Национальный исследовательский ядерный университет «МИФИ»
Институт интеллектуальных кибернетических систем
Кафедра №12 «Компьютерные системы и технологии»

ОТЧЕТ

О выполнении лабораторной работы №6

«Работа со структурами данных на основе списков»

Студент: Кругликова М. В.
Группа: Б22-504
Преподаватель: Комаров Т. И.

Москва – 2023

1. **Формулировка индивидуального задания**
   Вариант №15
   Осуществить циклический сдвиг на N символов влево в каждом слове строки
2. **Описание использованных типов данных**
   При выполнении данной лабораторной работы использовались такие типы данных, как:
   1. Встроенный тип данных int, предназначенный для работы с целыми числами
   2. Встроенный тип данных char, предназначенный для работы с символами
   3. Собственный тип данный Item, предназначенный для работы с структурами, содержащими поле для символа и адресное поле
   4. Собственный тип данных List, предназначенный для работы со списками из структур Item
3. **Описание использованного алгоритма**

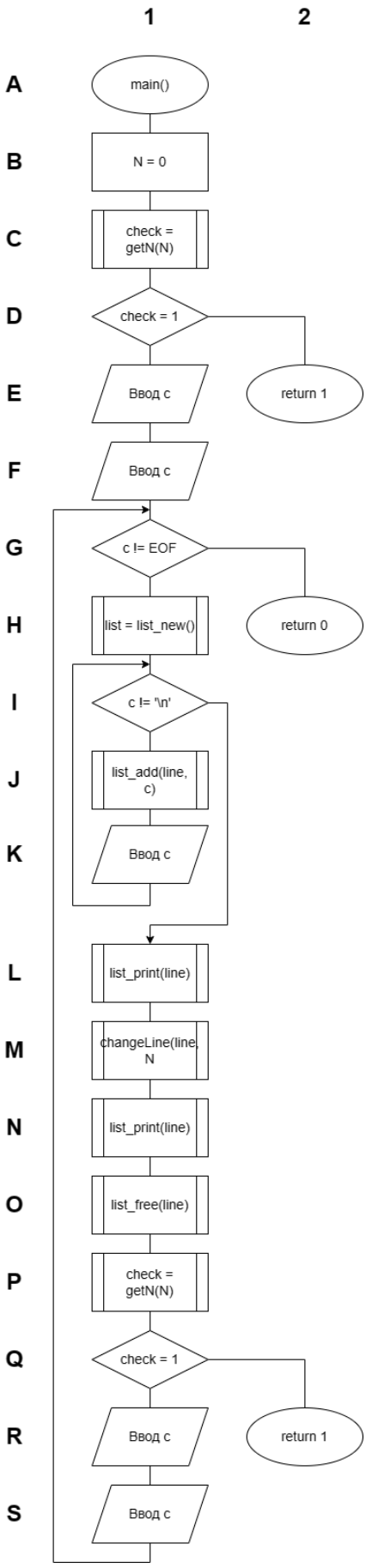

Рисунок 1. Блок-схема алгоритма работы функции main (файл main6.c)

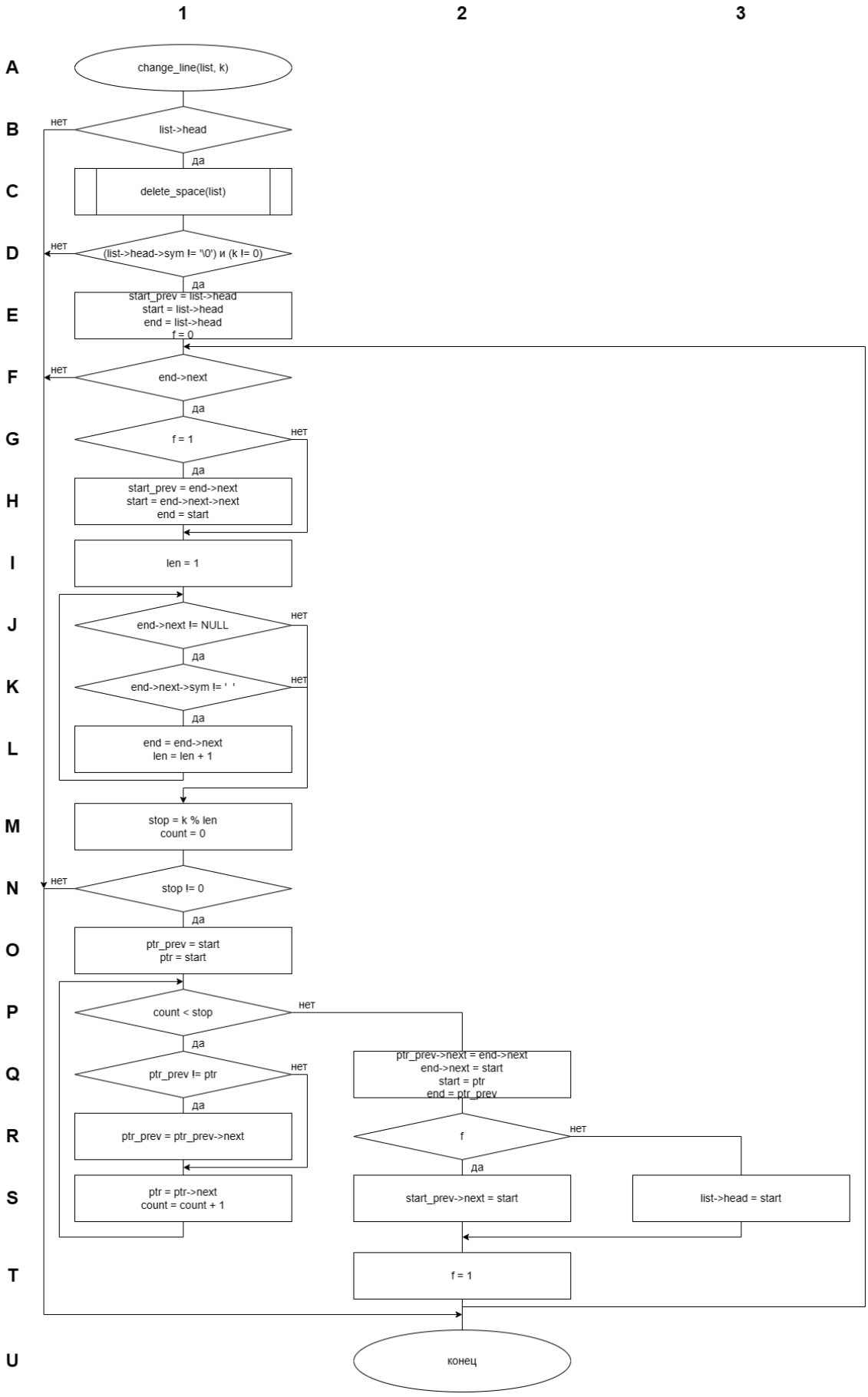

Рисунок 2. Блок-схема алгоритма работы функции change_line (файл changeLine.c)

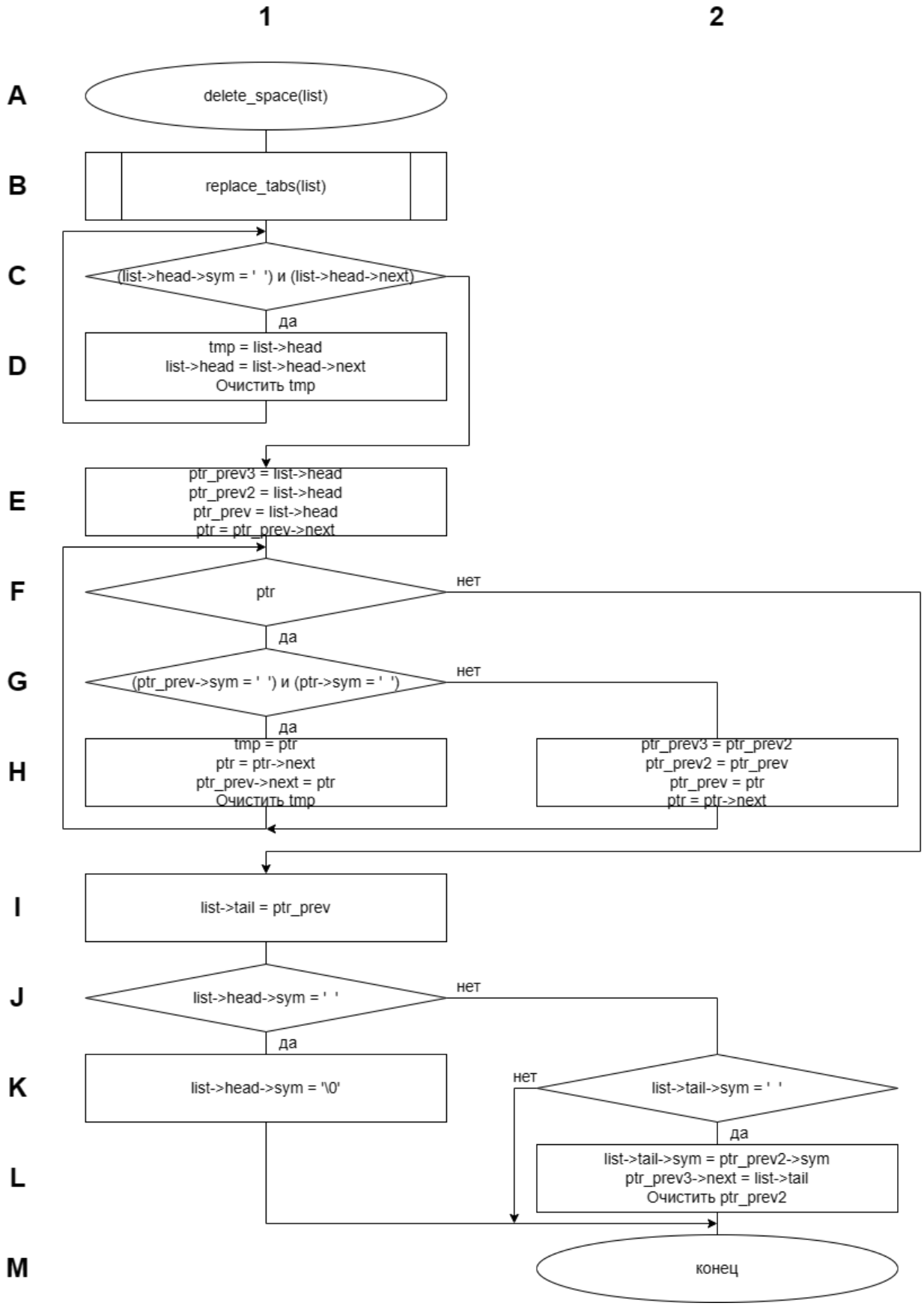

Рисунок 3. Блок-схема алгоритма работы функции delete_space (файл changeLine.c)

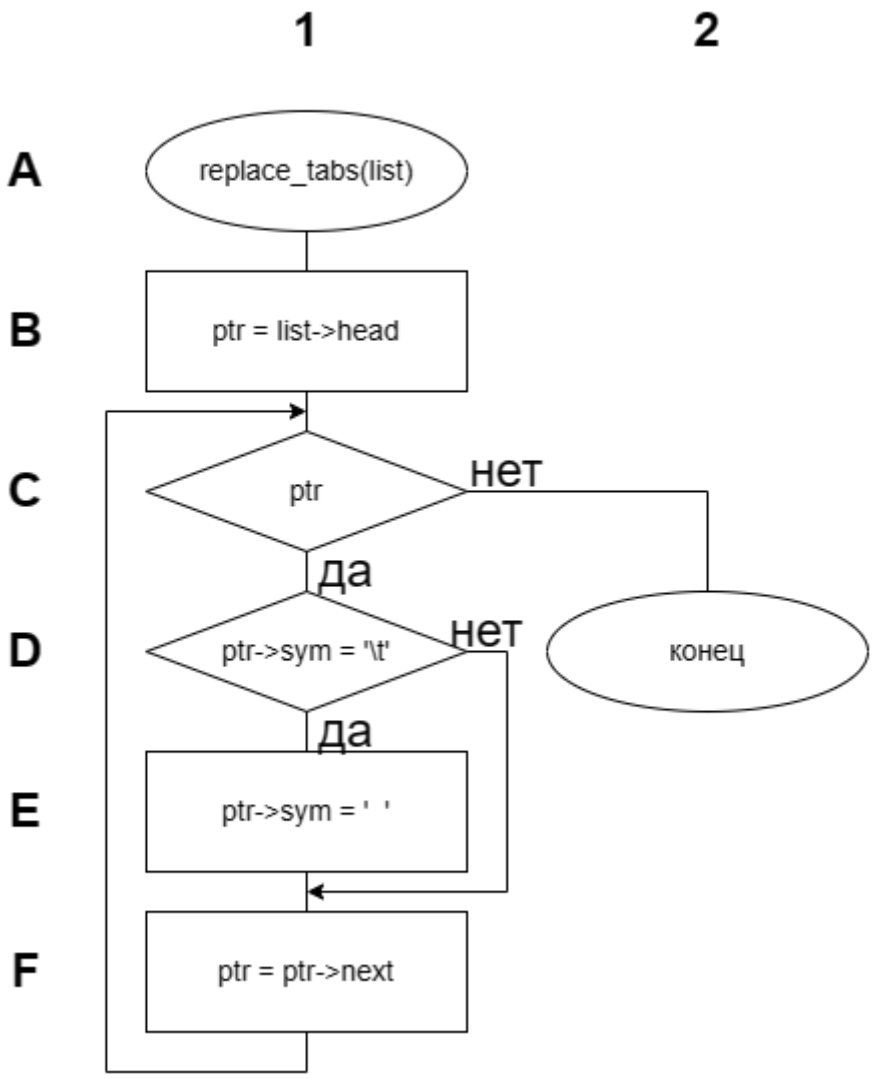

Рисунок 4. Блок-схема алгоритма работы функции replace_tabs (файл changeLine.c)

4. Исходные коды разработанных программ

Листинг 1. Исходные коды программы lab6 (файлы changeLine.c, changeLine.h, getN.c, getN.h, list.c, list.h, main6.c)

Файл changeLine.c

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"
#include "changeLine.h"

void change_line(List *list, int k) {
    if (list->head) {
        delete_space(list);
```

```
        if ((list->head->sym != '\0') && (k != 0))
{
            Item *start_prev = list->head;
            Item *start = list->head;
            Item *end = list->head;
            int f = 0;

            while (end->next) {
                if (f == 1) {
                    start_prev = end->next;
                    start = end->next->next;
                    end = start;
                }
                int len = 1;
                while (end->next != NULL) {
                    if (end->next->sym != ' ') {
                        end = end->next;
                        len++;
                    } else {
                        break;
                    }
                }

                int stop = k % len;
                int count = 0;
                if (stop != 0) {
                    Item *ptr_prev = start;
                    Item *ptr = start;
                    while (count < stop) {
                        if (ptr_prev != ptr) {
                            ptr_prev = ptr_prev-
>next;
                        }
                        ptr = ptr->next;
                        count++;
                    }
                    ptr_prev->next = end->next;

                    end->next = start;
                    start = ptr;
                    end = ptr_prev;
                    if (f) {
```

```
                        start_prev->next = start;
                    } else {
                        list->head = start;
                    }
                }

                f = 1;
            }
        }
    }
}

void delete_space(List *list) {
    replace_tabs(list);
    Item *tmp;
    while ((list->head->sym == ' ') && (list->head-
>next)) {
        tmp = list->head;
        list->head = list->head->next;
        free(tmp);
    }
    Item *ptr_prev3 = list->head;
    Item *ptr_prev2 = list->head;
    Item *ptr_prev = list->head;
    Item *ptr = ptr_prev->next;
    while (ptr) {
        if ((ptr_prev->sym == ' ') && (ptr->sym ==
' ')) {
            tmp = ptr;
            ptr = ptr->next;
            ptr_prev->next = ptr;
            free(tmp);
        } else {
            ptr_prev3 = ptr_prev2;
            ptr_prev2 = ptr_prev;
            ptr_prev = ptr;
            ptr = ptr->next;
        }
    }
    list->tail = ptr_prev;

    if (list->head->sym == ' ') {
```

```
            list->head->sym = '\0';
        } else if (list->tail->sym == ' ') {
            list->tail->sym = ptr_prev2->sym;
            ptr_prev3->next = list->tail;
            free(ptr_prev2);
        }
}

void replace_tabs(List *list) {
    Item *ptr = list->head;
    while(ptr) {
        if (ptr->sym == '\t') {
            ptr->sym = ' ';
        }
        ptr = ptr->next;
    }
}
```

Файл changeLine.h

```
#include "list.h"

#ifndef changeLine_H
#define changeLine_H

void change_line(List *list, int k);
void delete_space(List *list);
void replace_tabs(List *list);

#endif
```

Файл getN.c

```
#include <stdio.h>

int getN(int *n) {
    int ch = scanf("%d", n);
    while (ch != 1) {
        if (ch == EOF) {
            printf("Ошибка!\n");
            return 1;
        } else {
            printf("Некорректное значение. Введите
снова: ");
```

```
            scanf("%*[^\n]");
            ch = scanf("%d", n);
        }
    }
    return 0;
}
```

Файл getN.h

```
#ifndef getN_H
#define getN_H

int getN(int *n);

#endif
```

Файл list.c

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"

List *list_new() {
    return (List *) calloc(1, sizeof(List));
}

void list_print(const List *list) {
    Item *ptr = list->head;
    printf("\"");
    while (ptr) {
        printf("%c", ptr->sym);
        ptr = ptr->next;
    }
    printf("\"\n");
}

int list_add(List *list, char sym) {
    Item *new = (Item *) malloc(sizeof(Item));
    if (!new) {
        return 1;
    }
    new->sym = sym;
    new->next = NULL;
    if (!list->head) {
```

```
        list->head = new;
        list->tail = new;
    } else {
        list->tail->next = new;
        list->tail = new;
    }
    return 0;
}

int list_str(List **list, const char *str) {
    int ind = 0;
    while (str[ind] != '\0') {
        int ch = list_add((*list), str[ind]);
        if (ch == 1) {
            return 1;
        }
        ind++;
    }

    return 0;
}

void list_free(List *list) {
    Item *ptr = list->head;
    Item *ptr_prev;
    while(ptr) {
        ptr_prev = ptr;
        ptr = ptr->next;
        free(ptr_prev);
    }
    free(list);
}
```

Файл list.h

```
#ifndef LIST_H
#define LIST_H

typedef struct Item {
    char sym;
    struct Item *next;
} Item;
```

```
typedef struct List {
    Item *head;
    Item *tail;
} List;

List *list_new();

void list_print(const List *list);

int list_add(List *list, char sym);

int list_str(List **list, const char *str);

void list_free(List *list);

#endif
```

Файл main6.c

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"
#include "changeLine.h"
#include "getN.h"

int main() {
    int N = 0;
    printf("N = ");
    int check = getN(&N);
    if (check == 1) {
        return 1;
    }
    printf("Введите строку: ");
    char c = getchar();
    c = getchar();
    while (c != EOF) {
        List *line = list_new();
        while (c != '\n') {
            list_add(line, c);
            c = getchar();
        }
        printf("Введенная строка: ");
        list_print(line);
```

```
            change_line(line, N);
            printf("Измененная строка: ");
            list_print(line);
            list_free(line);

            printf("N = ");
            int check = getN(&N);
            if (check == 1) {
                return 1;
            }
            printf("Введите строку: ");
            c = getchar();
            c = getchar();
        }
        return 0;
    }
```

5. Описание тестовых примеров

Таблица 1. Тестовые примеры работы программы lab6

N	Введенная строка	Измененная строка (ожидание)	Измененная строка (реальность)
5	пустая	пустая	пустая
1	“ “	пустая	пустая
0	“ fdlksgj 4589yu j 34 “	“fdlksgj 4589yu j 34”	“fdlksgj 4589yu j 34”
3	“ kjh er qwjrter 5 “	“kjh re rterqwj 5”	“kjh re rterqwj 5”

При вводе ctrl+d программа завершается.

6. Скриншоты

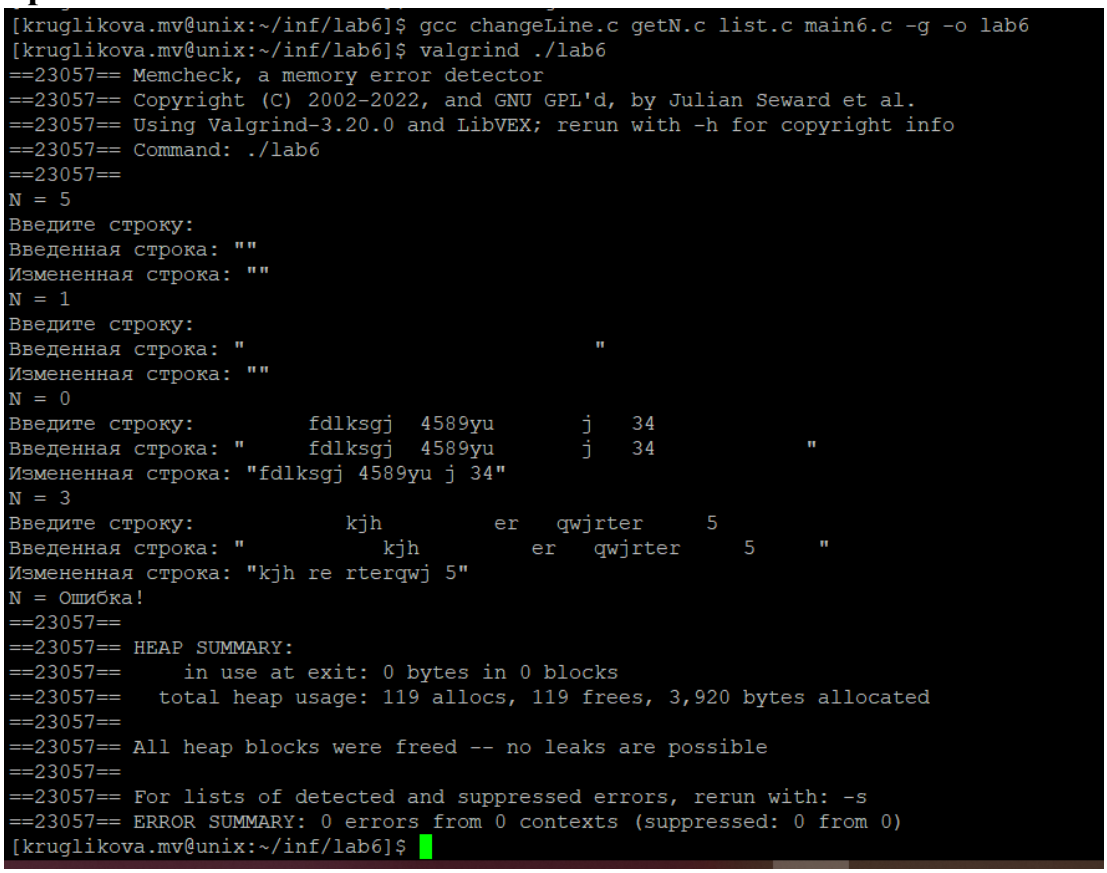

Рисунок 5. Запуск программы lab6

7. Выводы

В ходе данной лабораторной работы был изучен принцип представления строк в виде списков на физическом уровне. Также были изучены основы работы со списками и написаны несколько вспомогательных функций для упрощения работы с ними.