Державний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Алгоритмізація та програмування»

Тема «Програмування динамічної структури даних – однозв’язний список»

Студента 1 курсу АІ-215 групи
Спеціальності 122 – «Комп’ютерні науки»
_____Березанський Т.С._____
Керівник Іванов О. В.

Національна шкала _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

м. Одеса – 2022 рік

Державний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

студенту Березанському Тимофію Сергійовичу група АІ-215

1. Тема роботи

«Програмування динамічної структури даних – однозв’язний список»

2. Термін здачі студентом закінченої роботи 15.06.2022

3. Початкові дані до проекту (роботи)

Варіант 2

Реалізувати динамічну структуру даних (однозв’язний список), що містить наступну інформацію: номер автобуса; прізвище та ініціали водія; номер маршруту; наявність автобуса в парку або на виїзді; кількість порушень водієм правил експлуатації автобуса.

Програма повинна забезпечувати:

- додавання елемента;
- видалення елемента;
- можливість коригування даних;
- виведення всіх даних;
- формування списку автобусів, які виїхали з парку (що залишилися в парку);
- виведення даних про водія з найбільшим числом порушень;
- сортування даних за номерами автобусів;
- виведення всіх прізвищ водіїв, що починаються на задану букву.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)

Вступ. Теоретичні відомості про однозв’язний список. Програмна реалізація автопарк. Інструкція користувача. Висновки. Перелік використаних джерел. Додатки

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

Блок-схема алгоритму – 1 аркуш формату А1.

Завдання видано 28.03.22 _____
(підпис викладача)

Завдання прийнято до виконання 28.03.22 _____
(підпис студента)

АНОТАЦІЯ

Розглянуто теоретичний матеріал та написана динамічна структура (однозв’язний список). Розроблено код та 10 головних функцій для нього, код написаний на С без використання С++. Запропоновано оригінальні методи вирішення проблем. Розроблено також інструкцію для коду, щоб було легше працювати з ним .

ABSTRACT

Theoretical material is considered and a dynamic structure is written (single-linked list). Developed code and 10 main functions for it, code written in C without the use of C ++. Original problem-solving methods have been proposed. Code instructions have also been developed to make it easier to work with.

ЗМІСТ

ВСТУП

Мета курсової роботи - це показати навички, які здобув студент за рік навчання на 122 спеціальності. Кожному студент було видано особливе завдання.

Студент повинен розробити власний код та налагодити його роботу, зробити правильно записку до курсової роботи та захистити свою роботу на 13-15 тижнях.

Студенту запропоновано розробити динамічну структуру. Динамічна структура відрізняється від тієї структури, яку ми робили в першому семестрі (статичної). Вона може змінювати свої дані під час роботи. Мені здається що зручніше використовувати саме динамічну структуру, а не статичну. Також хочу зауважити, що існують декілька різновидів динамічної структури :

1) списки (однозв’язний та двозв’язний);
2) стек;
3) черга;
4) бінарні дерева .

Види динамічних структур розрізняються способами зв'язку окремих елементів і допустимими операціями. Динамічна структура на відміну від масиву або записи може займати несуміжні ділянки оперативної пам'яті. Динамічні структури широко застосовують і для більш ефективної роботи з даними, розмір яких відомий, особливо для вирішення завдань сортування (яке було запропоновано для вирішення).

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ОДНОЗВ’ЯЗНИЙ СПИСОК

1.1 Однозв’язний список

Лінійний список – це динамічна структура даних, кожний елемент якої за допомогою вказівника зв’язується з наступним елементом. З визначення випливає, що кожен елемент списку містить поле даних і поле посилання на наступний елемент (next). Поле посилання останнього елемента повинно містити порожній покажчик (NULL). Так як посилання лише одне (тільки на наступний елемент), то такий список є однозв’язним. Коли говорять про лінійний список, то, як правило, мають на увазі саме однозв’язний список.

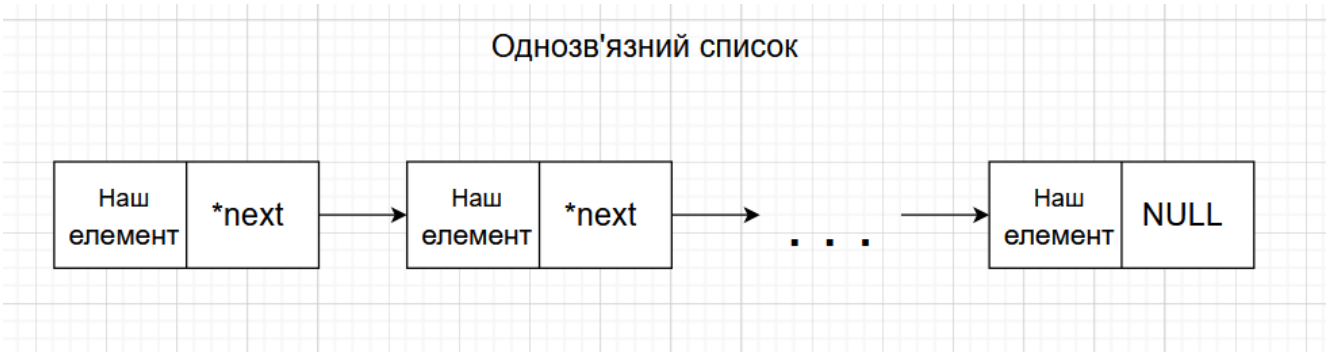

Рисунок 1.1 – Однозв’язний список

Розглянемо більш детально алгоритм додавання елементу в структуру (рис. 1.2).

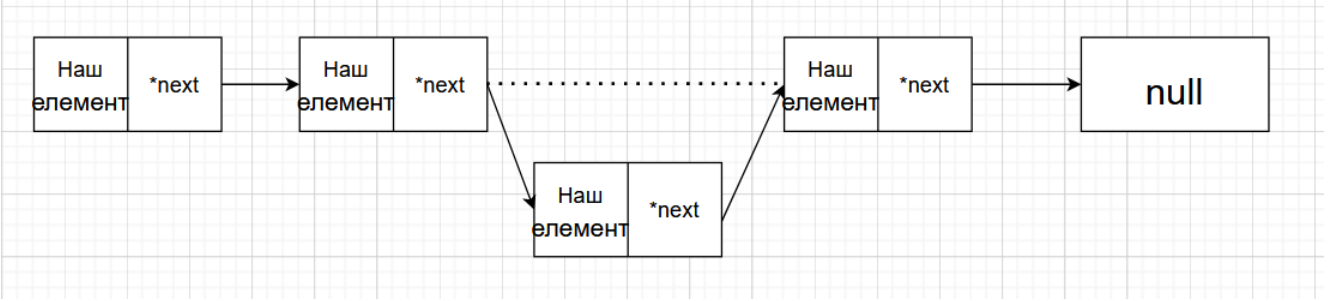

Рисунок 1.2 - Додавання елементу

Як бачимо, щоб додати новий елемент потрібно попрацювати з покажчиками .

Основна перевага зв'язаного списку перед звичайним масивом полягає в тому, що елементи списку можна легко вставити або видалити без перерозподілу або реорганізації всієї структури, оскільки елементи даних не потрібно зберігати безперервно в пам'яті або на диску, під час реструктуризації масиву на час виконання є набагато дорожчою операцією. Зв’язані списки дозволяють вставляти та видаляти вузли в будь-якій точці списку, і дозволяють робити це за допомогою постійної кількості операцій, зберігаючи посилання перед посиланням, яке додається або видаляється в пам’яті під час обходу списку. З іншого боку, оскільки прості зв’язані списки самі по собі не надають випадкового доступу до даних або будь-якої форми ефективного індексування, багато основних операцій, таких як отримання останнього вузла списку, пошук вузла, який містить задану дату, або визначення місця, куди слід вставити новий вузол, може вимагати повторення більшості або всіх елементів списку. Переваги та недоліки використання зв’язаних списків наведено нижче.

Однозв'язний лінійний список є рекурсивною структурою даних, оскільки містить покажчик на менший об’єкт того ж типу. З цієї причини багато операцій з однозв’язаними лінійними списками (наприклад, об’єднання двох списків або перерахування елементів у зворотному порядку) часто мають дуже прості рекурсивні алгоритми, набагато простіші, ніж будь-яке рішення з використанням ітераційних команд. Хоча ці рекурсивні рішення можна адаптувати для двозв’язаних і циркулярно зв’язаних списків, процедури зазвичай потребують додаткових аргументів і більш складних базових випадків.

Під час побудови зв’язаного списку постає вибір, чи зберігати дані списку безпосередньо у вузлах зв’язаного списку, які називаються внутрішньою пам’яттю, чи просто зберігати посилання на дані, які називаються зовнішніми сховищами. Внутрішня пам’ять має перевагу, що робить доступ до даних більш ефективним, вимагає менше пам’яті в цілому, має кращу локальність посилання та спрощує керування пам’яттю для списку (його дані виділяються та звільняються одночасно з вузлами списку).

Зовнішнє сховище, з іншого боку, має перевагу в тому, що воно більш загальне, оскільки для зв’язаного списку можна використовувати однакову структуру даних і машинний код незалежно від розміру даних. Це також полегшує розміщення одних і тих самих даних у кількох пов’язаних списках. Хоча за допомогою внутрішньої пам’яті одні й ті самі дані можна помістити в кілька списків, включивши кілька наступних посилань у структуру даних вузла, тоді буде необхідно створити окремі підпрограми для додавання або видалення комірок на основі кожного поля. Можна створити додаткові зв’язані списки елементів, які використовують внутрішнє сховище, використовуючи зовнішню пам’ять, і якщо клітинки додаткових зв’язаних списків зберігають посилання на вузли зв’язаного списку, що містить дані.

Загалом, якщо набір структур даних потрібно включити до пов’язаних списків, найкращим підходом є зовнішній сховище. Якщо набір структур даних потрібно включити лише в один зв’язаний список, то внутрішня пам’ять трохи краще, якщо не доступний загальний пакет зв’язаного списку, що використовує зовнішнє сховище.Також можна створити загальні підпрограми, які використовують мінімальну структуру для виконання операцій типу зв’язаного списку, але окремі підпрограми потім можуть обробляти конкретні дані.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ ОДНОЗВ'ЯЗНОГО СПИСКУ

2.1 Як працює програма

Програма працює з інформацією автопарка. Вона вміє додавати, видаляти елементи; показувати всю інформацію; показувати: хто присутній в автопарку, а хто ні; показує елемент з найбільшою кількістю порушень; сортує елементи; змінює інформацію; шукає людину або людей за першою літерою, записує всю інформацію в файл та працює з інформацією, яку виводить з файлу.

В структурі присутні: номер автобусу, ім'я, прізвище, по батькові, номер маршруту, присутність в автопарку та кількість зауважень.

```
typedef struct BUS{
    int num;                //number bus
    char name[20];          //fullname
    int trip;               //num of trip
    char av[2];             //availability
    int attention;          //count of attention

    BUS *next;              //pointer next
}BUS;
```

Я використовував в своєму коді прототипи void 'назва функції' (). На мою думку, з ними працювати швидше та зручніше. Тільки одна функція була з int ,вона повертає кількість елементів.

2.2 Прототипи та їх мета

1. int count_of_elements()- ця функція рахує кількість елементів та повертає число. Вона використовувалася в функціях видалення елементу та для заміни інформації. Треба ввести позицию елемента і продовжити дії далі;
2. void new_information(BUS *ptr) - ця функція для переходу між елементами. Використовується в функції add_information();
3. void delete_elements()- функція, щоб видалити елемент за позицією. Як я писав раніше, для цієї функції потрібна допоміжна функція - count_of_elements(), щоб знайти елемент за позицією;
4.void add_information()- функція додавання елементу (використовує new_information(BUS *ptr)). Хочу зауважити, що потрібно вводити коректні дані(не треба вводити менше за 0). Якщо дані менш за 0 - потрібно буде ввести знов поле;
5. void all_information() - функція, яка виводить присутню інформацію. Виводить у повному обсязі;
6. void availability_of_elements() - функція, яка виводить елементи, які підходять за значенням. Спочатку користувач вводить ‘+’ або ‘-’ і далі йому виводить інформацію за полем **av**;
7. void count_of_attention() - виводить елемент з найбільшою кількістю порушння. Елемент шукається за полем **attention**, зроблено у циклі **for**;
8. void sorting()- функція, щоб відсортувати усі присутні елементи за полем **num**. Використовує допоміжну функцію swap_char(), щоб сортувати дані char;
9. void swap_char(char *p,char *q)- допоміжна функція sorting(). Приймає значення char та за допомогою функції strcpy() копіює рядок 1 в рядок 2;
10. void write_new_infromation() - функція, щоб змінити присутню інформацію. Щоб почати заміну потрібно ввести позицію;
11. void first_letter() - на початку вводимо літеру ,за якою потрібно шукати. Далі за допомогою strncmp() шукаємо слова із заданою буквою за полем **name** та виводимо дані;

12. void write_to_file()- записує всю присутню інформацію у файл. Інформація зберігається у файлі з розширенням .txt;

13. void read_from_file()- функція, яка виводить інформацію з файлу, яка заздалегідь була занесена в нього. З цією інформацією користувач може працювати так само, як і з тією, що була б введена з клавіатури за допомогою add_information();

Також можно переглянути прототипи, як вони виглядають в самому коді (рис. 2.2).

```
17
18 int count_of_elements();              //кількість елементів
19 void new_information(BUS *ptr);       //перехід
20 void delete_elements();               //видалити елемент за позицією
21 void add_information();                //додати елемент
22 void all_information();                //уся присутня інформація
23 void availability_of_elements();      //наявність. + -> так , - ->ні
24 void count_of_attention();            //шукаєм елемент з найбільшою кількістю порушень
25 void sorting();                        //сортування
26 void swap_char(char *p,char *q);       //допоміжна функція для сорутвання
27 void write_new_infromation();          //змінити інформацію
28 void first_letter();                   //перша літера
29 void write_to_file();                  //запис у файл
30 void read_from_file();                 //інформація вже є в файлі
```

Рисунок 2.2 - Прототипи функцій

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

3.1 Інструкція

Програма запускається і одразу виводить меню (рис. 3.1). Можно вибрати 10 функцій або завершити, натиснувши 0 на клавіатурі . Якщо користувач натисне цифру ,яка не відповідає меню і не 0 йому буде запропоновано зробити це знову . Процес буде продовжуватися поки користувач не натисне правильну цифру .

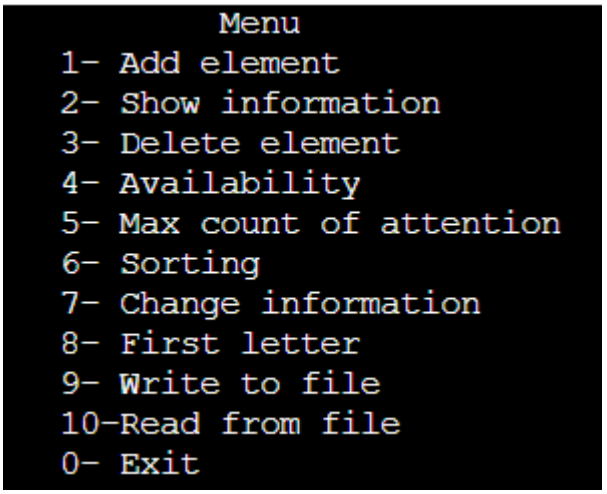

Рисунок 3.1 - Меню

3.2 Введення елементу

Коли користувач натисне 1 буде додано новий елемент . Цю функцію можно виконувати безліч разів. Потрібно вводити коректні дані, якщо буде введено число менше за 0 або 0 - буде запропоновано знову ввести поле даних (не стосується attention , attention >=0) (рис. 3.2).

Інформація додається в наступному порядку:

1)номер автобусу;

2)прізвище та ініціали;

3)номер поїздки;

4)наявність в автопарку;

5)кількість зауважень.

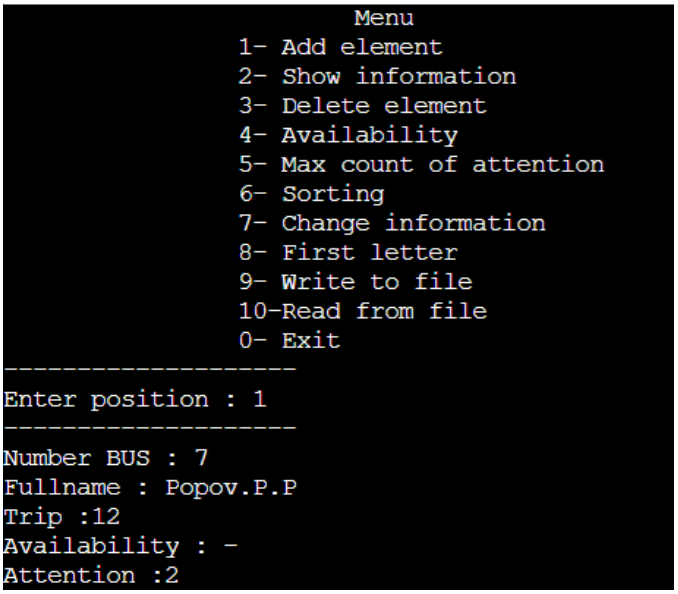

Рисунок 3.2 - Додавання елементу

3.3 Виведення елементу/ів.

Коли користувач натисне 2 йому виведе всю присутню інформацію про існуючі елементи (рис. 3.3).

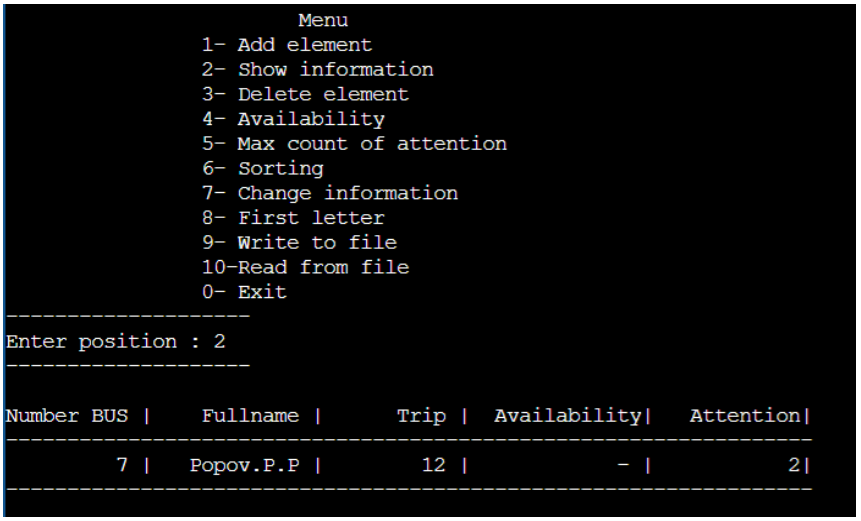

Рисунок 3.3 - Вивід інформації

3.4 Видалення елементу зі списку

Коли користувач натисне 3 йому буде запропоновано видалити елемент. Щоб це зробити потрібно спочатку вибрати позицію елемента для видалення (рис. 3.4).

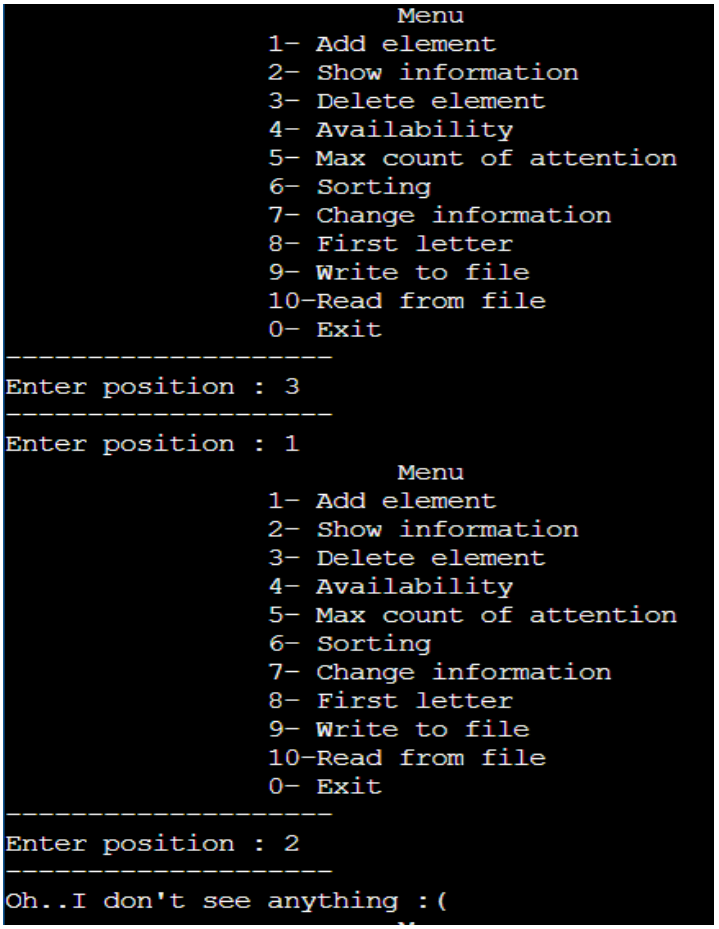

Рисунок 3.4 - Видалення елементу

3.5 Присутність в автопарку

Коли користувач натисне 4 йому буде потрібно ввести або ‘+’, або ‘-’ в залежності від того, що він хоче побачити . Зрозуміло, що ‘+’ - це присутність в автопарку, а ‘-’ - це відсутність (рис. 3.5).

Буде виведена уся інформація ,що буде мати задане значення характеристики

.

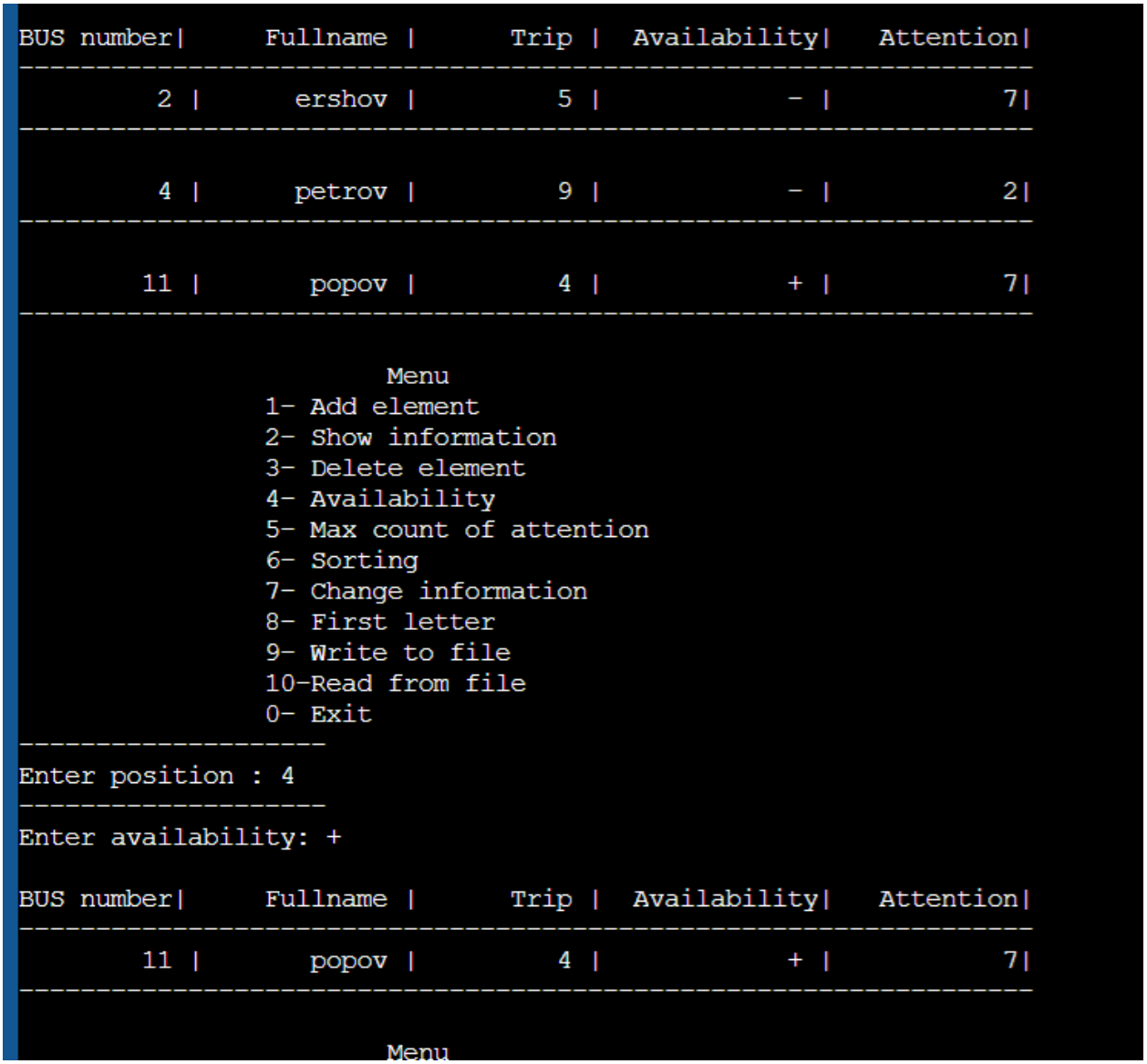

Рисунок 3.5 - Присутність в автопарку

3.6 Максимальна кількість зауважень

Коли користувач натисне 5 йому виведе елемент, який шукає за полем **attention** максимальну кількість порушення, а також виводить іншу інформацію про цей елемент(рис. 3.6) .

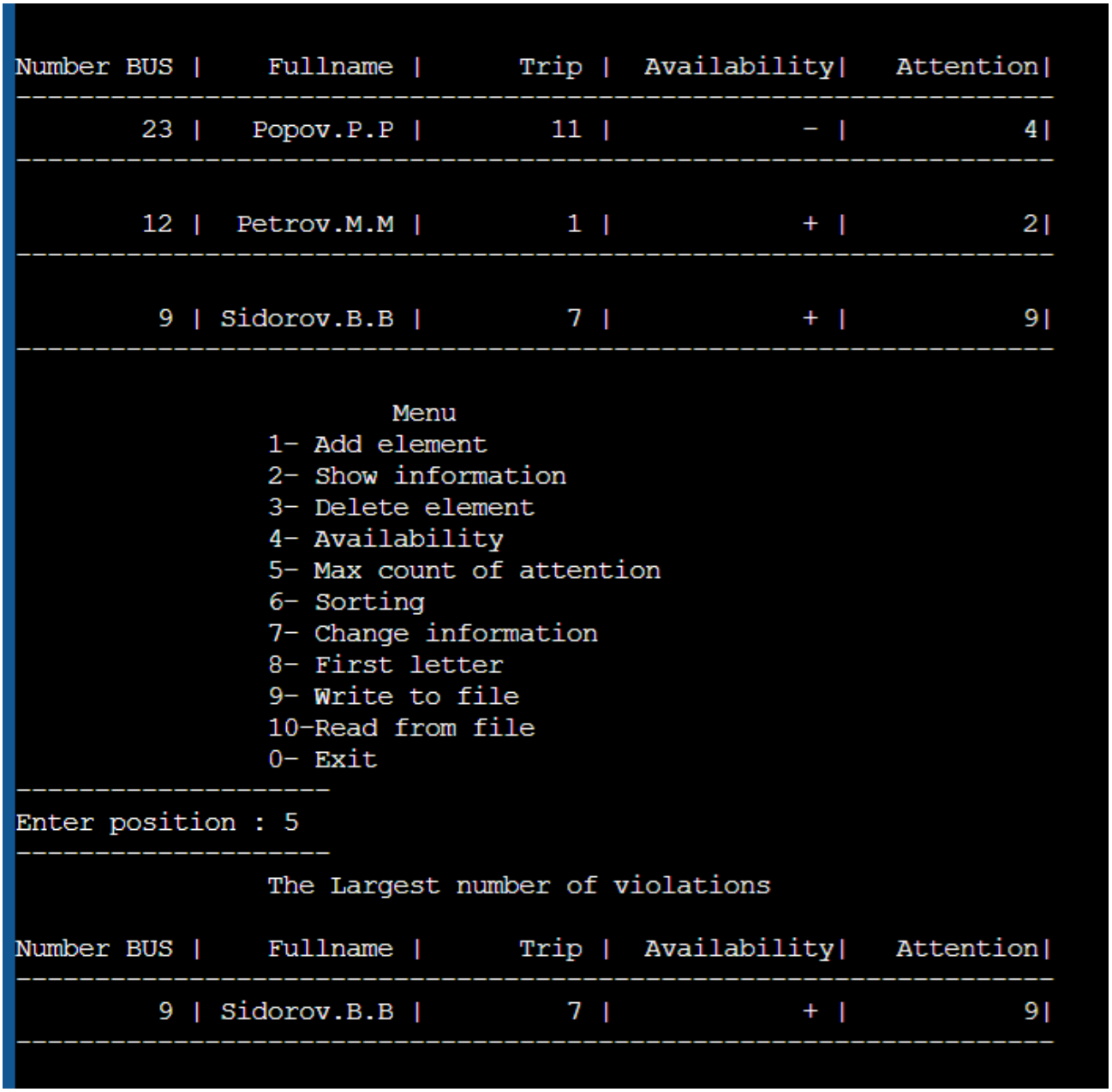

Рисунок 3.6 - Найбільша кількість зауважень

3.7 Сортування даних

Коли користувач натисне 6 йому виведе відсортований список за полем **num** .Сортування починається з найменшого елемента і закінчується найбільшим елементом за полем **num**(рис. 3.7).

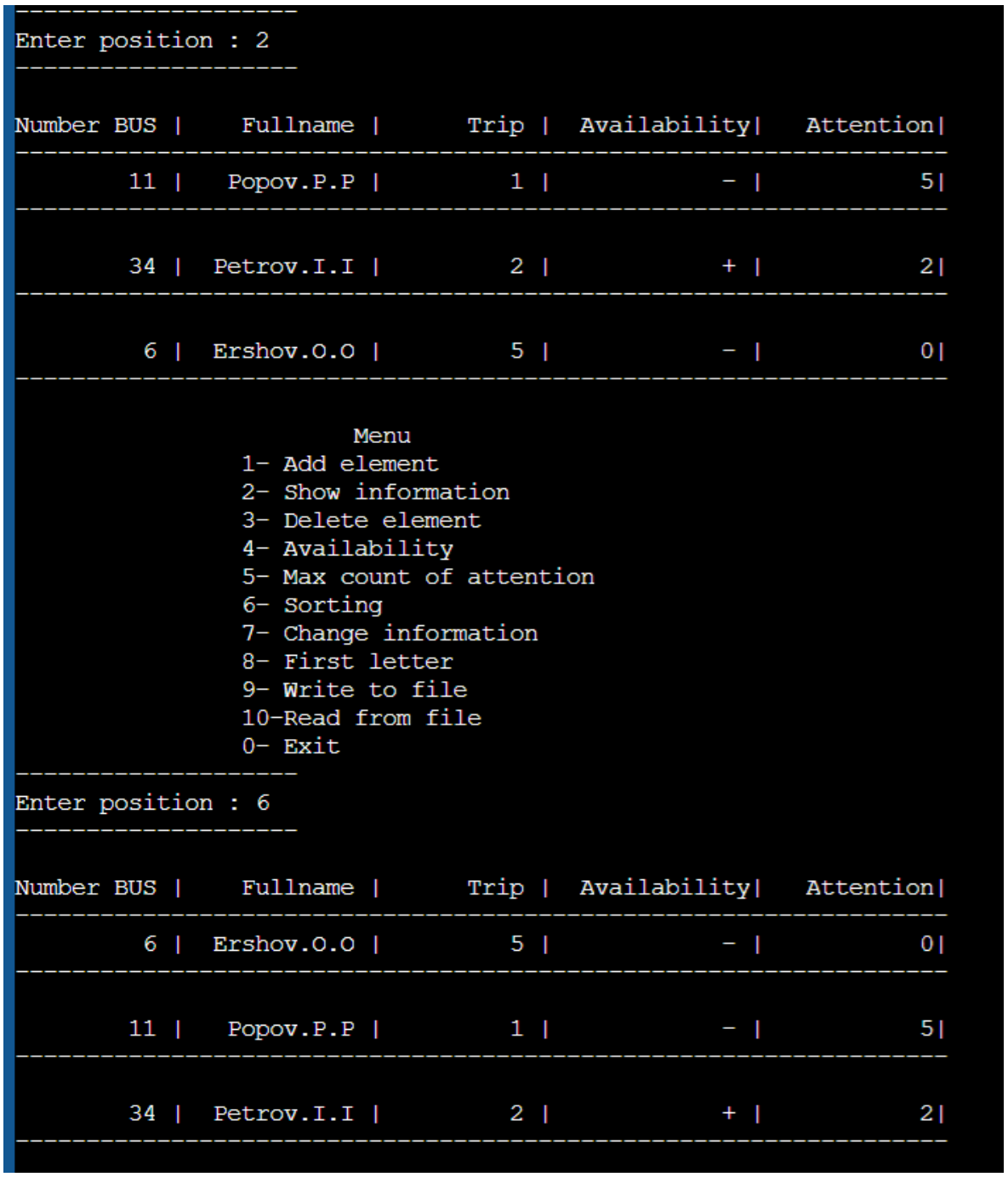

Рисунок 3.7 - Сортування даних

3.8 Зміна інформації

Коли користувач натисне 7 йому буде запропоновано ввести позицію елемента з яким потрібно працювати. Після цього користувач заново вводить всю інформацію та додається цей елемент.

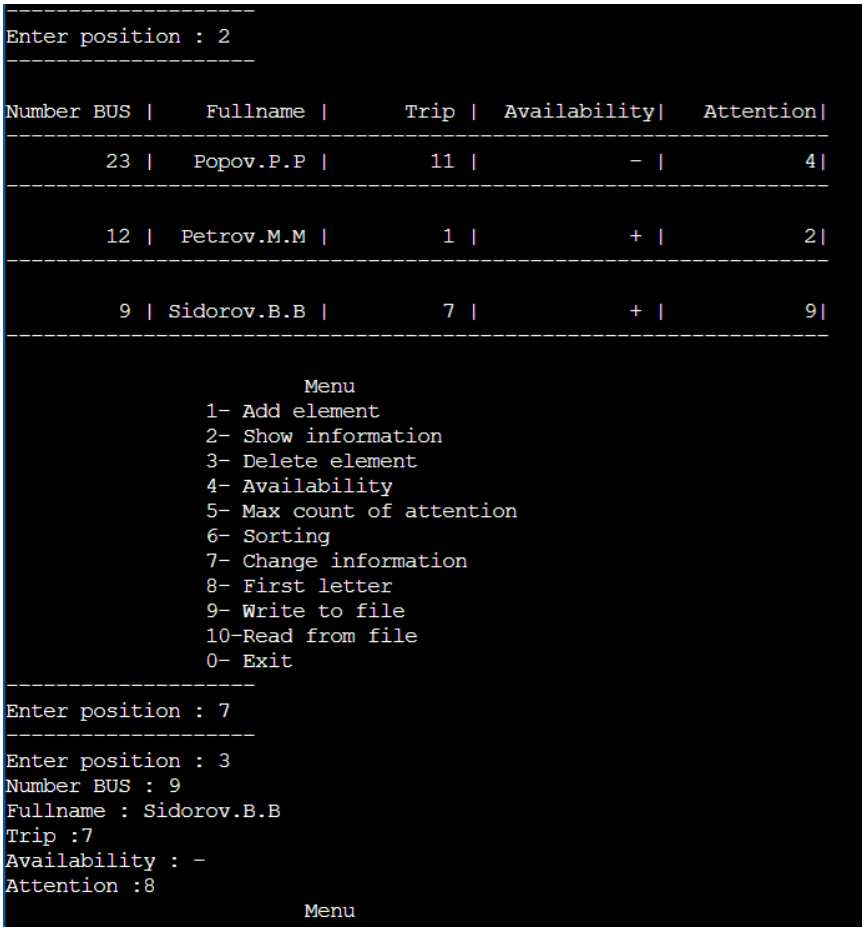

Рисунок 3.81 - Ввід нових даних

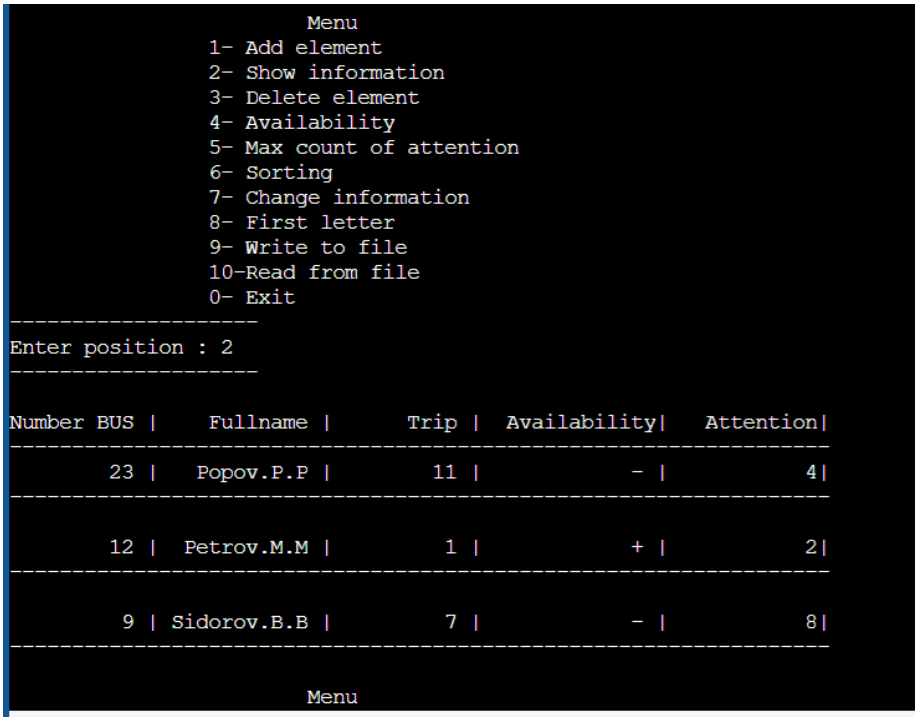

Рисунок 3.82 - Вивід нової інформації

3.9 Пошук за літерою

Коли користувач натисне 8 йому буде запропоновано ввести першу букву для пошуку елементу/ів. Після введення програма шукає та виводить усі існуючі елементи на задану букву за полем **name**.

```
Number BUS |    Fullname |      Trip |  Availability|   Attention|
-------------------------------------------------------------------
         6 |  Ershov.O.O |         5 |            - |           0|
-------------------------------------------------------------------

        11 |   Popov.P.P |         1 |            - |           5|
-------------------------------------------------------------------

        34 |  Petrov.I.I |         2 |            + |           2|
-------------------------------------------------------------------

                     Menu
                1- Add element
                2- Show information
                3- Delete element
                4- Availability
                5- Max count of attention
                6- Sorting
                7- Change information
                8- First letter
                9- Write to file
                10-Read from file
                0- Exit
--------------------
Enter position : 8
--------------------
Enter letter to search : P
People with such surnames start on ' P ' :

Number BUS |    Fullname |      Trip |  Availability|   Attention|
-------------------------------------------------------------------
        11 |   Popov.P.P |         1 |            - |           5|
-------------------------------------------------------------------

        34 |  Petrov.I.I |         2 |            + |           2|
-------------------------------------------------------------------
```

Рисунок 3.9 - Пошук за літерою

3.10 Запис у файл

Коли користувач натисне 9 вся існуюча інформація буде внесена у файл .

```
                    Menu
               1- Add element
               2- Show information
               3- Delete element
               4- Availability
               5- Max count of attention
               6- Sorting
               7- Change information
               8- First letter
               9- Write to file
               10-Read from file
               0- Exit
--------------------
Enter position : 2
--------------------

Number BUS |    Fullname |      Trip |  Availability|   Attention|
-------------------------------------------------------------------
        23 |   Popov.P.P |        11 |            - |           4|
-------------------------------------------------------------------

        12 |  Petrov.M.M |         1 |            + |           2|
-------------------------------------------------------------------

         9 | Sidorov.B.B |         7 |            - |           8|
-------------------------------------------------------------------

                     Menu
```

Рисунок 3.100 - Запис у файл

```
                    Menu
               1- Add element
               2- Show information
               3- Delete element
               4- Availability
               5- Max count of attention
               6- Sorting
               7- Change information
               8- First letter
               9- Write to file
               10-Read from file
               0- Exit
--------------------
Enter position : 9
--------------------
Information was written to the file.
```

Рисунок 3.101 - Запис у файл

main.c D:\f.txt

```
1    23    Popov.P.P    11    -    4
2    12    Petrov.M.M    1    +    2
3     9    Sidorov.B.B    7    -    8
4
```

Рисунки 3.102 - Дані в файлі

3.11 Вивести з файлу

Коли користувач натисне 10 буде виведена інформація ,яка записана заздалегідь у файл. Ця інформація працює так само ,як і та ,що була б введена з клавіатури .

main.c D:\f.txt

```
1    11    popov    4    +    7
2     4    petrov    9    -    2
3     2    ershov    5    -    7
4
```

Рисунок 3.111 - Дані в файлі

```
                    Menu
               1- Add element
               2- Show information
               3- Delete element
               4- Availability
               5- Max count of attention
               6- Sorting
               7- Change information
               8- First letter
               9- Write to file
               10-Read from file
               0- Exit
-------------------
Enter position : 10
--------------------

BUS number|     Fullname |      Trip |  Availability|   Attention|
-------------------------------------------------------------------
         2 |      ershov |         5 |            - |           7|
-------------------------------------------------------------------

         4 |      petrov |         9 |            - |           2|
-------------------------------------------------------------------

        11 |       popov |         4 |            + |           7|
-------------------------------------------------------------------

                    Menu
```

Рисунок 3.112 - Вивід даних

ВИСНОВКИ

Під час роботи було виконано завдання і створена динамічна структура (однозв’язний список). Робота виконана в повному обсязі.

Навчився більш ретельно вибирати функції для роботи зі строками. Зробив правильне сортування елементів списку. На мою думку треба звернути увагу на функцію виведення максимального значення за полем. Під час роботи виникли проблеми з цією функцією. Особливу увагу треба звернути на функцію зчитування з файлу. Було важко її реалізувати хоча в голові все було не складно.

Не раз виникали помилки з написанням коду ,але вони були вирішені, коли я прочитав додаткову літературу.

На мою думку, мій внесок в курсову роботу приблизно 80%.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Лінійний однозв'язний список. - Ерудит.нет: веб-сайт. URL:https://erudyt.net/navchalni-predmety/informatika/prohramuvannya/linijnyj-odnozvyaznyj-spysok.html (дата звернення : 28.05.2022).

2.ДИНАМІЧНІ СТРУКТУРИ ДАНИХ, ЛІНІЙНІ СПИСКИ ,БІНАРНІ ДЕРЕВА-Інформатика:веб-сайт.URL:https://stud.com.ua/97380/informatika/dinamichni_strukturi_danih (дата звернення 28.05.2022).

3.Linked list - Wikipedia: веб-сайт.URL:https://en.wikipedia.org/wiki/Linked_list (дата звернення 30.05.2022).

4. Interview Analysis System - Part-7 (Read CSV file data into LinkedList) - Read Topics . URL: https://youtu.be/GknT_DfgkoI (дата звернення :26.05.2022).

5.Конспект лекцій з дисципліни «Алгоритмізація та програмування» для студентів спеціальності 122 – «Комп'ютерні науки» /Укл.: О.Ю. Бабілунга. – Одеса: ОНПУ, 2017. – 138 с.

6..Практичне заняття 3(2 сем).

7.Методичні вказівки до лабораторних робіт з дисципліни «Алгоритмізація та програмування» для студентів спеціальності 122 «Комп’ютерні науки» всіх форм навчання / Укл.: О. Ю. Бабілунга, М. Г. Глава, Г. С. Панькіна. – Одеса: ОНПУ, 2018. – 83 с.

ДОДАТОК А
КОД ПРОГРАМИ

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct BUS BUS;
static BUS *front = NULL,*temp,*ptr;

typedef struct BUS{
	int num;			//number bus
	char name[20];			//fullanme
	int trip;			//num of trip
	char av[2];			//availability
	int attention;			//count of attention

	BUS *next;			//pointer next
}BUS;

int count_of_elements();		//кількість елементів
void new_information(BUS *ptr);	//перехід
void delete_elements();		//видалити елемент за позицією
void add_information();		//додати елемент
void all_information();		//уся присутня інформація
void availability_of_elements();	//наявність. + -> так , - ->ні
void count_of_attention();		//шукаєм елемент з найбільшою кількістю порушень
void sorting();			//сортування
void swap_char(char *p,char *q);	//допоміжна функція для сорутвання
void write_new_infromation();	//змінити інформацію
void first_letter();		//перша літера
void write_to_file();		//запис у файл
void read_from_file();		//інформація вже є в файлі

int main(){
	int a,replace;
	do {
	printf("\t\t\tMenu\n");
	printf("\t\t1- Add element \n");
	printf("\t\t2- Show information\n");
	printf("\t\t3- Delete element\n");
	printf("\t\t4- Availability\n");
	printf("\t\t5- Max count of attention\n");
	printf("\t\t6- Sorting\n");
	printf("\t\t7- Change information\n");
	printf("\t\t8- First letter\n");
	printf("\t\t9- Write to file\n");
	printf("\t\t10-Read from file\n");
```

```
        printf("\t\t0- Exit\n");
        printf("--------------------\n");
        printf("Enter position : ");
        scanf("%d",&a);
        printf("--------------------\n");

        switch(a){
        case 1:
                add_information();              //готово
                break;
        case 2:
                //count_of_elements();          //готово
                all_information();              //готово
                break;
        case 3:
                delete_elements();              //готово
                break;
        case 4:
                availability_of_elements();     //готово
                break;
        case 5:
                count_of_attention();           //готово
                break;
        case 6:
                sorting();                      //готово
                break;
        case 7:
                write_new_infromation();        //готово
                break;
        case 8:
                first_letter();                 //готово
                break;
        case 9:
                write_to_file();                //готово
                break;
        case 10:
                read_from_file();               //готово
                break;
        case 0:
                printf("Well...Exit...");
                exit(0);
        default:
                printf("Enter correct number.\n");
                break;
        }
        }while(1);{
        return 0;
        }
}

//////////////////////count of elements//////////////////
```

```
int count_of_elements(){
	int count=0;
	if(front == NULL){
	return count;
	}else{
	BUS *temp = front;
	while (temp != NULL){
	count++;
	temp = temp->next;
	}
	//printf("Count of elements %d .\n",count);
	return count;
	}
}

//////////////////////////next elements//////////////////////

void new_information(BUS *ptr){
	if(front == NULL){
	front = ptr;
	return;
	}else{
	BUS *temp = front;
	while(temp->next!=NULL){
	temp=temp->next;
	}
	temp->next=ptr;
	}
}

//////////////////////list of information//////////////////////

void all_information(){
	if(front == NULL){
	printf("Oh..I don't see anything :(\n");
	return;
	}else{
	BUS *temp = front;
	printf("\nBUS number|\tFullname |\tTrip |\tAvailability|\tAttention|\n");
	printf("-----------------------------------------------------------------\n");
		while (temp != NULL) {
printf("%9d|%12.20s|%10d|%13.5s|%12d|\n",temp->num,temp->name,temp->trip,temp->av,tem
p->attention);
		printf("-----------------------------------------------------------------\n\n");
		temp = temp->next;
	}
	}
}

//////////////////////information//////////////////////////
```

```
void add_information(){
	BUS *ptr = malloc(sizeof(BUS));
	ptr->next = NULL;

	/*Числа ,які дорівнюють нулю або менше
	нуля не підходять . Потрібно знову
	ввести значення*/

	do{
	printf("BUS number: ");
	scanf("%d",&ptr->num);			//number bus
	}while(ptr->num<=0);

	printf("Fullname : ");			//fullanme
	scanf("%s",ptr->name);

	do{
	printf("Trip :");			//number trip
	scanf("%d",&ptr->trip);
	}while(ptr->trip<=0);

	printf("Availability : ");		//availability , + -> yes, - ->no
	scanf("%s",ptr->av);

	do{
	printf("Attention :");			//number of attention
	scanf("%d",&ptr->attention);
	}while(ptr->attention<0);

	new_information(ptr);
}

//////////////////////delete elements//////////////////////////

void delete_elements(){
	int value, position, i = 0;
	value = count_of_elements();
	temp=front ;
	if(front == NULL){
	printf("Oh..I don't see anything :(\n");
	return ;
	}else{
	//number to delete
	printf("Enter position : ");
	scanf("%d",&position);

	if (position > 0 && position <= value) {
	if (position == 1) {
	temp = temp -> next;
	front = temp;
```

```
        } else {
        while (temp != 0) {
        if (i == (position - 1)) {
        ptr->next = temp->next;
        break;
        } else {
        i++;
        ptr = temp;
        temp = temp -> next;
        }
        }
        }
        }else{
        printf("I don't see this elements\n");
        }
        }
}

//////////////////////sorting//////////////////////
void swap_char(char *p,char *q){     //функція ,яка приймає значення і міняє місцями
        char t[20] = "";
        strcpy(t,p);
        strcpy(p,q);
        strcpy(q,t);
}

void sorting(){
        if(front == NULL){
        printf("Oh..I don't see anything :(\n");
        return;
        }else{
        BUS *temp = front,*index=NULL;
        int sort,sort1,sort2;

        while(temp!=NULL){
        index=temp->next;
        while(index!=NULL){
                if(temp->num > index->num){
                sort = temp->num;
                sort1 = temp->trip;
                sort2 = temp->attention;

                temp->num = index->num;
                temp->trip = index->trip;
                temp->attention = index->attention;

                swap_char(index->name,temp->name);
                swap_char(index->av,temp->av);

                index->num = sort;
                index->trip = sort1;
```

```
                index->attention = sort2;
                }index=index->next;
        }temp=temp->next;
        }
        }all_information();
}

////////////////BUS + - ////////////////////////////////////////

void availability_of_elements(){
        if(front == NULL){
        printf("Oh..I don't see anything :(\n");
        return;
        }
        BUS *temp = front;
        char *availability;
        printf("Enter availability: ");//вводимо характеристику ,яка цікавить
        scanf("%s",availability);
        printf("\nBUS number|\tFullname |\tTrip |\tAvailability|\tAttention|\n");
        printf("-----------------------------------------------------------------\n");
        while (temp!= NULL) {
        if(strcmp(temp->av,availability)==0){
printf("%9d|%12.20s|%10d|%13.5s|%12d|\n",temp->num,temp->name,temp->trip,temp->av,tem
p->attention);
                printf("-----------------------------------------------------------------\n\n");
        }
        temp = temp->next;
        }
}

//////////////////////////////max attention////////////////

void count_of_attention(){

        if(front == NULL){
        printf("Oh..I don't see anything :(\n");
        return;
        }
        BUS *temp = front;
        int max = 0;
        printf("\t\tThe Largest number of violations\n");
        printf("\nBUS number|\tFullname |\tTrip |\tAvailability|\tAttention|\n");
        printf("-----------------------------------------------------------------\n");

        for(;temp!=NULL;temp=temp->next){
        if(max<temp->attention){
        max=temp->attention;
        }
        }
        BUS *index = front;
        for(;index!=NULL;index=index->next){
```

```
        if(index->attention == max){
printf("%9d|%12.20s|%10d|%13.5s|%12d|\n",index->num,index->name,index->trip,index->av,in
dex->attention);
        printf("-----------------------------------------------------------------\n\n");
        }
        }
}

//////////////////////write new informayion////////////////

void write_new_infromation(){

        int value, position, i = 0;
        value = count_of_elements();
        temp=front ;
        if(front == NULL){
        printf("Oh..I don't see anything :(\n");
        return ;
        }else{
        printf("Enter position : ");
        scanf("%d",&position);

        if (position > 0 && position <= value) {
        if (position == 1) {
        temp = temp -> next;
        front = temp;
        } else {
        while (temp != 0) {
        if (i == (position - 1)) {
        ptr->next = temp->next;
        break;
        } else {
        i++;
        ptr = temp;
        temp = temp -> next;
        }
        }
        }add_information();
        }else{
        printf("Element to change not found\n");
        }
        }
}

//////////////////first letter////////////////////////////

void first_letter(){
        char *letter;

        if(front == NULL){
        printf("Oh..I don't see anything :(\n");
```

```
        return;
        }else{
        BUS *temp = front;

        printf("Enter letter to search : ");                    // вводимо букву для пошуку
        scanf("%s",letter);
        printf("People with such surnames start on ' %s ' :\n",letter);

        printf("\nBUS number|\tFullname |\tTrip |\tAvailability|\tAttention|\n");
        printf("-----------------------------------------------------------------\n");
        while(temp!=NULL){
                if(strncmp(temp->name, letter,1)==0){
printf("%9d|%12.20s|%10d|%13.5s|%12d|\n",temp->num,temp->name,temp->trip,temp->av,tem
p->attention);
                printf("-----------------------------------------------------------------\n\n");
                }
                temp = temp->next;
        }
        }
}

//////////////////////file//////////////////////////////

void write_to_file(){
        if(front == NULL){
        printf("Nothing to write to file...\n");
        return;
        }else{
        FILE *f = fopen("D:\\f.txt","w");

        if(f == NULL){
        printf("File is not available.\n");
        }else{
        BUS *temp = front;
        while(temp!=NULL){
fprintf(f,"%9d%12.20s%10d%13.5s%12d\n",temp->num,temp->name,temp->trip,temp->av,tem
p->attention);
                temp = temp->next;
        }
        printf("Information was written to the file.\n");
        fclose(f);
        }
        }
}

void read_from_file(){
   FILE* f;
        f = fopen("D:\\f.txt", "r");
        if (f == NULL) {
        printf("File is not available. \n");
        return ;
```

```
        }else{
        BUS *node = (BUS*)malloc(sizeof(BUS));
while(fscanf(f,"%d%s%d%s%d",&node->num,node->name,&node->trip,node->av,&node->atte
ntion)!=EOF) {
        node->next = front;
        front=node;
        node = malloc(sizeof(BUS));
        }all_information();
        }
        fclose(f);
}
```