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: Титов Иван Андреевич.

Преподаватель: Уваров М.П.

Группа: Б23-901

г. Москва – 2023

I. **Подготовительная часть**

Лабораторная работа № 6.1 «Работа со структурами данных на основе списков»

Вариант №34

Введение

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`).

Отчёт о выполнении лабораторной работы должен включать:

1. Блок-схемы основных алгоритмов работы программы.
2. Исходные коды программы.
3. Тестовые наборы для программы.
4. Выводы.

Индивидуальное задание

Удалить из строки слова, содержащие хотя бы один из N наиболее часто встречающихся в строке непробельных символов.

Цель: организовать программу, отвечающую ряду требований и обладающую необходимым функционалом.

Необходимый функционал:

1. Ввод строки
2. Ввод параметра N числа самых часто встречающихся в строке символов

3. Обработка строки следующим образом: а)нахождение N самых часто встречающихся символов строки, б)нахождение слова, в которых присутствуют эти N самых часто встречающихся символов строки, в)удаление слова, отвечающего вышеуказанной характеристике
4. Вывод в терминал обработанной строки

Требования к реализации:

1. Строка на физическом уровне есть список символов. Конкретнее для данной реализации – односвязный список
2. Ввод строки должен быть реализован посимвольно (функцией getchar())
3. Выходная строка должна иметь следующий формат: не больше одного пробельного символа между словами, отсутствие пробельных символов в конце и в начале строки
4. Запрещается создавать новый список при обработке строки-списка
5. Структурировать код программы (разбить по функциям и файлам)

Примечания:

1. Ввод строки, так как он должен постоянно повторяться до момента завершения по команде пользователя выполнения программы организован следующим образом: а)Ввод символов getchar-ом продолжается до тех пор, пока не введён пустой символ. В таком случае функция ввода Input() возвращает в основную функцию специальное значение, обозначающее прерывание ввода строки б)Завершение программы осуществляется при вводе строки, путём ввода значения EOF.
2. Для вычисления N самых часто встречающихся членов строки создаётся отдельный массив структур chFreq, отображающий для каждого уникального символа строки значение частоты его повторений в этой строке. Массив сортируется по полю значений частоты и таким образом N самых часто встречающихся членов строки находятся как крайние элементы массива.

II. **Алгоритмическая часть**

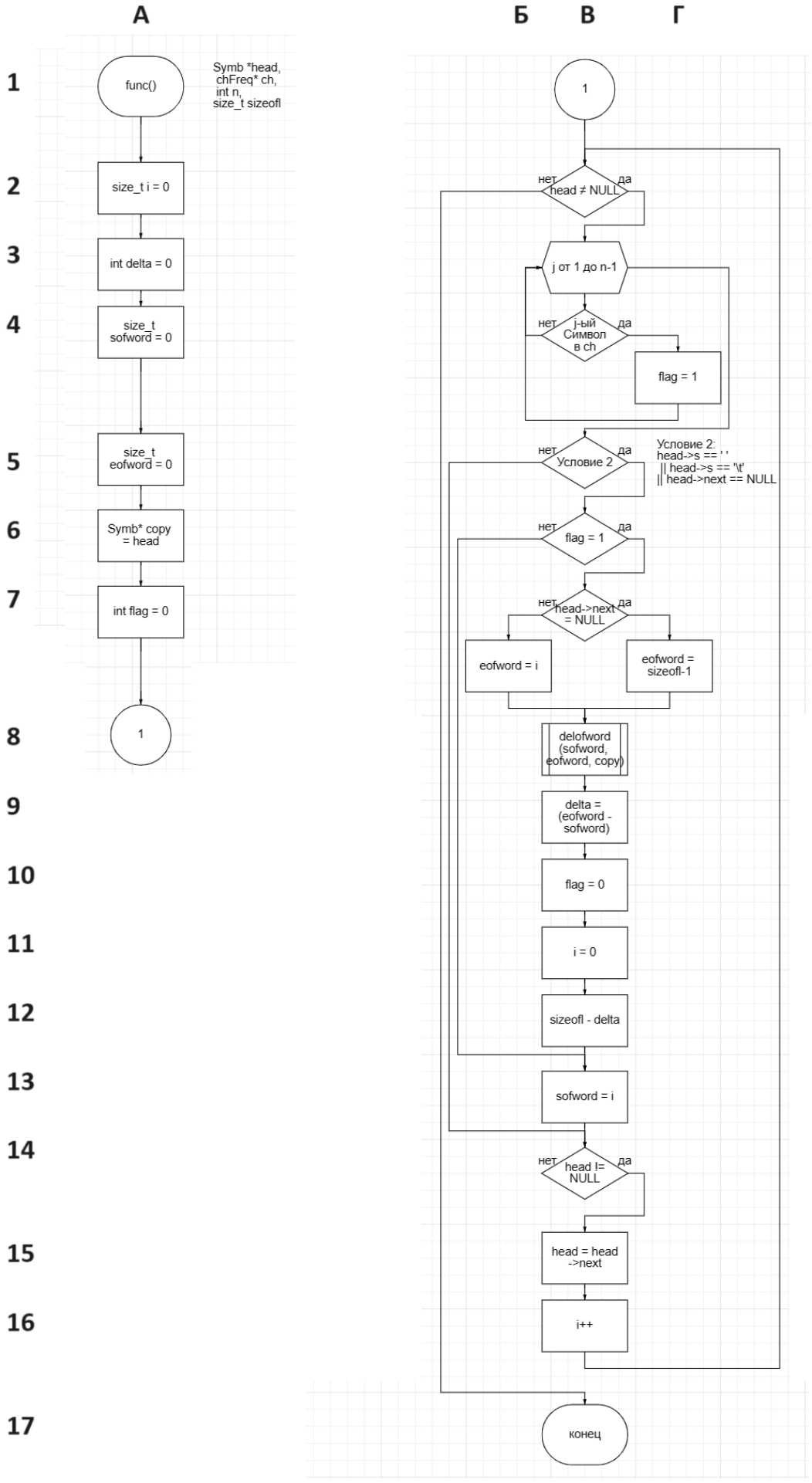
Программа имеет следующую структуру:

1. Заголовочный файл prog.h
2. Файл с исходным кодом prog.c, в том числе его функции:

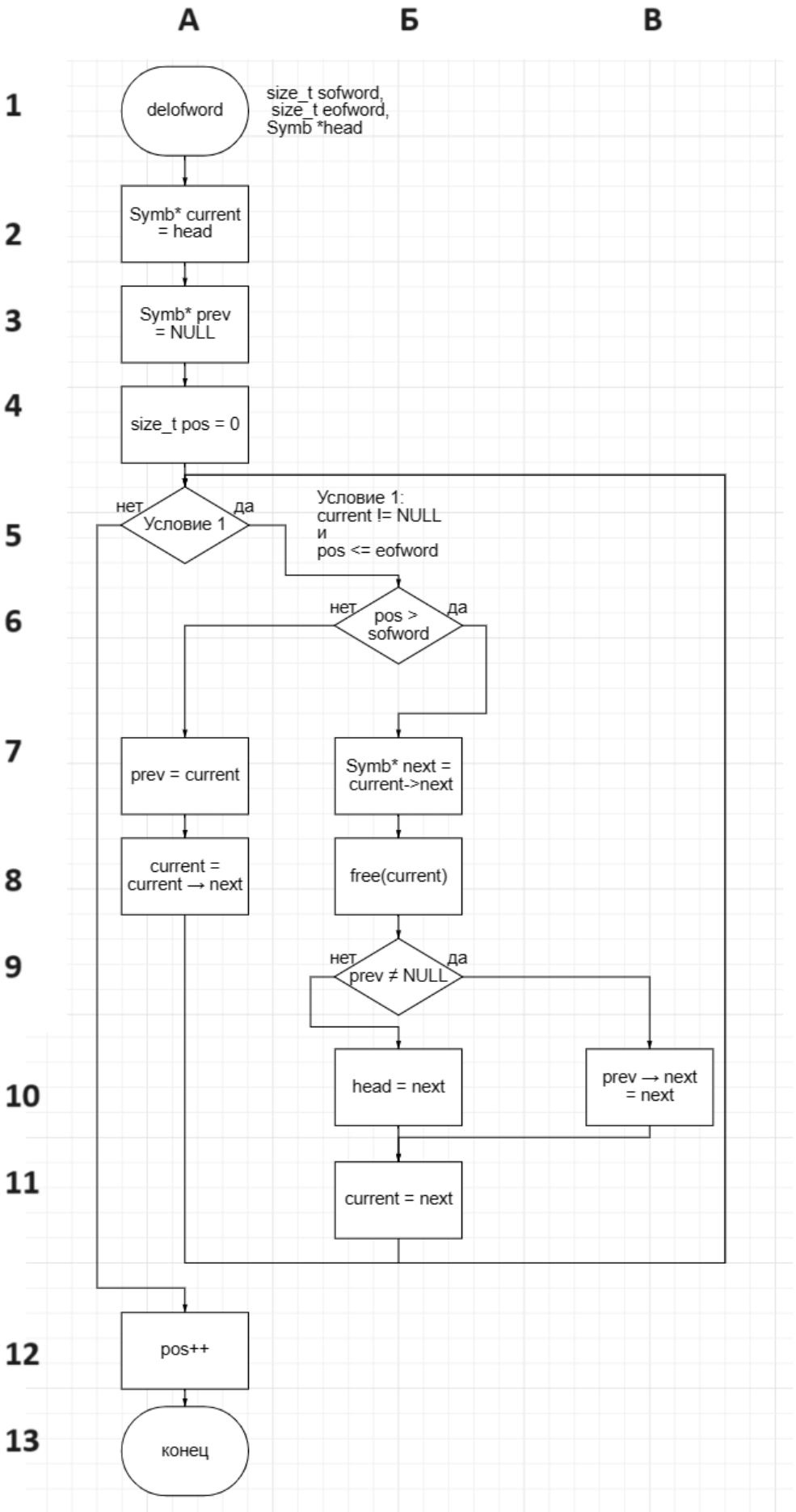
А) create () – осуществляет посимвольный ввод строки. После ввода строки возвращает длину введённой строки. В случае ввода EOF возвращает специальное (отрицательное) значение. На вход функции подаётся указатель на *head – начало списка. Соответственно в функции он меняется, за счёт чего изменённый (с заполненными значениями узлов) сохраняется
Б) printList () – осуществляет вывод в терминал значения узлов списков с начала списка, сам список не меняет
В) Nmaxs() – пробегается по списку и заполняет соответствующие ячейки массива структур ch (формат структуры chFreq включает в себя символ и частоту его встречаемости в списке), а также сортирует ch. Важно отметить, что пробельные символы в ch не записываются, т.к. их удалять не нужно
Г) func () - на вход подаются массив структур ch, а также head списка. Функция проходится по списку и обновляет значения sofword и eofword. Они в свою очередь необходимы для вызываемой здесь же функции delofword().
Д) delofword() – на вход подаются значения sofword и eofword. Из списка удаляется промежуток (интервал, т.е. значения sofword и eofword не удаляются), соответствующий входным индексам.
Е) delofsp() – специальная функция, приводящая строку к необходимому формату. В частности, удаляются лишние пробельные символы, оставшиеся после удаления слов а также присутствующие в строке с самого момента их ввода
Д) freeList() – функция, очищающая узлы списка.
Е) main() – функция, в которой содержатся последовательные вызовы функций create(), Nmaxs(), func(), delofsp(), freeList(). В данной функции также осуществляется ввод параметра N, замер времени выполнения программы, обработка ошибок.

Блок схемы:

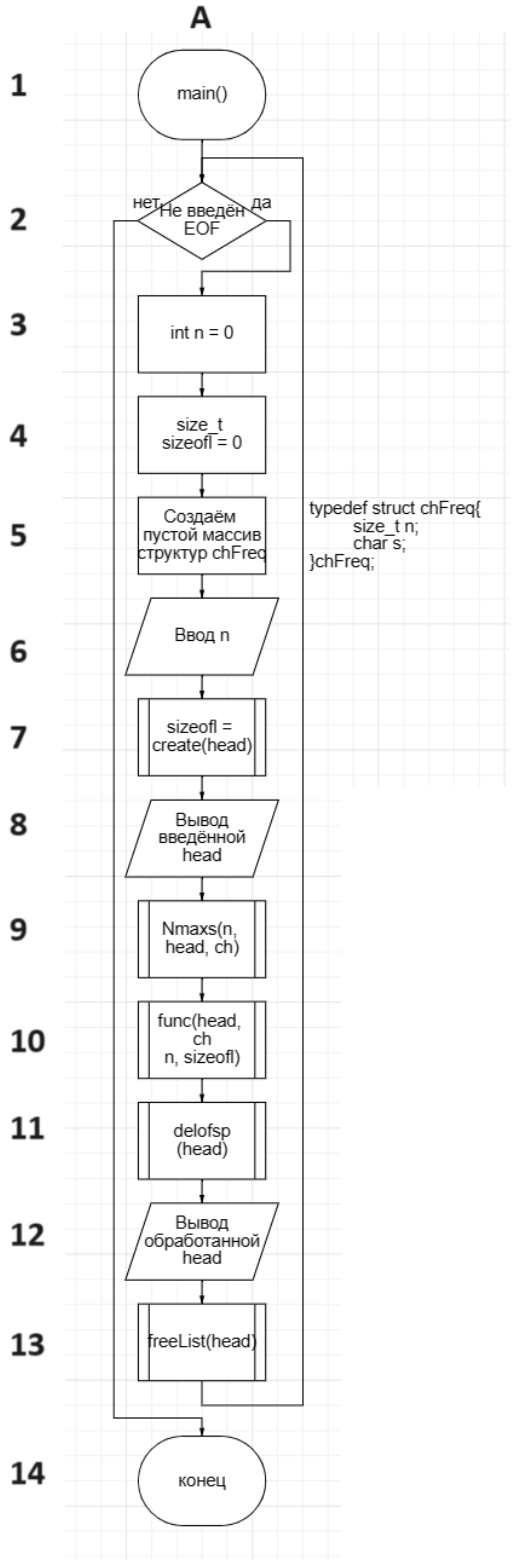
1. func(Symb *head, chFreq* ch, int n, size_t sizeofl)



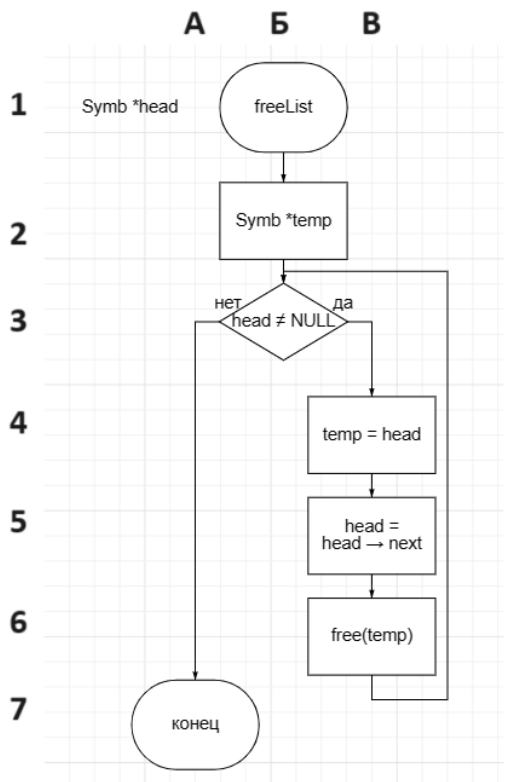
2. delofword(size_t sofword, size_t eofword, Symb *head)



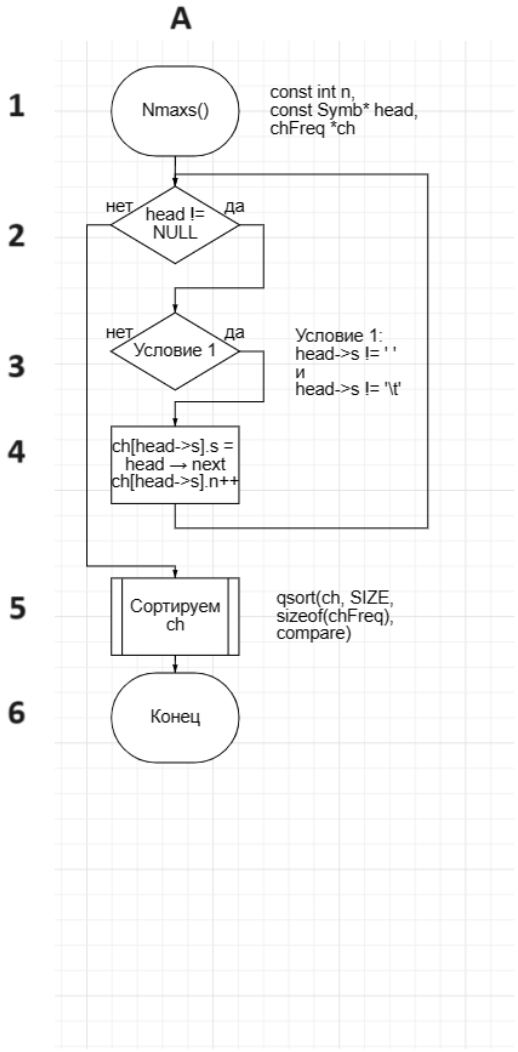
3. main()



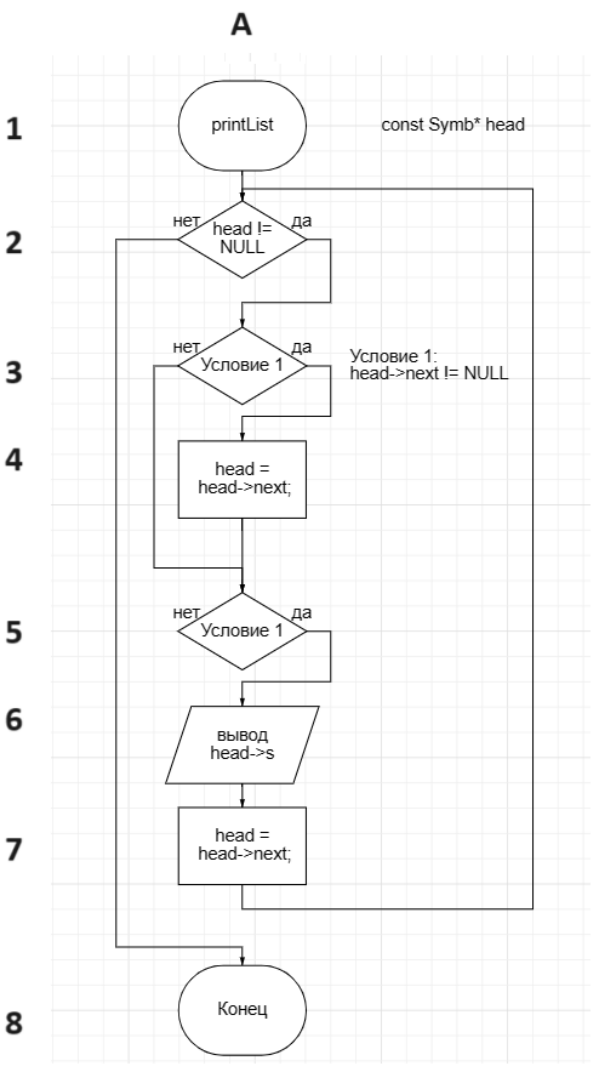
4. freeList (Symb *head)



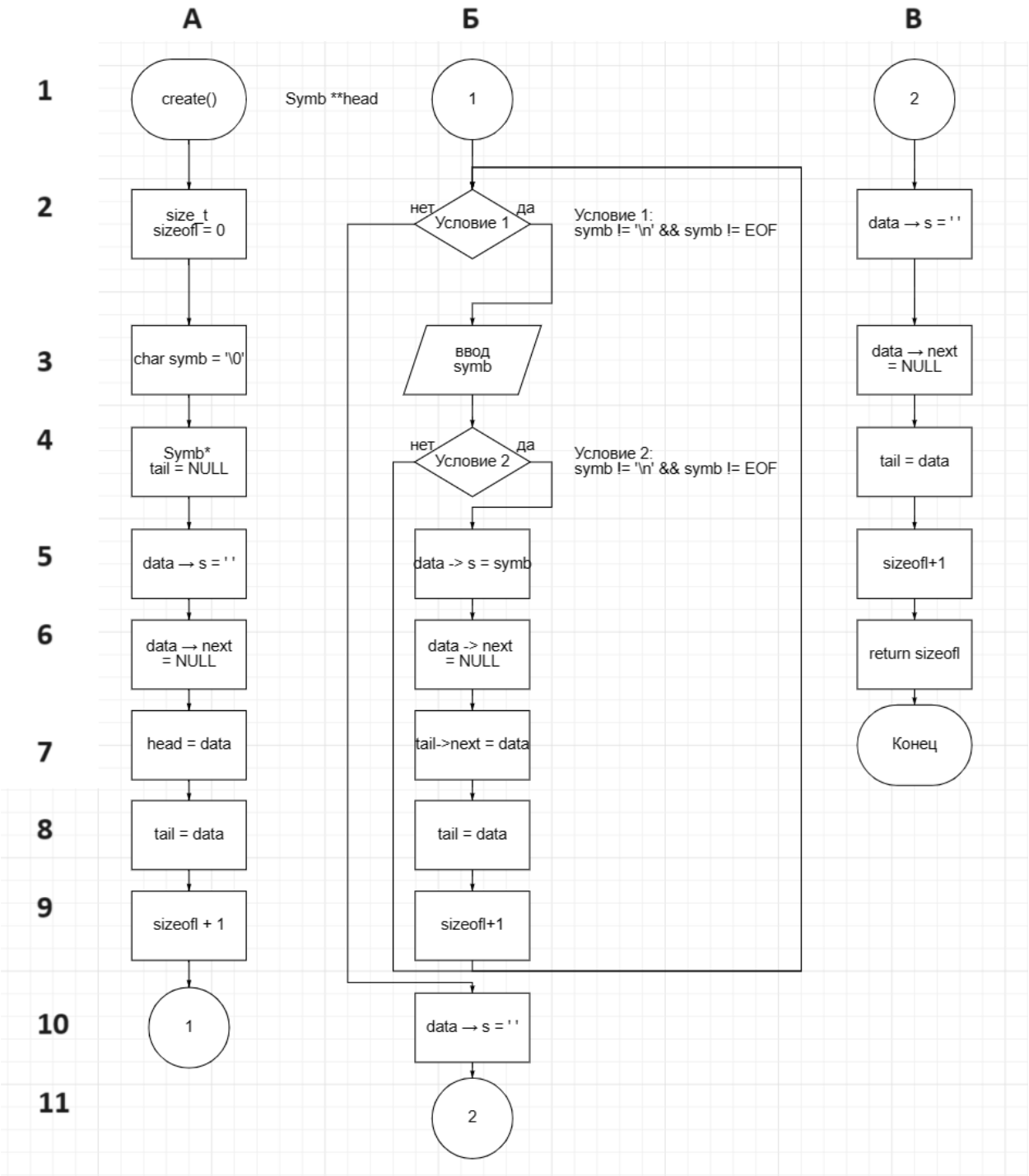
5.Nmaxs(const int n, const Symb* head, chFreq *ch)



6. printList (Symb *head)



7.Create(Symb **head)



III. Код

1. Prog.h

```
#ifndef POINT_H
#define POINT_H

typedef struct Symb{
        struct Symb* next;
        char s;
}Symb;
typedef struct chFreq{
         size_t n;
         char s;
}chFreq;

Symb* delofword(size_t sofword, size_t eofword, Symb *head);

int compare(const void* a, const void* b);

size_t searcheofword(Symb *head, size_t sofword, size_t sizeofl);

void printList(const Symb* head);

size_t create(Symb **head);
#endif
```

2. Prog.c

1. main()

```
int main(){
        printf("В следующей программе вам необходимо вводить посимвольно
строки. Ввод пустого символа завершает ввод строки,\nEOF - ввод строк...
        while(1){
                double sumtime = 0;
                Symb* head = NULL;
                int n = 0; size_t sizeofl = 0;
                chFreq ch[SIZE] = {0,0};
                sizeofl = create(&head);
                if (sizeofl < 0){
                        freeList(head);
                        break;
                }
                printf("Введите число N\n");
                if (scanf("%d", &n) <= 0){
                        freeList(head);
                        fprintf(stderr, "Ввод N пропущен\n");
                        return 0;
                }
                printf("\nВаша строка:\n");
                printList(head);
                clock_t begin = clock();
                Nmaxs(n, head, ch);
                func(head,ch, n, sizeofl);
                delofsp(head);
                clock_t end = clock();
                printf("\nВаша обработанная строка:\n");
                if (head != NULL) printList(head);
                else printf (" ");
                printf("\n");
                freeList(head);
                sumtime += (double)(end - begin)/CLOCKS_PER_SEC;
                printf ("Время выполнения: %lf секунд\n", sumtime);
                scanf("%*[^\n]");
                scanf("%*c");
```

```
        }
}
```

2. delofword()

```
Symb* delofword(size_t sofword, size_t eofword, Symb *head){
    Symb* current = head;
    Symb* prev = NULL;
    size_t pos = 0;

    while(current != NULL && pos <= eofword){
        if(pos > sofword){
                Symb* next = current->next;
                free(current);
                if(prev != NULL) {
                        prev->next = next;
                }
                else {
                        head = next;
                }
                current = next;
        }
        else { //двигаемся к началу удаляемого слова
                prev = current;
                current = current->next;
        }
        pos++;
    }
    return head;
}
```

3. func()

```
void func(Symb *head, chFreq* ch, int n, size_t sizeofl){
    size_t i = 0; int delta = 0;
    size_t sofword = 0; size_t eofword = 0;
    Symb* copy = head;
    int flag = 0;
    while(head != NULL){
        for (size_t j = 0; j<n; j++){
            if (head->s == ch[j].s){
                flag = 1; //флаг указывает на то, принадлежит ли символ
числу самых часто встречающихся
                break;
            }
        }
        if (head->s == ' ' || head->s == '\t' || head->next == NULL){
            if (flag) {//начинаем удалять слово, там, где оно завершается
                if (head->next == NULL){
                    eofword = sizeofl-1;
                }
                else eofword = i;
                head = delofword(sofword, eofword, copy); //удаляем слово
по индексам
                delta = (eofword - sofword);
                flag = 0;
                i = 0;
                sizeofl -= delta;
            }
            sofword = i;
        }
        if (head != NULL) {
            head = head->next;
        }
        i++;
```

```
    }
}
```

4. Nmaxs()

```
void Nmaxs(const int n, const Symb* head, chFreq *ch){
        while(head != NULL){
                if (head->s != ' ' && head->s != '\t'){
                        ch[(unsigned char)head->s].s = head -> s;
                        ch[(unsigned char)head->s].n ++;
                }
                head = head -> next;
        }
        qsort(ch, SIZE, sizeof(chFreq), compare);
        //for (int i = 0; i < n; i++){
        //      printf("\n%zu %c\n", ch[i].n, ch[i].s);
        //}
}
```

5. printList()

```
void printList(const Symb* head) {
    if (head != NULL) {
        printf("\"");
        if (head->next != NULL) head = head->next;
        while (head->next != NULL) {
            printf("%c", head->s);
            head = head->next;
        }
        printf("\"");
    }
}
```

6. create()

```
size_t create(Symb **head){
        size_t sizeofl = 0;
        char symb = '\0';
        Symb* tail = NULL; //tail указывает на последний узел
        Symb* data = (Symb*) calloc(1, sizeof (Symb));
        data -> s = ' '; //первый узел пробел
        data -> next = NULL;
        *head = data;
        tail = data;
        sizeofl++;

        while (symb != '\n' && symb != EOF) {
                printf("Введите символ\n");
                symb = getchar();
                if (symb != '\n' && symb != EOF) {
                        Symb* data = (Symb*) calloc(1, sizeof (Symb));
                        data -> s = symb;
                        data -> next = NULL;
                        tail->next = data;
                        tail = data;
                        sizeofl++;
                        scanf("%*[^\n]");
                        scanf("%*c");
                }
                if (symb == EOF){ //прерываем ввод строк
                        sizeofl = -100;
                        break;
                }
        }
        data = (Symb*) calloc(1, sizeof (Symb));
        data -> s = ' '; //последний узел пробел
```

```
        data -> next = NULL;
        tail->next = data;
        sizeofl++;
        return sizeofl;
}
```

7. delofsp()

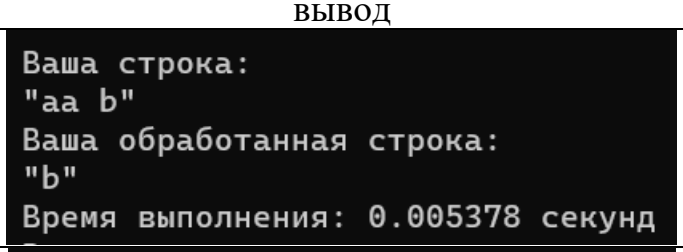
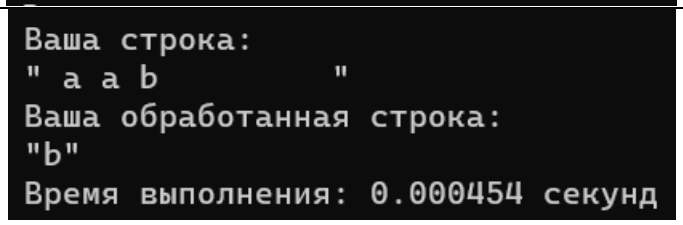
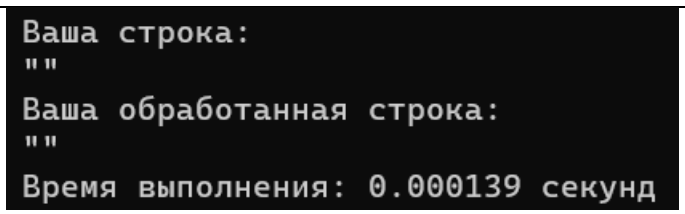
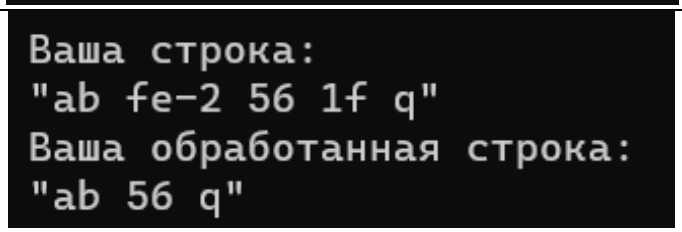
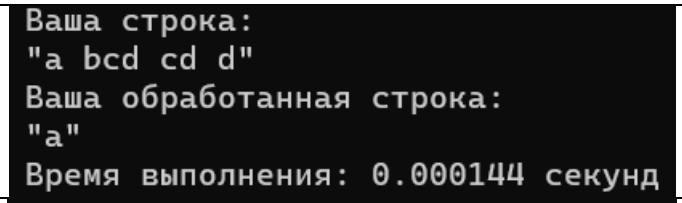
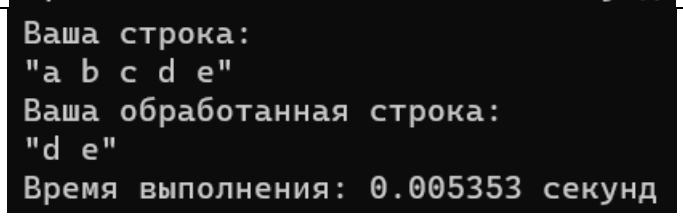
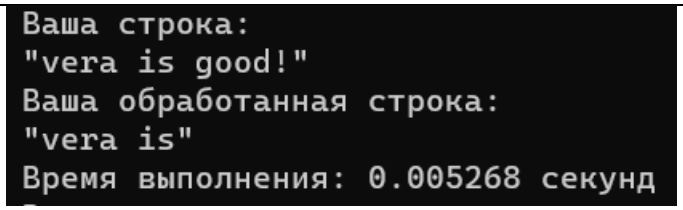
```
void delofsp (Symb *head){
        Symb *current = head;
        Symb *prev = NULL;
        Symb *temp = NULL; //просто буффер

        while(current != NULL){
                if(prev == NULL){
                        prev = current;
                        current = current -> next;
                }
                else if ((current -> s == ' ' || current -> s ==
'\t')&&(prev -> s == ' ' || prev -> s == '\t')){
                        temp = current -> next;
                        prev -> next = current -> next;
                        free(current);
                        current = temp;
                }
                else{
                        prev = current;
                        current = current -> next;
                }
        }
}
```

8. freeList()

```
void freeList(Symb *head) {
        Symb *temp;
        while (head != NULL) {
                temp = head;
                head = head->next;
                free(temp);
        }
}
```

IV. Тесты

Ввод (строка/N)	Ожидаемый вывод	Фактический вывод
«aa b»/1	b	
« a a b »/1	b	
«»/1	пустая строка	
"ab fe-2 56 1f q"/2	строка с удалёнными f и ещё каким-то символом, встречающимся единожды	
"a bcd cd d"/2	Только «a», т.к. удаляются слова с d и c	
«a b c d e»/3	«d e»	
"vera is good!"/1	"vera is"	

V. Выводы

Работая над лабораторной работой №6, автор освоил работу со списками. В данной лабораторной работе был использован односвязный список. Альтернативный ему двусвязный список работал бы аналогично, за исключением работы пары функций. Поэтому можно сказать, что в целом списки автором освоены.

Итоговое тестирование программы показало, что программа работает корректно при самых разнообразных случаях. В частности, не возникает проблем при вводе лишних пробельных символов, при обработке натуральных N, при вводе пустой строки. Время же выполнения программы в целом коррелирует с длиной строки, за некоторыми исключениями.

Вполне вероятно, что логику программы возможно улучшить. В частности, функции delofsp и func можно объединить, делая в процессе одного прохода и удаление символов, и удаление необходимых слов. Конечно, повышение эффективности программы на один проход не является критичным. К тому же, как видно по замерам времени, при вводе пустой строки затрачивается некоторое время на выполнение программы. Логика программы такова, что эта пустая строка сначала попадает в функцию func(), а уже затем вызывает по условию в main() соответствующий вывод. Следовательно, можно улучшить программу, изменив последовательность вызываемых функций.