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: Гатченко А.С.

Группа: Б22-525

Преподаватель: Половнева Ю. А.

Москва – 2022

1. Формулировка индивидуального задания

К каждому слову строки, начинающемуся на согласную букву, добавить некоторый константный префикс.

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовались встроенные типы данных int и char, предназначенные для работы с целыми числами, символами и строками, а также структуры данных.

3. Описание использованного алгоритма

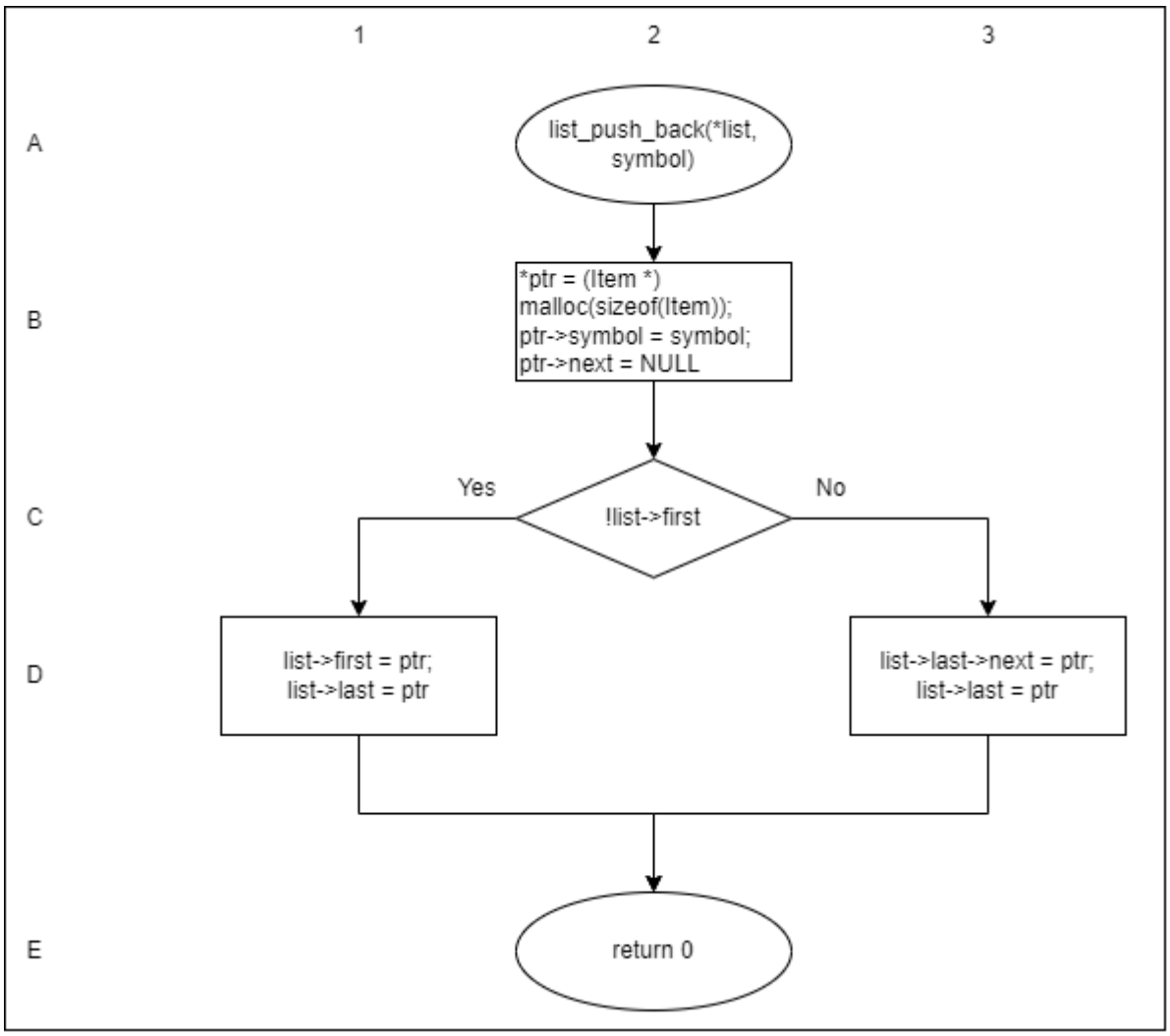
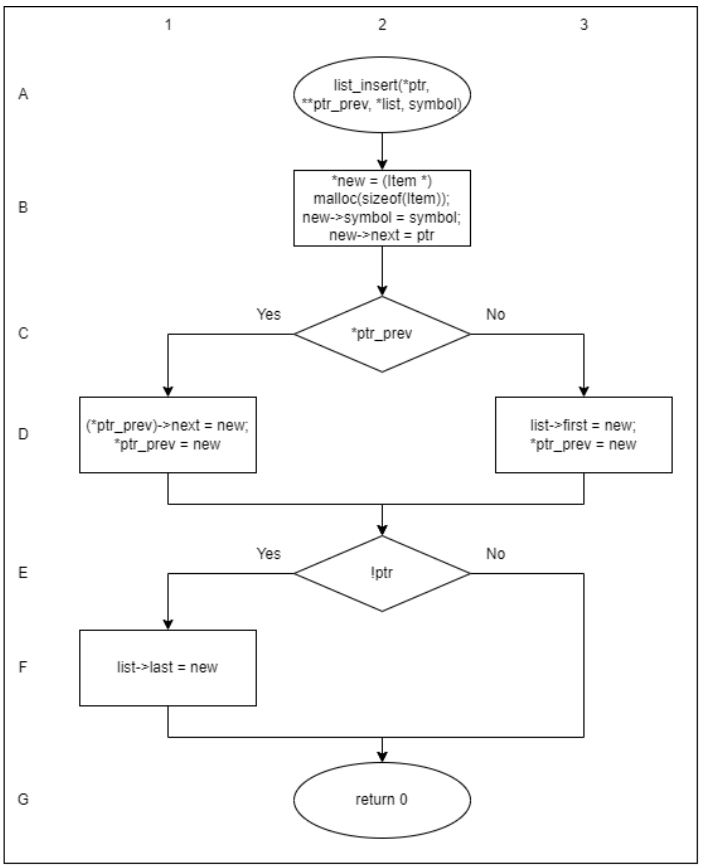


Рис. 1: Блок-схема алгоритма работы функции `list_push_back()`



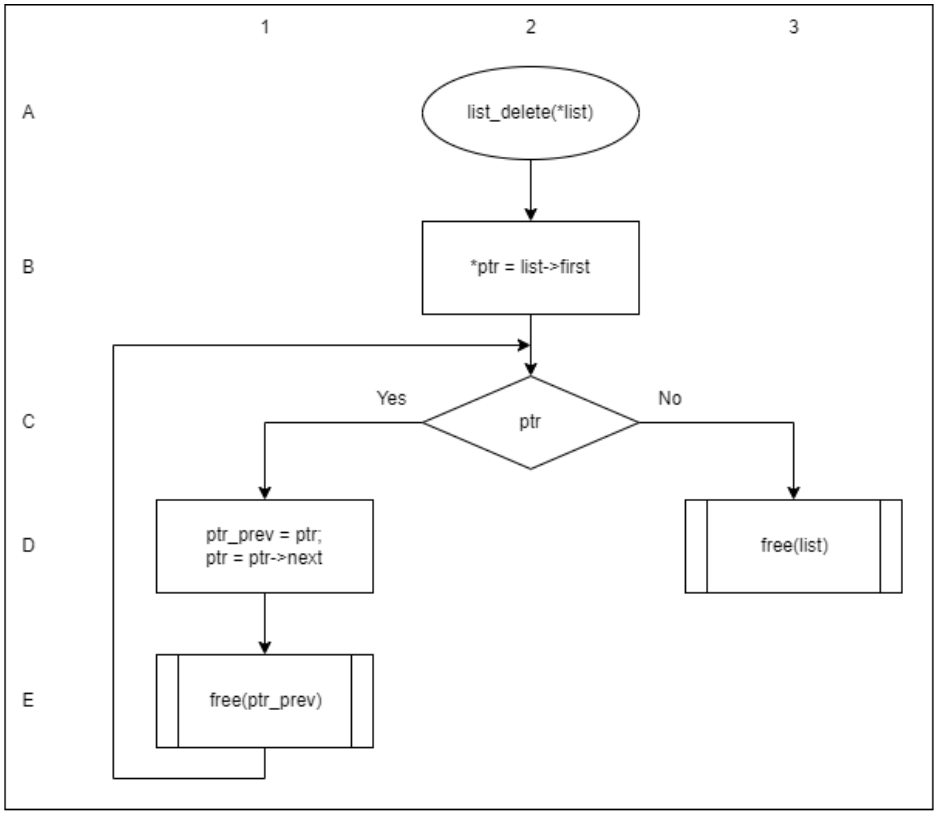
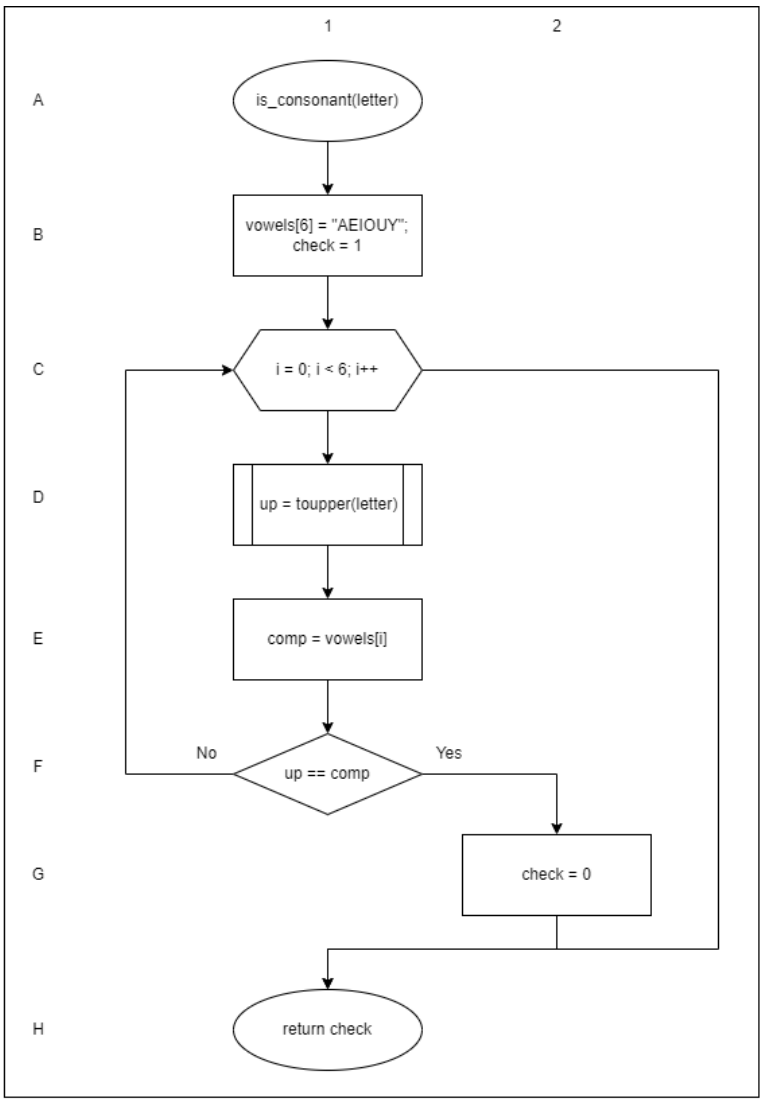


Рис. 2: Блок-схема алгоритма работы функции `list_insert()`

Рис. 3: Блок-схема алгоритма работы функции `list_delete()`



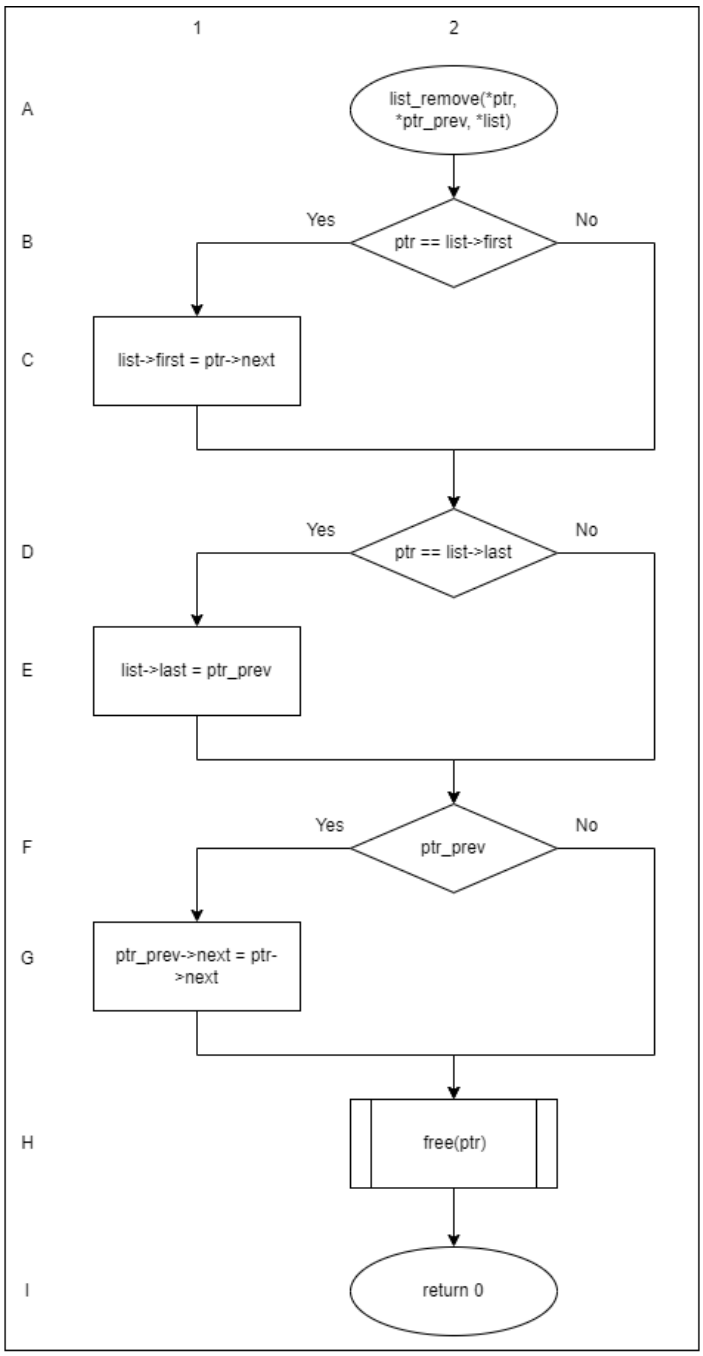


Рис. 4: Блок-схема алгоритма работы функции `is_consonant()`

Рис. 5: Блок-схема алгоритма работы функции `list_remove()`

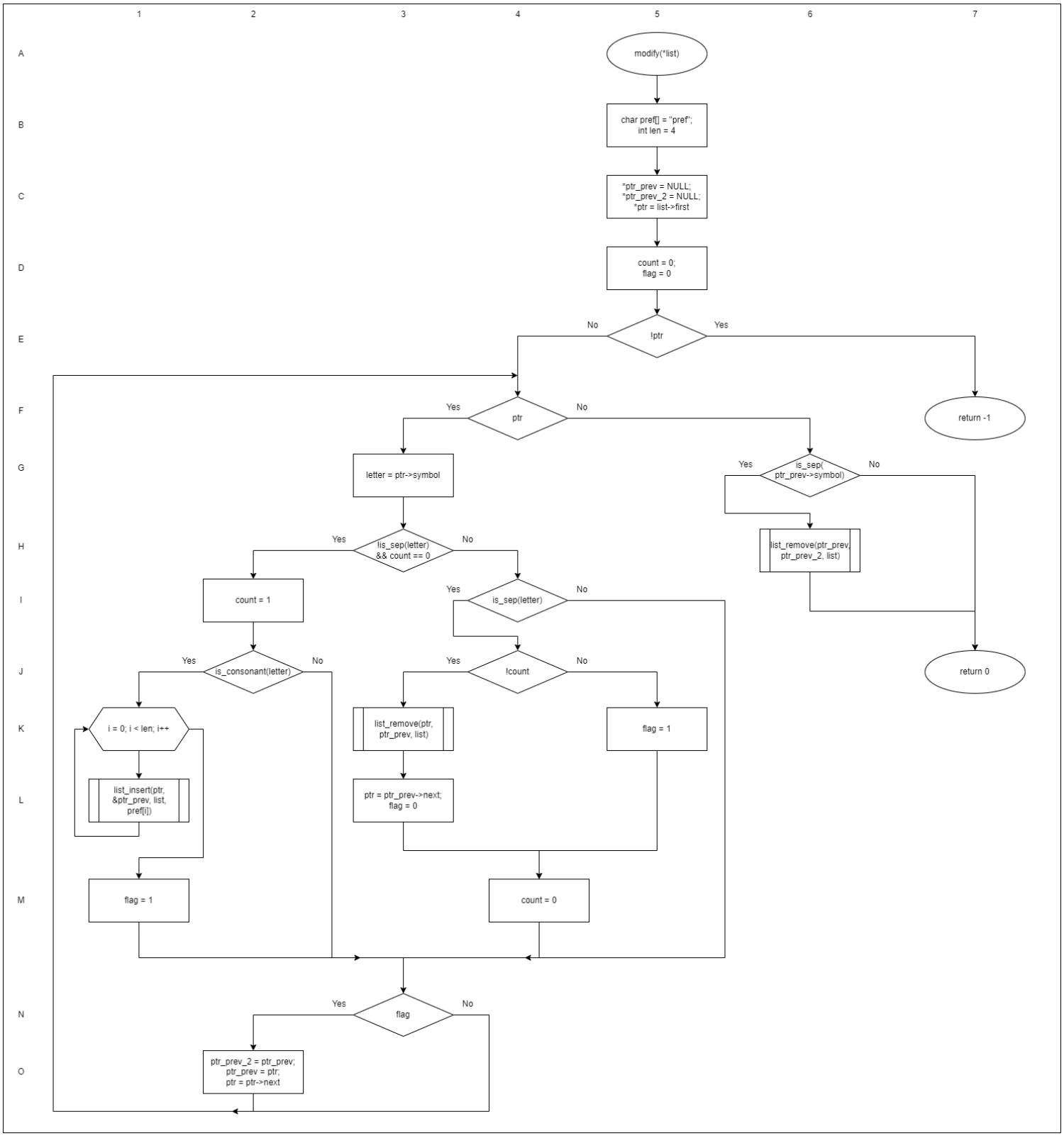


Рис. 6: Блок-схема алгоритма работы функции modify ( )

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы lab6

1) Файл: prog.c

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

int input(List *list){
    printf("Input some words:\n");
    int letter = getchar();

    while (letter == ' ' || letter == '\t'){
        letter = getchar();
    }

    while (letter != '\n' && letter != EOF){
        list_push_back(list, letter);
        letter = getchar();
    }
    if (letter == EOF){
        return -1;
    }
    return 0;
}

int main(){
    List *list = list_new();
    int check = input(list);

    while (check != -1){
        printf("Initialized list:\n");
        list_print(list);

        printf("New list:\n");
        modify(list);
        list_print(list);

        list_delete(list);
        list = list_new();
        check = input(list);
    }

    list_delete(list);
    return 0;
}
```

2) Файл: list.c

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


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

void list_print(const List *list){
    Item *ptr = list->first;
    printf("'");
```

```
    while (ptr){
        printf("%c", ptr->symbol);
        ptr = ptr->next;
    }
    printf("'");
    printf("\n");
}

int list_push_back(List *list, char symbol){
    Item *ptr = (Item *) malloc(sizeof(Item));
    ptr->symbol = symbol;
    ptr->next = NULL;

    if (!list->first){
        list->first = ptr;
        list->last = ptr;
    }
    else {
        list->last->next = ptr;
        list->last = ptr;
    }
    return 0;
}

int list_insert(Item *ptr, Item **ptr_prev, List *list, const char symbol){
    Item *new = (Item *) malloc(sizeof(Item));
    new->symbol = symbol;
    new->next = ptr;

    if (*ptr_prev){
        (*ptr_prev)->next = new;
        *ptr_prev = new;
    }
    else {
        list->first = new;
        *ptr_prev = new;
    }
    if (!ptr){
        list->last = new;
    }
    return 0;
}

int list_insert_chunk(Item *ptr, Item **ptr_prev, List *list, char *text, int
len){
    List *new_list = list_new();

    for (int i = 0; i < len; i++){
        list_push_back(new_list, text[i]);
    }

    if (!(*ptr_prev)){
        *ptr_prev = new_list->first;
        list->first = *ptr_prev;
    }
    else {
        (*ptr_prev)->next = new_list->first;
    }

    new_list->last->next = ptr;
    *ptr_prev = new_list->last;
    return 0;
}
```

```
int list_remove(Item *ptr, Item *ptr_prev, List *list){
    if (ptr == list->first){
        list->first = ptr->next;
    }
    if (ptr == list->last){
        list->last = ptr_prev;
    }
    if (ptr_prev){
        ptr_prev->next = ptr->next;
    }
    free(ptr);
    return 0;
}

int is_sep(const char letter){
   return (letter == ' ' || letter == '\t');
}

int is_consonant(const char letter){
    char vowels[6] = "AEIOUY";
    int check = 1;
    for (int i = 0; i < 6; i++){
        char up = toupper(letter);
        char comp = vowels[i];

        if (up == comp){
            check = 0;
            break;
        }
    }
    return check;
}

int modify(List *list){
    char pref[] = "pref";
    int len = 4;
    Item *ptr_prev = NULL;
    Item *ptr_prev_2 = NULL;
    Item *ptr = list->first;
    int count = 0;
    int flag = 0;
    char letter;
    if (!ptr){
        return -1;
    }
    while (ptr){
        letter = ptr->symbol;
        if (!is_sep(letter) && count == 0){
            count = 1;
            if (is_consonant(letter)){
                #ifdef chunk
                list_insert_chunk(ptr, &ptr_prev, list, pref, len);
                #else
                for (int i = 0; i < len; i++){
                    list_insert(ptr, &ptr_prev, list, pref[i]);
                }
                #endif
            }
            flag = 1;
        }
        else if (is_sep(letter)){
            if (!count){
```

```
                list_remove(ptr, ptr_prev, list);
                ptr = ptr_prev->next;
                flag = 0;
            }
            else {
                flag = 1;
            }
            count = 0;
        }
        if (flag){
            ptr_prev_2 = ptr_prev;
            ptr_prev = ptr;
            ptr = ptr->next;
        }
    }
    if (is_sep(ptr_prev->symbol)){
        list_remove(ptr_prev, ptr_prev_2, list);
    }
    return 0;
}

void list_delete(List *list){
    Item *ptr = list->first, *ptr_prev;
    while (ptr){
        ptr_prev = ptr;
        ptr = ptr->next;
        free(ptr_prev);
    }
    free(list);
}
```

3) Файл: list.h

```
#ifndef LIST_H
#define LIST_H

typedef struct Item {
    char symbol;
    struct Item *next;
} Item;

typedef struct List {
    Item *first;
    Item *last;
} List;

List *list_new();
void list_print(const List *list);
int modify(List *list);
int list_push_back(List *list, char symbol);
void list_delete(List *list);

#endif
```

5. Описание тестовых примеров

Таблица 1: Тестовые примеры lab6

Введенная строка	Ожидаемый результат после обработки	Полученный результат после обработки
' wejkfhwuhwijfkj ik a ldwak ng kesjkfn kj n g efsef ht'	'prefwejkfhwuhwijfkj ik a prefldwak prefng prefkesjkfn prefkj prefn prefg efsef prefht'	'prefwejkfhwuhwijfkj ik a prefldwak prefng prefkesjkfn prefkj prefn prefg efsef prefht'
'aaaaaa iiiiii k k klklakd w wa wadfr d '	'aaaaaa iiiiii prefk prefk prefklklakd prefw prefwa prefwadfr prefd'	'aaaaaa iiiiii prefk prefk prefklklakd prefw prefwa prefwadfr prefd'
'jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj'	'prefjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj'	'prefjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj'
'uoooooooouoooooooooooooooooooooooooooooooooooou'	'uoooooooouoooooooooooooooooooooooooooooooooooou'	'uoooooooouoooooooooooooooooooooooooooooooooooou'

6. Скриншоты

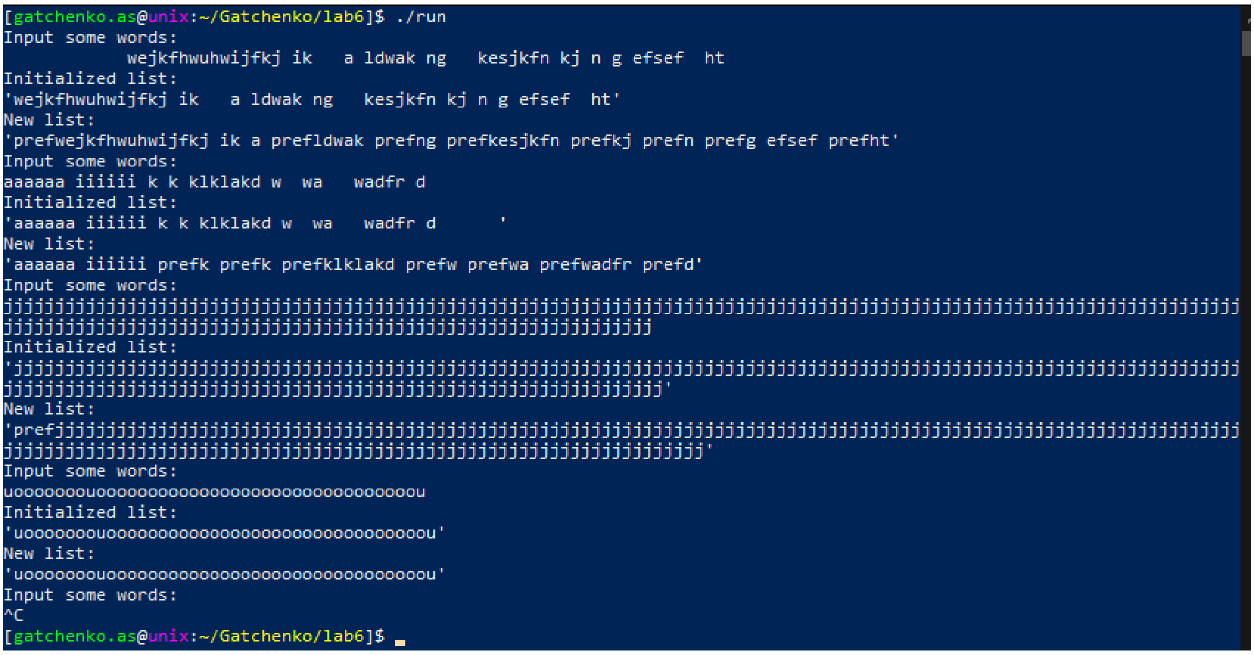


Рис. 7: Работа программы lab6

7. Выводы

В ходе выполнения данной работы на примере программы, выполняющей обработку строк на основе списков структур данных, были рассмотрены базовые принципы работы построения программ на языке С, обработки строк и использования структур:

1. Организация ввода/вывода, а также проверка корректности ввода.
2. Разработка функций.
3. Объявление и использование переменных.
4. Выполнение простейших арифметических операций над целочисленными и дробными операндами.
5. Использование циклов и условий.
6. Использование структур данных.
7. Обработка строк.
8. Использование указателей (параметров) на структуры данных.
9. Разбиение программы на несколько файлов.
10. Работа с памятью.