**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 8**

**по дисциплине «Программирование»**

Тема: «Линейные односвязные списки»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3311 |  | Шарпинский Д. А. |
| Преподаватель |  | Хахаев И. А. |

Санкт-Петербург

2024

**Цель работы.**

Научиться работать с линейными односвязными списками в языке С.

**Задание (вариант 14)**

Разработать подалгоритм и написать функцию, удаляющую в односвязном списке элемент перед элементом с указанным номером. Если указан номер первого элемента, вывести сообщение о невозможности удаления.

**Постановка задачи и описание решения**

Целью написания программы является демонстрация работы с односвязным списком пользователей, включая основные операции, такие как чтение из файла, добавление, удаление, сортировка и фильтрация данных. Жизненный цикл программы начинается с инициализации списка пользователей путём чтения данных из файла CSV.

Структура User, используемая для хранения информации о пользователях, представлена следующим образом:

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Описание |
| id | int | Уникальный идентификатор пользователя |
| fullName | char\* | Полное имя пользователя |
| age | int | Возраст пользователя |
| profession | char\* | Профессия пользователя |
| friendsRating | float | Рейтинг среди друзей |
| publicRating | float | Общественный рейтинг |
| friendsCount | int | Количество друзей |
| friendsId | int\* | Массив идентификаторов друзей |
| next | User\* | Указатель на следующего пользователя |

А также важной частью программы является структура Head, которая служит для управления списком пользователей:

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Описание |
| last\_id | int | Идентификатор последнего пользователя |
| isFriendsSorted | int | Флаг сортировки списка по количеству друзей |
| first | User\* | Указатель на первого пользователя в списке |
| last | User\* | Указатель на последнего пользователя в списке |

После инициализации данных пользователь встречается с меню, предлагающим различные опции для управления списком:

1. вывод всего списка
2. сортировка по количеству друзей
3. сортировка по идентификатору
4. добавление нового пользователя
5. фильтрация по имени или профессии
6. удаление пользователя перед указанным id
7. очистка списка
8. выход из программы

Для сортировки используется алгоритм пузырьковой сортировки, адаптированный для работы со связными списками, что позволяет упорядочивать элементы списка согласно заданным критериям.

Одним из ключевых аспектов программы является управление памятью. Благодаря использованию функций для динамического выделения и освобождения памяти, в программе отсутствуют утечки памяти, что было проверено с использованием инструмента Valgrind на операционной системе Linux.

**Описание переменных**

**Функция main()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | head | Head\* | Указатель на голову списка, хранит основные данные о списке |
| 2 | user | User\* | Вспомогательный указатель для работы с пользователями |
| 3 | slen | int | Длина строки, используется при чтении данных из файла |
| 4 | i | int | Индекс в цикле, общее использование в различных циклах for |
| 5 | n | int | Количество строк (записей) в файле CSV |
| 6 | count | int | Счётчик успешно добавленных пользователей |
| 7 | option | int | Переменная для хранения выбранного пользователем варианта действия в меню |
| 8 | temp | char[] | Временная строка для чтения данных из файла |
| 9 | splitArray | char\*\* | Массив строк, полученный в результате разбиения строки из файла |
| 10 | file | FILE\* | Указатель на файл, открытый для чтения данных |

**Функция makeNode()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | str | char\*\* | Массив строк с данными для нового пользователя |

**Функция addNode()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | my\_head | Head\* | Указатель на голову списка |
| 2 | new\_node | User\* | Указатель на добавляемого пользователя |

**Функция selectId()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | my\_head | Head\* | Указатель на голову списка |
| 2 | id | int | Идентификатор пользователя, которого требуется удалить |

**Функция deleteNode()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | my\_head | Head\* | Указатель на голову списка |
| 2 | current\_node | User\* | Указатель на удаляемого пользователя |

**Функция deleteById()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | head | Head\* | Указатель на голову списка |

**Функция addUser()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | head | Head\* | Указатель на голову списка |

**Функция freeStruct()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | user | User\* | Указатель структуру, для которой требуется провести операцию очистки памяти |

**Функция freeList()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | my\_head | Head\* | Указатель на голову списка |

**Функция bubbleSortByField()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | my\_head | Head\* | Указатель на голову списка |
| 2 | desc | char\* | Строка с названием поля, по которому производится сортировка |

**Функция swapNodes()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | prevNode | User\*\* | Указатель на предыдущий элемент в списке |
| 2 | a | User\* | Указатель на текущий элемент для обмена |
| 3 | b | User\* | Указатель на следующий элемент для обмена |

**Функция startsWithIgnoreCase()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | str | char\* | Строка, в которой ищется подстрока |
| 2 | prefix | char\* | Подстрока, по которой производится поиск |

**Функция filterList()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | head | Head\* | Указатель на голову списка |

**Функция clearList()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | head | Head\* | Указатель на голову списка |

**Функция simpleSplit()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | str | char\* | Строка для разбиения |
| 2 | length | int | Длина строки |

**Функция simpleSplitInt()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | user | User\* | Указатель структуру, для которой требуется провести операцию очистки памяти |
| 2 | str | char\* | Строка для разбиения |
| 3 | isManual | int | Флаг, указывающий, требуется ли сопоставление количества друзей, введенных в строке, и количества, указанного пользователем |
| 4 | idList | int[] | Массив всех идентификаторов пользователей |
| 5 | usersCount | int | Количество всех пользователей |

**Функция trim()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | str | char\* | Строка для обрезки |

**Контрольные примеры**

**Пример:**

Choose option and press ENTER:

1. Print all users

2. Sort users by friends count

3. Sort users by id

4. Add new user

5. Filter users by name or profession

6. Delete user before specified id

7. Clear list

8. Exit

Option: 1

ID Full Name Age Profession Friends Rating Public Rating Friends Count Friends IDs

1 John Doe 30 teacher 4.5 3.9 3 [2, 5, 7]

2 Jane Smith 25 engineer 3.8 4.1 2 [1, 3]

3 Alice Johnson 28 driver 4.2 3.7 4 [1, 2, 6, 8]

4 Michael Brown 33 pilot 3.9 4.0 5 [3, 6, 9, 10, 2]

5 Emily Davis 27 dentist 4.1 3.8 3 [1, 2, 3]

6 David Wilson 35 actor 4.0 4.2 2 [5, 2]

7 Linda Martinez 32 actor 3.9 3.7 4 [4, 6, 5, 1]

8 Robert White 29 teacher 4.3 3.8 3 [1, 2, 3]

9 Sarah Taylor 31 teacher 4.0 4.1 5 [8, 5, 6, 3, 1]

10 James Anderson 34 pilot 4.2 3.9 2 [1, 2]

11 Davidios Morgan 20 teacher 2.0 1.0 0 []

12 Casey Taylor 28 pilot 3.7 1.7 5 [6, 8, 13, 10, 15]

13 Jamie Jones 27 dentist 4.4 1.3 5 [8, 3, 11, 1, 7]

14 Charlie Williams 26 engineer 3.4 3.5 0 []

15 Sam Jones 26 driver 2.8 4.4 3 [3, 5, 2]

16 Jordan Miller 35 engineer 3.0 2.0 4 [1, 13, 4, 14]

17 Chris Moore 20 pilot 2.7 4.4 0 []

18 Charlie Williams 32 actor 3.1 2.9 1 [14]

19 Casey Wilson 20 dentist 3.1 2.7 0 []

20 Taylor Johnson 32 driver 4.1 2.8 4 [12, 7, 20, 10]

21 Cameron Moore 28 pilot 2.4 2.1 2 [7, 10]

22 Casey Moore 35 actor 3.9 1.8 2 [13, 15]

23 Charlie Moore 24 teacher 3.4 1.7 5 [10, 3, 25, 18, 22]

24 Chris Wilson 27 teacher 4.4 3.6 3 [7, 14, 20]

25 Cameron Wilson 32 dentist 4.3 2.6 5 [8, 14, 15, 23, 29]

26 Chris Johnson 21 pilot 2.5 1.5 0 []

27 Jamie Moore 27 dentist 2.9 4.4 5 [25, 1, 11, 31, 27]

28 Jamie Moore 24 driver 2.2 1.1 0 []

29 Jordan Jones 31 engineer 3.9 3.7 2 [27, 16]

30 Sam Wilson 26 driver 2.3 4.3 0 []

Press ENTER to continue

Choose option and press ENTER:

1. Print all users

2. Sort users by friends count

3. Sort users by id

4. Add new user

5. Filter users by name or profession

6. Delete user before specified id

7. Clear list

8. Exit

Option: 2

Sorted!

Press ENTER to continue

Choose option and press ENTER:

1. Print all users

2. Sort users by friends count

3. Sort users by id

4. Add new user

5. Filter users by name or profession

6. Delete user before specified id

7. Clear list

8. Exit

Option: 4

Enter full name: new User

Enter age: 12

Enter profession: pilot

Enter friends rating (float number less than 5): 4.23

Enter public rating (float number less than 5): 4.1

Enter friends count (less than 30): 12

Enter friends IDs (example: 1,2,3,4)

Available IDs: [4,9,12,13,23,25,27,3,7,16,20,1,5,8,15,24,2,6,10,21,22,29,18,11,14,17,19,26,28,30]

IDs: 1,2,3

It seems that the number of entered IDs does not correspond to the specified number of friends -> updating friends count: 3

New user successfully added!

Press ENTER to continue

Choose option and press ENTER:

1. Print all users

2. Sort users by friends count

3. Sort users by id

4. Add new user

5. Filter users by name or profession

6. Delete user before specified id

7. Clear list

8. Exit

Option: 6

Enter the users id to remove the user BEFORE it

(enter 0 if you don't want to do it)

Here is list of ids:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Example:

id: 2 -> user with id 1 will be removed

id: 2

Success: user with id 1 has been removed!

Press ENTER to continue

Choose option and press ENTER:

1. Print all users

2. Sort users by friends count

3. Sort users by id

4. Add new user

5. Filter users by name or profession

6. Delete user before specified id

7. Clear list

8. Exit

Option: 6

Enter the users id to remove the user BEFORE it

(enter 0 if you don't want to do it)

Here is list of ids:

[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Example:

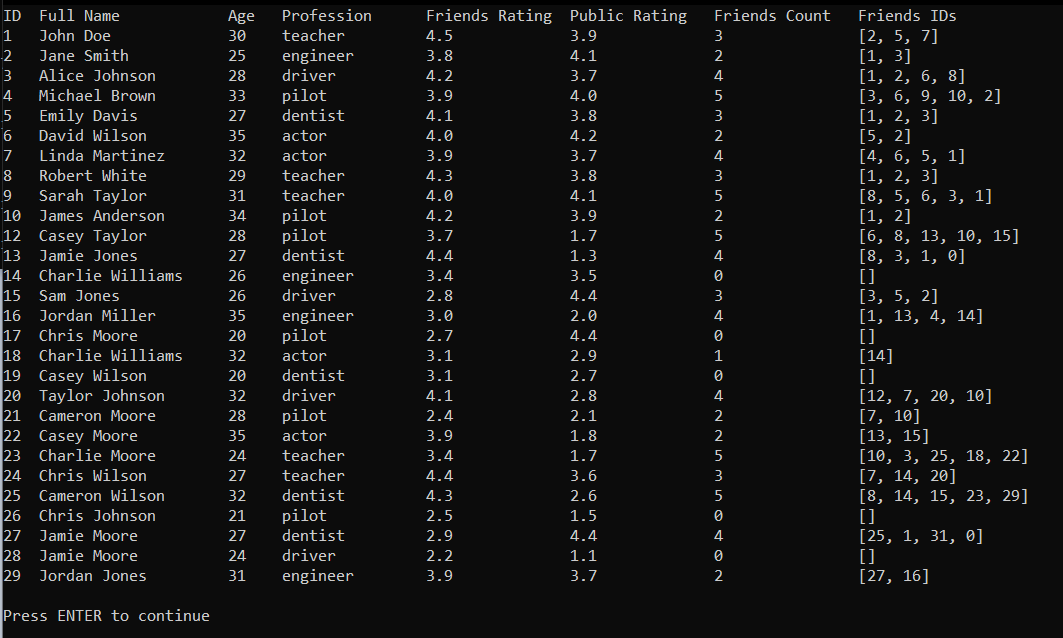
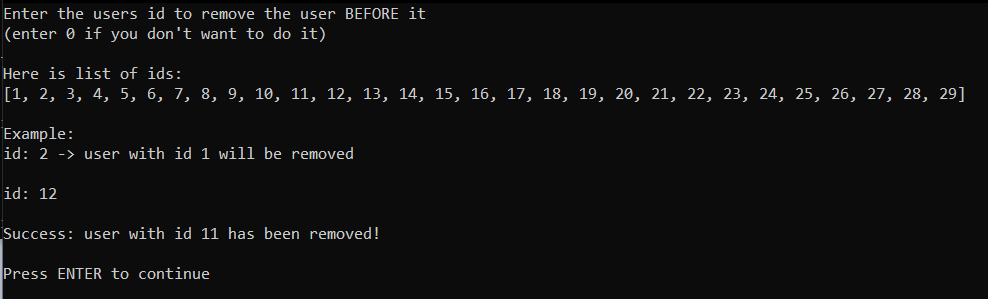
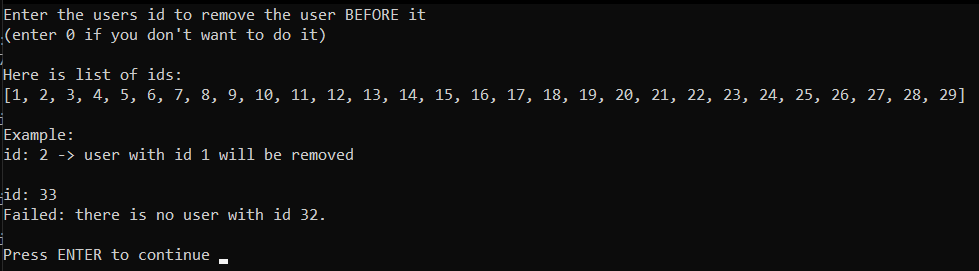
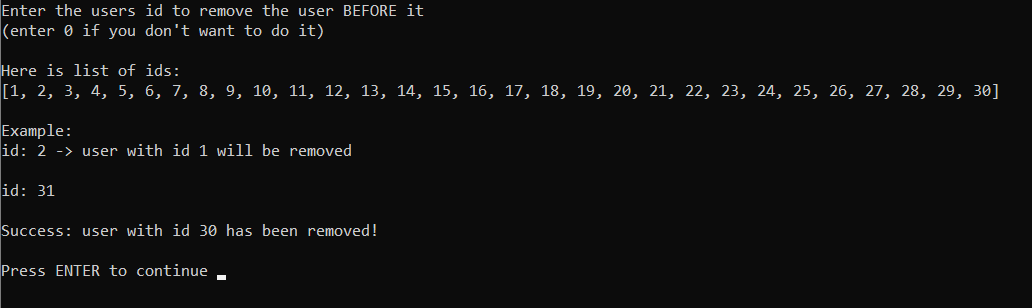
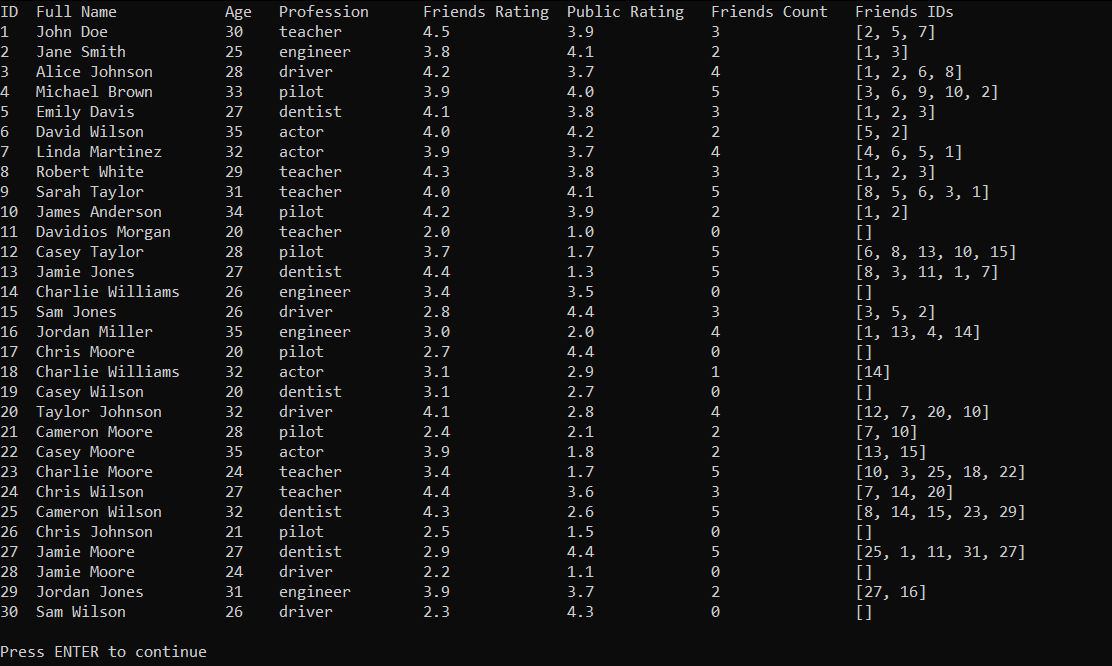
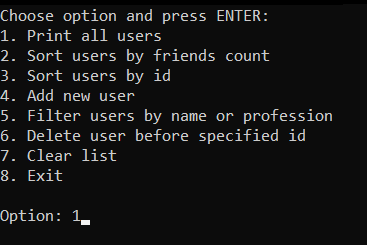
id: 2 -> user with id 1 will be removed

id: 2

Failed: there is no user with id 1.

Press ENTER to continue

**Примеры выполнения программы**

****

**Выводы.**

В ходе выполнения работы были получены практические навыки в работе с линейными односвязными списками в языке С.