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: Кукса И. В..
Группа: Б23-503
Преподаватель: Бабалова И. Ф.

Москва — 2023

1. Формулировка индивидуального задания

Сгруппировать слова в строке таким образом, чтобы в начале строки были представлены группы слов, состоящих из одинаковых символов, а затем — все прочие. При сравнении символов регистр букв игнорируется, но при формировании выходной строки он сохраняется. Например, строка «Listen cat listen sIlent tac dog» преобразуется в строку «Listen listen sIlent cat tac dog».

2. Описание использованных типов данных

Необходимо спроектировать и разработать на языке С программу, осуществляющую обработку строковых данных, на физическом уровне представленных в виде списков символов.

Из входного потока вводится произвольное количество строк произвольной длины. Каждая строка в общем случае содержит одно или более слов, разделенных пробелами и/или знаками табуляции. Завершение ввода определяется концом файла.

Каждая выходная строка формируется путем модификации исходной строки в соответствии с требованиями, предъявляемыми индивидуальным заданием. В полученной строке слова разделяются только одним пробелом. Исходная и полученная строки выводятся в кавычках на экран.

Примечания:

1. Каждая строка представлена списком. Элементы списка имеют по два поля, первое из которых содержит символ, а второе — указатель на следующий элемент списка или NULL. При желании возможно использование двусвязного списка.

2. Выходная строка должна формироваться путем модификации исходной строки (т.е. путем модификации исходного списка, без создания нового).

3. Ввод строк должен быть организован с помощью функции `getchar()`, каждый считываемый из входного потока символ должен сразу добавляться в формируемый список.

4. Логически законченные части алгоритма решения задачи должны быть оформлены в виде отдельных функций с параметрами. Использование глобальных переменных не допускается.

5. Программа должна корректным образом работать с памятью, для проверки необходимо использовать соответствующие программные средства, например: valgrind (при тестировании и отладке программы её необходимо запускать командой вида `valgrind ./lab6`).

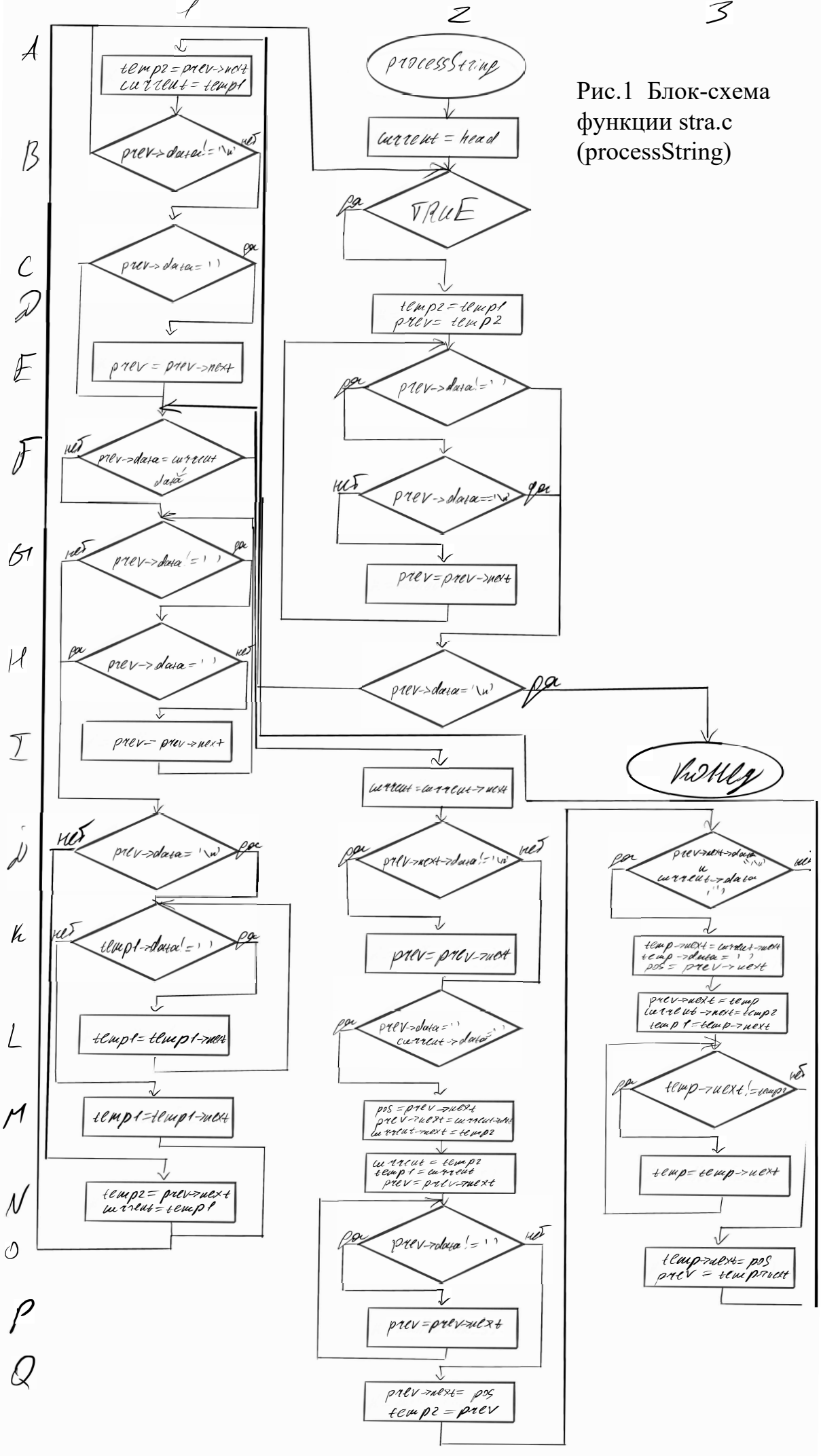


Рис.1 Блок-схема функции stra.c (processString)

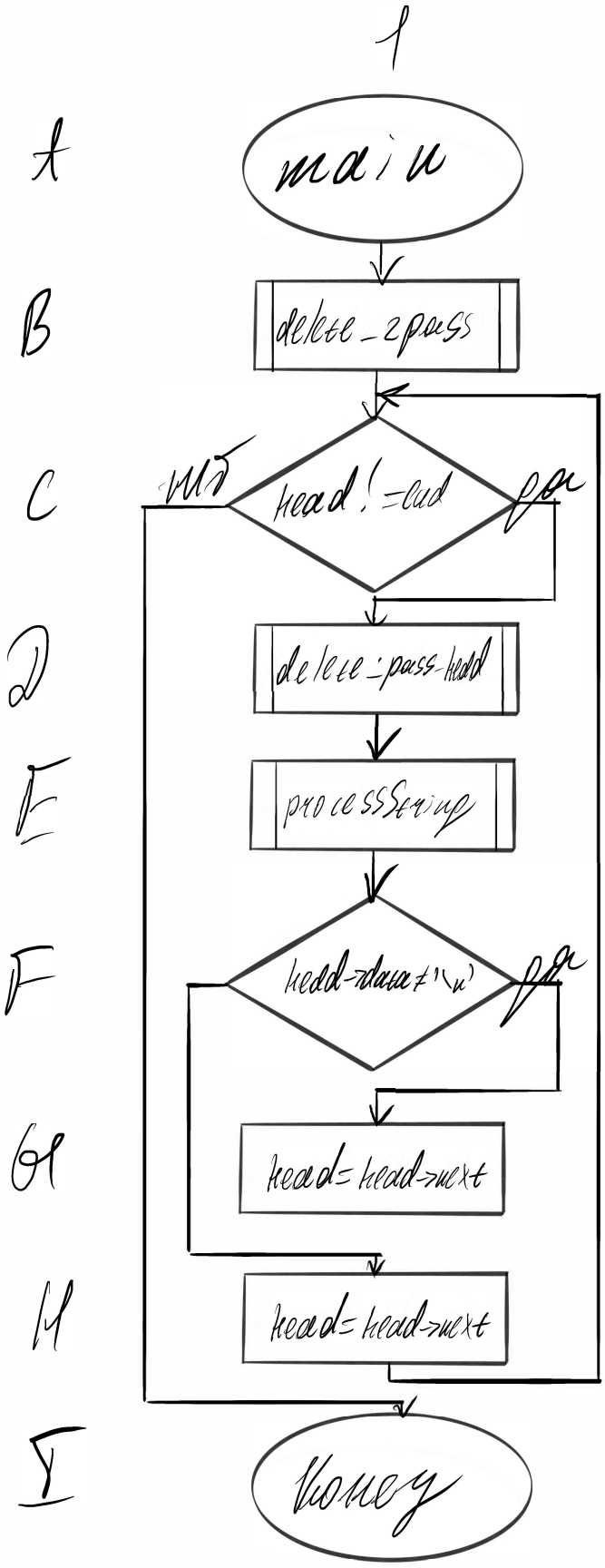
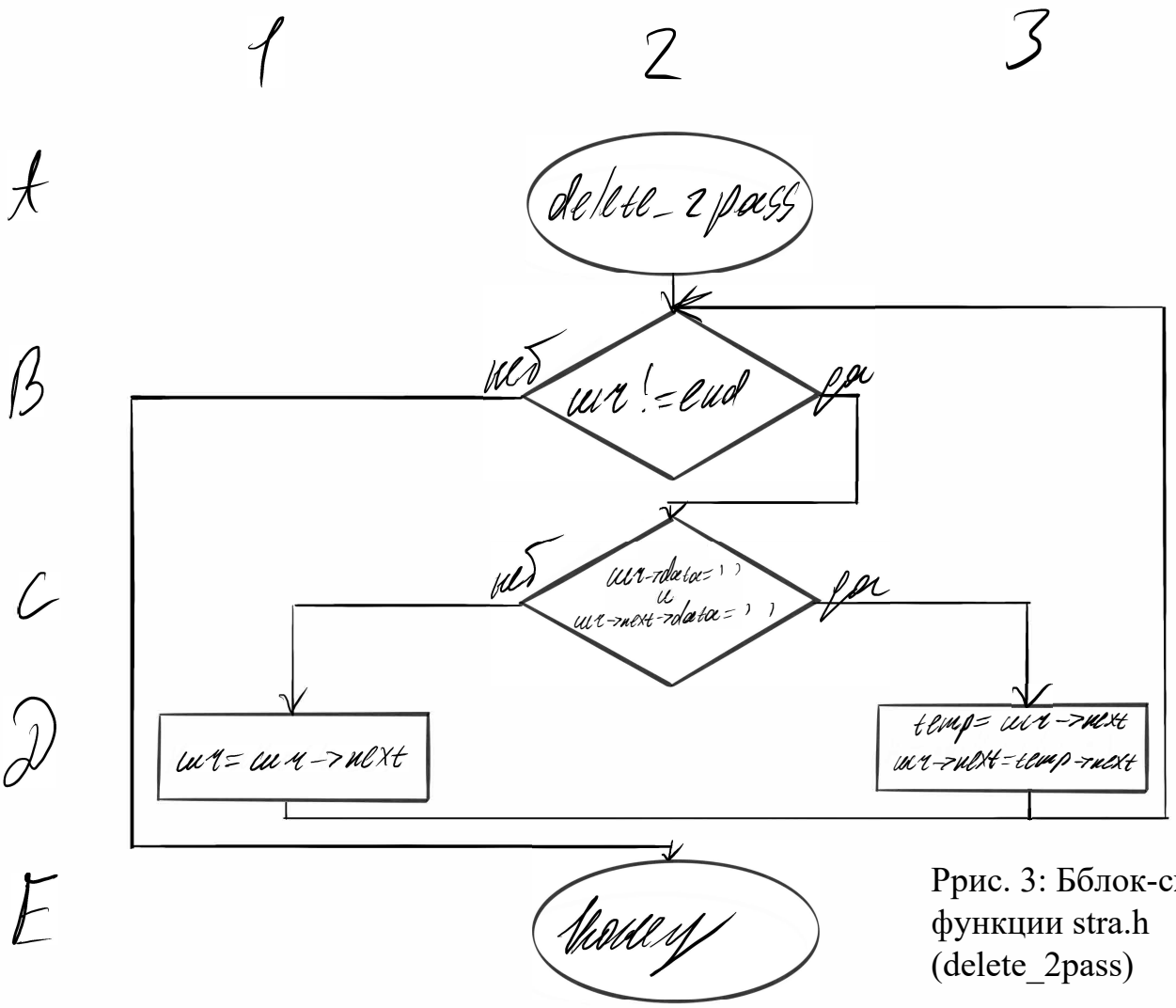
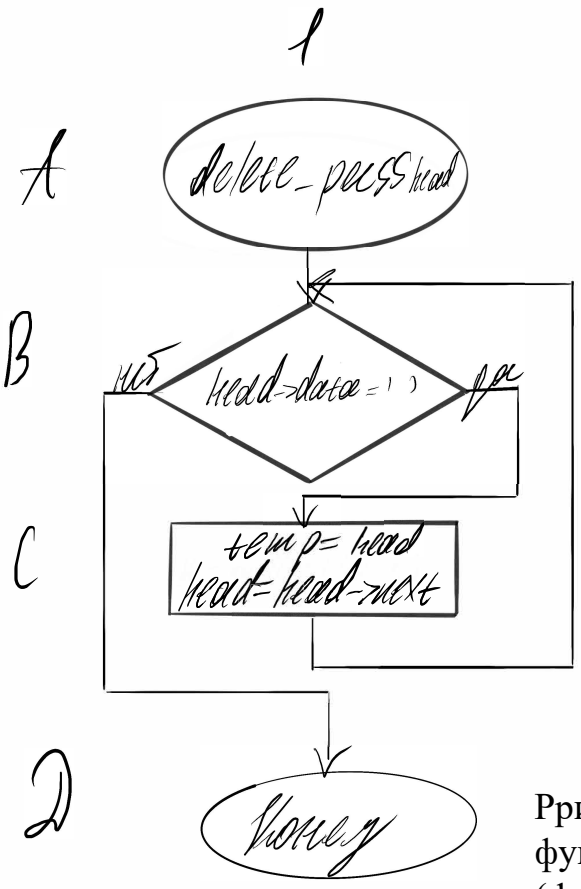


Рис. 2: Блок-схема функции lab6.c (main)



Ррис. 3: Бблок-схема функции stra.h (delete_2pass)



Ррис. 4: Бблок-схема функции stra.h (delete_head_pass)

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы Lab6 (файл: lab6.c)

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



int main() {
    Node* head = NULL;
    Node* current = NULL;
    char ch;

    while ((ch = getchar()) != EOF) {
        Node* newNode = createNode(ch);
        if (head == NULL) {
            head = newNode;
            current = newNode;
        } else {
            current->next = newNode;
            newNode->prev=current;
            current = newNode;
        }
    }

    List* list=malloc(sizeof(List)) ;
    list->head=head;
    Node* ss = NULL;
    while(head!=NULL){
        list->head=head;
        printf("\"");
        processString(list);


        printList(list);
        printf("\"\n");
        head=list->head;
        while(head->data!='\n'){
            ss=head;
            head=head->next;
            free(ss);
        }
            ss=head;
            head=head->next;
            head->prev=NULL;
            free(ss);
    }

    free(list);

}
```

Листинг 2: Исходные коды программы Lab6 (файл: stra.h)

```h
#ifndef _LIB_ARRAY_
#define _LIB_ARRAY_
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>


typedef struct Node {
    char data;
    struct Node* next;
    struct Node* prev;
} Node;


typedef struct List{
    Node *head;
} List;

Node* createNode(char data) ;
void printList(List* list) ;
void new_list(List* f,Node* head);
void processString(List* list);




#endif
```

Листинг 3: Исходные коды программы Lab6 (файл: stra.c)

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

Node* createNode(char data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}


void printList(List* list) {
    Node* head =list->head;
    Node* current = head;
    while (current != NULL) {
        if(current->data=='\n'){
            break;
        }
        printf("%c", current->data);
        current = current->next;
    }

}
void new_list(List* f,Node* head){
    Node* t=NULL;
    t=f->head;

    while(t!=NULL){
        Node* r=t;
        t=t->next;
        free(r);
    }


    Node* c=NULL;
    Node* h=NULL;



    while (( head->data!=' ') && head->data!='\n'){
        if(head==NULL){
            break;
        }
        Node* n=createNode(head->data);
        if(h==NULL){
            h=n;
            c=n;


        }else{
        c->next=n;
        n->prev=c;
        c=c->next;



        }
        head=head->next;


    }

    f->head=h;
}

```

```
void processString(List* list) {
    Node* head=list->head;
    Node* current = head;




    while (head != NULL && (head->data == ' ' || head->data == '\t')) {
        Node* temp = head;
        head = head->next;
        head->prev=NULL;
        free(temp);
    }
    list->head=head;

    current = head;
    while (current != NULL) {
        if ((current->data == ' ' || current->data == '\t') && (current->next != NULL && ((current->next)->data == ' ' || current->next->data == '\t'))) {
            Node* temp = current->next;
            current->next = temp->next;
            temp->next->prev=current;
            free(temp);
        } else {
            current = current->next;
        }
    }


    head=list->head;
    List* f=malloc(sizeof(List));
    Node* h=malloc(sizeof(Node));
    h->next=NULL;
    f->head=h;
    new_list(f,head);


    Node* pr1=head;
    while(head->data!=' '){
        head=head->next;
    }
    head=head->next;
    List* f2=malloc(sizeof(List));
    Node* h2=malloc(sizeof(Node));
    h2->next=NULL;
    f2->head=h2;
    new_list(f2,head);


    char AA[]="qwertyuiopasdfghjklzxcvbnm";
    int ll=strlen(AA);
    int *a1=malloc(sizeof(int)*ll);
    int *a2=malloc(sizeof(int)*ll);

    for (int i=0;i<ll;i++){
        a1[i]=0;
        a2[i]=0;

    }
    Node* ht1=NULL;
    Node* ht2=NULL;
    ht1=f->head;
    ht2=f2->head;
    int fl=0;


    int flag=0;



    while(pr1->data!='\n'){
            for (int i=0;i<ll;i++){

                a1[i]=0;

    }
        while(ht1!=NULL){
            for (int i=0;i<ll;i++){
                if(ht1->data==AA[i]){
                    a1[i]+=1;
                    break;
                }

            }
        }
```

```
        ht1=ht1->next;
    }


    while(f2->head->data!='\n'){
        for (int i=0;i<ll;i++){

            a2[i]=0;

}
        while(ht2!=NULL){
        for (int i=0;i<ll;i++){
            if(ht2->data==AA[i]){
                a2[i]+=1;
                break;
            }

        }
        ht2=ht2->next;
    }

    for (int i=0;i<ll;i++){
        if(a1[i]!=a2[i]){
            fl=1;
            break;
        }
    }
```

```
        Node* tr=NULL;
        if(fl!=1){
            Node* hh2=NULL;
            hh2=f2->head;
            while (hh2!=NULL){
                printf("%c",hh2->data);
                hh2=hh2->next;
            }
            printf(" ");

            flag=1;
                for (int i=0;i<ll;i++){

                a2[i]=0;

    }




        tr=head;
        tr=tr->prev;
        while(head->data!=' ' && head->data!='\n'){
            Node* ffr=NULL;
            ffr=head;
            head=head->next;
            free(ffr);
        }
        if(head->data!='\n'){
        Node* ffr=NULL;
        ffr=head;
        head=head->next;
        free(ffr);
        tr->next=head;
        head->prev=tr;
        }else{
            tr=tr->prev;
            free(tr->next);
            tr->next=head;
            head->prev=tr;
        }
        }else{
            while(head->data!=' ' && head->data!='\n'){
                head=head->next;
            }
            if(head->data!='\n'){
            head=head->next;
            }
        }
        fl=0;
        if(head->data=='\n'){
            if(flag==1){

            Node* hh1=NULL;
            hh1=f->head;
            while (hh1!=NULL){
                printf("%c",hh1->data);
                hh1=hh1->next;
            }
            printf(" ");
            flag=0;


            Node* ttr=NULL;
            ttr=pr1;
            ttr=ttr->prev;
            while(pr1->data!=' ' && pr1->data!='\n'){
            Node* ffr=NULL;
            ffr=pr1;
            pr1=pr1->next;
            free(ffr);
        }

        if(pr1->data!='\n'){
        Node* fffr=NULL;
        fffr=pr1;
        pr1=pr1->next;
        free(fffr);


        }
        if(ttr!=NULL){
```

```
        ttr->next=pr1;
        pr1->prev=ttr;
        }else{
            pr1->prev=NULL;
        }


            }else{
                while(pr1->data!=' ' && pr1->data!='\n'){
                    pr1=pr1->next;
                }
                if(pr1->data!='\n'){
                    pr1=pr1->next;
                }
            }

            break;
        }


        new_list(f2,head);

        ht2=f2->head;

        }
        if(pr1->data=='\n'){
            break;
        }
    new_list(f,pr1);

    ht1=f->head;

    head=pr1;
    while (head->data!=' ' && head->data!='\n'){
        head=head->next;
    }
    if(head->data=='\n'){

        break;
    }else{
        head=head->next;
    }

    new_list(f2,head);

    ht2=f2->head;

    }


    while(pr1->prev!=NULL){
            pr1=pr1->prev;
        }
    list->head=pr1;


    Node* k=NULL;
    k=f->head;
    while (k!=NULL){
        Node* y=k;
        k=k->next;
        free(y);
    }
    free(f);
    k=f2->head;
    while(k!=NULL){
        Node* u=k;
        k=k->next;
        free(u);
    }
    free(f2);
    free(a1);
    free(a2);



}
```

```
[kuksa.iv@unix lab6]$ gcc stra.c lab6.c -o lab6
```

```
tt ttt tt
rr rrr rr
yu iy uy
"tt tt ttt"
"rr rr rrr"
"uy yu iy"
```

Запуск и сборка программы lab6

```
==29011== HEAP SUMMARY:
==29011==     in use at exit: 0 bytes in 0 blocks
==29011==   total heap usage: 14 allocs, 14 frees, 2,240 bytes allocated
==29011==
==29011== All heap blocks were freed -- no leaks are possible
==29011==
==29011== For lists of detected and suppressed errors, rerun with: -s
==29011== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
tt yy yyh TT
"TT tt yy yyh"
```

```
uyuyu oooT jddcj ioio uuu iioo ToOo
"ToOo oooT iioo ioio uyuyu jddcj uuu"
```

```
emfeopiwsmf fepsofp nn sofejp posefpo NNomsefpo
"emfeopiwsmf fepsofp nn sofejp posefpo NNomsefpo"
```

```
TTTTTTT hds shD ff ff gg gg hds
"shD hds hds ff ff gg gg TTTTTTT "
```

7. Выводы

В ходе выполнения данной работы на примере программы, выполняющей работу со списками были реализованы следующие моменты:

1. Список символов
2. Ввод списка символов
3. Обработка списка символов (удаление пробелов в начале, 2-ых пробелов в середине, перемещение и сравнение слов)