Министерство цифрового развития, связи и массовых коммуникаций РФ

Федеральное государственное бюджетное общеобразовательное учреждение

высшего образования

“Сибирский Государственный Университет Телекоммуникаций и Информатики”

(СибГУТИ)

Кафедра телекоммуникационных систем и вычислительных средств

Отчет по лабораторной работе на тему: Контейнерные типы

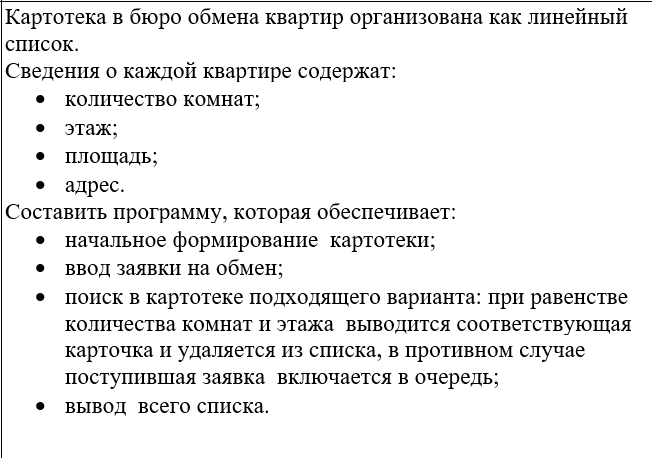
Выполнил: студент 1 курса группы ИА-231

Зырянов Иван Александрович

Проверил: Вейлер Андрей Игоревич

Новосибирск, 2023

**ЗАДАНИЕ**

**Задание.** 

**Код**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Определение структуры квартиры

struct apartment {

    int num\_rooms;

    int floor;

    float area;

    char address[100];

};

// Определение элемента списка квартир

struct node {

    struct apartment data;

    struct node\* next;

};

// Определение очереди заявок

struct queue {

    struct apartment data;

    struct queue\* next;

};

// Функция для создания нового элемента списка

struct node\* create\_node(struct apartment apt) {

    struct node\* new\_node = (struct node\*) malloc(sizeof(struct node));

    new\_node->data = apt;

    new\_node->next = NULL;

    return new\_node;

}

// Функция для создания новой заявки

struct queue\* create\_request(struct apartment apt) {

    struct queue\* new\_request = (struct queue\*) malloc(sizeof(struct queue));

    new\_request->data = apt;

    new\_request->next = NULL;

    return new\_request;

}

// Функция для добавления элемента в конец списка

void append(struct node\*\* head\_ref, struct apartment apt) {

    struct node\* new\_node = create\_node(apt);

    struct node\* current = \*head\_ref;

    if (\*head\_ref == NULL) {

        \*head\_ref = new\_node;

    } else {

        while (current->next != NULL) {

            current = current->next;

        }

        current->next = new\_node;

    }

}

// Функция для добавления заявки в конец очереди

void enqueue(struct queue\*\* rear\_ref, struct apartment apt) {

    struct queue\* new\_request = create\_request(apt);

    if (\*rear\_ref == NULL) {

        \*rear\_ref = new\_request;

    } else {

        (\*rear\_ref)->next = new\_request;

        \*rear\_ref = new\_request;

    }

}

// Функция для удаления элемента из списка по индексу

void delete\_node(struct node\*\* head\_ref, int index) {

    struct node\* current = \*head\_ref;

    struct node\* prev = NULL;

    if (index == 0) {

        \*head\_ref = current->next;

        free(current);

    } else {

        for (int i = 0; i < index; i++) {

            prev = current;

            current = current->next;

        }

        prev->next = current->next;

        free(current);

    }

}

// Функция для вывода списка квартир

void print\_list(struct node\* head) {

    struct node\* current = head;

    printf("kartoteka kvartir:\n");

    while (current != NULL) {

        printf("kolichestvo komnat: %d, etash: %d, ploshad: %.2f, address: %s\n",

            current->data.num\_rooms, current->data.floor, current->data.area, current->data.address);

        current = current->next;

    }

}

// Определение структуры квартиры, состоящей из четырех полей: num\_rooms, floor, area и address.

int main() {

    struct node\* head = NULL;

    struct queue\* rear = NULL;

    int choice = 0;

    do {

        printf("\n vyberite: \n");

        printf("1. formirovanie kartoteki kvartir\n");

        printf("2. Vvod zayavki na obmen\n");

        printf("3. Poisk podhodyyashey kvartiry\n");

        printf("4. Vidod vsego spiska\n");

        printf("5. Vihod\n");

        scanf("%d", &choice);

        switch (choice) {

            case 1: // Начальное формирование картотеки

                printf("Vvedite dannie kvartiry:\n");

                struct apartment apt;

                printf("kolichestvo komnat: ");

                scanf("%d", &apt.num\_rooms);

                printf("etash: ");

                scanf("%d", &apt.floor);

                printf("ploshad: ");

                scanf("%f", &apt.area);

                printf("address: ");

                scanf(" %[^\n]s", apt.address);

                append(&head, apt);

                printf("Krvartira dobavlena v kartoteky.\n");

                break;

            case 2: // Ввод заявки на обмен

                printf("Vvedite dannie zaiyavki:\n");

                struct apartment request;

                printf("kolichestvo komnat: ");

                scanf("%d", &request.num\_rooms);

                printf("etash: ");

                scanf("%d", &request.floor);

                printf("ploshad: ");

                scanf("%f", &request.area);

                printf("address: ");

                scanf(" %[^\n]s", request.address);

                enqueue(&rear, request);

                printf("zauyavka dobavlena v ochered.\n");

                break;

            case 3: // Поиск подходящей квартиры

                if (rear == NULL) {

                    printf("ochered zauyavok pusta.\n");

                } else {

                    int found = 0;

                    struct node\* current = head;

                    struct node\* prev = NULL;

                    struct queue\* temp = NULL;

                    while (current != NULL) {

                        if (current->data.num\_rooms == rear->data.num\_rooms &&

                            current->data.floor == rear->data.floor) {

                            printf("Naydena podhodyyashey kvartiry:\n");

                            printf("kolichestvo komnat: %d, etash: %d, ploshad: %.2f, address: %s\n",

                                current->data.num\_rooms, current->data.floor,

                                current->data.area, current->data.address);

                            found = 1;

                            if (prev == NULL) {

                                head = current->next;

                            } else {

                                prev->next = current->next;

                            }

                            free(current);

                            temp = rear;

                            rear = rear->next;

                            free(temp);

                            break;

                        } else {

                            prev = current;

                            current = current->next;

                        }

                    }

                    if (!found) {

                        printf("podhodyyashey kvartiry ne Naydena.\n");

                    } else {

                        printf("kvartira ydalena iz spiska.\n");

                    }

                }

                break;

            case 4: // Вывод всего списка

                printf("spisok kvartir v kartoteki:\n");

                print\_list(head);

                break;

            case 5: // Выход

                printf("Do svidaniya!\n");

                exit(0);

            default:

                printf("Neverniy vvod. poprobyi ehse raz.\n");

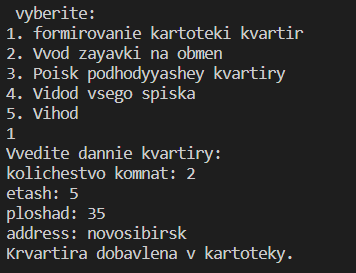
        }

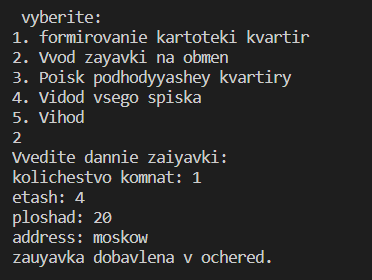
    } while (choice != 5);

    return 0;

}

**Выводы программы:**

****

****

