**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 № 3 з дисципліни

«Проектування алгоритмів»

„ **Проектування структур даних**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-15 Поліщук Валерій*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2022

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)

# Мета лабораторної роботи

Мета роботи – вивчити основні підходи проектування та обробки складних структур даних.

# 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 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.

Зробити висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |
| --- | --- |
| **№** | **Структура даних** |
| 1 | Файли з щільним індексом з перебудовою індексної області, бінарний пошук |
| 2 | Файли з щільним індексом з областю переповнення, бінарний пошук |
| 3 | Файли з не щільним індексом з перебудовою індексної області, бінарний пошук |
| 4 | Файли з не щільним індексом з областю переповнення, бінарний пошук |
| 5 | АВЛ-дерево |
| 6 | Червоно-чорне дерево |
| 7 | B-дерево t=10, бінарний пошук |
| 8 | B-дерево t=25, бінарний пошук |
| 9 | B-дерево t=50, бінарний пошук |
| 10 | B-дерево t=100, бінарний пошук |
| 11 | Файли з щільним індексом з перебудовою індексної області, однорідний бінарний пошук |
| 12 | Файли з щільним індексом з областю переповнення, однорідний бінарний пошук |
| 13 | Файли з не щільним індексом з перебудовою індексної області, однорідний бінарний пошук |
| 14 | Файли з не щільним індексом з областю переповнення, однорідний бінарний пошук |
| 15 | АВЛ-дерево |
| 16 | Червоно-чорне дерево |
| 17 | B-дерево t=10, однорідний бінарний пошук |
| 18 | B-дерево t=25, однорідний бінарний пошук |
| 19 | B-дерево t=50, однорідний бінарний пошук |
| 20 | B-дерево t=100, однорідний бінарний пошук |
| 21 | Файли з щільним індексом з перебудовою індексної області, метод Шарра |
| 22 | Файли з щільним індексом з областю переповнення, метод Шарра |
| 23 | Файли з не щільним індексом з перебудовою індексної області, метод Шарра |
| 24 | Файли з не щільним індексом з областю переповнення, метод Шарра |
| 25 | АВЛ-дерево |
| 26 | Червоно-чорне дерево |
| 27 | B-дерево t=10, метод Шарра |
| 28 | B-дерево t=25, метод Шарра |
| 29 | B-дерево t=50, метод Шарра |
| 30 | B-дерево t=100, метод Шарра |
| 31 | АВЛ-дерево |
| 32 | Червоно-чорне дерево |
| 33 | B-дерево t=250, бінарний пошук |
| 34 | B-дерево t=250, однорідний бінарний пошук |
| 35 | B-дерево t=250, метод Шарра |

# Виконання

## Псевдокод алгоритмів

**private static int Search(long key)**

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

List<String> indexLines = File.ReadAllLines("records.ind").ToList();

blockNum = int.Parse(key.ToString()[0].ToString());

**Повторити** для і від blockSize\*(blockNum-1) до blockSize\*blockNum

**Якщо** (indexLines[i]= «»)

**То**

*Зупинити*

**Інакше**

fields = indexLines[i].Split(' ')

indexes.Add(new Index(Convert.ToInt64(fields[0]), Convert.ToInt32(fields[1])))

**Все Якщо**

**Все Повторити**

long[] index\_arr = new long[indexes.Count];

inc = 0;

**Для кожного** елемента ind у списку indexes

index\_arr[inc] = ind.key;

inc++;

**Все для кожного**

count\_rf = 0

*Повернути* SharSort.SharSearch(index\_arr, key, ref count\_rf)

**private static void Delete(long key)**

List<string> indexLines = File.ReadAllLines("records.ind").ToList();

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

position = Search(key)

foundRecNum = indexes[position].number;

indexLines[WorkingWithFiles.blockSize \* (blockNum - 1) + position] = "";

**Повторити** для і від blockSize \* (blockNum - 1) + position до blockSize \* blockNum

**Якщо** (indexLines[i+1]= «»)

**То**

*Зупинити*

**Інакше**

temp = indexLines[i + 1];

indexLines[i + 1] = indexLines[i];

indexLines[i] = temp;

**Все Якщо**

**Все Повторити**

records[foundRecNum].deleted = true;

File.WriteAllLines("records.ind", indexLines);

WorkingWithFiles.WriteInFile("records.dat", records);

**private static void Edit(long key, name, surname, phoneNum)**

List<string> indexLines = File.ReadAllLines("records.ind").ToList();

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

position = Search(key)

foundRecNum = indexes[position].number;

records[foundRecNum].name = name;

records[foundRecNum].surname = surname;

records[foundRecNum].phoneNumber = phoneNum;

WorkingWithFiles.WriteInFile("records.dat", records);

**private static void Add(name, surname, phoneNum)**

List<string> indexLines = File.ReadAllLines("records.ind").ToList();

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

string skey = rnd.Next(1, 9).ToString() + DateTime.Now.Ticks.ToString();

long key = Convert.ToInt64(skey);

Record newRecord = new Record(key, name, surname, phoneNum, false);

records.Add(newRecord);

int blockNum = int.Parse(key.ToString()[0].ToString());

List<Index> indexes = new List<Index>();

**Повторити** для і від blockSize\*(blockNum-1) до blockSize\*blockNum

**Якщо** (indexLines[i]= «»)

**То**

*Зупинити*

**Інакше**

fields = indexLines[i].Split(' ')

indexes.Add(new Index(Convert.ToInt64(fields[0]), Convert.ToInt32(fields[1])))

**Все Якщо**

**Все Повторити**

**Якщо (**indexes.Count >blockSize**)**

**То**

List<string> newIndexLines = new List<string>()

**Повторити** для і від 1 до 9

**Повторити** для j від blockSize \* (i - 1) до blockSize \* i

newIndexLines.Add(indexLines[j])

**Все Повторити**

**Повторити** для l від 0 до blockSize

newIndexLines.Add("")

**Все Повторити**

**Все Повторити**

blockSize\*=2;

indexLines = newInexLines;

**Все Якщо**

Index newInd = new Index(key, records.Count - 1);

indexes.Add(newInd);

indexes.Sort((p, q) => p.key.CompareTo(q.key));

List<string> newBlockInd = new List<string>();

**Для кожного** елемента item у списку indexes

newBlockInd.Add(item.ToString());

**Все для кожного**

**Повторити** для l від 0 до blockSize - indexes.Count

newBlockInd.Add("");

**Все повторити**

Index = 0

**Повторити** для i від blockSize \* (blockNum - 1) до blockSize \* blockNum

indexLines[i] = newBlockInd[index];

index++;

**Все повторити**

WorkingWithFiles.WriteInFile("records.dat", records);

File.WriteAllLines("records.ind", indexLines);

## Часова складність пошуку

O()

## Програмна реалізація

### Вихідний код

**MainForm.cs**

using System.Xml.Linq;

namespace DenseIndex

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

private void MainForm\_Load(object sender, EventArgs e)

{

listView1.HideSelection = false;

listView1.FullRowSelect = true;

listView1.MultiSelect = false;

LoadListView();

WorkingWithFiles.blockSize = GetBlockSize();

Console.WriteLine();

}

private void LoadListView()

{

listView1.Items.Clear();

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

foreach (Record record in records)

{

if (!record.deleted)

{

string[] row = { record.key.ToString(), record.name, record.surname, record.phoneNumber };

var listItem = new ListViewItem(row);

listView1.Items.Add(listItem);

}

}

}

private int GetBlockSize()

{

//List<string> lines = File.ReadAllLines("records.ind").ToList();

//char curNum = '-';

//int num = 0;

//foreach (var line in lines)

//{

// if (line != "" && line != "/n" && line != "\r" && line is not null)

// {

// curNum = line.Split(' ')[0][0];

// num = lines.FindIndex(x => x == line);

// break;

// }

//}

//if (curNum == '-')

//{

// return 0;

//}

//int count = 1;

//for (int i = num + 1; i < lines.Count; i++)

//{

// if (lines[i] != "" && lines[i] != "/n" && lines[i] != "\r" && lines[i] is not null)

// {

// string[] fields = lines[i].Split(' ');

// char ch = fields[0][0];

// if (ch != curNum)

// {

// return count;

// }

// }

// count++;

//}

//return count;

List<string> lines = File.ReadAllLines("records.ind").ToList();

return lines.Count / 8;

}

private void button4\_Click(object sender, EventArgs e)

{

string key\_s = textBox\_key.Text;

bool keyVal = ValidateKey(key\_s);

if (!keyVal)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"Key is incorrect. It must be 19 digits number",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

long key = Convert.ToInt64(key\_s);

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

List<string> indexLines = File.ReadAllLines("records.ind").ToList();

int blockNum = int.Parse(key.ToString()[0].ToString());

List<Index> indexes = new List<Index>();

for (int i = WorkingWithFiles.blockSize \* (blockNum - 1); i < WorkingWithFiles.blockSize \* blockNum; i++)

{

if (indexLines[i] == "\n" || indexLines[i] == "\r" || indexLines[i] == "" || indexLines[i] is null)

{

break;

}

else

{

string[] fields = indexLines[i].Split(' ');

indexes.Add(new Index(Convert.ToInt64(fields[0]), Convert.ToInt32(fields[1])));

}

}

long[] index\_arr = new long[indexes.Count];

int inc = 0;

foreach (var ind in indexes)

{

index\_arr[inc] = ind.key;

inc++;

}

int count\_rf = 0;

int position = SharSort.SharSearch(index\_arr, key, ref count\_rf);

if (index\_arr[position]!=key || position==-1)

{

position = Array.BinarySearch(index\_arr, key);

}

if (position < 0)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"The record with specified key was not found",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

int foundRecNum = indexes[position].number;

int delcount = 0;

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

{

if (records[i].deleted)

{

delcount++;

}

}

listView1.Focus();

listView1.Items[foundRecNum - delcount].Selected = true;

listView1.EnsureVisible(foundRecNum - delcount);

}

private void button1\_Click(object sender, EventArgs e)

{

//DELETE

DialogResult delAns = MessageBox.Show(

"Are you sure you want to delete the entry",

"Alert",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button2,

MessageBoxOptions.DefaultDesktopOnly);

if (delAns == DialogResult.Yes)

{

bool selected = listView1.SelectedItems.Count > 0;

if (!selected)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"The record is not selected",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns==DialogResult.OK)

{

Activate();

}

return;

}

this.Activate();

ListViewItem item = listView1.SelectedItems[0];

long key = Convert.ToInt64(item.Text.Split(' ')[0]);

List<string> indexLines = File.ReadAllLines("records.ind").ToList();

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

int blockNum = int.Parse(key.ToString()[0].ToString());

List<Index> indexes = new List<Index>();

for (int i = WorkingWithFiles.blockSize \* (blockNum - 1); i < WorkingWithFiles.blockSize \* blockNum; i++)

{

if (indexLines[i] == "\n" || indexLines[i] == "\r" || indexLines[i] == "" || indexLines[i] is null)

{

break;

}

else

{

string[] fields = indexLines[i].Split(' ');

indexes.Add(new Index(Convert.ToInt64(fields[0]), Convert.ToInt32(fields[1])));

}

}

long[] index\_arr = new long[indexes.Count];

int inc = 0;

foreach (var ind in indexes)

{

index\_arr[inc] = ind.key;

inc++;

}

int count\_rf = 0;

int position = SharSort.SharSearch(index\_arr, key, ref count\_rf);

if (index\_arr[position] != key || position == -1)

{

position = Array.BinarySearch(index\_arr, key);

}

int foundRecNum = indexes[position].number;

indexLines[WorkingWithFiles.blockSize \* (blockNum - 1) + position] = "";

for (int i = WorkingWithFiles.blockSize \* (blockNum - 1) + position; i < WorkingWithFiles.blockSize \* blockNum; i++)

{

if (indexLines[i + 1] == "" || indexLines[i + 1] == "/n" || indexLines[i + 1] == "/r" || indexLines[i + 1] is null)

{

break;

}

string temp = indexLines[i + 1];

indexLines[i + 1] = indexLines[i];

indexLines[i] = temp;

}

records[foundRecNum].deleted = true;

File.WriteAllLines("records.ind", indexLines);

WorkingWithFiles.WriteInFile("records.dat", records);

LoadListView();

}

else

{

Activate();

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void textBox\_surname\_TextChanged(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

//EDIT

DialogResult editAns = MessageBox.Show(

"Are you sure you want to edit the entry",

"Alert",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button2,

MessageBoxOptions.DefaultDesktopOnly);

if (editAns == DialogResult.Yes)

{

#region prep

Activate();

string name = textBox\_name.Text;

string surname = textBox\_surname.Text;

string phoneNum = textBox\_phone\_num.Text;

bool nameVal = ValidateName(name);

bool surnameVal = ValidateSurname(surname);

bool phoneVal = ValidatePhoneNum(phoneNum);

if (listView1.SelectedItems.Count < 1)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"The record is not selected",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

if (!nameVal)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"Name is incorrect. Fitst letter must be upper case, it must not contain spaces, maximum 30 chars",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

if (!surnameVal)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"Suranme is incorrect. Fitst letter must be upper case, it must not contain spaces, maximum 30 chars",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

if (!phoneVal)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"Phone number is incorrect. Fitst symbol must +, other chars must be digits, length must be 12",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

#endregion

ListViewItem item = listView1.SelectedItems[0];

long key = Convert.ToInt64(item.Text);

List<string> indexLines = File.ReadAllLines("records.ind").ToList();

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

int blockNum = int.Parse(key.ToString()[0].ToString());

List<Index> indexes = new List<Index>();

for (int i = WorkingWithFiles.blockSize \* (blockNum - 1); i < WorkingWithFiles.blockSize \* blockNum; i++)

{

if (indexLines[i] == "\n" || indexLines[i] == "\r" || indexLines[i] == "" || indexLines[i] is null)

{

break;

}

else

{

string[] fields = indexLines[i].Split(' ');

indexes.Add(new Index(Convert.ToInt64(fields[0]), Convert.ToInt32(fields[1])));

}

}

long[] index\_arr = new long[indexes.Count];

int inc = 0;

foreach (var ind in indexes)

{

index\_arr[inc] = ind.key;

inc++;

}

int count\_rf = 0;

int position = SharSort.SharSearch(index\_arr, key, ref count\_rf);

if (index\_arr[position] != key || position == -1)

{

position = Array.BinarySearch(index\_arr, key);

}

int foundRecNum = indexes[position].number;

records[foundRecNum].name = name;

records[foundRecNum].surname = surname;

records[foundRecNum].phoneNumber = phoneNum;

WorkingWithFiles.WriteInFile("records.dat", records);

LoadListView();

}

else

{

Activate();

}

}

private bool ValidateName(string name)

{

if (name.Length == 0 || name == "/n" || name is null || name == "\r" || name == "")

{

return false;

}

if (name.Contains(' '))

{

return false;

}

if (name.Length > 30)

{

return false;

}

if (name.Any(x => !char.IsLetter(x)))

{

return false;

}

if (!Char.IsUpper(name[0]))

{

return false;

}

return true;

}

private bool ValidateSurname(string surname)

{

if (surname.Length == 0 || surname == "/n" || surname is null || surname == "\r" || surname == "")

{

return false;

}

if (surname.Contains(' '))

{

return false;

}

if (surname.Length > 30)

{

return false;

}

if (surname.Any(x => !char.IsLetter(x)))

{

return false;

}

if (!Char.IsUpper(surname[0]))

{

return false;

}

return true;

}

private bool ValidatePhoneNum(string num)

{

if (num is null)

{

return false;

}

if (num.Length != 12)

{

return false;

}

if (num[0] != '+')

{

return false;

}

for (int i = 1; i < 12; i++)

{

if (!char.IsDigit(num[i]))

{

return false;

}

}

return true;

}

private bool ValidateKey(string key)

{

if (key is null)

{

return false;

}

if (key.Length != 19)

{

return false;

}

if (key.Any(x => !char.IsDigit(x)))

{

return false;

}

if (key[0] == '0' || key[0] == '9')

{

return false;

}

return true;

}

private void button3\_Click(object sender, EventArgs e)

{

//ADD

DialogResult editAns = MessageBox.Show(

"Are you sure you want to add the entry",

"Alert",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button2,

MessageBoxOptions.DefaultDesktopOnly);

if (editAns == DialogResult.Yes)

{

#region prep

Activate();

string name = textBox\_name.Text;

string surname = textBox\_surname.Text;

string phoneNum = textBox\_phone\_num.Text;

bool nameVal = ValidateName(name);

bool surnameVal = ValidateSurname(surname);

bool phoneVal = ValidatePhoneNum(phoneNum);

if (!nameVal)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"Name is incorrect. Fitst letter must be upper case, it must not contain spaces, maximum 30 chars",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

if (!surnameVal)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"Suranme is incorrect. Fitst letter must be upper case, it must not contain spaces, maximum 30 chars",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

if (!phoneVal)

{

Activate();

DialogResult infoAns = MessageBox.Show(

"Phone number is incorrect. Fitst symbol must +, other chars must be digits, length must be 12",

"Alert",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if (infoAns == DialogResult.OK)

{

Activate();

}

return;

}

#endregion

List<string> indexLines = File.ReadAllLines("records.ind").ToList();

List<Record> records = WorkingWithFiles.ReadFromFile("records.dat");

Random rnd = new Random();

string skey = rnd.Next(1, 9).ToString() + DateTime.Now.Ticks.ToString();

long key = Convert.ToInt64(skey);

Record newRecord = new Record(key, name, surname, phoneNum, false);

records.Add(newRecord);

int blockNum = int.Parse(key.ToString()[0].ToString());

List<Index> indexes = new List<Index>();

for (int i = WorkingWithFiles.blockSize \* (blockNum - 1); i < WorkingWithFiles.blockSize \* blockNum; i++)

{

if (indexLines[i] == "\n" || indexLines[i] == "\r" || indexLines[i] == "" || indexLines[i] is null)

{

break;

}

else

{

string[] fields = indexLines[i].Split(' ');

indexes.Add(new Index(Convert.ToInt64(fields[0]), Convert.ToInt32(fields[1])));

}

}

if (indexes.Count < WorkingWithFiles.blockSize)

{

Index newInd = new Index(key, records.Count - 1);

indexes.Add(newInd);

indexes.Sort((p, q) => p.key.CompareTo(q.key));

List<string> newBlockInd = new List<string>();

foreach (var item in indexes)

{

newBlockInd.Add(item.ToString());

}

for (int l = 0; l < WorkingWithFiles.blockSize - indexes.Count; l++)

{

newBlockInd.Add("");

}

int index = 0;

for (int i = WorkingWithFiles.blockSize \* (blockNum - 1); i < WorkingWithFiles.blockSize \* blockNum; i++)

{

indexLines[i] = newBlockInd[index];

index++;

}

WorkingWithFiles.WriteInFile("records.dat", records);

File.WriteAllLines("records.ind", indexLines);

LoadListView();

int addeddRecNum = records.Count - 1;

int delcount = 0;

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

{

if (records[i].deleted)

{

delcount++;

}

}

listView1.Focus();

listView1.Items[addeddRecNum - delcount].Selected = true;

listView1.EnsureVisible(addeddRecNum - delcount);

}

else

{

List<string> newIndexLines = new List<string>();

for (int i = 1; i <= 8; i++)

{

for (int j = WorkingWithFiles.blockSize \* (i - 1); j < WorkingWithFiles.blockSize \* i; j++)

{

newIndexLines.Add(indexLines[j]);

}

for (int l = 0; l < WorkingWithFiles.blockSize; l++)

{

newIndexLines.Add("");

}

}

WorkingWithFiles.blockSize \*= 2;

Index newInd = new Index(key, records.Count - 1);

indexes.Add(newInd);

indexes.Sort((p, q) => p.key.CompareTo(q.key));

List<string> newBlockInd = new List<string>();

foreach (var item in indexes)

{

newBlockInd.Add(item.ToString());

}

for (int l = 0; l < WorkingWithFiles.blockSize - indexes.Count; l++)

{

newBlockInd.Add("");

}

int index = 0;

for (int i = WorkingWithFiles.blockSize \* (blockNum - 1); i < WorkingWithFiles.blockSize \* blockNum; i++)

{

newIndexLines[i] = newBlockInd[index];

index++;

}

WorkingWithFiles.WriteInFile("records.dat", records);

File.WriteAllLines("records.ind", newIndexLines);

LoadListView();

int addeddRecNum = records.Count - 1;

int delcount = 0;

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

{

if (records[i].deleted)

{

delcount++;

}

}

listView1.Focus();

listView1.Items[addeddRecNum - delcount].Selected = true;

listView1.EnsureVisible(addeddRecNum - delcount);

}

}

else

{

Activate();

}

}

}

}

**Record.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DenseIndex

{

public class Record

{

public long key;

public string? name;

public string? surname;

public string? phoneNumber;

public bool deleted;

public Record(long \_key, string? \_name, string? \_surname, string? \_phoneNumber, bool \_deleted)

{

key = \_key;

name = \_name;

surname = \_surname;

phoneNumber = \_phoneNumber;

deleted = \_deleted;

}

public override string ToString()

{

return key.ToString() + " " + name + " " + surname + " " + phoneNumber + " " + deleted.ToString();

}

}

}

**Index.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DenseIndex

{

public class Index

{

public long key;

public int number;

public Index(long \_key, int \_number)

{

key = \_key;

number = \_number;

}

public override string ToString()

{

return key.ToString() + " " + number.ToString();

}

}

}

**SharSort.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DenseIndex

{

public class SharSort

{

private static int UniformBinarySearch(long[] arr, long key, int i, int omega, ref int count)

{

count = 0;

while (omega > 0)

{

count++;

if (i < arr.Length && i >= 0)

{

if (arr[i] == key)

{

return i;

}

else

{

if (arr[i] > key)

{

i = i - (omega / 2 + 1);

if (omega > 1)

{

omega /= 2;

}

}

else

{

i = i + (omega / 2 + 1);

if (omega > 1)

{

omega /= 2;

}

}

}

}

else

{

if (i < 0)

{

i = i + (omega / 2 + 1);

if (omega > 1)

{

omega /= 2;

}

}

else

{

i = i - (omega / 2 + 1);

if (omega > 1)

{

omega /= 2;

}

}

}

}

if (arr[i] == key)

{

return i;

}

return -1;

}

public static int SharSearch(long[] arr, long key, ref int count)

{

int k = Convert.ToInt32(Math.Truncate(Math.Log2(arr.Length)));

long keyI = arr[Convert.ToInt32(Math.Pow(2, k)) - 1];

if (key == keyI)

{

return Convert.ToInt32(Math.Pow(2, k)) - 1;

}

else

{

if (key < keyI)

{

return UniformBinarySearch(arr, key, Convert.ToInt32(Math.Pow(2, k) - 0), 2 \* Convert.ToInt32(Math.Pow(2, (k - 1))), ref count); // i: -1

}

else

{

int l = Convert.ToInt32(Math.Log2(arr.Length - Math.Pow(2, k) + 1));

return UniformBinarySearch(arr, key, Convert.ToInt32(arr.Length + 1 - Math.Pow(2, l) - 0), 2 \* Convert.ToInt32(Math.Pow(2, l - 1)), ref count); // i:-1

}

}

}

}

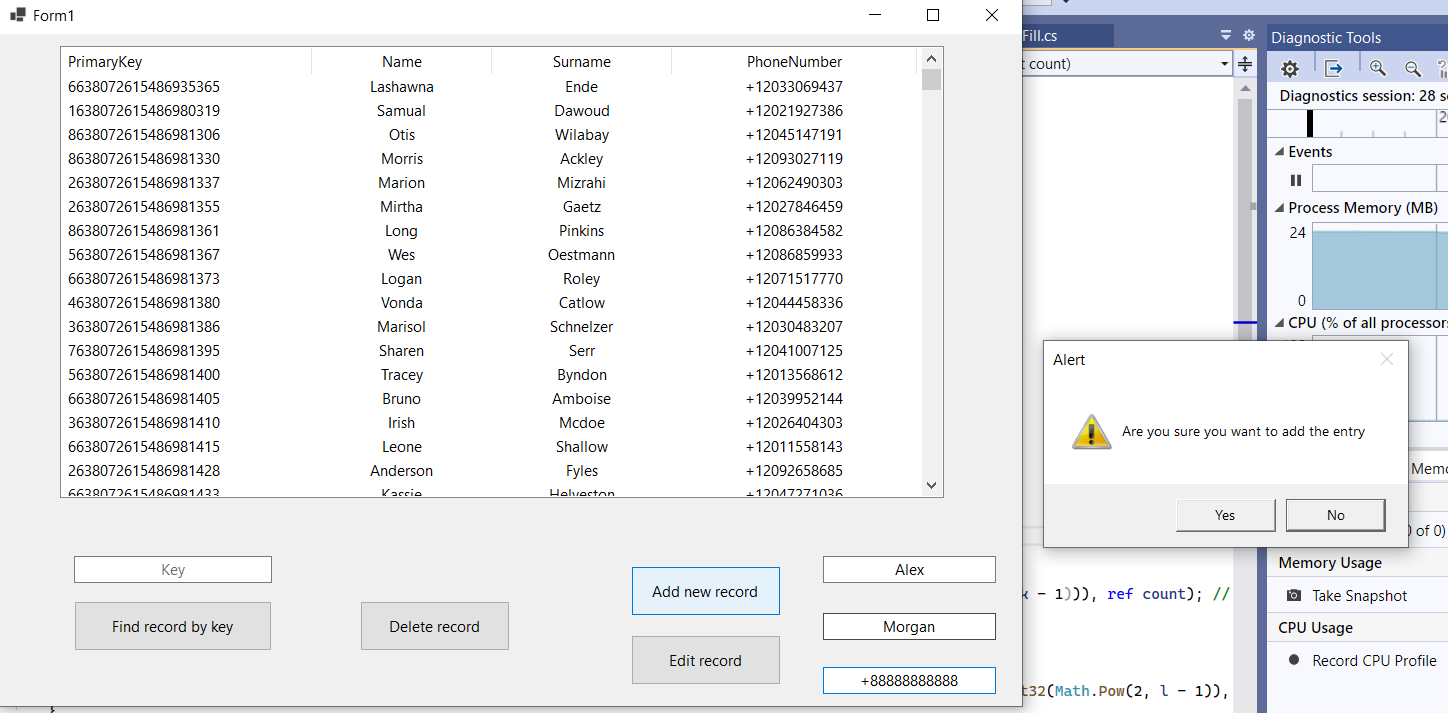
}

**WorkingWithFiles.CS**

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для додавання і пошуку запису.

Рисунок 3.1 –Додавання запису



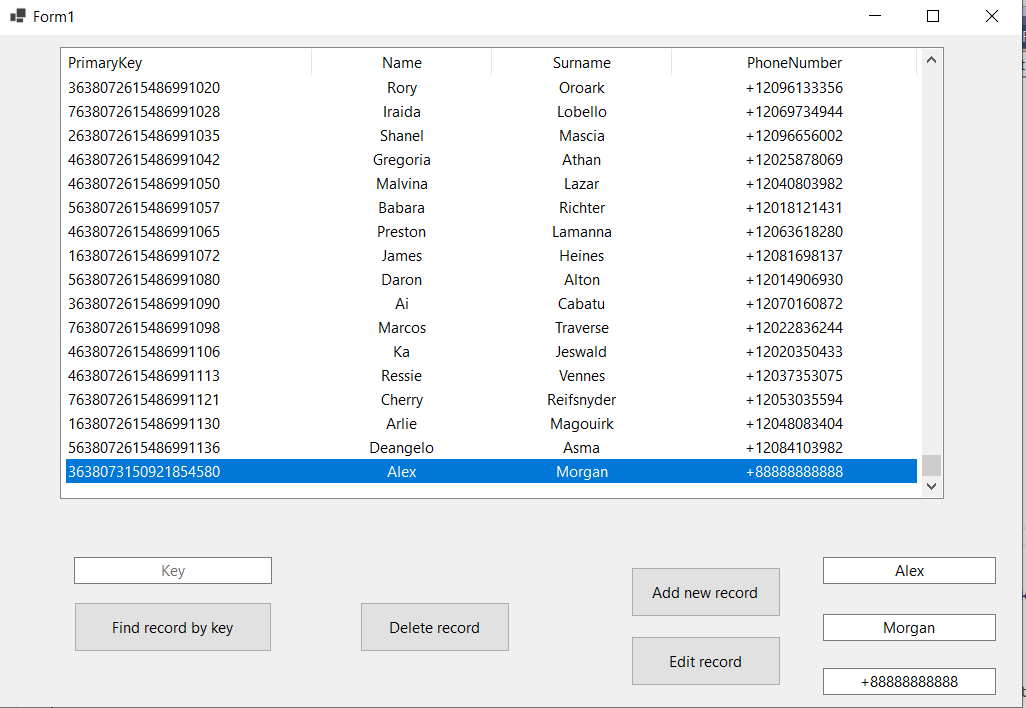
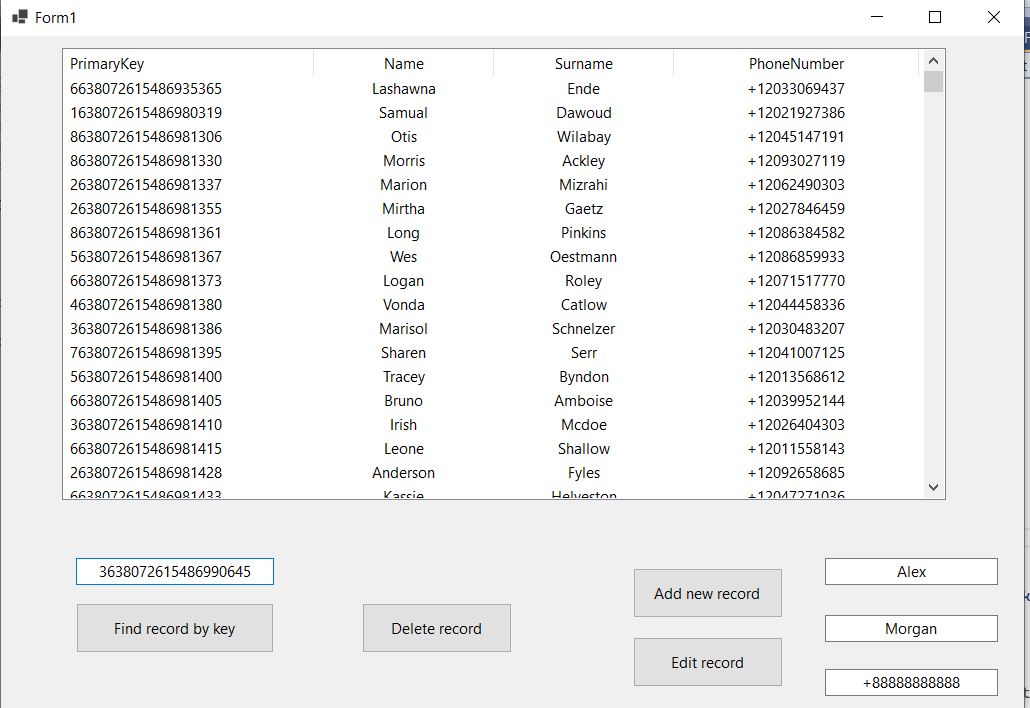
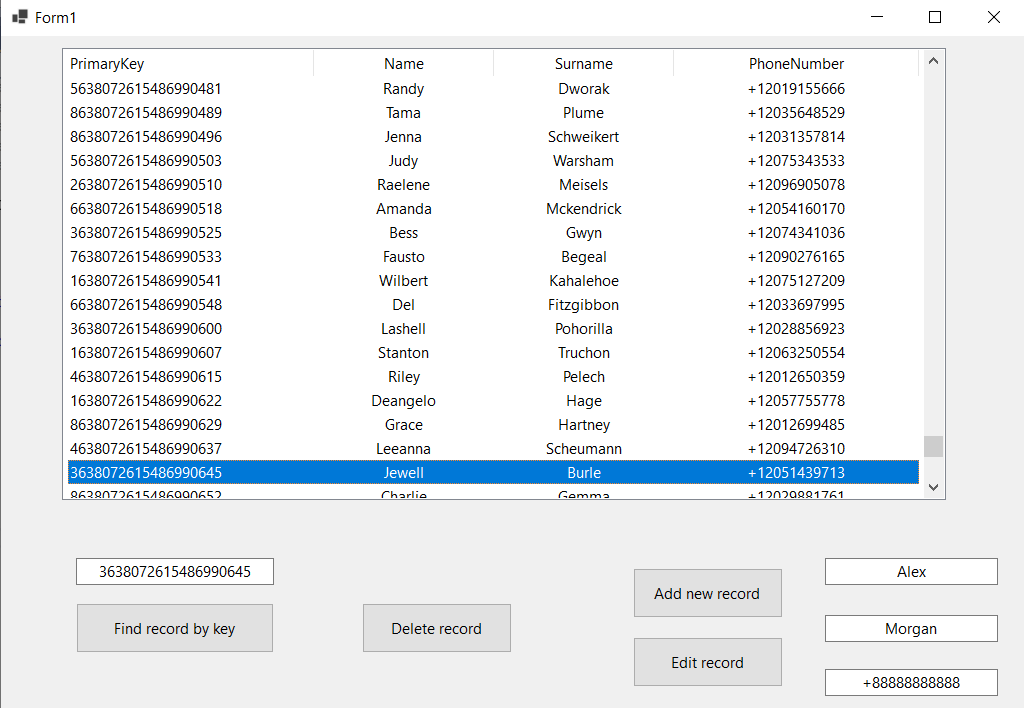


Рисунок 3.2 – Пошук запису





## Тестування алгоритму

### Часові характеристики оцінювання

В таблиці 3.1 наведено кількість порівнянь для 15 спроб пошуку запису по ключу.

Таблиця 3.1 – Число порівнянь при спробі пошуку запису по ключу

|  |  |
| --- | --- |
| Номер спроби пошуку | Число порівнянь |
| 1 | 13 |
| 2 | 15 |
| 3 | 15 |
| 4 | 14 |
| 5 | 14 |
| 6 | 12 |
| 7 | 15 |
| 8 | 14 |
| 9 | 11 |
| 10 | 12 |
| 11 | 13 |
| 12 | 14 |
| 13 | 12 |
| 14 | 15 |
| 15 | 15 |

Висновок

В рамках лабораторної роботи я спроектував складну структуру даних, що представляє собою файл з щільним індексом з перебудовою індексної області. Я виконав програмну реалізацію невеликої СУБД з графічним інтерфейсом користувача та реалізував операції створення, пошуку, редагування та видалення записів. Виконуючу лабораторну роботу, я вивчив основні підходи проектування та обробки складних структур даних.

Критерії оцінювання

За умови здачі лабораторної роботи до 13.11.2022 включно максимальний бал дорівнює – 5. Після 13.11.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

* псевдокод алгоритму – 15%;
* аналіз часової складності – 5%;
* програмна реалізація алгоритму – 65%;
* тестування алгоритму – 10%;
* висновок – 5%.

+1 додатковий бал можна отримати за реалізацію графічного зображення структури ключів.