

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)

ІП-15 Поліщук Валерій
(шифр, прізвище, ім'я, по батькові)

Перевірів

Головченко М.Н.
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1	МЕТА ЛАБОРАТОРНОЇ РОБОТИ	3
2	ЗАВДАННЯ	4
3	ВИКОНАННЯ	7
	3.1 ПСЕВДОКОД АЛГОРИТМІВ.....	7
	3.2 ЧАСОВА СКЛАДНІСТЬ ПОШУКУ.....	9
	3.3 ПРОГРАМНА РЕАЛІЗАЦІЯ	10
	3.3.1 Вихідний код.....	10
	3.3.2 Приклади роботи.....	39
	3.4 ТЕСТУВАННЯ АЛГОРИТМУ.....	42
	3.4.1 Часові характеристики оцінювання.....	42
	ВИСНОВОК	43
	КРИТЕРІЇ ОЦІНЮВАННЯ	44

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

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

2 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	В-дерево t=10, бінарний пошук
8	В-дерево t=25, бінарний пошук
9	В-дерево t=50, бінарний пошук
10	В-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	В-дерево $t=10$, однорідний бінарний пошук
18	В-дерево $t=25$, однорідний бінарний пошук
19	В-дерево $t=50$, однорідний бінарний пошук
20	В-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	В-дерево $t=10$, метод Шарра
28	В-дерево $t=25$, метод Шарра
29	В-дерево $t=50$, метод Шарра
30	В-дерево $t=100$, метод Шарра
31	АВЛ-дерево
32	Червоно-чорне дерево
33	В-дерево $t=250$, бінарний пошук
34	В-дерево $t=250$, однорідний бінарний пошук

3.1 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 i від $\text{blockSize} * (\text{blockNum} - 1) + \text{position}$ до $\text{blockSize} * \text{blockNum}$

Якщо $(\text{indexLines}[i+1] = \ll \gg)$

То

Зупинити

Інакше

$\text{temp} = \text{indexLines}[i + 1];$

$\text{indexLines}[i + 1] = \text{indexLines}[i];$

$\text{indexLines}[i] = \text{temp};$

Все Якщо

Все Повторити

$\text{records}[\text{foundRecNum}].\text{deleted} = \text{true};$

$\text{File}.\text{WriteAllLines}(\text{"records.ind"}, \text{indexLines});$

$\text{WorkingWithFiles}.\text{WriteInFile}(\text{"records.dat"}, \text{records});$

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

$\text{List}\langle\text{string}\rangle \text{indexLines} = \text{File}.\text{ReadAllLines}(\text{"records.ind"}).\text{ToList}();$

$\text{List}\langle\text{Record}\rangle \text{records} = \text{WorkingWithFiles}.\text{ReadFromFile}(\text{"records.dat"});$

$\text{position} = \text{Search}(\text{key})$

$\text{foundRecNum} = \text{indexes}[\text{position}].\text{number};$

$\text{records}[\text{foundRecNum}].\text{name} = \text{name};$

$\text{records}[\text{foundRecNum}].\text{surname} = \text{surname};$

$\text{records}[\text{foundRecNum}].\text{phoneNumber} = \text{phoneNum};$

$\text{WorkingWithFiles}.\text{WriteInFile}(\text{"records.dat"}, \text{records});$

private static void Add(name, surname, phoneNum)

$\text{List}\langle\text{string}\rangle \text{indexLines} = \text{File}.\text{ReadAllLines}(\text{"records.ind"}).\text{ToList}();$

$\text{List}\langle\text{Record}\rangle \text{records} = \text{WorkingWithFiles}.\text{ReadFromFile}(\text{"records.dat"});$

$\text{string} \text{skey} = \text{rnd}.\text{Next}(1, 9).\text{ToString}() + \text{DateTime}.\text{Now}.\text{Ticks}.\text{ToString}();$

$\text{long} \text{key} = \text{Convert}.\text{ToInt64}(\text{skey});$

$\text{Record} \text{newRecord} = \text{new Record}(\text{key}, \text{name}, \text{surname}, \text{phoneNum}, \text{false});$

$\text{records}.\text{Add}(\text{newRecord});$

$\text{int} \text{blockNum} = \text{int}.\text{Parse}(\text{key}.\text{ToString}()[0].\text{ToString}());$

$\text{List}\langle\text{Index}\rangle \text{indexes} = \text{new List}\langle\text{Index}\rangle();$

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 = newIndexLines;

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);

3.2 Часова складність пошуку

$O(\log_2 n)$

3.3 Програмна реалізація

3.3.1 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.ReadFile("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. First 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. First 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(
        "Surname is incorrect. First 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;
                }
            }
        }
    }
}
```

```

        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

3.3.2 Приклади роботи

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

Рисунок 3.1 –Додавання запису

Form1

PrimaryKey	Name	Surname	PhoneNumber
6638072615486935365	Lashawna	Ende	+12033069437
1638072615486980319	Samual	Dawoud	+12021927386
8638072615486981306	Otis	Wilabay	+12045147191
8638072615486981330	Morris	Ackley	+12093027119
2638072615486981337	Marion	Mizrahi	+12062490303
2638072615486981355	Mirtha	Gaetz	+12027846459
8638072615486981361	Long	Pinkins	+12086384582
5638072615486981367	Wes	Oestmann	+12086859933
6638072615486981373	Logan	Roley	+12071517770
4638072615486981380	Vonda	Catlow	+12044458336
3638072615486981386	Marisol	Schnelzer	+12030483207
7638072615486981395	Sharen	Serr	+12041007125
5638072615486981400	Tracey	Byndon	+12013568612
6638072615486981405	Bruno	Amboise	+12039952144
3638072615486981410	Irish	Mcdoe	+12026404303
6638072615486981415	Leone	Shallow	+12011558143
2638072615486981428	Anderson	Fyles	+12092658685
6638072615486981433	Kacie	Halveston	+12047271036

Key

Find record by key

Add new record

Edit record

Alex

Morgan

+88888888888

Delete record

Diagnostic Tools

Diagnostics session: 28 s

Events

Process Memory (MB)

CPU (% of all processors)

Alert

Are you sure you want to add the entry

Yes

No

Memory Usage

Take Snapshot

CPU Usage

Record CPU Profile

Form1

PrimaryKey	Name	Surname	PhoneNumber
3638072615486991020	Rory	Oroark	+12096133356
7638072615486991028	Iraida	Lobello	+12069734944
2638072615486991035	Shanel	Mascia	+12096656002
4638072615486991042	Gregoria	Athan	+12025878069
4638072615486991050	Malvina	Lazar	+12040803982
5638072615486991057	Babara	Richter	+12018121431
4638072615486991065	Preston	Lamanna	+12063618280
1638072615486991072	James	Heines	+12081698137
5638072615486991080	Daron	Alton	+12014906930
3638072615486991090	Ai	Cabatu	+12070160872
7638072615486991098	Marcos	Traverse	+12022836244
4638072615486991106	Ka	Jeswald	+12020350433
4638072615486991113	Ressie	Vennes	+12037353075
7638072615486991121	Cherry	Reifsnnyder	+12053035594
1638072615486991130	Arlie	Magouirk	+12048083404
5638072615486991136	Deangelo	Asma	+12084103982
3638073150921854580	Alex	Morgan	+88888888888

Key

Find record by key

Add new record

Edit record

Alex

Morgan

+88888888888

Delete record

Рисунок 3.2 – Пошук запису

Form1

PrimaryKey	Name	Surname	PhoneNumber
6638072615486935365	Lashawna	Ende	+12033069437
1638072615486980319	Samual	Dawoud	+12021927386
8638072615486981306	Otis	Wilabay	+12045147191
8638072615486981330	Morris	Ackley	+12093027119
2638072615486981337	Marion	Mizrahi	+12062490303
2638072615486981355	Mirtha	Gaetz	+12027846459
8638072615486981361	Long	Pinkins	+12086384582
5638072615486981367	Wes	Oestmann	+12086859933
6638072615486981373	Logan	Roley	+12071517770
4638072615486981380	Vonda	Catlow	+12044458336
3638072615486981386	Marisol	Schnelzer	+12030483207
7638072615486981395	Sharen	Serr	+12041007125
5638072615486981400	Tracey	Byndon	+12013568612
6638072615486981405	Bruno	Amboise	+12039952144
3638072615486981410	Irish	Mcdoe	+12026404303
6638072615486981415	Leone	Shallow	+12011558143
2638072615486981428	Anderson	Fyles	+12092658685
6638072615486981433	Kacie	Helveston	+12047271036

3638072615486990645

Find record by key

Delete record

Add new record

Edit record

Alex

Morgan

+88888888888

Form1

PrimaryKey	Name	Surname	PhoneNumber
5638072615486990481	Randy	Dworak	+12019155666
8638072615486990489	Tama	Plume	+12035648529
8638072615486990496	Jenna	Schweikert	+12031357814
5638072615486990503	Judy	Warsham	+12075343533
2638072615486990510	Raelene	Meisels	+12096905078
6638072615486990518	Amanda	Mckendrick	+12054160170
3638072615486990525	Bess	Gwyn	+12074341036
7638072615486990533	Fausto	Begeal	+12090276165
1638072615486990541	Wilbert	Kahalehoe	+12075127209
6638072615486990548	Del	Fitzgibbon	+12033697995
3638072615486990600	Lashell	Pohorilla	+12028856923
1638072615486990607	Stanton	Truchon	+12063250554
4638072615486990615	Riley	Pelech	+12012650359
1638072615486990622	Deangelo	Hage	+12057755778
8638072615486990629	Grace	Hartney	+12012699485
4638072615486990637	Leeanna	Scheumann	+12094726310
3638072615486990645	Jewell	Burle	+12051439713
8638072615486990652	Charlie	Gemma	+12029881761

3638072615486990645

Find record by key

Delete record

Add new record

Edit record

Alex

Morgan

+88888888888

3.4 Тестування алгоритму

3.4.1 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.