

Національний Технічний Університет України  
«Київський Політехнічний Інститут»  
Інститут Прикладного Системного Аналізу  
Кафедра Системного Проектування

*Курсова робота*  
дисципліна  
*“Алгоритмізація та програмування”*

Керівник:  
Романов В.В.

Виконавець:  
Лобач Г.Ю.

“Захист дозволено”

гр. ДА-82

“ \_\_\_\_ ” \_\_\_\_\_ 2019 р.

/підпис керівника/

/підпис/

Захищено з оцінкою

**Національний технічний університет України “КПІ”**  
**Інститут Прикладного Системного Аналізу**

Кафедра \_\_\_\_\_ системного проектування \_\_\_\_\_

Дисципліна \_\_\_\_\_ алгоритмізація та програмування \_\_\_\_\_

Спеціальність \_\_\_\_\_ Комп’ютерні науки \_\_\_\_\_

Курс \_\_\_\_\_ I \_\_\_\_\_ Група \_\_\_\_\_ КА-86 \_\_\_\_\_ Семестр \_\_\_\_\_ 2 \_\_\_\_\_

**ЗАВДАННЯ**

**на курсову роботу студента**

Лобач Гліб Юрійович

(прізвище, ім’я, по батькові)

1. Тема роботи \_\_\_\_\_ База даних \_\_\_\_\_

2. Строк здачі студентом закінченого проекту (роботи) \_\_\_\_\_ 28.05.2019 \_\_\_\_\_

3. Вихідні дані до проекту (роботи) \_\_\_\_\_ мова програмування \_\_\_\_\_  
C++

Технічна література

4. Зміст розрахунково – пояснювальної записки (перелік питань, які підлягають розробці)

Обґрунтування вибору алгоритму

Розробка програми

Керівництво користувачу

Керівництво розробнику

Отримані результати

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

*Скріншоти роботи програми*

*Теоретичні додатки*

6. Дата видачі завдання \_\_\_\_\_ 21.02.2019 \_\_\_\_\_

## ЗМІСТ

Вступ.....	4
Обґрунтування й вибір алгоритму .....	5
Розробка програми .....	9
Керівництво користувачу .....	12
Керівництво розробнику .....	17
Результат роботи програм .....	22
Висновки .....	26
Використана література .....	27
Додатки. Лістинг програми .....	28

## Вступ

Використання баз даних є однією з характерних рис більшості сучасних інформаційних систем. По своїй суті бази даних є тим, навколо чого і будується інформаційна система будь-якого підприємства. Тому теорії створення та практиці використання баз даних приділяється достатня увага протягом періоду функціонування інформаційних систем. Досить тривалий час основним типом були реляційні бази даних, які на сьогодні вже вважаються класичними. Проте розвиток інформаційних систем поставив перед сучасними базами даних завдання, вирішення яких неможливе в межах використання тільки реляційних баз даних. Крім класичних завдань, сучасні бази даних повинні забезпечувати багатомашинну обробку та зберігання великих обсягів інформації, оперативний аналіз даних.

В даному курсовому проекті за предметну область взято базу даних, яка буде зберігати усю необхідну інформацію про працівників ІТ компанії створену на C++ та за основу взято зв'язні списки. Щоб користувачі мали змогу одержати для себе зручний доступ до інформації про робітників необхідно створити базу даних, яка зберігатиме усі необхідні дані та реалізувати функції роботи всередині бази даних.

База даних повинна бути зручною у використанні та задовольняти усім стандартам розробки баз даних. Однією з найважливіших функцій бази даних є швидкість обробки запитів від користувача та швидке видання інформації для нього.

База даних має мати зручний інтерфейс з користувачем. Завдяки цьому користувач може зручно виконувати пошук інформації яка його зацікавить та допоможе обрати/оновити інформацію про шуканого працівника.

## **1. Обґрунтування та вибір алгоритму**

### **1.1. Теоретичні відомості.**

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД.

Головним завданням БД є гарантоване збереження значних обсягів інформації (т.зв. записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин : збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Дані – це інформація, відомості, показники, необхідні для ознайомлення з ким-, чим-небудь, для характеристики когось, чогось або для прийняття певних висновків, рішень. В базах даних важливу роль відіграє інформація - Інформація — абстрактне поняття, що має різні значення залежно від контексту.

Кожна створена база даних має свою предметну область. Предметна область – це необхідний для розробки бази даних об'єкт, який має в собі дані, які будуть зберігатися в базі даних.

Модель даних — абстрактне представлення реального світу, що відображає тільки ті об'єкти, що безпосередньо стосуються програми. Це, як правило, визначає специфічну групу об'єктів, їх атрибутивне значення і відношення між ними.

Відомі два підходи до організації інформаційних масивів: файлова організація та організація у вигляді бази даних. Файлова організація передбачає спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але характеризується рядом недоліків. Характерна риса файлового підходу - вузька спеціалізація як обробних програм, так і файлів даних, що служить причиною великої надлишковості, тому що ті самі елементи даних зберігаються в різних системах. Оскільки керування здійснюється різними особами (групами осіб),

відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для задоволення запитів користувачів, які перекривають дві і більше області. Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів передання даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає необхідність відокремити дані від їхнього опису, визначити таку організацію збереження даних з обліком існуючих зв'язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох застосувань. Вказані причини обумовили появу баз даних. База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об'єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполученні "файл-програма", вона явним чином декларується в базі.

База даних орієнтована на інтегровані запити, а не на одну програму, яку випадку файлового підходу, і використовується для інформаційних потреб багатьох користувачів. В зв'язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації. Перехід від структури БД до потрібної структури в програмі користувача відбувається автоматично за допомогою систем управління базами даних (СУБД).

## **1.2. Постановка задачі курсової роботи**

В курсовому проєкті за мету взято створення бази даних основану на зв'язних списках працівників ІТ компанії, яка має надати можливість зберігання великої кількості резюме працівників, забезпечувати швидкий пошук співробітників. База даних повинна реалізовувати змогу оновлення та додавання, або видалення даних які в ній зберігаються, а також зручне та надійне користування, та доступ усіх необхідних програм до неї.

В базі даних має зберігатися ім'я та прізвище працівника, його рівень професіоналізму (тобто Senior/Medium/ Junior) та його резюме.

Також потрібно реалізувати зручний пошук по прізвищам, по першій літері прізвища, по рівню професіоналізму та відображувати усі підходящі варіанти. Більш за це, треба реалізувати точний пошук за ім'ям та прізвищем. І врешті решт додати змогу модифікувати данні знайдених співробітників.



Тому було поставлено питання як зберегти швидкість, але не втратити в використанні пам'яті. Після дослідження було вирішено використати хеш-функцію, що буди обирати комірку за першою літерою, але деякі виключені літери будуть усі поміщатися в одну комірку. Ці виключені літери або можна задати самому в меню програми, або використати стандартні налаштування, що було виведено зі статистики.

Більш того, для зберігання швидкості було створено особливий вузол зв'язного списку, що зберігає не тільки інформацію про працівника, а й вказівник на наступну літеру. Тобто у комірці з маловживаним літерами на початку прізвища пошук були не на багато повільніший ніж через хеш-функцію.

Несумнівним плюсом цієї програми є гнучкість та адаптація під користувача. Можливість змінювати виключні літери та перекалібрування дає змогу задовільнити користувача максимальною ефективністю. Також, якщо користувачу потрібна максимальна швидкість, та не має різниці скільки пам'яті буде використано, то він може задати порожню множину у налаштуваннях.

Ще додати до цього треба, що лише змінивши хеш-функцію можливо досягати навіть більшою ефективності. Наприклад, за допомогою неї можливо розкидати літери, що зустрічаються занадто часто у декілька комірок, що збільшить швидкість досить сильно, і при цьому програму не доведеться змінювати за виключенням незначних налаштувань у складних випадках.

Взявши все до уваги, остаточно було обрано метод з хеш-таблицею з вирішенням колізій методом ланцюгів з хеш-функцією, що обирає комірку за першою літерою прізвища та виключні літери записує у одну комірку, де послідовність з літер зберігається, використовуючи особливий вузол з вказівником на працівника, прізвище якого починається з наступної за алфавітом літери.



## 2. Розробка програми

Програма написана в середовищі Dev-C++ мовою програмування C++ та працює на базі операційної системи Microsoft Windows.

В процесі розробки програми необхідно було створити базу даних на основі зв'язних списків. Також потрібно реалізувати зручний пошук по прізвищам, по першій літері прізвища, по рівню професіоналізму та відображувати усі підходящі варіанти. Більш за це, треба реалізувати точний пошук за ім'ям та прізвищем. І врешті решт додати змогу модифікувати данні знайдених співробітників. Тобто: змінювати рівень професіоналізму, змінювати резюме та можливість видалити працівника з бази даних, та створити функцію експорту даних з програми у текстовий файл та можливість імпорту даних з файлу у базу даних.

Для реалізації ідеї створення бази даних на основі хеш-таблиці з підтримкою вузлів у списку, що зберігають переміщення через перші літери у прізвищі та хеш-функції, що обирає комірку за першою літерою у прізвищі, а якщо перша літера є однією з виключених літер (літер, що було задано у налаштуваннях), то хеш-функція повертає значення останньої комірки в хеш-таблиці було утворено функцію `list_minor * add_node(list_minor *first, char *name, char *surname, string cv, int type = 1)` та `list_minor * create_node(char name[15], char surname[20], string cv, int type = 1)`, що ввідсортовано вставляють елементи в список порівнюючи прізвища за допомогою `strcmp()`.

`void list_insert_unit(list_hash_table *lht, char *name, char *surname, string cv, int type = 1)` була створена для того, щоб управляти функціями `add_node` та `create_node`. Вона замінює при потребі замінює перший елемент, та перевіряє чи потрібно створювати новий список, чи потрібно лише додати елемент.

Структура бази даних зображено нижче у Рис.2.1



Рис. 2.1.

Головна програма керує меню бази даних, її структуру зображено на блок-схемі нижче (Рис 2.2.):

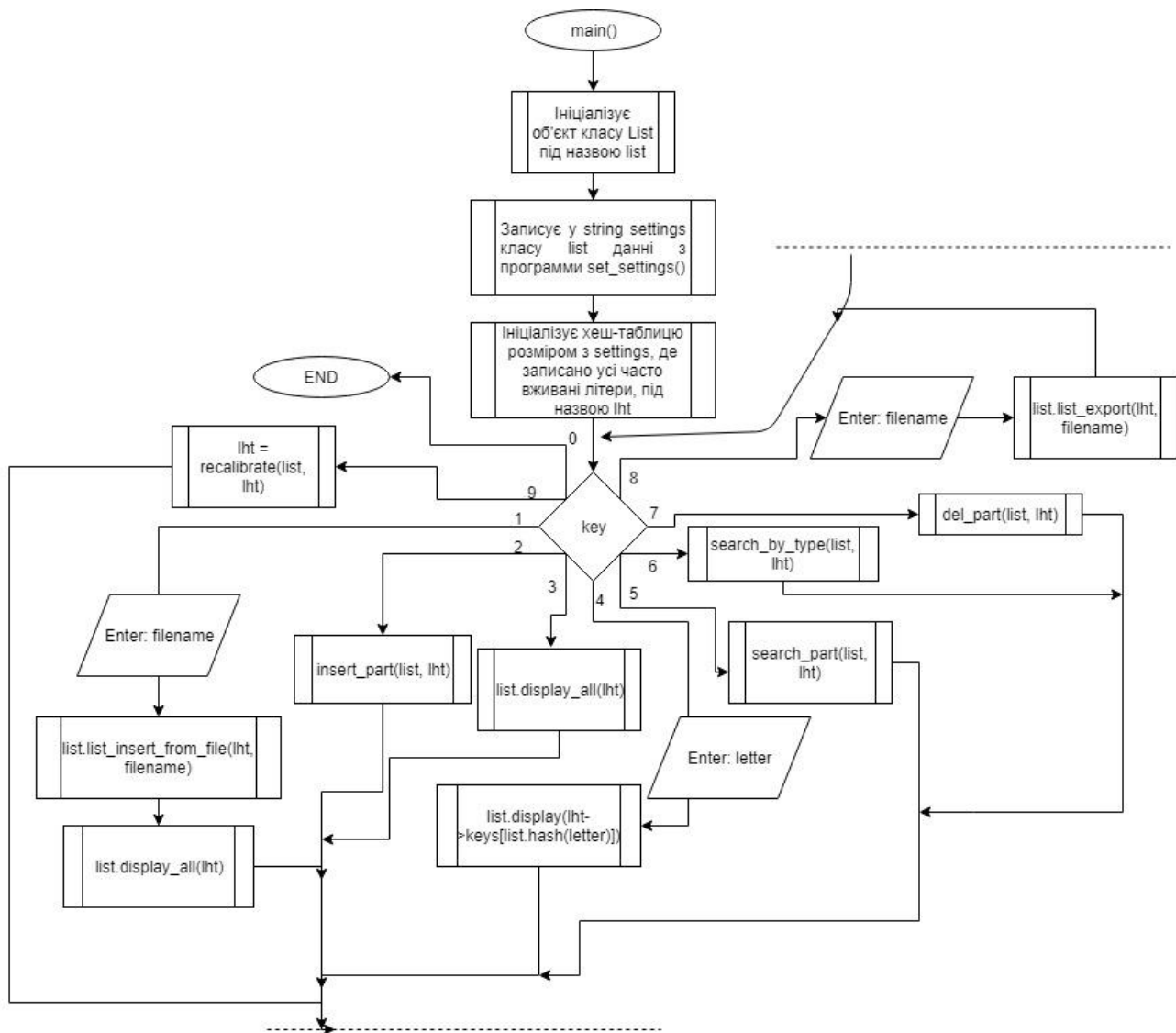


Рис. 2.2.

### 3. Керівництво користувачу

Програма працює у консолі.

При старті програми, вона питає які налаштування встановити (див. Рис. 3.1.)

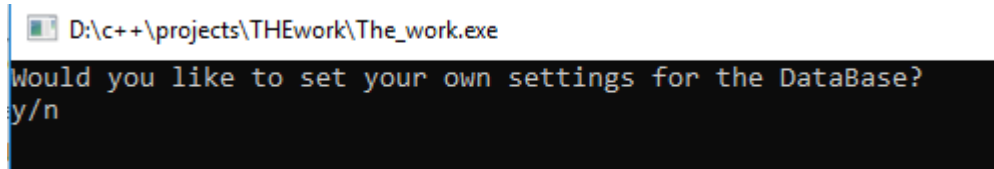


Рис. 3.1.

Користувач може обрати або стандартні налаштування натиснувши «n», або встановити свої натиснувши «y». Якщо користувач хоче встановити свою налаштування, то, після натискання «y», програма запитає в користувача, які букви, з яких починається прізвище, є найрідкіснішими серед працівників (див. Рис. 3.2.).

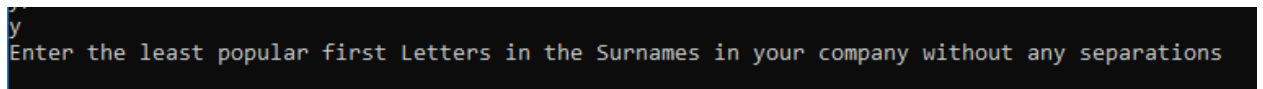


Рис. 3.2.

Після налаштування програми відриється Меню можливих дій. (див. Рис. 3.3.)

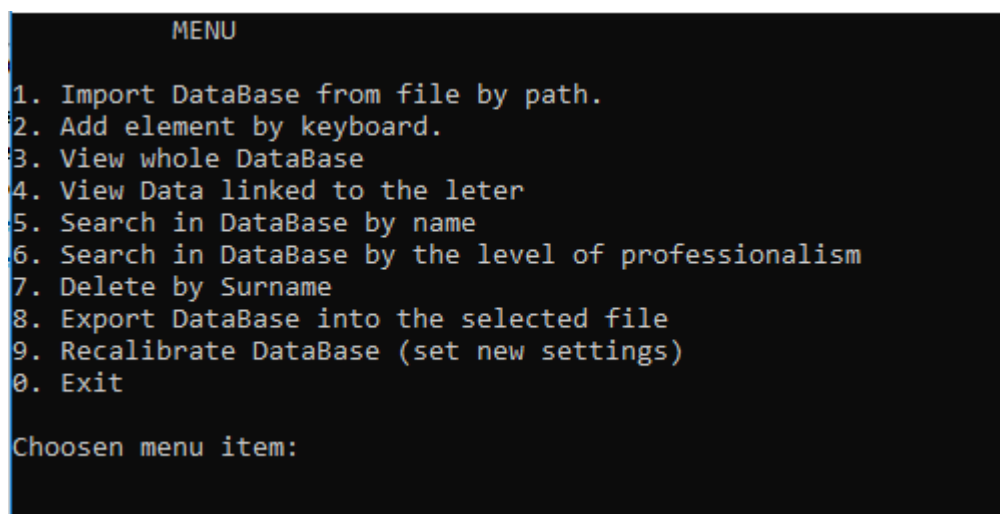


Рис. 3.3. Меню

Користувач обирає дію вводячи цифри від 0 – 9.

Для додавання працівників з файлу потрібно обрати в меню відповідну цифру («1») та виконати інструкції, що будуть виводитися в консоль. Якщо введений файл відкриється, то елементи з нього буде внесено в базу даних. Зауваження: файл з якого будуть експортуватися данні має утримувати дані про робітників у форматі:

```
Surname Name Level_of_professionalism
CV
Surname2 Name2 Level_of_professionalism2
CV2
....
І так далі.
```

Для додавання працівників вручну потрібно обрати в меню відповідну цифру («2») та виконати інструкції, що будуть виводитися в консоль.

Приклад додавання елемента:

```
enter NAME
Igor
enter SURNAME
Laplaski
enter the CV
I am a student of KPI university. I managed to become a Medium developer and decided to become a game developer. I am us
ing C#, C++, Python.
enter Professionalism level
Medium
Press any key to continue . . .
```

Для перегляду бази даних потрібно обрати в меню відповідну цифру («3»).

```
Leter is L
Worker 1:
    Igor Laplaski. Professionalism level is Middle
CV:
    I am a student of KPI university. I managed to become a Medium developer and decided to become a game developer.
    I am using C#, C++, Python.
Worker 2:
    Ira Lebedeva. Professionalism level is Senior
CV:
    Well i guess you're alone after all
Worker 3:
    Gleb Lobach. Professionalism level is Junior
CV:
    I am just a juy after all
Leter is V
Worker 4:
    Boris Vinovnik. Professionalism level is Middle
CV:
    Hi, i am a medium C# developer
Leter is H
Worker 5:
    Gleb Hleb. Professionalism level is Middle
CV:
    Hi, i am a medium C++ developer
Leter is J
Worker 6:
    Misha Junior. Professionalism level is Junior
CV:
    I am a Junior programmer
Press any key to continue . . .
```

Для перегляду усіх працівників, прізвища котріх починається на введену літеру потрібно обрати в меню відповідну цифру («4»).

Приклад:

```
Enter the letter
L
Leter is L
Worker 1:
    Ira Lebedeva. Professionalism level is Senior
CV:
    Well i guess you're alone after all
Worker 2:
    Gleb Lobach. Professionalism level is Junior
CV:
    I am just a juy after all
Press any key to continue . . .
```

Для пошуку усіх працівників за прізвищем або прізвищем та ім'ям («5»).

```
Gleb Lobach. Professionalism level is Junior
CV:
    I am just a juy after all
Maria Lobach. Professionalism level is Senior
CV:
    I am a Senior C# programmer
which one would you like to modify? enter his index number or enter 0 not to modify anything
```

Примітка: Для модифікації елемента потрібно обрати, який з шуканих елементів користувач хоче змінити.

```
MENU
1. Delete this worker from DataBase
2. Change this worker`s level of professionalism.
3. Change this worker`s CV.
4. End modifying
3
enter new CV
Actually, i am a pro developer
```

```
Gleb Lobach. Professionalism level is Junior
CV:
    Actually, i am a pro developer
```

Для пошуку усіх працівників за рівнем професіоналізму («б»).

Приклад пошуку:

```
Enter the level of professionalism you are looking for.
Senior
```

```
Ira Lebedeva. Professionalism level is Senior
CV:
    Well i guess you`re alone after all
Maria Lobach. Professionalism level is Senior
CV:
    I am a Senior C# programmer
which one would you like to modify? enter his index number or enter 0 not to modify anything
```

Для видалення працівників за прізвищем та подальшим вибором працівника«7».

Приклад:

```
Write Surname, which you want to Delete
Lebedeva
The elem Lebedeva Ira was DELETED
No worker with such Surname was found
Press any key to continue . . .
```

Для експорту бази даних у заданий файл «8».

Приклад:

```
Enter the filepath  
3.txt  
Press any key to continue . . .
```

```
File Edit Format View Help  
Boris Vinovnik Middle  
Hi, i am a medium C# developer  
Gleb Hleb Middle  
Hi, i am a medium C++ developer  
Misha Junior Junior  
I am a Junior programmer
```

Для переналаштування хеш-таблиці «9».

Приклад:

```
Would you like to set your own settings for the DataBase?  
y/n  
y  
Enter the least popular first Letters in the Surnames in your company without any separations  
FGHAQ  
Your preference was amplified  
Press any key to continue . . .
```

Для завершення програми «0».



#### 4. Керівництво розробнику

Для успішного функціонування програми файлу `main.cpp` необхідно підключити наступні бібліотеки:

```
#include <iostream>
#include <cstring>
#include <vector>
#include <sstream>
```

Та підключити файл `List.h`

```
#include "List.h"
```

Для якого підключити бібліотеки:

```
#include <iostream>
#include <stdlib.h>
#include <cstring>
#include <fstream>
#include <sstream>
#include <vector>
```

Також, вверху програми за допомогою директиви «define», записана назва файлу в лапках, в який будуть записуватися дані при переналаштуванні бази даних. Стандартно цей файл називається `forbidden.txt`. Зауваження: цю назву не можна використовувати в експорті програми, так як можливі втрати інформації.

Довжина прізвища та довжина ім'я не має перевищувати 20 та 15 символів відповідно.

Функція `main()` починається з налаштування програми, де користувач обирає виключні літери, тобто літери з яких досить рідко починаються прізвища. Та після налаштування відкриває меню програми, в якому користувач обирає одну з перерахованих дій (див. Рис. 2).

```
MENU

1. Import DataBase from file by path.
2. Add element by keyboard.
3. View whole DataBase
4. View Data linked to the leter
5. Search in DataBase by name
6. Search in DataBase by the level of professionalism
7. Delete by Surname
8. Export DataBase into the selected file
9. Recalibrate DataBase (set new settings)
0. Exit

Chooosen menu item:
```

Рис. 2. Меню.

У файлі main.cpp містяться такі функції:

`void insert_part(List list, List::list_hash_table *lht)` —

ця функція відповідає за додавання елементів, які вводяться з клавіатури.

`string set_settings()` —

ця функція відповідає за налаштування хеш-функції та хеш-таблиці відповідно до вводу користувача, або стандарту.

`void modify(List list, List::list_hash_table * lht, List::found_el * answer)` —

ця функція відповідає за зміну даних у знайдених елементах.

`void search_part(List list, List::list_hash_table *lht)` —

ця функція відповідає за пошук працівників за прізвищем або прізвищем та ім'ям.

`void search_by_type(List list, List::list_hash_table *lht)` —

ця функція відповідає за пошук робітників за рівнем професіоналізму.

`void del_part(List list, List::list_hash_table *lht)` —

ця функція відповідає за видалення працівників з бази даних за прізвищем та подальшим вибором.

`List::list_hash_table * recalibrate(List list, List::list_hash_table *lht)` —

ця функція відповідає за переналаштування хеш-функції та хеш-таблиці відповідно до вводу користувача, або стандарту.

У файлі List.h міститься клас List.

У класі List містяться такі структури:

```
typedef struct list{  
    char name[15];      — ім'я  
    char surname[20]; — прізвище  
    string cv;          — резюме  
    int type;           — рівень професіоналізму  
    struct list * next; — вказівник на наступний елемент  
} list; —
```

ця структура відповідає за вузол списку, що утримає дані про працівника.

```
typedef struct list_minor{  
    char name[15];      — ім'я  
    char surname[20];  — прізвище  
    string cv;          — резюме  
    int type;           — рівень професіоналізму  
    list * next;        — вказівник на наступний елемент  
    struct list_minor * next_L; — вказівник на елемент з наступною літерою  
} list_minor; —
```

ця структура відповідає за вузол списку, що утримає дані про працівника та вказівник на елемент з наступною літерою.

```
typedef struct found_el{  
    vector<list*> list_el;      — усі знайдені робітники  
    vector<list_minor*> minor_el; — усі знайдені робітники  
}found_el; —
```

Це структура, яка використовується для повернення знайдених елементів.

```
public: typedef struct list_hash_table{  
    list_minor **keys;      — масив списків  
    int max;                — довжина масиву  
}list_hash_table; —
```

хеш-таблиця.

У класі List.h містяться такі функції:

Зауваження: функції з приміткою \*\* є private.

```
int hash(char* unit) —
```

хеш-функція.

```
**list_minor * add_node(list_minor *first, char *name, char *surname, string  
cv, int type = 1) —
```

функція, що відсортовано додає елементи у зв'язний список.

```
**list_minor * create_node(char name[15], char surname[20], string cv, int  
type = 1) —
```

функція, що створює перший вузол списку.

```
void list_insert_unit(list_hash_table *lht, char *name, char *surname, string cv,  
int type = 1) —
```

функція, що додає елементи у хеш-таблицю.

```
void display(list_minor * first) —
```

функція, що виводить інформацію про працівників у списку.

```
**void export_by_letter(list_minor * first, string filename) —
```

функція, що експортує данні в заданий файл (поміжна функція)

```
void list_export(list_hash_table * lht, string filename) —
```

функція, що експортує данні в заданий файл.

```
void display_one(list * n) —
```

функція, що показує поданий елемент стандартного вузла.

```
void display_one(list_minor * n) —
```

перевантажена функція, що показує поданий елемент вузла з вказівником на літеру.

```
void display_all(list_hash_table * lht) —
```

функція, що виводить інформацію про працівників у всій хеш-таблиці.

```
list_hash_table* list_init_table(int n) —
```

функція, що ініціалізує хеш-таблицю.

```
int list_insert_from_file(list_hash_table *lht, string filename) —
```

функція, що імпортує дані в базу даних з заданого файлу.

`list * list_search_unit(list* first, char *name, char* surname) —`

функція, що шукає робітника за ім'ям та прізвищем.

`found_el * list_search_unit(list_hash_table *lht, char* unit) —`

функція, що шукає робітника за прізвищем.

`found_el * list_search_unit(list_hash_table *lht, char* surname, char*name) —`

перевантажена функція, що шукає робітника за ім'ям та прізвищем.

`**void list_search_type_ad(list_minor * root, int t, found_el * answer) —`

функція, що шукає елемент за рівнем професіоналізму (поміжна функція).

`void list_search_type(list_hash_table * lht, int t, found_el * answer) —`

функція, що шукає елемент за рівнем професіоналізму.

`**int list_del_unit(list_hash_table *lht, char* surname, char* name) —`

функція видаляє робітника з бази даних за ім'ям та прізвищем (для видалення стандартного вузла) (поміжна функція).

`**int list_del_unit(list_hash_table *lht, char* surname, char* name, int version) —`

функція видаляє робітника з бази даних за ім'ям та прізвищем (для видалення вузла з вказівником на наступну літеру)(поміжна функція).

`list_del_unit(list_hash_table * lht, list_minor * minor_el) —`

функція видаляє робітника з бази даних за ім'ям та прізвищем (для видалення стандартного вузла)

`list_del_unit(list_hash_table * lht, list * list_el) —`

функція видаляє робітника з бази даних за ім'ям та прізвищем (для видалення вузла з вказівником на наступну літеру)(поміжна функція).

## 5. Результат роботи програм

Старт програми:

```
D:\c++\projects\THEwork\The_work.exe
Would you like to set your own settings for the DataBase?
y/n
```

Налаштування:

```
y
Enter the least popular first Letters in the Surnames in your company without any separations
```

Меню:

```

MENU

1. Import DataBase from file by path.
2. Add element by keyboard.
3. View whole DataBase
4. View Data linked to the letter
5. Search in DataBase by name
6. Search in DataBase by the level of professionalism
7. Delete by Surname
8. Export DataBase into the selected file
9. Recalibrate DataBase (set new settings)
0. Exit

Chooosen menu item:
```

Результат імпорту даних у базу даних:

База даних порожня:

```
Press any key to continue . . .
```

Файл 1.txt:

```
1 - Notepad
File Edit Format View Help
Gleb Lobach Junior
I am just a juy after all
Ira Lebedeva Senior
Well i guess you`re alone after all
Boris Vinovnik Medium
Hi, i am a medium C# developer
Gleb Hleb Medium
Hi, i am a medium C++ developer
Misha Junior Junior
I am a Junior programmer
```

Проведено імпорт:

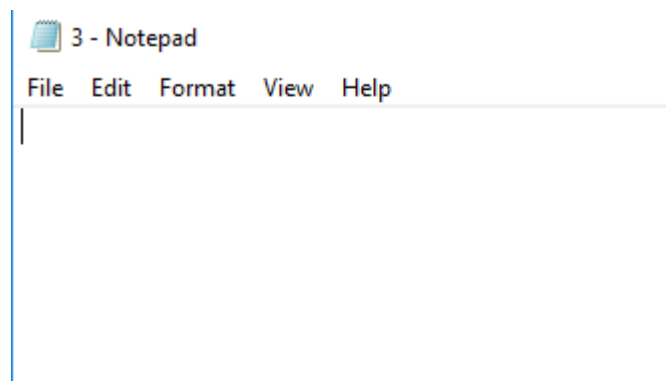
```
Enter the filepath
1.txt
Leter is L
Worker 1:
    Ira Lebedeva. Professionalism level is Senior
CV:
    Well i guess you`re alone after all
Worker 2:
    Gleb Lobach. Professionalism level is Junior
CV:
    I am just a juy after all
Leter is V
Worker 3:
    Boris Vinovnik. Professionalism level is Middle
CV:
    Hi, i am a medium C# developer
Leter is H
Worker 4:
    Gleb Hleb. Professionalism level is Middle
CV:
    Hi, i am a medium C++ developer
Leter is J
Worker 5:
    Misha Junior. Professionalism level is Junior
CV:
    I am a Junior programmer
Press any key to continue . . .
```

Результат роботи експорту у файл:

Елементи в базі даних:

```
Leter is E
Worker 1:
    Test Export. Professionalism level is Senior
CV:
    It is made specially for export test
Leter is L
Worker 2:
    Ira Lebedeva. Professionalism level is Senior
CV:
    Well i guess you're alone after all
Worker 3:
    Gleb Lobach. Professionalism level is Junior
CV:
    I am just a juy after all
Leter is V
Worker 4:
    Boris Vinovnik. Professionalism level is Middle
CV:
    Hi, i am a medium C# developer
Leter is H
Worker 5:
    Gleb Hleb. Professionalism level is Middle
CV:
    Hi, i am a medium C++ developer
Leter is J
Worker 6:
    Misha Junior. Professionalism level is Junior
CV:
    I am a Junior programmer
Press any key to continue . . .
```

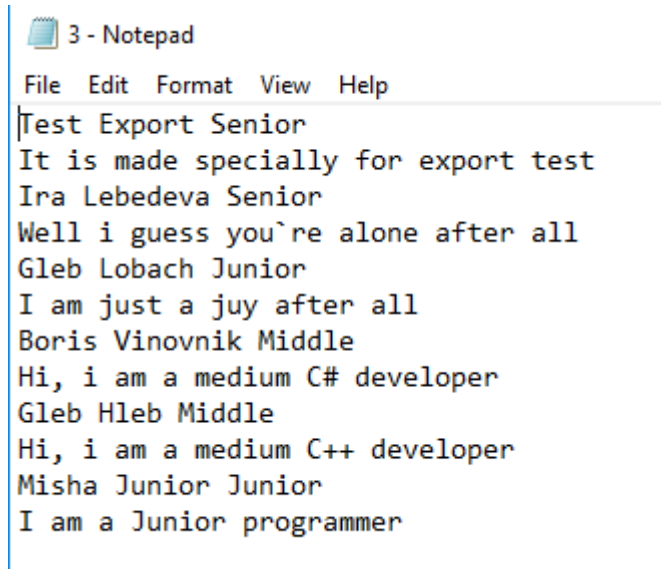
Файл 3.txt до експорту:



```
Enter the filepath
3.txt
Press any key to continue . . .
```



Файл 3.txt після експорту:



```
3 - Notepad
File Edit Format View Help
Test Export Senior
It is made specially for export test
Ira Lebedeva Senior
Well i guess you`re alone after all
Gleb Lobach Junior
I am just a juy after all
Boris Vinovnik Middle
Hi, i am a medium C# developer
Gleb Hleb Middle
Hi, i am a medium C++ developer
Misha Junior Junior
I am a Junior programmer
```

## Висновок

В даному курсовому проєкті за предметну область було взято базу даних, яка буде зберігати усю необхідну інформацію про працівників ІТ компанії створену на C++ та за основу взято зв'язні списки. Щоб користувачі мали змогу одержати для себе зручний доступ до інформації про робітників було створено базу даних, яка зберігатиме усі необхідні дані та реалізовано функції роботи всередині бази даних.

Створена база даних є зручною у використанні та задовольняє усім стандартам розробки баз даних. Функції бази даних є швидкими в обробці запитів від користувача та швидко повертає інформації для нього.

База даних має зручний та зрозумілий інтерфейс с користувачем. Завдяки цьому користувач може зручно виконувати пошук інформації яка його зацікавить та допоможе обрати/оновити інформацію про шуканого працівника.

Завдяки методу з хеш-таблицею з вирішенням колізій методом ланцюгів з хеш-функцією, що обирає комірку за першою літерою прізвища та виключні літери записує у одну комірку, де послідовність з літер зберігається, використовуючи особливий вузол з вказівником на працівника, прізвище якого починається з наступної за алфавітом літери було збережено швидкість пошуку через хеш-функцію (завдяки особливому вузлу) та видалено можливість порожніх комірок завдяки виключним літерам, які задаються при налаштуванні.

## Використана література

- Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебное пособие.-М: Финансы и статистика, 1998.-600 с.
- Проценко В.С., Чаленко П.Й., Ставровський А.Б. Техніка програмування мовою Сі.-К.: Либідь, 1993.-224 с.
- Скиена С. Алгоритмы. Руководство по разработке. – 2-е изд.: Пер. с англ.-СПб.: БХВ-Петербург, 2011. – 720 с.
- Хусаинов Б.С. Структуры и алгоритмы обработки данных. Примеры на языке Си. –М.: Финансы и статистика, 2004.
- Список общерусских фамилий Балановской.
- Шилдт Г. Полный справочник по С. Москва: Вильямс, 4 издание. — 2004.
- Скиена С. Алгоритмы. Руководство по разработке. – 2-е изд.: Пер. с англ.-СПб.: БХВ-Петербург, 2011. – 720 с.
- <https://habr.com/ru/company/oleg-bunin/blog/358984/>

## Додатки

### Додаток 1. Лістинг (код) програми List.h

```
#include <iostream>
#include <stdlib.h>
#include <cstring>
#include <fstream>
#include <sstream>
#include <vector>

using namespace std;
class List{
public: string settings;
private: int w_number = 1;
public: void restart_number(){
    w_number = 1;
}
public: int count_settings(){
    return settings.length();
}
public: void settings_out(){
    cout<<settings<<endl;
}

public : int hash(char* unit){
    // Hashes by the first letter
    char b = unit[0];

    if(strchr(settings.c_str(), b)){
        return  strchr(settings.c_str(), b) - settings.c_str();

    }
    else{
        return settings.length();
    }
}

private: char types[3][7] = {"Junior", "Middle", "Senior"};

public: typedef struct list{
    char name[15];
    char surname[20];
    string cv;
    the worker
    int type;
    professionalism level { 1, 2, 3 } = {Junior, Middle, Senior}

    struct list * next;

} list;
public:
typedef struct list_minor{
    char name[15];
```

//a standart node

// Text about

// the

// Alphabet order

//the node for the surnames with the pointer to the next letter

```

        char surname[20];
        string cv; // Text about
the worker
        int type; // the professionalism level { 1, 2, 3 } = { Junior, Middle, Senior }

        list * next; // Alphabet order
        struct list_minor * next_L; // Shortcut forward through the Alphabet

    } list_minor;

public: typedef struct found_el{
    vector<list*> list_el;
    vector<list_minor*> minor_el;
}found_el;

private : list_minor * add_node(list_minor *first, char *name, char *surname, string cv, int type = 1){
    //Adds an element into the list

    list_minor * n = first;
    list_minor * prev = n;
    while(n->next_L){
        if(surname[0]<n->next_L->surname[0]) break;
        prev = n;
        n = n->next_L;
    }
    if((strcmp(n->surname, surname) == 0) && strcmp(n->name, name)==0){
        cout<<"Worker "<< surname<<" was already registred. Would you like to update his CV?
y/n"<<endl;
        char ch;
        cin>>ch;
        if(ch=='y'){
            n->cv =cv;
            n->type = type;
            cout<<n->surname<<"'s CV was updated"<<endl;
            return first;
        }
        else return first;
    }
    else if(list * elem = list_search_unit(n->next, name, surname)){
        cout<<"Worker "<< surname<<" was already registred. Would you like to update his CV?
y/n";
        char ch;
        cin>>ch;
        if(ch=='y'){
            elem->cv =cv;
            elem->type = type;
            cout<<elem->surname<<"'s CV was updated"<<endl;
            return first;
        }
        else return first;
    }
}

```

```

    if(surname[0]!=n->surname[0] && !n->next_L && strcmp(n->surname, surname)<0){
        prev = n;
        n = n->next_L;
        n = create_node(name, surname, cv, type);
        prev->next_L = n;
        return first;
    }
    if(surname[0]==n->surname[0]){
//cheks if The letter
was already registered
        if(strcmp(n->surname, surname)>0 || (strcmp(n->surname, surname) == 0) && strcmp(n->name,
name)>0 ){

            list_minor *new_el = new list_minor;
            strcpy(new_el->name, name);
            strcpy(new_el->surname,surname);
            new_el->type = type;
            new_el->cv = cv;
            new_el->next_L = n->next_L;
            prev->next_L = new_el;
            if(first == n)
                first = new_el;
            list*temp = new list;
            strcpy(temp->name, n->name);
            strcpy(temp->surname,n->surname);
            temp->next = n->next;
            temp->type = n->type;
            temp->cv = n->cv;
            new_el->next = temp;
            free(n);
            return first;
        }
        list *nextel;
        list *new_el = new list;
        strcpy(new_el->name, name);
        strcpy(new_el->surname,surname);
        new_el->next = NULL;
        new_el->type = type;
        new_el->cv = cv;
        list *n = first->next;
        if((n == NULL) || (strcmp(n->surname, new_el->surname)>0)){
            new_el->next = n;
            first->next = new_el;
            return first;
        }
        if(strcmp(n->surname, new_el->surname)== 0){
            if(strcmp(n->name, new_el->name)>0){
                new_el->next = n->next;
                n->next = new_el;
                return first;
            }
        }
    }
    do{
        if(n->next)nextel = n->next;
        else nextel= NULL;

        if((nextel == NULL) ||(strcmp(nextel->surname, new_el->surname)>0)){
            new_el->next = nextel;

```

```

        n->next = new_el;
        return first;
    }
    if(strcmp(nextel->surname, new_el->surname)== 0){
        if(strcmp(nextel->name, new_el->name)>0){
            new_el->next = nextel->next;
            n->next = new_el;
            return first;
        }
    }
    n = n->next;
}while(n );
cerr<<"Wrong Insert in the DATABASE"<<endl;
exit(-1);
}
else{
    if(strcmp(n->surname, surname)>0 || (strcmp(n->surname, surname) == 0) && strcmp(n->name,
name)>0 ){
        list_minor *new_el = new list_minor;
        strcpy(new_el->name, name);
        strcpy(new_el->surname,surname);
        new_el->type = type;
        new_el->cv = cv;
        new_el->next_L = n;
        new_el->next = NULL;
        if(prev!=n)prev->next_L = new_el;
        if(first = n)
            first = new_el;
        return first;
    }
    list * nextel;
    list_minor *new_el = new list_minor;
    strcpy(new_el->name, name);
    strcpy(new_el->surname,surname);
    new_el->next = NULL;
    new_el->type = type;
    new_el->cv = cv;
    new_el->next_L = n->next_L;
    prev->next_L = new_el;
    n->next_L = new_el;
    do{
        nextel = n->next;

        if((nextel == NULL) ||(strcmp(nextel->surname, new_el->surname)>0)){
            new_el->next = nextel;
            n->next = NULL;
            return first;
        }
    }
    else if(strcmp(nextel->surname, new_el->surname)== 0){
        if(strcmp(nextel->name, new_el->name)>0){
            new_el->next = nextel->next;
            n->next = NULL;
            return first;
        }
    }
}
}

```

```

        list *n = n->next;
    }while(n );
    cerr<<"Wrong Insert in the DATABASE"<<endl;
    exit(-1);
}

}

private: list_minor * create_node(char name[15], char surname[20], string cv, int type = 1){

    list_minor *new_el = new list_minor;
    strcpy(new_el->name, name);

    strcpy(new_el->surname, surname);
    new_el->cv = cv;
    new_el->next = NULL;
    new_el->next_L = NULL;
    new_el->type = type;
    return new_el;
}

public: void display(list_minor * first){           //displays data about the worker in the list
    list_minor *n = first;

    while(n){
        list * elem = n->next;
        cout<<"Leter is "<<n->surname[0]<<endl;
        cout<<"Worker "<<w_number++<<":"<<endl;
        cout<<"\t"<<n->name<<" "<<n->surname<<". Professionalism level is "<<types[n->type-
1]<<endl;
        cout<<"CV:\n\t"<<n->cv<<endl;
        while(elem){
            cout<<"Worker "<<w_number++<<":"<<endl;
            cout<<"\t"<<elem->name<<" "<<elem->surname<<". Professionalism level is
"<<types[elem->type - 1] <<endl;
            cout<<"CV:\n\t"<<elem->cv<<endl;
            elem=elem->next;
        }

        n=n->next_L;
    }

};

private: void export_by_letter(list_minor * first, string filename){
    list_minor *n = first;
    ofstream fw(filename, ios_base::app);
    string line;

    if(!fw.is_open()){
        cerr<<"Erron in file open"<<endl;
        return;
    }
    while(n){
        list * elem = n->next;
        string s(types[n->type- 1]);
        string name(n->name), surname(n->surname);

```



```

        line = name + ' ' + surname + ' ' + s;
        fw<<line<<endl;
        fw<<n->cv<<endl;
        while(elem){
            string s(types[elem->type- 1]);
            string name(elem->name), surname(elem->surname);
            line = name + ' ' + surname + ' ' + s;
            fw<<line<<endl;
            fw<<elem->cv<<endl;
            elem=elem->next;
        }

        n=n->next_L;
    }
    fw.close();
};

public: void display_one(list * n){                                     // displays
data from a standart node
    if(n){
        cout<<n->name<<" "<<n->surname<<". Professionalism level is "<<types[n->type-
1]<<endl;
        cout<<"CV:\n\t"<<n->cv<<endl;
    }
    else cout<<"Not Found"<<endl;
}

public: void display_one(list_minor * n){                             // overloaded
function, which displays data from a node with a letter-pointer
    if(n){
        cout<<n->name<<" "<<n->surname<<". Professionalism level is "<<types[n->type-
1]<<endl;
        cout<<"CV:\n\t"<<n->cv<<endl;
    }
    else cout<<"Not Found"<<endl;
}
/*public: void del(list * first){
    while(first){
        first = delfirst(first);
    }
};*/

// -----Hash Part -----

public: typedef struct list_hash_table{                               //Hash table structure
    list_minor **keys;                                                //Array of lists
    int max;                                                          //Length of
the array
}list_hash_table;

public: void display_all(list_hash_table * lht){                      // displays all the elements from the hash
table
    list_minor ** keys = lht->keys;
    w_number = 1;

```

```

        for(int i = 0; i<lht->max; i++){

            display(keys[i]);

        }

};

public: void list_export(list_hash_table * lht, string filename){
    list_minor ** keys = lht->keys;
    for(int i = 0; i<lht->max; i++){

        export_by_letter(keys[i], filename);

    }

};

public: list_hash_table* list_init_table(int n){
    //Initializes the Hash Table

    list_hash_table *lht = (list_hash_table *)malloc(sizeof(list_hash_table));
    lht->max=n;
    lht->keys = (list_minor**)malloc(sizeof(list_minor)*lht->max);
    for(int i = 0; i<lht->max; i++) lht->keys[i] = NULL;
    return lht;
}

public : void list_insert_unit(list_hash_table *lht, char *name, char *surname, string cv, int type = 1){
//Inserts the DATA into the DATABASE
    if(!(lht->keys[hash(surname)])){
        lht->keys[hash(surname)] = create_node(name, surname, cv, type);

    }
    else {
        lht->keys[hash(surname)] = add_node(lht->keys[hash(surname)], name, surname, cv, type);
    }

}

public: int list_insert_from_file(list_hash_table *lht, string filename){

    ifstream fr;
    fr.open(filename);
    if(!fr.is_open()){
        cerr<<"Erron in file open"<<endl;
        return 0;
    }
    while(!fr.eof()){

        string name, surname,cv, temp, type;
        int t;
        getline(fr, temp);
        std::istringstream ist(temp);
        ist>>name;
        ist>>surname;
        ist>>type;
        if(!name.length()) break;
        if(type[0] == 'J') t = 1;
        else if(type[0] == 'M') t = 2;

```

```

else if (type[0] == 'S')t = 3;
getline(fr , cv);

char namef[15] , surnamef[20];
strcpy(namef, name.c_str());
strcpy(surnamef, surname.c_str());
list_insert_unit(lht, namef, surnamef, cv, t);
}
fr.close();
return 1;

}

list * list_search_unit(list* first, char *name, char* surname){           //search by name and surname
    list * n = first;
    while(n){
        if(!strcmp(n->surname, surname) && !strcmp(n->name, name)){
            return n;
        }
        if(n->next)n=n->next;
        else n = NULL;
    }
    return NULL;
}

public: found_el * list_search_unit(list_hash_table *lht, char* unit){     //search by surname

    found_el * answer = new found_el;

    list_minor* root = lht->keys[hash(unit)];

    while(root && root->surname[0] != unit[0]){
        if(!strcmp(root->surname, unit)){
            answer->minor_el.push_back(root);
        }
        root = root->next_L;
    }
    if(!root) {
        return answer;
    }
    if(!strcmp(root->surname, unit)){
        answer->minor_el.push_back(root);
    }
    if(root->surname[0] != unit[0]){
        return answer;
    }
    list*regular;
    if(root->next)regular = root->next;
    else regular = NULL;
    while(regular){
        if(!strcmp(regular->surname, unit)){
            answer->list_el.push_back(regular);
        }
        regular = regular->next;
    }
}

```

```

        return answer;
};

public: found_el * list_search_unit(list_hash_table *lht, char* surname, char*name){ //search by name and
surname

    found_el * answer = new found_el;

    list_minor* root = lht->keys[hash(surname)];

    while(root && root->surname[0] != surname[0]){
        if(!strcmp(root->surname, surname)&& !strcmp(root->name, name)){
            answer->minor_el.push_back(root);

        }
        root = root->next_L;
    }
    if(!root) {

        return answer;
    }
    if(!strcmp(root->surname, surname)&& !strcmp(root->name, name)){
        answer->minor_el.push_back(root);
    }
    if(root->surname[0] != surname[0]){
        return answer;
    }
    list*regular;
    if(root->next)regular = root->next;
    else regular = NULL;
    while(regular){
        if(!strcmp(regular->surname, surname)&& !strcmp(regular->name, name)){
            answer->list_el.push_back(regular);
        }
        regular = regular->next;
    }

    return answer;
};

private: void list_search_type_ad(list_minor * root, int t, found_el * answer){ //searches by the
professionalism level (helping function)

    while(root){
        if(root->type ==t){
            answer->minor_el.push_back(root);

        }
        list*regular;
        if(root->next)regular = root->next;
        else regular = NULL;
        while(regular){
            if(regular->type == t){
                answer->list_el.push_back(regular);
            }
            regular = regular->next;
        }
    }
}

```

```

        root = root->next_L;
    }
};

public: void list_search_type(list_hash_table *lht, int t, found_el * answer){           //searches by the
professionalism level
    for(int i =0; i<lht->max; i++){
        list_search_type_ad(lht->keys[i], t, answer);
    }
};

public: int list_del_unit(list_hash_table *lht, char* surname, char* name){
    list_minor* root = lht->keys[hash(surname)];
    list_minor * prev= root;
    while(root && root->surname[0] != surname[0]){
        prev= root;
        root = root->next_L;
    }
    if(!root){
        cout<<"Not Found"<<endl;
        return 0;
    }
    if(!strcmp(root->surname, surname) && !strcmp(root->name, name)){
        if(root->next){
            list_minor *new_el = new list_minor;
            strcpy(new_el->name, root->next->name);
            strcpy(new_el->surname,root->next->surname);
            new_el->type = root->next->type;
            new_el->cv = root->next->cv;
            new_el->next_L = root->next_L;
            prev->next_L = new_el;
            if(root->next->next) new_el->next=root->next->next;
            else new_el = NULL;
            if(root == lht->keys[hash(surname)]) lht->keys[hash(surname)] =
new_el;

            free(root->next);
            free(root);
            return 1;
        }
        else{
            prev->next_L = root->next_L;
            cout<<"The elem "<<surname<<' '<<name<<" was
DELETED"<<endl;

            free(root);
            if(root == lht->keys[hash(surname)]) lht->keys[hash(surname)] =
NULL;

            return 1;
        }
    }
    if(root->surname[0] != surname[0]){
        cout<<"Not Found"<<endl;
        return 0;
    }

    list*regular;
    if(root->next)regular = root->next;
    else regular = NULL;

```

```

list * prev_l = regular;

while(regular){

    if(!strcmp(regular->surname, surname) && !strcmp(regular->name, name)){
if(regular == root->next) root->next = root->next->next;
else
    prev_l->next = regular ->next;
cout<<"The elem "<<surname<<' '<<name<<" was DELETED"<<endl;
free(regular);

        return 1;
    }
    prev_l= regular;
    regular = regular->next;
}
cout<<"Not Found"<<endl;
return 0;
};
public: int list_del_unit(list_hash_table *lht, char* surname, char* name, int version){
    list_minor * n = lht->keys[hash(surname)];
    list_minor * prev = n;
    while(n->surname[0] != surname[0]){
        prev=n;
        n= n->next_L;
    }
    if(prev==n){
        if(n->next){
            list_minor * next= create_node(n->next->name, n->next->surname, n->next-
>cv, n->next->type);

            next->next_L = n->next_L;
            if(n->next->next)next->next = n->next->next;
            else next->next = NULL;
            lht->keys[hash(surname)] = next;
            cout<<"The elem "<<surname<<' '<<name<<" was DELETED"<<endl;
            free(n);
            return 1;
        }
        else{
            lht->keys[hash(surname)] = NULL;
            cout<<"The elem "<<surname<<' '<<name<<" was DELETED"<<endl;
            free(n);
            return 1;
        }
    }
    else{
        if(n->next){
            list_minor * next= create_node(n->next->name, n->next->surname, n->next-
>cv, n->next->type);

            next->next_L = n->next_L;
            if(n->next->next)next->next = n->next->next;
            else next->next = NULL;
            prev->next_L = next;
            lht->keys[hash(surname)] = next;
            cout<<"The elem "<<surname<<' '<<name<<" was DELETED"<<endl;
            free(n);
            return 1;
        }
    }
}

```

```

        else{
            prev->next_L = n->next_L;
            cout<<"The elem "<<surname<<' '<<name<<" was DELETED"<<endl;
            free(n);
            return 1;
        }

    }
    cerr<<"Not Found"<<endl;
    return -1;
};

public: list_del_unit(list_hash_table * lht, list_minor * minor_el){
    list_del_unit(lht, minor_el->surname, minor_el->name, 1);
};

public: list_del_unit(list_hash_table * lht, list * list_el){
    list_del_unit(lht, list_el->surname, list_el->name);
};

};

```

## Додаток 2. Лістинг (код) програми main.cpp

```

#include <iostream>
#include <cstring>
#include <vector>
#include <sstream>
#include "List.h"

#define forbidden_name "forbidden.txt" //filepath for recalibration? it is important not to use it in exporting or
the Data may be lost

void insert_part(List list, List::list_hash_table *lht){
    char a[15],b[20], type[7];
    string cv;
    int t;
    cout<<"enter NAME"<<endl;
    cin>>a;
    cout<<"enter SURNAME"<<endl;
    cin>>b;
    cout<<"enter the CV"<<endl;
    cin>>ws;
    getline(cin, cv);

    cout<<"enter Professionalism level"<<endl;
    cin>>type;

    if(type[0] == 'J') t = 1;
    else if(type[0] == 'M') t = 2;
    else if (type[0] == 'S')t = 3;
    list.list_insert_unit(lht, a, b, cv, t);
}

```

```

string set_settings(){
    char ch;
    string settings, temp;
    cout<<"Would you like to set your own settings for the DataBase? \ny/n"<<endl;
    cin>>ch;
    if(ch == 'y'){
        cout<<"Enter the least popular first Letters in the Surnames in your company without any
separations"<<endl;
        cin>>settings;
        cout<<"Your preference was amplified"<<endl;
    }
    else {
        settings = "HJQUWXYZ";
        cout<<"The Standart settings were amplified"<<endl;
    }
    for(int i = (int)('A'); i<(int)('Z');i++) {
        if(!(strchr(settings.c_str(), (char)(i))))temp+=(char)(i);
    }
    return temp;
}

```

```

void modify(List list, List::list_hash_table * lht, List::found_el * answer){
    system("cls");
    int choice, what;
    if(answer->list_el.size() || answer->minor_el.size()){
        for(int i = 0; i<answer->minor_el.size(); i++){
            list.display_one(answer->minor_el[i]);
        }
        for(int i = 0; i<answer->list_el.size(); i++){
            list.display_one(answer->list_el[i]);
        }

        cout<<"which one would you like to modify? enter his index number or enter 0 not to
modify anything"<<endl;

        cin>>choice;

        }
        int exit = 1;

        if(choice == 0) return;

        choice--;
        while(exit){
            system("cls");
            cout<<"      MENU \n"<<endl;
            cout<<"1. Delete this worker from DataBase"<<endl;
            cout<<"2. Change this worker`s level of professionalism."<<endl;
            cout<<"3. Change this worker`s CV."<<endl;
            cout<<"4. End modifying"<<endl;
            cin>>what;
            switch(what){
                case 1:{

                    if(answer->minor_el.size() && choice == 0){

```



```

        if(answer->minor_el[0])list.list_del_unit(lht, answer->minor_el[0]);
        else return;
    }
    else {
        if(answer->list_el[choice])list.list_del_unit(lht, answer->list_el[choice]);
        else return;
    }
    return;
    break;
}
case 2:{
    string type;
    int t;
    if(answer->minor_el.size() && choice == 0){
        if(answer->minor_el[0]){
            cout<<"enter Professionalism level"<<endl;
            cin>>type;
            if(type[0] == 'J') t = 1;
            else if(type[0] == 'M') t = 2;
            else if (type[0] == 'S')t = 3;
            answer->minor_el[0]->type = t;
        }
        else{
            cout<<"Nothing to modify"<<endl;
            return;
        }
    }
    else {
        if(answer->list_el[choice]){
            cout<<"enter Professionalism level"<<endl;
            cin>>type;
            if(type[0] == 'J') t = 1;
            else if(type[0] == 'M') t = 2;
            else if (type[0] == 'S')t = 3;
            answer->list_el[choice]->type = t;
        }
        else{
            cout<<"Nothing to modify"<<endl;
            return;
        }
    }
    break;
}
case 3:{
    string type;
    if(answer->minor_el.size() && choice == 0){
        if(answer->minor_el[0]){
            cout<<"enter new CV"<<endl;
            cin>>ws;
            getline(cin, type);
            answer->minor_el[0]->cv= type;
        }
        else{
            cout<<"Nothing to modify"<<endl;
            return;
        }
    }
}

```

```

        }
        else {
            if(answer->list_el[choice]){
                cout<<"enter new CV"<<endl;
                cin>>ws;
                getline(cin, type);
                answer->list_el[choice]->cv = type;
            }
            else{
                cout<<"Nothing to modify"<<endl;
                return;
            }
        }
        break;
    }

    case 4:{
        exit = 0;
        break;
    }

}

}

}

void search_part(List list, List::list_hash_table *lht){
//***** a function to search by name + surname + add a
feature to modify founded elemets
    string temp, name, surname;
    List::found_el * answer;
    cout<<"Enter the worker`s surname and name if it is known, which you want to find.\nIn Template:
Surname Name"<<endl;
    cin>>ws;
    getline(cin, temp);
    std::istringstream ist(temp);
    ist>>surname;
    ist>>name;

    if(surname.length() && name.length()){
        char surnamef[20], namef[15];
        strcpy(namef, name.c_str());
        strcpy(surnamef, surname.c_str());
        answer = list.list_search_unit(lht, surnamef,namef);
    }
    else if(surname.length()){
        char surnamef[20];
        strcpy(surnamef, surname.c_str());
        answer = list.list_search_unit(lht, surnamef);
    }
    else {
        cout<<"Blanc Input"<<endl;
        return;
    }
}

```

```

        if(answer->list_el.size() || answer->minor_el.size()){
            for(int i = 0; i<answer->minor_el.size(); i++){
                list.display_one(answer->minor_el[i]);
            }
            for(int i = 0; i<answer->list_el.size(); i++){
                list.display_one(answer->list_el[i]);
            }
            modify(list, lht, answer);
        }

        else cout<<"No worker with such Surname was found"<<endl;

    }

void search_by_type(List list, List::list_hash_table *lht){
//***** a function to search by name + surname + add a
feature to modify founded elemets
    string type;
    List::found_el * answer = new List::found_el;
    cout<<"Enter the level of professionalism you are looking for."<<endl;
    cin>>type;
    if(type.length()){
        int t;
        if(type[0] == 'J') t = 1;
        else if(type[0] == 'M') t = 2;
        else if (type[0] == 'S')t = 3;
        list.list_search_type(lht, t, answer);
    }
    else {
        cout<<"Blanc Input"<<endl;
        return;
    }

    if(answer->list_el.size() || answer->minor_el.size()){
        for(int i = 0; i<answer->minor_el.size(); i++){
            list.display_one(answer->minor_el[i]);
        }
        for(int i = 0; i<answer->list_el.size(); i++){
            list.display_one(answer->list_el[i]);
        }
        modify(list, lht, answer);
    }
    else cout<<"No worker with such level of professionalism"<<endl;

}

void del_part(List list, List::list_hash_table *lht){
    char b[20];
    int choice;
    cout<<"Write Surname, which you want to Delete"<<endl;
    cin>>b;
    List::found_el * answer = list.list_search_unit(lht, b);

    if(answer->list_el.size() || answer->minor_el.size()){

```

```

        if((answer->list_el.size() + answer->minor_el.size()) == 1){
            choice = 1;
        }
        else{
            for(int i = 0; i<answer->minor_el.size(); i++){
                list.display_one(answer->minor_el[i]);
            }
            for(int i = 0; i<answer->list_el.size(); i++){
                list.display_one(answer->list_el[i]);
            }
            cout<<"which one would you like to delete? enter his index number or enter 0 not to delete
anything" <<endl;

            cin>>choice;
            if(choice == 0) return;
        }
        choice--;

        if(answer->minor_el.size() && choice == 0){
            list.list_del_unit(lht, answer->minor_el[0]); // function for minor_el (a must
to create) *****
        }
        else list.list_del_unit(lht, answer->list_el[choice]);

    }
    else {
        cout<<"No worker with such Surname was found" <<endl;
        return;

    }
    answer->list_el.clear();
    answer->minor_el.clear();
    answer = list.list_search_unit(lht, b);
    if(answer->list_el.size() || answer->minor_el.size()){
        for(int i = 0; i<answer->minor_el.size(); i++){
            list.display_one(answer->minor_el[i]);
        }
        for(int i = 0; i<answer->list_el.size(); i++){
            list.display_one(answer->list_el[i]);
        }
    }
    else cout<<"No worker with such Surname was found" <<endl;

}

List::list_hash_table * recalibrate(List list, List::list_hash_table *lht){
    ofstream fw(forbidden_name);
    fw.close();
    list.list_export(lht, forbidden_name);
    list.settings = set_settings();
    List::list_hash_table *lht2 = list.list_init_table(list.count_settings() + 1);
    list.list_insert_from_file(lht2, forbidden_name);
    free(lht);
    return lht2;

}

int main()

```

```

{
    List list;
    list.settings = set_settings(); //Required settings
    List::list_hash_table *lht = list.list_init_table(list.count_settings() + 1);

while (1)
{
    system("pause");
    system("cls");
    cout<<"    MENU \n"<<endl;
    cout<<"1. Import DataBase from file by path."<<endl;
    cout<<"2. Add element by keyboard."<<endl;
    cout<<"3. View whole DataBase"<<endl;
    cout<<"4. View Data linked to the letter"<<endl;
    cout<<"5. Search in DataBase by name"<<endl;
    cout<<"6. Search in DataBase by the level of professionalism"<<endl;
    cout<<"7. Delete by Surname"<<endl;
    cout<<"8. Export DataBase into the selected file"<<endl;
    cout<<"9. Recalibrate DataBase (set new settings)"<<endl;
    cout<<"0. Exit"<<endl;
    cout<<endl;
    int key;
    cout<<"Chosen menu item:";
    cin>>key;
    system("cls");
    switch(key)
    {
        case 1: {string filename;cout<<"Enter the filepath"<<endl;cin>>filename;list.list_insert_from_file(lht,
filename);list.display_all(lht);break;}
        case 2: {insert_part(list, lht);break;}
        case 3: {list.display_all(lht);break;}
        case 4: {char letter[1];cout<<"Enter the letter"<<endl;cin>>letter;list.restart_number();list.display(lht-
>keys[list.hash(letter)]);break;}
        case 5: {search_part(list, lht);break;}
        case 6: {search_by_type(list, lht);break;}
        case 7: {del_part(list, lht);break;}
        case 8: {string filename;cout<<"Enter the filepath"<<endl;cin>>filename;list.list_export(lht,
filename);break;}
        case 9: {lht = recalibrate(list, lht);break;}
        case 0: {exit (0); break;}
        default: {cout<<"Try another time...";system("pause");break;}
    }
}

return 0;
}

```