Open:

Закрити підключення

Звільнити ресурси підключення

Метод Search(key)

Виконати SQL-запит для пошуку запису за ключем

Якщо знайдено:

Повернути знайдений запис

Інакше повернути null

Метод Insert(key, data)

Почати транзакцію

Спробувати:

Виконати SQL-запити для додавання даних у таблиці data\_records та dense\_index

Підтвердити транзакцію

Якщо помилка:

Відкотити транзакцію

Метод Delete(key)

Почати транзакцію

Спробувати:

Видалити дані з таблиць dense\_index та data\_records за ключем

Підтвердити транзакцію

Якщо помилка:

Відкотити транзакцію

Метод Update(key, newData)

Виконати SQL-запит для оновлення значення даних у таблиці data\_records

Метод GenerateRandomData(count)

Для i від 0 до count:

Згенерувати випадковий ключ і дані

Спробувати виконати Insert

Якщо помилка:

Зменшити i на 1

Метод MeasureAverageComparisons

Змінна totalComparisons ← 0

Для i від 0 до 15:

Згенерувати випадковий ключ

Виміряти час пошуку запису

Додати час до totalComparisons

Повернути totalComparisons / 15

Метод GetAllRecords(limit)

Якщо connection = null, викинути помилку "Підключення до бази даних не ініціалізоване"

Виконати SQL-запит для отримання всіх записів з обмеженням у limit рядків

Повернути результуючу таблицю

3.3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.3.1 Вихідний код

namespace Laba3;

**public class DataRecord**

{

public int Key { get; set; }

public string Data { get; set; }

}

using System.Data;

using Npgsql;

namespace Laba3;

**public class DataService : IDisposable**

{

private const string ConnectionString =

"Host=localhost;Port=5432;Database=homework;Username=postgres;Password=2025";

public NpgsqlConnection? connection;

public void InitializeDatabase()

{

try

{

connection = new NpgsqlConnection(ConnectionString);

connection.Open();

CreateTablesIfNotExist();

}

catch (Exception ex)

{

throw new ApplicationException("Failed to initialize database", ex);

}

}

private void CreateTablesIfNotExist()

{

if (connection == null) throw new InvalidOperationException("Database connection not initialized");

using var cmd = new NpgsqlCommand(@"

CREATE TABLE IF NOT EXISTS data\_records (

key\_id INTEGER PRIMARY KEY,

data\_value VARCHAR(255)

);

CREATE TABLE IF NOT EXISTS dense\_index (

key\_id INTEGER PRIMARY KEY,

record\_pointer INTEGER,

FOREIGN KEY (record\_pointer) REFERENCES data\_records(key\_id)

);", connection);

cmd.ExecuteNonQuery();

}

public void Dispose()

{

if (connection != null)

{

if (connection.State == ConnectionState.Open)

{

connection.Close();

}

connection.Dispose();

connection = null;

}

}

public DataRecord Search(int key)

{

using (var cmd = new NpgsqlCommand())

{

cmd.Connection = connection;

cmd.CommandText = @"

SELECT d.key\_id, d.data\_value

FROM data\_records d

INNER JOIN dense\_index i ON d.key\_id = i.record\_pointer

WHERE d.key\_id = @key";

cmd.Parameters.AddWithValue("@key", key);

using (var reader = cmd.ExecuteReader())

{

if (reader.Read())

{

return new DataRecord

{

Key = reader.GetInt32(0),

Data = reader.GetString(1)

};

}

}

}

return null;

}

public void Insert(int key, string data)

{

using (var transaction = connection.BeginTransaction())

{

try

{

using (var cmd = new NpgsqlCommand())

{

cmd.Connection = connection;

cmd.Transaction = transaction;

cmd.CommandText = "INSERT INTO data\_records (key\_id, data\_value) VALUES (@key, @data)";

cmd.Parameters.AddWithValue("@key", key);

cmd.Parameters.AddWithValue("@data", data);

cmd.ExecuteNonQuery();

}

using (var cmd = new NpgsqlCommand())

{

cmd.Connection = connection;

cmd.Transaction = transaction;

cmd.CommandText = "INSERT INTO dense\_index (key\_id, record\_pointer) VALUES (@key, @key)";

cmd.Parameters.AddWithValue("@key", key);

cmd.ExecuteNonQuery();

}

transaction.Commit();

}

catch

{

transaction.Rollback();

throw;

}

}

}

public void Delete(int key)

{

using (var transaction = connection.BeginTransaction())

{

try

{

using (var cmd = new NpgsqlCommand())

{

cmd.Connection = connection;

cmd.Transaction = transaction;

cmd.CommandText = "DELETE FROM dense\_index WHERE key\_id = @key";

cmd.Parameters.AddWithValue("@key", key);

cmd.ExecuteNonQuery();

}

using (var cmd = new NpgsqlCommand())

{

cmd.Connection = connection;

cmd.Transaction = transaction;

cmd.CommandText = "DELETE FROM data\_records WHERE key\_id = @key";

cmd.Parameters.AddWithValue("@key", key);

cmd.ExecuteNonQuery();

}

transaction.Commit();

}

catch

{

transaction.Rollback();

throw;

}

}

}

public void Update(int key, string newData)

{

using (var cmd = new NpgsqlCommand())

{

cmd.Connection = connection;

cmd.CommandText = "UPDATE data\_records SET data\_value = @data WHERE key\_id = @key";

cmd.Parameters.AddWithValue("@key", key);

cmd.Parameters.AddWithValue("@data", newData);

cmd.ExecuteNonQuery();

}

}

public void GenerateRandomData(int count)

{

Random random = new Random();

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

{

int key = random.Next(1, 100000);

string data = $"Data\_{random.Next(1000000)}";

try

{

Insert(key, data);

}

catch (Exception)

{

i--;

}

}

}

public double MeasureAverageComparisons()

{

Random random = new Random();

int numberOfSearches = 15;

int totalComparisons = 0;

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

{

int randomKey = random.Next(1, 100000);

var startTime = DateTime.Now;

Search(randomKey);

var endTime = DateTime.Now;

totalComparisons += (endTime - startTime).Milliseconds;

}

return totalComparisons / (double)numberOfSearches;

}

public DataTable GetAllRecords(int limit = 1000)

{

if (connection == null) throw new InvalidOperationException("Database connection not initialized");

var table = new DataTable();

using var command = new NpgsqlCommand(@"

SELECT d.key\_id as ""Key"",

d.data\_value as ""Data""

FROM data\_records d

INNER JOIN dense\_index i ON d.key\_id = i.record\_pointer

ORDER BY d.key\_id

LIMIT @limit", connection);

command.Parameters.AddWithValue("@limit", limit);

using var adapter = new NpgsqlDataAdapter(command);

adapter.Fill(table);

return table;

}

}

using System.Data;

using Npgsql;

namespace Laba3;

public partial class MainForm : Form

{

private readonly DataService dataService;

public MainForm()

{

InitializeComponent();

dataService = new DataService();

try

{

dataService.InitializeDatabase();

lblStatus.Text = "Connected to database successfully";

}

catch (Exception ex)

{

MessageBox.Show($"Database initialization failed: {ex.Message}", "Error",

MessageBoxButtons.OK, MessageBoxIcon.Error);

lblStatus.Text = "Database connection failed";

}

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

base.OnFormClosing(e);

dataService?.Dispose();

}

private void BtnSearch\_Click(object sender, EventArgs e)

{

try

{

if (!int.TryParse(txtKey.Text, out int key))

{

MessageBox.Show("Please enter a valid integer key");

return;

}

var record = dataService.Search(key);

if (record != null)

{

txtData.Text = record.Data;

lblStatus.Text = "Record found";

}

else

{

lblStatus.Text = "Record not found";

}

}

catch (Exception ex)

{

MessageBox.Show($"Error during search: {ex.Message}");

}

}

private void BtnAdd\_Click(object sender, EventArgs e)

{

try

{

if (!int.TryParse(txtKey.Text, out int key))

{

MessageBox.Show("Please enter a valid integer key");

return;

}

string data = txtData.Text;

if (string.IsNullOrEmpty(data))

{

MessageBox.Show("Please enter data");

return;

}

dataService.Insert(key, data);

lblStatus.Text = "Record added successfully";

RefreshDataGrid();

}

catch (Exception ex)

{

MessageBox.Show($"Error adding record: {ex.Message}");

}

}

private void BtnUpdate\_Click(object sender, EventArgs e)

{

try

{

if (!int.TryParse(txtKey.Text, out int key))

{

MessageBox.Show("Please enter a valid integer key");

return;

}

string newData = txtData.Text;

if (string.IsNullOrEmpty(newData))

{

MessageBox.Show("Please enter data");

return;

}

dataService.Update(key, newData);

lblStatus.Text = "Record updated successfully";

RefreshDataGrid();

}

catch (Exception ex)

{

MessageBox.Show($"Error updating record: {ex.Message}");

}

}

private void BtnDelete\_Click(object sender, EventArgs e)

{

try

{

if (!int.TryParse(txtKey.Text, out int key))

{

MessageBox.Show("Please enter a valid integer key");

return;

}

dataService.Delete(key);

lblStatus.Text = "Record deleted successfully";

RefreshDataGrid();

}

catch (Exception ex)

{

MessageBox.Show($"Error deleting record: {ex.Message}");

}

}

private void BtnGenerateData\_Click(object sender, EventArgs e)

{

try

{

dataService.GenerateRandomData(10000);

lblStatus.Text = "Generated 10,000 random records";

RefreshDataGrid();

}

catch (Exception ex)

{

MessageBox.Show($"Error generating data: {ex.Message}");

}

}

private void BtnMeasure\_Click(object sender, EventArgs e)

{

try

{

double avgComparisons = dataService.MeasureAverageComparisons();

lblStatus.Text = $"Average search time: {avgComparisons:F2} ms";

}

catch (Exception ex)

{

MessageBox.Show($"Error measuring performance: {ex.Message}");

}

}

private void RefreshDataGrid()

{

try

{

dataGridView.DataSource = dataService.GetAllRecords();

foreach (DataGridViewColumn column in dataGridView.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

lblStatus.Text = $"Data refreshed successfully. Showing {dataGridView.RowCount} records.";

}

catch (Exception ex)

{

MessageBox.Show("Failed to refresh data grid", "Error",

MessageBoxButtons.OK, MessageBoxIcon.Error);

lblStatus.Text = $"Failed to refresh data: {ex.Message}";

}

}

}

namespace Laba3;

using System.ComponentModel;

using System.ComponentModel;

using System.Windows.Forms;

partial class MainForm

{

private IContainer components = null;

private TextBox txtKey;

private TextBox txtData;

private Button btnSearch;

private Button btnAdd;

private Button btnUpdate;

private Button btnDelete;

private Button btnGenerateData;

private DataGridView dataGridView;

private Label lblKey;

private Label lblData;

private Label lblStatus;

private GroupBox groupBoxOperations;

private GroupBox groupBoxData;

private Button btnMeasure;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

this.txtKey = new System.Windows.Forms.TextBox();

this.txtData = new System.Windows.Forms.TextBox();

this.btnSearch = new System.Windows.Forms.Button();

this.btnAdd = new System.Windows.Forms.Button();

this.btnUpdate = new System.Windows.Forms.Button();

this.btnDelete = new System.Windows.Forms.Button();

this.btnGenerateData = new System.Windows.Forms.Button();

this.dataGridView = new System.Windows.Forms.DataGridView();

this.lblKey = new System.Windows.Forms.Label();

this.lblData = new System.Windows.Forms.Label();

this.lblStatus = new System.Windows.Forms.Label();

this.groupBoxOperations = new System.Windows.Forms.GroupBox();

this.groupBoxData = new System.Windows.Forms.GroupBox();

this.btnMeasure = new System.Windows.Forms.Button();

//

// Form setup

//

this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 15F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Text = "Dense Index Database";

//

// GroupBox Operations

//

this.groupBoxOperations.Location = new System.Drawing.Point(12, 12);

this.groupBoxOperations.Size = new System.Drawing.Size(300, 150);

this.groupBoxOperations.Text = "Operations";

//

// Labels

//

this.lblKey.Location = new System.Drawing.Point(20, 25);

this.lblKey.Size = new System.Drawing.Size(70, 23);

this.lblKey.Text = "Key:";

this.lblData.Location = new System.Drawing.Point(20, 55);

this.lblData.Size = new System.Drawing.Size(70, 23);

this.lblData.Text = "Data:";

//

// TextBoxes

//

this.txtKey.Location = new System.Drawing.Point(100, 25);

this.txtKey.Size = new System.Drawing.Size(180, 23);

this.txtData.Location = new System.Drawing.Point(100, 55);

this.txtData.Size = new System.Drawing.Size(180, 23);

//

// Buttons

//

this.btnSearch.Location = new System.Drawing.Point(20, 90);

this.btnSearch.Size = new System.Drawing.Size(80, 23);

this.btnSearch.Text = "Search";

this.btnAdd.Location = new System.Drawing.Point(110, 90);

this.btnAdd.Size = new System.Drawing.Size(80, 23);

this.btnAdd.Text = "Add";

this.btnUpdate.Location = new System.Drawing.Point(200, 90);

this.btnUpdate.Size = new System.Drawing.Size(80, 23);

this.btnUpdate.Text = "Update";

this.btnDelete.Location = new System.Drawing.Point(20, 120);

this.btnDelete.Size = new System.Drawing.Size(80, 23);

this.btnDelete.Text = "Delete";

this.btnGenerateData.Location = new System.Drawing.Point(110, 120);

this.btnGenerateData.Size = new System.Drawing.Size(170, 23);

this.btnGenerateData.Text = "Generate Random Data";

//

// DataGridView

//

this.dataGridView.Location = new System.Drawing.Point(12, 170);

this.dataGridView.Size = new System.Drawing.Size(776, 230);

this.dataGridView.AllowUserToAddRows = false;

this.dataGridView.AllowUserToDeleteRows = false;

this.dataGridView.ReadOnly = true;

this.dataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

//

// Status Label

//

this.lblStatus.Location = new System.Drawing.Point(12, 410);

this.lblStatus.Size = new System.Drawing.Size(776, 23);

this.lblStatus.Text = "Ready";

//

// Measure Button

//

this.btnMeasure.Location = new System.Drawing.Point(320, 12);

this.btnMeasure.Size = new System.Drawing.Size(150, 23);

this.btnMeasure.Text = "Measure Performance";

//

// Add controls to form

//

this.groupBoxOperations.Controls.Add(this.lblKey);

this.groupBoxOperations.Controls.Add(this.lblData);

this.groupBoxOperations.Controls.Add(this.txtKey);

this.groupBoxOperations.Controls.Add(this.txtData);

this.groupBoxOperations.Controls.Add(this.btnSearch);

this.groupBoxOperations.Controls.Add(this.btnAdd);

this.groupBoxOperations.Controls.Add(this.btnUpdate);

this.groupBoxOperations.Controls.Add(this.btnDelete);

this.groupBoxOperations.Controls.Add(this.btnGenerateData);

this.Controls.Add(this.groupBoxOperations);

this.Controls.Add(this.dataGridView);

this.Controls.Add(this.lblStatus);

this.Controls.Add(this.btnMeasure);

// Wire up event handlers

this.btnSearch.Click += new System.EventHandler(this.BtnSearch\_Click);

this.btnAdd.Click += new System.EventHandler(this.BtnAdd\_Click);

this.btnUpdate.Click += new System.EventHandler(this.BtnUpdate\_Click);

this.btnDelete.Click += new System.EventHandler(this.BtnDelete\_Click);

this.btnGenerateData.Click += new System.EventHandler(this.BtnGenerateData\_Click);

this.btnMeasure.Click += new System.EventHandler(this.BtnMeasure\_Click);

}

}

namespace Laba3;

using System;

using System.Windows.Forms;

public static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

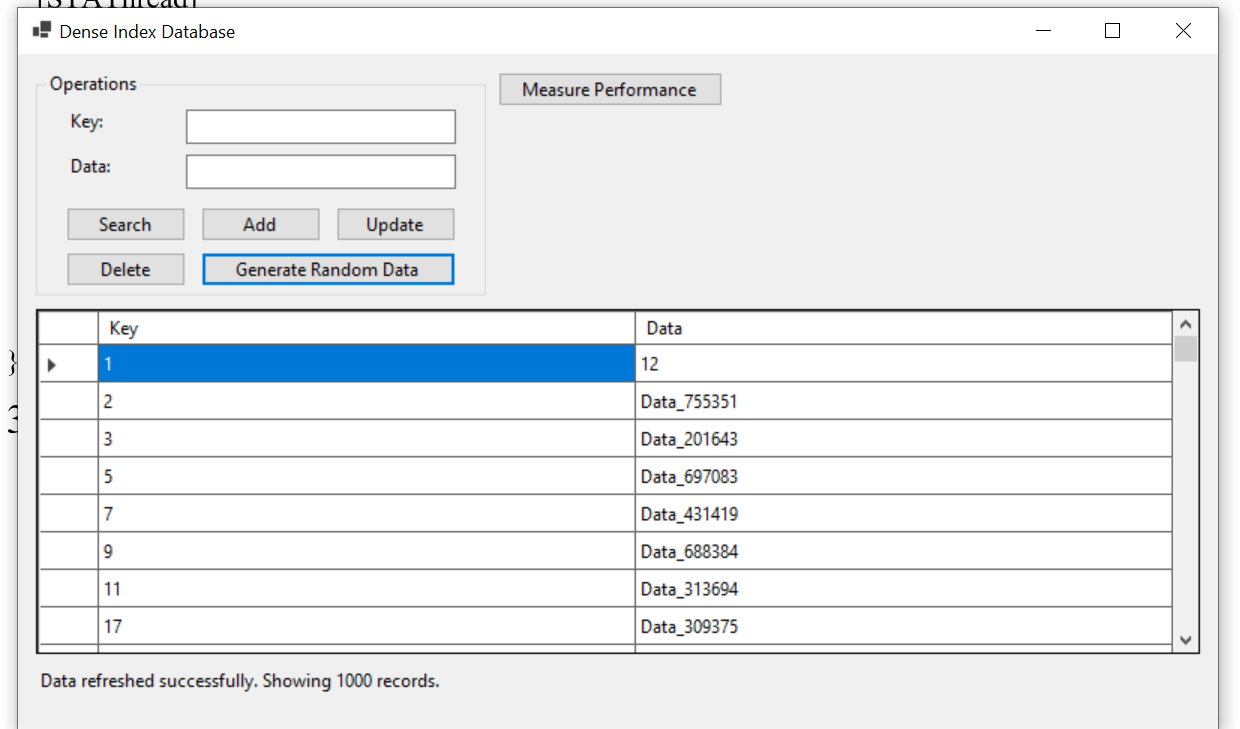
Application.Run(new MainForm());

}

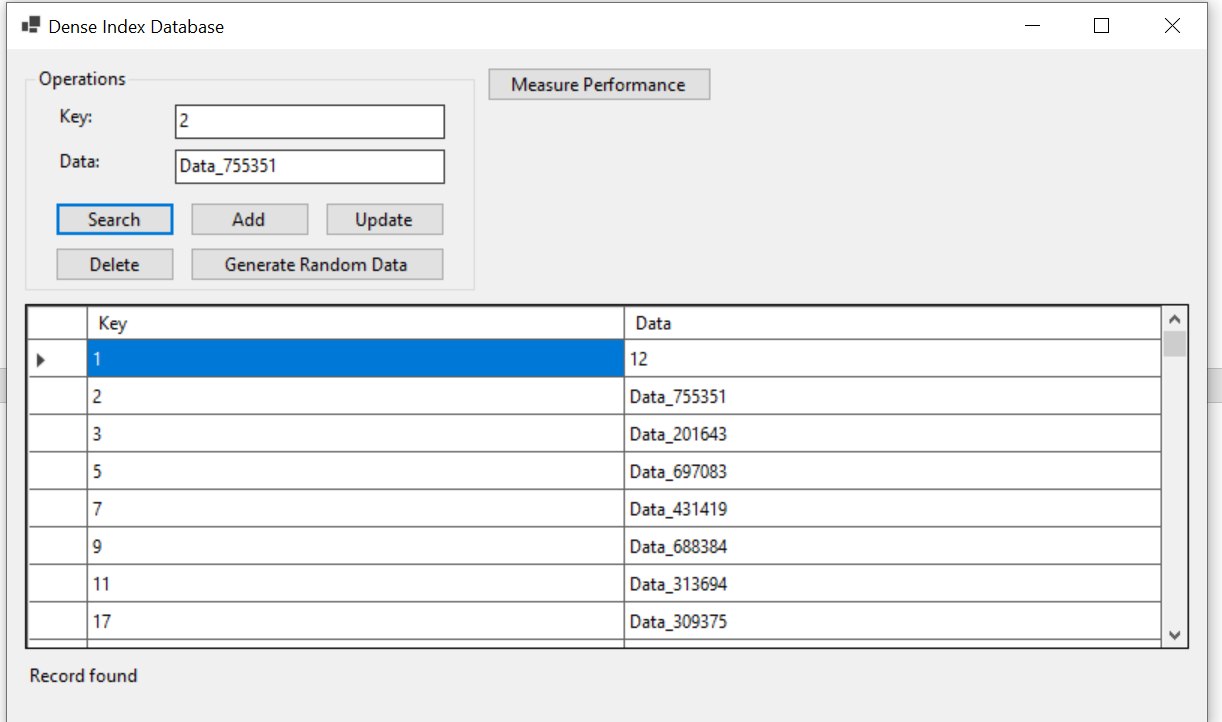
}

3.3.2 Приклади роботи

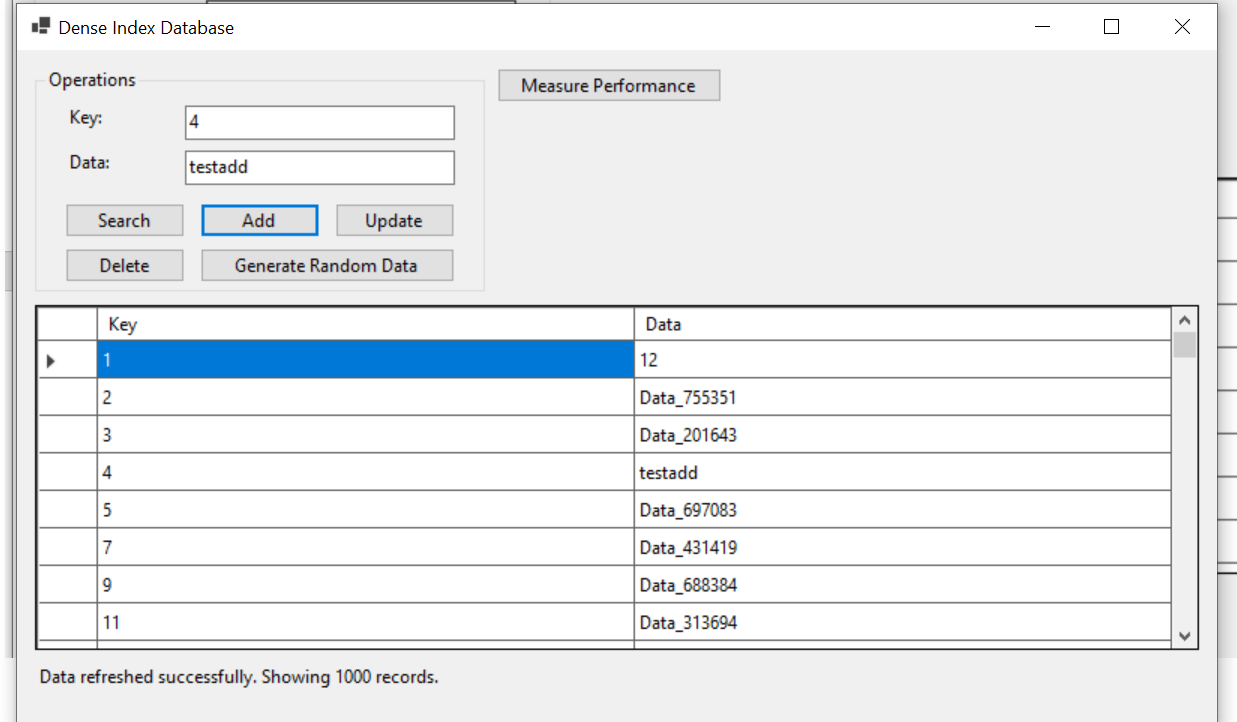
Генерація данних



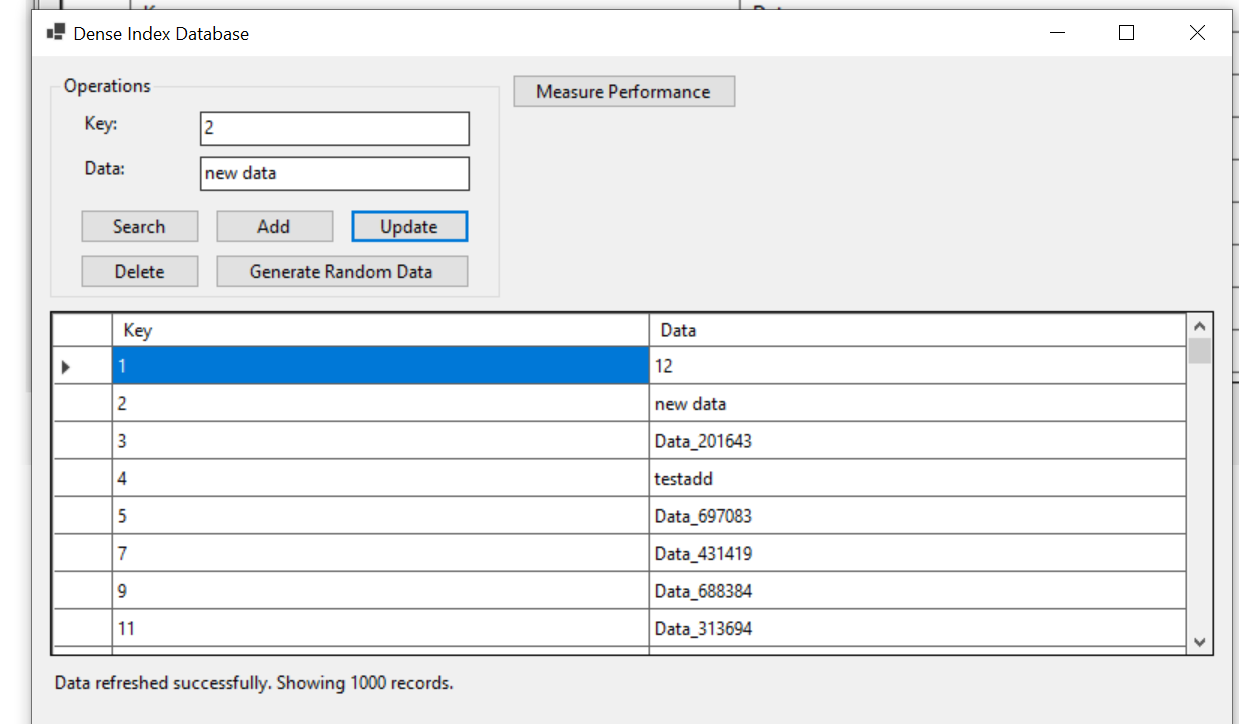
Пошук



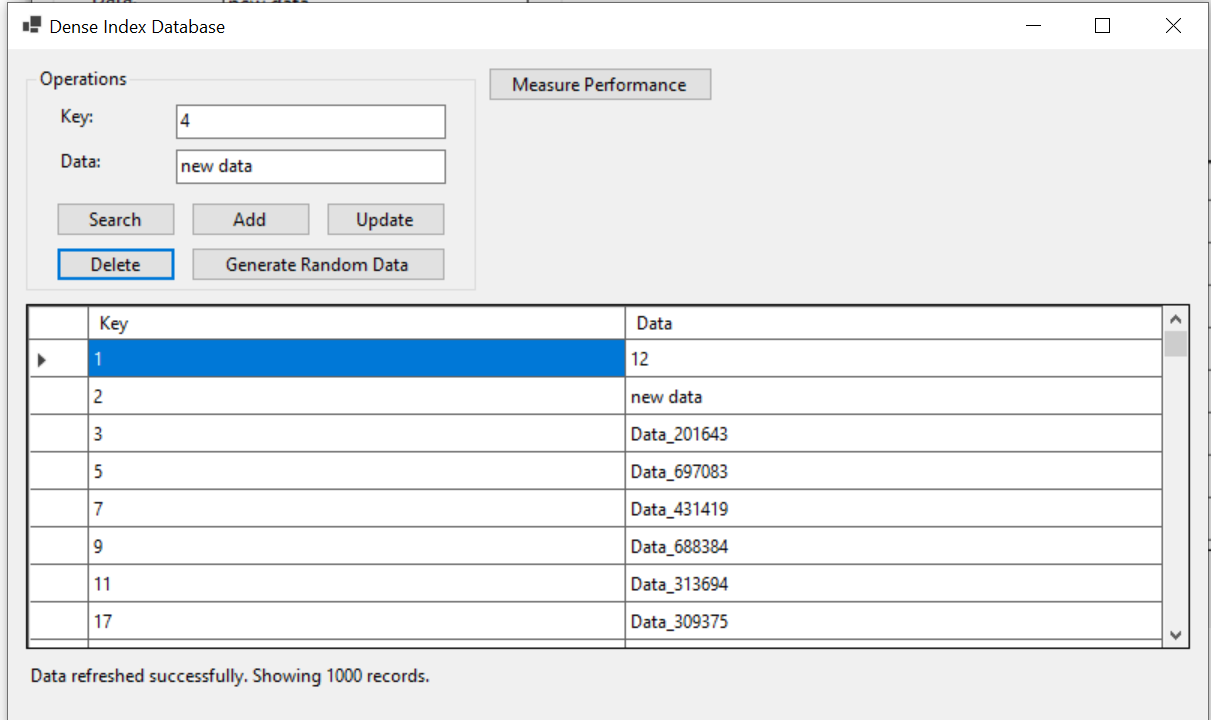
Додавання



Оновлення



Видалення

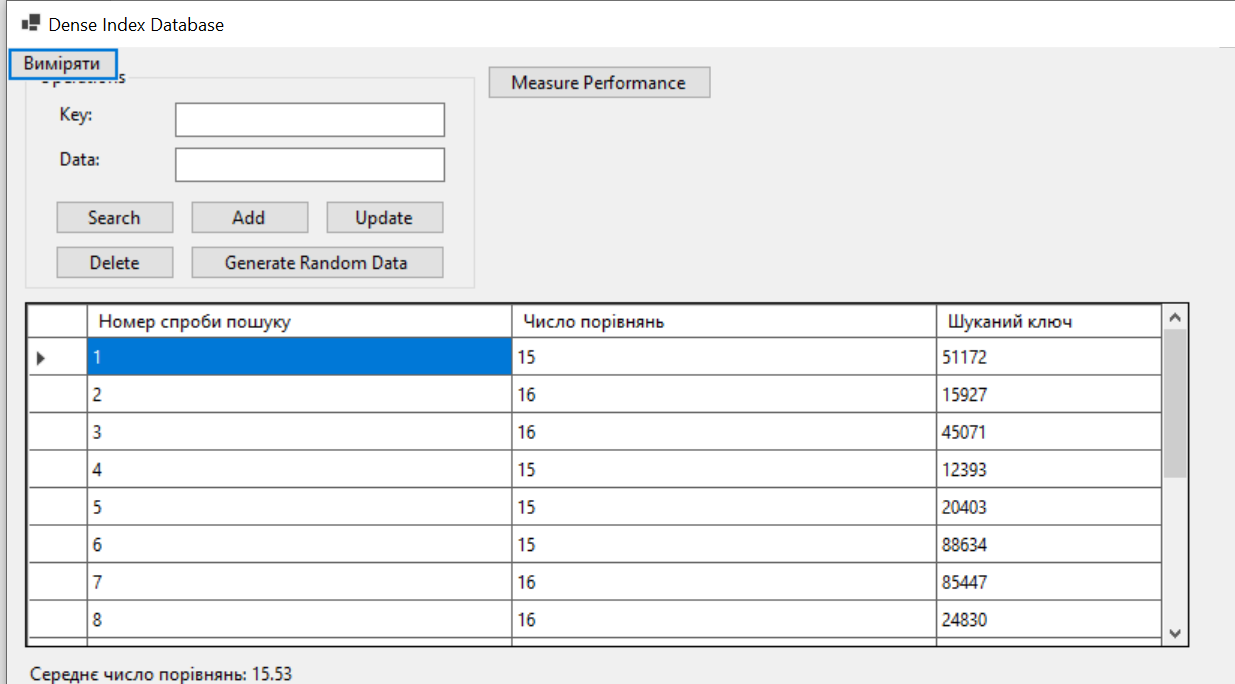


Середій час пошуку



3.4 ТЕСТУВАННЯ АЛГОРИТМУ

3.4.1 Часові характеристики оцінювання



ВИСНОВОК