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 Д.И.
Группа: Б22-505
Преподаватель: Вавренюк А.Б.

Москва — 2022

1. Формулировка индивидуального задания

Вариант №22

Задание

Упорядочить слова в строке по алфавиту.

2. Описание использованных типов данных

При выполнении данной работы были использованы типы данных int, char, собственный тип данных Node, указатели на этот тип данных для описания узла списка, Stack и указатель на него для описания самого списка, а также Coordinate.

3. Описание использованного алгоритма

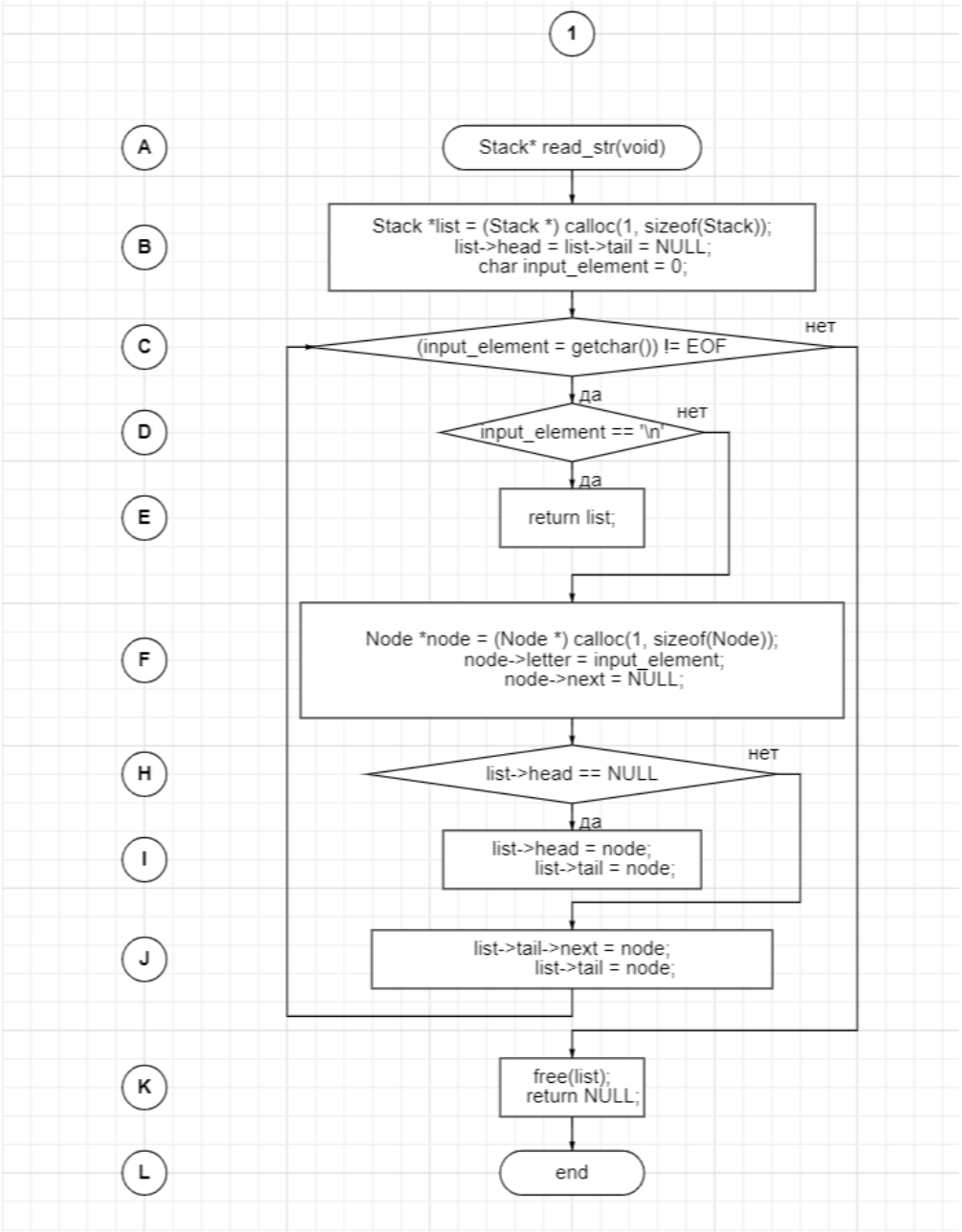

Рис. 1: Блоксхема алгоритма работы функции read_str

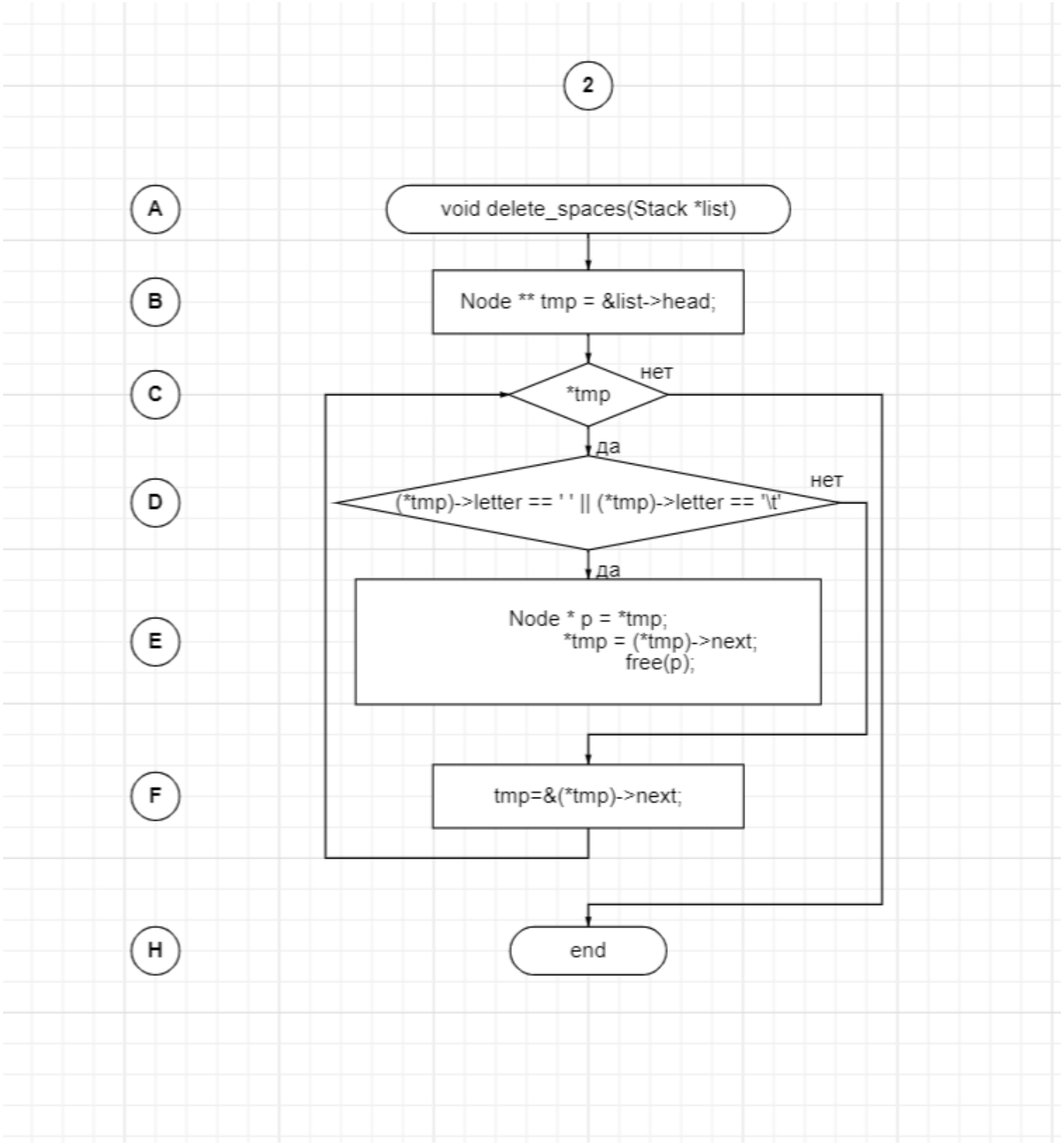

Рис.2: Блоксхема алгоритма работы функции delete_spaces

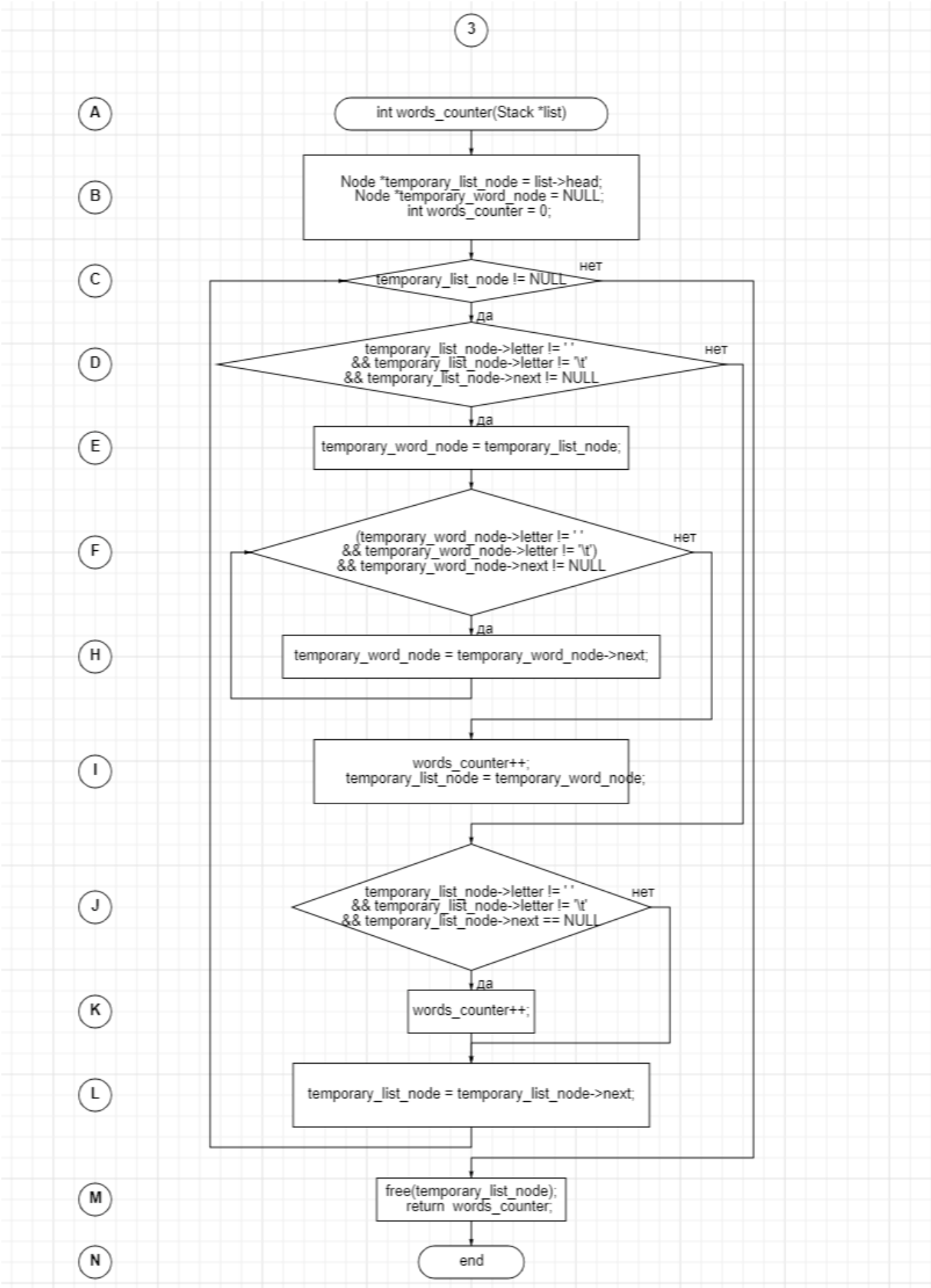

Рис.3: Блоксхема алгоритма работы функции words_counter

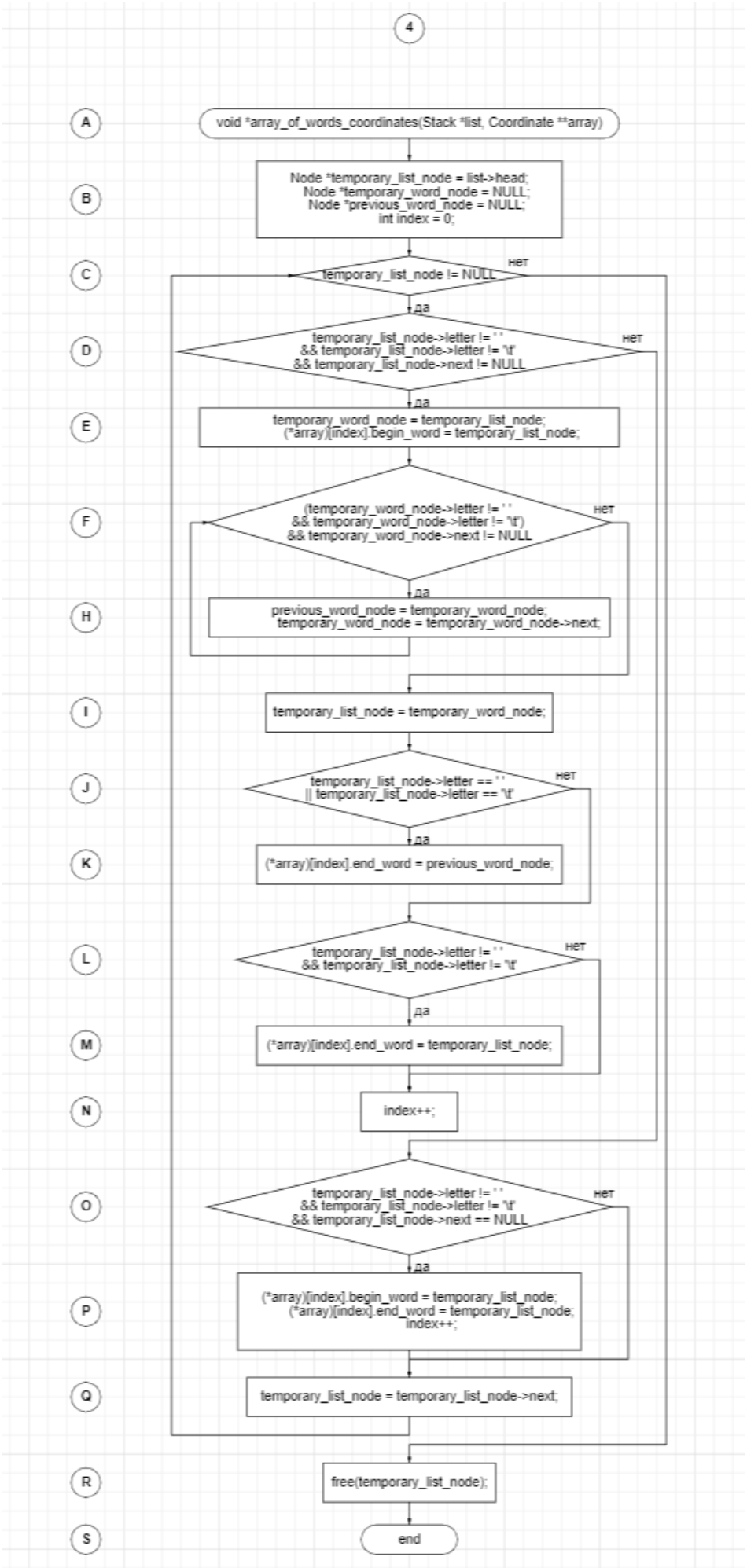

Рис.4: Блоксхема алгоритма работы функции array_of_words_coordinate

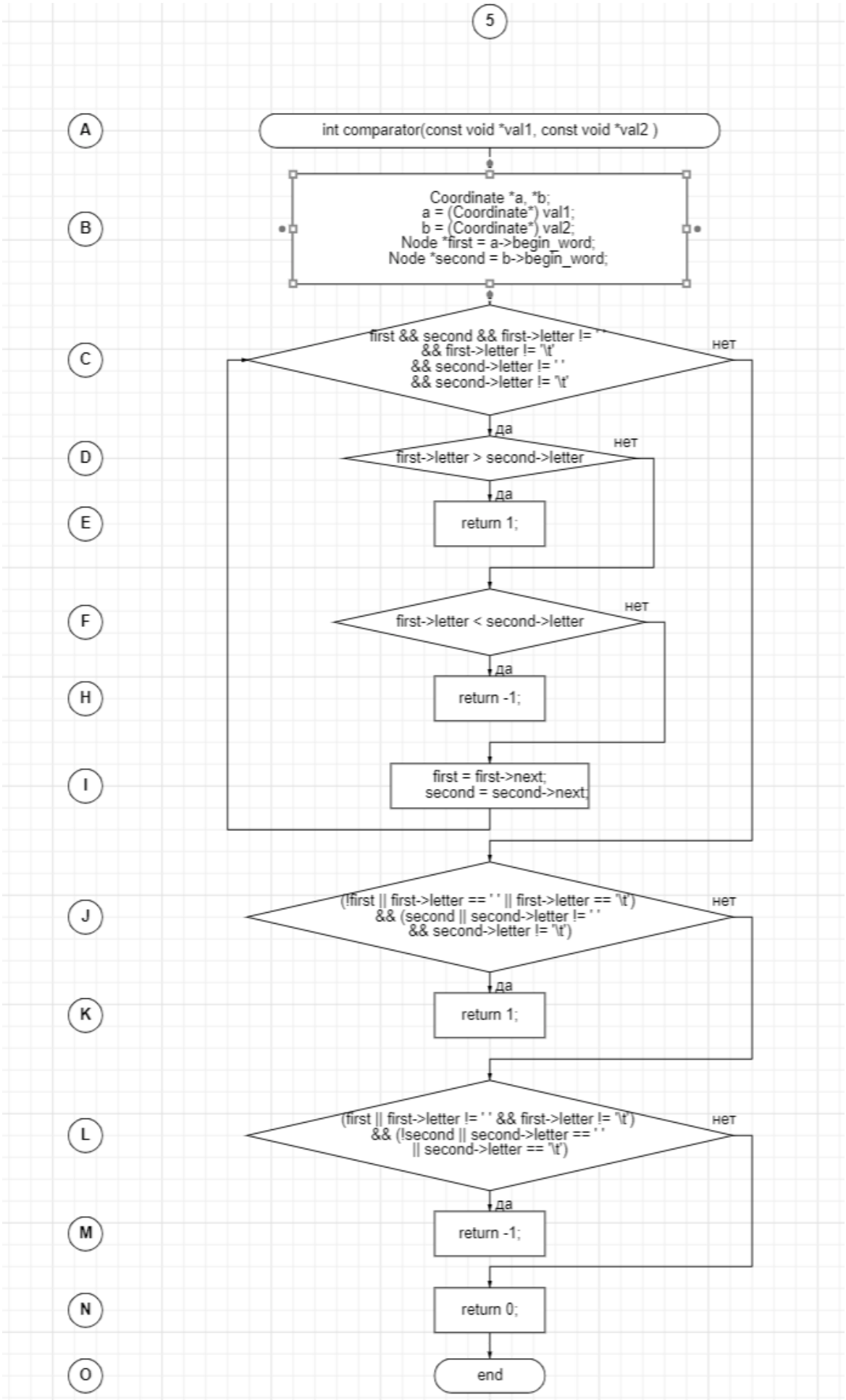

Рис.5: Блоксхема алгоритма работы функции comparator

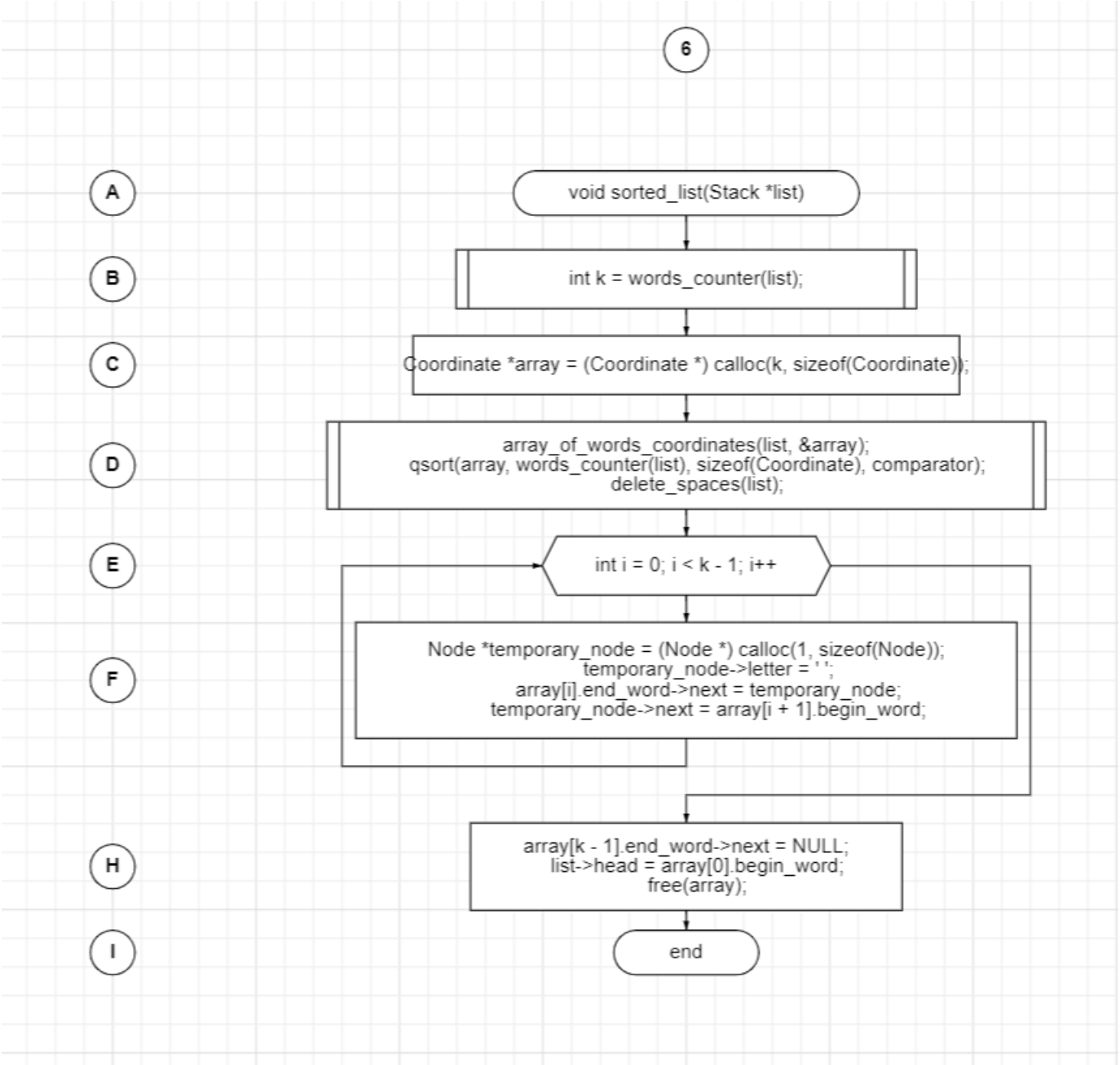

Рис.4: Блоксхема алгоритма работы функции sorted_list

4. Исходные коды разработанных программ

```c
#include <stdio.h>
#include <stdlib.h>
#include "functions.h"

int main() {
    Stack *list;
    while ((list = read_str()) != NULL) {
        printf("======\n");
        sorted_list(list);
        print_list(list);
        delete_list(list);
        printf("======\n");
    }
    free(list);
    return 0;
}
```

Листинг 1: исходные кода программы prog (файл: main.c)

```c
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

typedef struct Node {
    char letter;
    struct Node *next;
}Node;

typedef struct {
    Node *head;
    Node *tail;
}Stack;

typedef struct {
    Node *begin_word;
    Node *end_word;
}Coordinate;

Stack* read_str(void);
void delete_spaces(Stack *list);
void print_list(Stack *list);
void delete_list(Stack *list);
int words_counter(Stack *list);
void *array_of_words_coordinates(Stack *list, Coordinate **array);
int comparator(const void *val1, const void *val2);
void sorted_list(Stack *list);

#endif
```

Листинг 2: исходные кода программы prog (файл: functions.h)

```c
#include <stdio.h>
#include <stdlib.h>
#include "functions.h"

Stack* read_str(void) {
    Stack *list = (Stack *) calloc(1, sizeof(Stack));
    list->head = list->tail = NULL;
    char input_element = 0;
    while ((input_element = getchar()) != EOF) {
        if (input_element == '\n') {
            return list;
        }
        Node *node = (Node *) calloc(1, sizeof(Node));
        node->letter = input_element;
        node->next = NULL;
        if (list->head == NULL) {
            list->head = node;
            list->tail = node;
        } else {
            list->tail->next = node;
            list->tail = node;
        }
    }
    free(list);
    return NULL;
}
```

Листинг 3: исходные кода программы prog (файл: functions.c)

```
27
28 void delete_spaces(Stack *list) {
29         Node ** tmp = &list->head;
30         while (*tmp)
31                 if ((*tmp)->letter == ' ' || (*tmp)->letter == '\t') {
32                         Node * p = *tmp;
33                         *tmp = (*tmp)->next;
34                         free(p);
35                 } else {
36                         tmp=&(*tmp)->next;
37                 }
38 }
39
40 void print_list(Stack *list) {
41     Node *temporary_node = list->head;
42     printf("\"");
43     while (temporary_node != NULL) {
44         printf("%c", temporary_node->letter);
45         temporary_node = temporary_node->next;
46     }
47     printf("\"\n");
48 }
49
50 void delete_list(Stack *list) {
51     Node *temporary_node = list->head;
52     while (temporary_node != NULL) {
53         Node *pointer_previous = temporary_node;
54         temporary_node = temporary_node->next;
55         free(pointer_previous);
56     }
57     free(list);
58 }
59
60 int words_counter(Stack *list) {
61     Node *temporary_list_node = list->head;
62     Node *temporary_word_node = NULL;
63     int words_counter = 0;
64     while (temporary_list_node != NULL) {
65         if (temporary_list_node->letter != ' ' && temporary_list_node->letter != '\t' && temporary_list_node->next != NULL) {
66             temporary_word_node = temporary_list_node;
67             while ((temporary_word_node->letter != ' ' && temporary_word_node->letter != '\t') && temporary_word_node->next != NULL) {
68                 temporary_word_node = temporary_word_node->next;
69             }
70             words_counter++;
71             temporary_list_node = temporary_word_node;
72         } else if (temporary_list_node->letter != ' ' && temporary_list_node->letter != '\t' && temporary_list_node->next == NULL) {
73             words_counter++;
74         }
75         temporary_list_node = temporary_list_node->next;
76     }
77     free(temporary_list_node);
78     return  words_counter;
79 }
80
```

Листинг 4: исходные кода программы prog (файл: functions.c)

```c
void *array_of_words_coordinates(Stack *list, Coordinate **array) {
    Node *temporary_list_node = list->head;
    Node *temporary_word_node = NULL;
    Node *previous_word_node = NULL;
    int index = 0;
    while (temporary_list_node != NULL) {
        if (temporary_list_node->letter != ' ' && temporary_list_node->letter != '\t' && temporary_list_node->next != NULL) {
            temporary_word_node = temporary_list_node;
            (*array)[index].begin_word = temporary_list_node;
            while (temporary_word_node->letter != ' ' && temporary_word_node->letter != '\t' && temporary_word_node->next != NULL) {
                previous_word_node = temporary_word_node;
                temporary_word_node = temporary_word_node->next;
            }
            temporary_list_node = temporary_word_node;
            if (temporary_list_node->letter == ' ' || temporary_list_node->letter == '\t') {
                (*array)[index].end_word = previous_word_node;
            } else if (temporary_list_node->letter != ' ' && temporary_list_node->letter != '\t') {
                (*array)[index].end_word = temporary_list_node;
            }
            index++;
        } else if (temporary_list_node->letter != ' ' && temporary_list_node->letter != '\t' && temporary_list_node->next == NULL) {
            (*array)[index].begin_word = temporary_list_node;
            (*array)[index].end_word = temporary_list_node;
            index++;
        }
        temporary_list_node = temporary_list_node->next;
    }
    free(temporary_list_node);
}

int comparator(const void *val1, const void *val2 ) {
    Coordinate *a, *b;
    a = (Coordinate*) val1;
    b = (Coordinate*) val2;
    Node *first = a->begin_word;
    Node *second = b->begin_word;
    while (first && second && first->letter != ' ' && first->letter != '\t' && second->letter != ' ' && second->letter != '\t') {
        if (first->letter > second->letter) {
            return 1;
        }
        if (first->letter < second->letter) {
            return -1;
        }
        first = first->next;
        second = second->next;
    }
    if ((!first || first->letter == ' ' || first->letter == '\t') && (second || second->letter != ' ' && second->letter != '\t')) {
        return 1;
    }
    else if ((first || first->letter != ' ' && first->letter != '\t') && (!second || second->letter == ' ' || second->letter == '\t')) {
        return -1;
    }
    else {
        return 0;
    }
}
```

Листинг 5: исходные кода программы prog (файл: functions.c)

```c

void sorted_list(Stack *list) {
    int k = words_counter(list);
    Coordinate *array = (Coordinate *) calloc(k, sizeof(Coordinate));
    array_of_words_coordinates(list, &array);
    qsort(array, words_counter(list), sizeof(Coordinate), comparator);
    delete_spaces(list);
    for (int i = 0; i < k - 1; i++) {
        Node *temporary_node = (Node *) calloc(1, sizeof(Node));
        temporary_node->letter = ' ';
        array[i].end_word->next = temporary_node;
        temporary_node->next = array[i + 1].begin_word;
    }
    array[k - 1].end_word->next = NULL;
    list->head = array[0].begin_word;
    free(array);
}
```

Листинг 6: исходные кода программы prog (файл: functions.c)

5. Описание тестовых примеров и их скриншоты

Входные данные	Ожидаемый результат	Полученный результат
""	""	""
'\n'	""	""
""\t""	""	""
" "	""	""
" babb baac"	"baac babb"	"baac babb"

Таблица 1: работа программы prog

```
[titov.di@unix:~/inf/lab6]$ valgrind ./prog
==6647== Memcheck, a memory error detector
==6647== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==6647== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==6647== Command: ./prog
==6647==

======
""
======

======
""
======

======
""
======

======
""
======
    babb baac
======
"baac babb"
======
==6647==
==6647== HEAP SUMMARY:
==6647==     in use at exit: 0 bytes in 0 blocks
==6647==   total heap usage: 34 allocs, 34 frees, 2,576 bytes allocated
==6647==
==6647== All heap blocks were freed -- no leaks are possible
==6647==
==6647== For lists of detected and suppressed errors, rerun with: -s
==6647== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Рис.6: Тестовые примеры работы программы prog

6. Выводы

При выполнении данной лабораторной работы я научился работать с такой сложной структурой данных как списки, а также выполнив инд задание научился их сортировать, что является не самой простой задаче, также были улучшены навыки работы с указателями и усвоены положительные черты списка в виде отсутствия необходимости хранить индексы элементов.