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)

С использованием структуры, созданной при выполнении лабораторной работы №7 (по выбранной предметной области), создать односвязный линейный список и выполнить задание в соответствии с вариантом.

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.

ПРЕДМЕТНАЯ ОБЛАСТЬ:

- **Пользователи социальной сети**

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