

**Министерство науки и высшего образования Российской Федерации**  
федеральное государственное автономное образовательное учреждение высшего  
образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**


**Отчет**

по Лабораторная работа No 4

Работа с файлами и структуры данных на основе указателей

Вариант 3-8-3-5.

Автор: Киселёва Анна Николаевна

подпись 

Факультет: ФБИТ

Группа: N3147

Преподаватели:

Грозов Владимир Андреевич



Санкт-Петербург 2025

### **Содержимое Makefile:**

.PHONY: all clean

APP=prg4ankN3147

CFLAGS=-Wall -Wextra -Werror -g

all: \$(APP)

\$(APP): \$(APP).c

gcc -o \$(APP) \$(CFLAGS) \$(APP).c

clean:

rm \$(APP)

Примеры работы программы на различных входных данных:

1) Генерация строк в файл

```
anna@Huaweianna:~/projects/prg4ankN3147$ python3 prg4ankN3147_gen.py -n 3 file
0 26 b'\x13\x00\x00\x00' 16:01:19 10.06.1145
1 49 b'\x13\x00\x00\x00' 20:46:07 30.09.6717
2 72 b'\x13\x00\x00\x00' 06:12:01 02.07.5071
```

2) Опция -v в файле на питоне

```
anna@Huaweianna:~/projects/prg4ankN3147$ python3 prg4ankN3147_gen.py -v
Анна Николаевна Киселёва, гр N3147
Вариант 3-8-3-5
```

3) Команды push\_front, push\_back, pop\_front, pop\_back

```
anna@Huaweianna:~/projects/prg4ankN3147$ ./prg4ankN3147 file
push_front 07:33:22 27.02.2334
push_back 02:57:29 18.10.9121
pop_front
pop_back
[0] "06:12:01 02.07.5071" <^> [1] "20:46:07 30.09.6717" <^> [2] "16:01:19 10.06.1145" <^> NULL
command: push_front
[0] "07:33:22 27.02.2334" <^> [1] "06:12:01 02.07.5071" <^> [2] "20:46:07 30.09.6717" <^> [3] "16:01:19 10.06.1145" <^> NULL
command: push_back
[0] "07:33:22 27.02.2334" <^> [1] "06:12:01 02.07.5071" <^> [2] "20:46:07 30.09.6717" <^> [3] "16:01:19 10.06.1145" <^> [4] "02:57:29 18.10.9121" <^> NULL
command: pop_front
[0] "06:12:01 02.07.5071" <^> [1] "20:46:07 30.09.6717" <^> [2] "16:01:19 10.06.1145" <^> [3] "02:57:29 18.10.9121" <^> NULL
command: pop_back
[0] "06:12:01 02.07.5071" <^> [1] "20:46:07 30.09.6717" <^> [2] "16:01:19 10.06.1145" <^> NULL
```

4) Команды dump / dump\_file, exit

```
^[[Anna@Huaweianna:~/projects/prg4ankN3147$ ./prg4ankN3147 file
dump
dump dump_file
exit
[0] "03:51:49 18.01.8430" <^> [1] "19:27:28 24.03.5870" <^> [2] "07:04:17 13.01.4410" <^> NULL
command: dump
0x0 0x55f30fd24a40 03:51:49 18.01.8430
0x55f30fd24a60 0x40 19:27:28 24.03.5870
0x55f30fd24a40 0x55f30fd24a40 07:04:17 13.01.4410
command: dump_file
command: exit
```

файлы dump\_file и exit создаются в папке проекта.

5) Команда delete N

```
anna@Huaweianna:~/projects/prg4ankN3147$ ./prg4ankN3147 file
delete 1
[0] "03:51:49 18.01.8430" <^> [1] "19:27:28 24.03.5870" <^> [2] "07:04:17 13.01.4410" <^> NULL
command: delete
список до удаления элемента [1]
[0] "03:51:49 18.01.8430" <^> [1] "19:27:28 24.03.5870" <^> [2] "07:04:17 13.01.4410" <^> NULL
список после удаления элемента [1]
[0] "03:51:49 18.01.8430" <^> [1] "07:04:17 13.01.4410" <^> NULL
```

## Обработка ошибок:

### 1) Неподходящий аргумент команд

```
anna@Huaweianna:~/projects/prg4ankN3147$ ./prg4ankN3147 file
push_front lalala
[0] "03:51:49 18.01.8430" <^> [1] "19:27:28 24.03.5870" <^> [2] "07:04:17 13.01.4410" <^> NULL
command: push_front
Ошибка: неподходящий формат данных
```

```
anna@Huaweianna:~/projects/prg4ankN3147$ ./prg4ankN3147 file
delete lala
[0] "03:51:49 18.01.8430" <^> [1] "19:27:28 24.03.5870" <^> [2] "07:04:17 13.01.4410" <^> NULL
command: delete
Ошибка: неподходящий формат индекса
```

### 2) Неподходящая команда

```
anna@Huaweianna:~/projects/prg4ankN3147$ ./prg4ankN3147 file
lala
[0] "03:51:49 18.01.8430" <^> [1] "19:27:28 24.03.5870" <^> [2] "07:04:17 13.01.4410" <^> NULL
Ошибка: неизвестная команда lala
```

### 3) Неподходящее имя файла

```
Ошибка: неизвестная команда lala
anna@Huaweianna:~/projects/prg4ankN3147$ ./prg4ankN3147 lala
Ошибка: не удалось открыть файл: No such file or directory
anna@Huaweianna:~/projects/prg4ankN3147$ |
```

Код программы:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <inttypes.h>
#include <assert.h>
#include <ctype.h>
#include <stdbool.h>

#define DATE_LENGTH 19

typedef struct list_item_t {
    uintptr_t link;
    void *data;
} list_item_t;

typedef struct list_t {
    struct list_item_t *head, *tail;
} list_t;

list_t lst = {NULL, NULL};

bool date(const char *str) {
    if (strlen(str) != DATE_LENGTH) return 0;

    // H:MM:SS

    if (!isdigit(str[0]) || !isdigit(str[1]) || (str[2] != ':') || !isdigit(str[3]) || !isdigit(str[4]) ||
        (str[5] != ':') || (!isdigit(str[6]) || !isdigit(str[7])) || (str[8] != ' ')) return 0;

    //DD.MM.YYYY
```

```

        if (!isdigit(str[9]) || !isdigit(str[10]) || (str[11] != '.') || (!isdigit(str[12]) ||
!isdigit(str[13])) || (str[14] != '.') || (!isdigit(str[15]) || !isdigit(str[16]) || !isdigit(str[17])
|| !isdigit(str[18]))) return 0;

    return 1;
}

```

```

char** data_in_file(FILE *f, size_t *count) {
    char buffer[DATE_LENGTH + 1] = {0};
    char **dates = NULL;
    *count = 0;
    int pos = 0;
    int c;

    while ((c = fgetc(f)) != EOF) {
        if (pos < DATE_LENGTH){
            buffer[pos++] = (char)c;
            buffer[pos] = '\0';

            if (pos == DATE_LENGTH && date(buffer)) {
                dates = realloc(dates, (*count + 1) * sizeof(char*));
                dates[*count] = malloc(DATE_LENGTH + 1);
                strcpy(dates[*count], buffer);
                (*count)++;
                pos = 0;
            }
        }
        else{
            memmove(buffer, buffer + 1, DATE_LENGTH - 1);
            pos = DATE_LENGTH - 1;
            buffer[pos++] = (char)c;

```

```

        buffer[pos] = '\0';

        if (date(buffer)){
            dates = realloc(dates, (*count + 1) * sizeof(char*));
            dates[*count] = malloc(LENGTH + 1);
            strcpy(dates[*count], buffer);
            (*count)++;
            pos = 0;
        }
    }
}

return dates;
}

void print_list(list_t *lst) {
    if (!lst->head) {
        printf("<empty list>\n");
        exit(0);
    }

    int idx = 0;

    list_item_t *li = lst->head, *pi = NULL, *tmp = NULL;
    for (; li != NULL; ) {
        printf("\e[31m[%d]\e[m \e[34m\"%s\"\e[m <^> ", idx, (char*)li->data);
        tmp = li;
        li = (list_item_t*)(li->link ^ (uintptr_t)pi);
        pi = tmp;
        idx++;
    }
}

```

```

    printf("NULL\n");
}

void push_front(list_t *lst, char *s) {
    list_item_t *new_item = malloc(sizeof(*new_item));
    new_item->link = 0;
    new_item->data = s;

    if (lst->head) {
        lst->head->link = (uintptr_t)lst->head->link ^ (uintptr_t)new_item;
        new_item->link = (uintptr_t)lst->head;
        lst->head = new_item;
    }
    else {
        assert(!lst->tail);
        lst->tail = lst->head = new_item;
    }
}

void push_back(list_t *lst, char *s) {
    list_item_t *new_item = malloc(sizeof(*new_item));
    new_item->link = 0;
    new_item->data = s;

    if (lst->tail) {
        lst->tail->link = (uintptr_t)lst->tail->link ^ (uintptr_t)new_item;
        new_item->link = (uintptr_t)lst->tail;
        lst->tail = new_item;
    }
    else {
        assert(!lst->head);

```



```

        lst->head = lst->tail = new_item;
    }
}

```

```

void pop_back(list_t *lst){
    if (!lst->tail) {
        printf("<empty list>\n");
        exit(0);
    }
    list_item_t *tail = lst -> tail;

    if (tail->link == (uintptr_t)NULL) {
        free(tail);
        lst->head = lst->tail = NULL;
    }
    else {
        list_item_t *prev = (list_item_t*)(tail->link ^ (uintptr_t)NULL);
        prev->link ^= (uintptr_t)tail ^ (uintptr_t)NULL;
        lst->tail = prev;
        free(tail);
    }
}

void pop_front(list_t *lst){
    if(lst ->head == NULL) return;
    list_item_t *head = lst -> head;
    if(head -> link == (uintptr_t)NULL){
        free(head);
        lst->tail = lst -> head = NULL;
    }
}

```

```

else{
    list_item_t *prev = (list_item_t*)((uintptr_t)NULL ^ head->link);
    prev->link ^= (uintptr_t)NULL ^ (uintptr_t)head;
    lst->head = prev;
    free(head);
}
}

void delete(list_t *lst) {
    list_item_t *li = lst->tail, *ni = NULL, *tmp = NULL;
    while (li) {
        tmp = li;
        li = (list_item_t*)(li->link ^ (uintptr_t)ni);
        ni = tmp;
        free(tmp);
    }
    lst->head = lst->tail = NULL;
}

void dump(list_t *lst){
    if (!lst->head) {
        printf("<empty list>\n");
        exit(0);
    }
    list_item_t *li = lst->head, *prev = NULL, *tmp;

    while (li != NULL) {
        list_item_t *next = (list_item_t*)(li->link ^ (uintptr_t)prev);
        printf("0x%lx 0x%lx %s\n", (uintptr_t)prev, li->link, (char*)li->data);
        tmp = li;
        li = next;
    }
}

```

```

        prev = tmp;
    }
}

void dump_in_file(list_t *lst, char *file) {
    FILE *f = fopen(file, "wb");

    if (!f) {
        perror("Ошибка: не удалось открыть файл");
        exit(1);
    }

    if (!lst->head) {
        fprintf(f, "<empty list>\n");
        fclose(f);
        exit(0);
    }

    list_item_t *li = lst->head, *prev = NULL, *tmp;
    while (li != NULL) {
        list_item_t *next = (list_item_t*)(li->link ^ (uintptr_t)prev);
        fprintf(f, "0x%lx 0x%lx %s\n", (uintptr_t)prev, li->link, (char*)li->data);
        tmp = li;
        li = next;
        prev = tmp;
    }
    fclose(f);
}

void delete_N(list_t *lst, int N) {
    if (!lst->head) {
        printf("<empty list>\n");
        exit(0);
    }
}

```

```

    }

    list_item_t *li = lst->head;
    list_item_t *prev = NULL;
    list_item_t *tmp;
    int li_id = 0;

    while (li != NULL && li_id < N) {
        tmp = li;
        li = (list_item_t*)(li->link ^ (uintptr_t)prev);
        prev = tmp;
        li_id++;
    }
    list_item_t *next = (list_item_t*)(li->link ^ (uintptr_t)prev);

    if (prev)prev->link ^= (uintptr_t)li ^ (uintptr_t)next;
    else lst->head = next;

    if (next) next->link ^= (uintptr_t)li ^ (uintptr_t)prev;
    else lst->tail = prev;

    free(li);
}

void exit_func(list_t *lst){
    FILE *file;
    file = fopen("exit.txt", "w");
    if (file == NULL){
        printf("Ошибка: не удалось создать файл");
        exit(1);
    }
}

```

```

int idx = 0;

list_item_t *li = lst->head, *pi = NULL, *tmp = NULL;

for (; li != NULL; ) {
    fprintf(file, "[%d] %s\n", idx, (char*)li->data);

    tmp = li;

    li = (list_item_t*)(li->link ^ (uintptr_t)pi);

    pi = tmp;

    idx++;
}

fprintf(file, "NULL\n");

fclose(file);
}

void def_commands(char **commands, size_t count) {
    for (size_t i = 0; i < count; i++) {

        char *cmd_copy = strdup(commands[i]);

        char *arg = strtok(cmd_copy, " ");

        if (!arg) continue;

        if (strcmp(arg, "pop_front") == 0) {
            printf("command: %s\n", arg);

            pop_front(&lst);

            print_list(&lst);
        }

        else if (strcmp(arg, "pop_back") == 0) {
            printf("command: %s\n", arg);

            pop_back(&lst);

            print_list(&lst);
        }
    }
}

```

```

}
else if (strcmp(arg, "delete") == 0) {
    printf("command: %s\n", arg);
    arg = strtok(NULL, " ");
    if (arg) {
        int flag = 1;
        for (char *i = arg; *i; i++) {
            if (!isdigit(*i)) {
                flag = 0;
                printf("Ошибка: неподходящий формат индекса\n");
                exit(1);
            }
        }
        if (flag) {
            int n = strtol(arg, NULL, 10);
            printf("список до удаления элемента [%d]\n", n);
            print_list(&lst);
            delete_N(&lst, n);
            printf("список после удаления элемента [%d]\n", n);
            print_list(&lst);
        }
    }
}
else if (strcmp(arg, "dump") == 0) {
    arg = strtok(NULL, " ");
    if (arg) {
        printf("command: %s\n", arg);
        dump_in_file(&lst, arg);
    }
}

```

```

else{
    printf("command: dump\n");
    dump(&lst);

}
}
else if (strcmp(arg, "push_front") == 0) {
    printf("command: %s\n", arg);
    char *datetime = strtok(NULL, "\"");
    if (datetime) {
        char *data = strdup(datetime);
        if (date(data)) {
            push_front(&lst, data);
            print_list(&lst);
        }
        else{
            printf("Ошибка: неподходящий формат данных\n");
            exit(1);
        }
    }
}
else if (strcmp(arg, "push_back") == 0) {
    printf("command: %s\n", arg);

    char *datetime = strtok(NULL, "\"");
    if (datetime) {
        char *data = strdup(datetime);
        if (date(data)) {
            push_back(&lst, data);

```

```

        print_list(&lst);
    }
    else{
        printf("Ошибка: неподходящий формат данных\n");
        exit(1);
    }
}
}
else if(strcmp(arg, "exit") == 0){
    printf("command: %s\n", arg);
    exit_func(&lst);
}
else {
    printf("Ошибка: неизвестная команда %s\n", arg);
    exit(1);
}

    free(cmd_copy);
}
}

int main(int argc, char *argv[]) {
    (void)argc;
    char *DEBUG = getenv("LAB4DEBUG");
    char *file = NULL;
    FILE *f = NULL;
    if (DEBUG){
        fprintf(stderr, "Включен вывод отладочных сообщений\n");
    }

```



```

if (strcmp(argv[1], "-v") == 0){
    printf("Анна Николаевна Киселёва, гр N3147\nВариант 3-8-3-5\n");
    exit(0);
}

file = argv[1];
f = fopen(file, "r");
if (!f){
    perror("Ошибка: не удалось открыть файл");
    exit(1);
}

char **commands = NULL;
size_t count_l = 0;
char *line = NULL;
size_t len = 0;
ssize_t read;

while ((read = getline(&line, &len, stdin)) != -1){

    if (line[read - 1] == '\n') {
        line[read - 1] = '\0';
        read--;
    }

    char **tmp = realloc(commands, (count_l + 1) * sizeof(char*));
    if (!tmp) {
        perror("Ошибка: не удалось выделить память");
        exit(1);
    }

    commands = tmp;
    commands[count_l] = strdup(line);

```

```

        count_l++;
    }
    size_t count = 0;
    char **data = data_in_file(f, &count);
    for(size_t i = 0; i < count; i++){
        push_front(&lst, data[i]);
    }
    print_list(&lst);
    def_commands(commands, count_l);

    //освобождение памяти и удаление данных
    free(line);
    for (size_t i = 0; i < count_l; i++){
        free(commands[i]);
    }
    free(commands);
    for (size_t i = 0; i < count; i++){
        free(data[i]);
    }
    free(data);
    delete(&lst);
    fclose(f);
    exit(0);
}

```

