

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: Баранов А. Т.

Группа: Б22-534

Преподаватель: Широких Т. А.

Москва — 2022

1. Формулировка индивидуального задания

Вариант №21

Задание

Заменить в каждом слове строки последовательности повторяющихся подряд одинаковых символов на единичные вхождения данных символов.

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовался встроенный тип данных `char`, предназначенный для работы с символами.

3. Описание использованного алгоритма

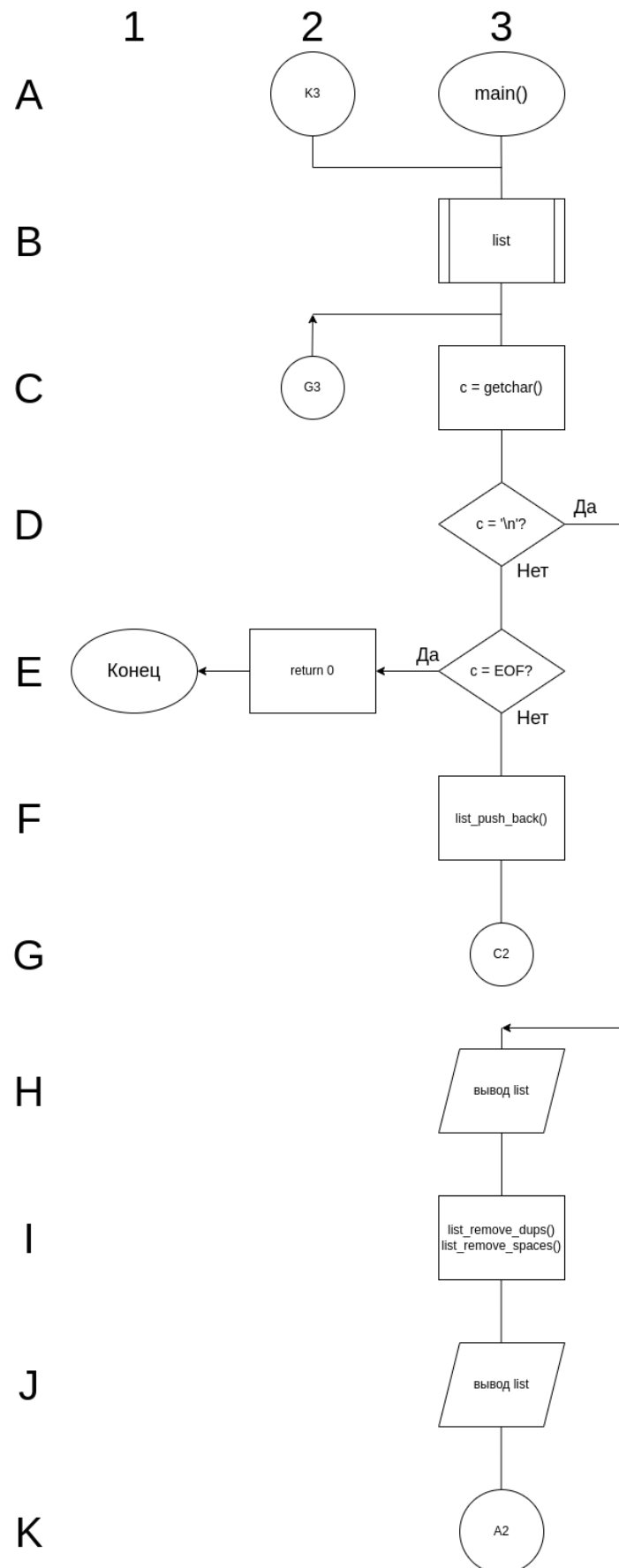


Рис. 1: Блок- схема алгоритма работы функции main

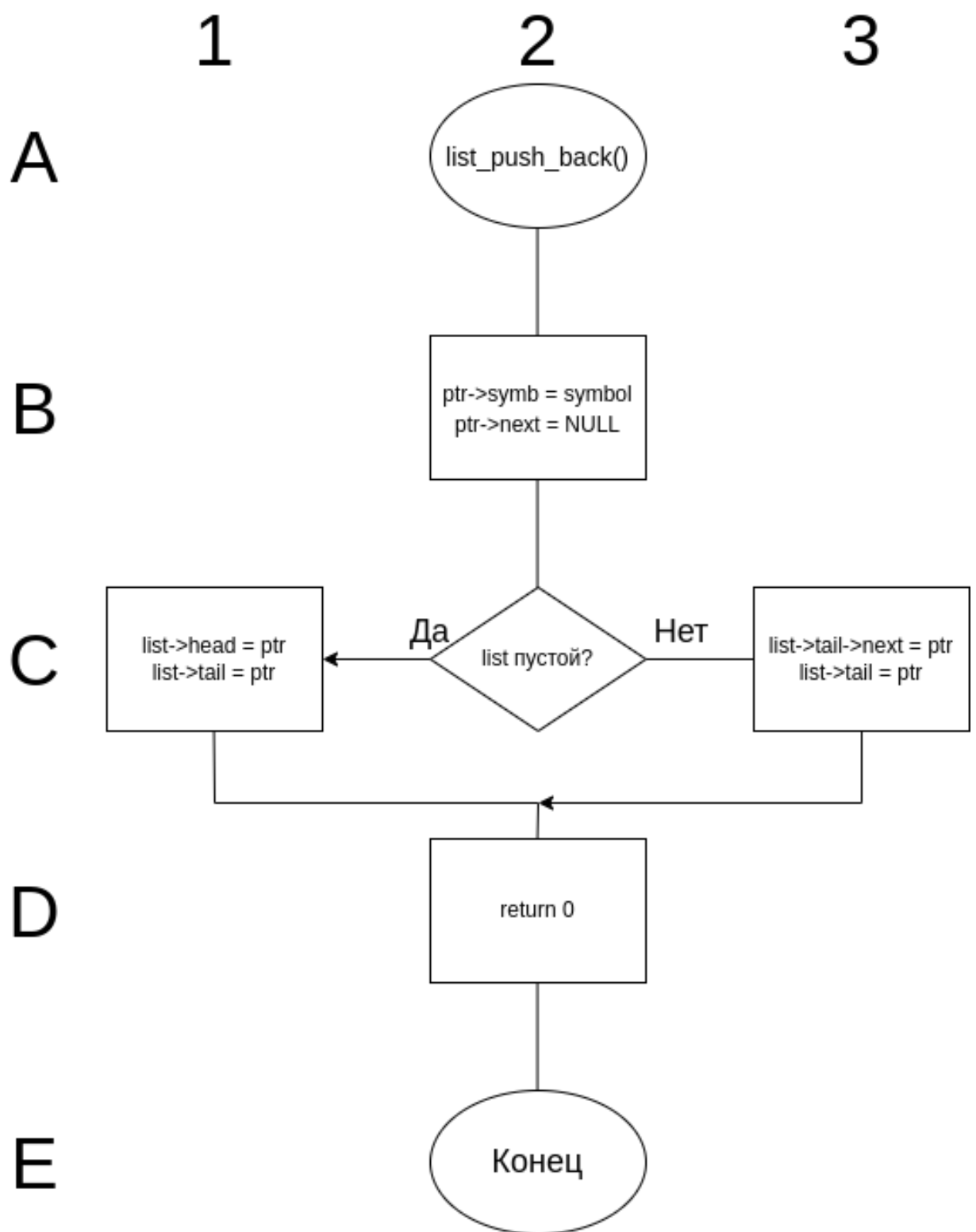


Рис. 2: Блок- схема алгоритма работы функции `list_push_back`

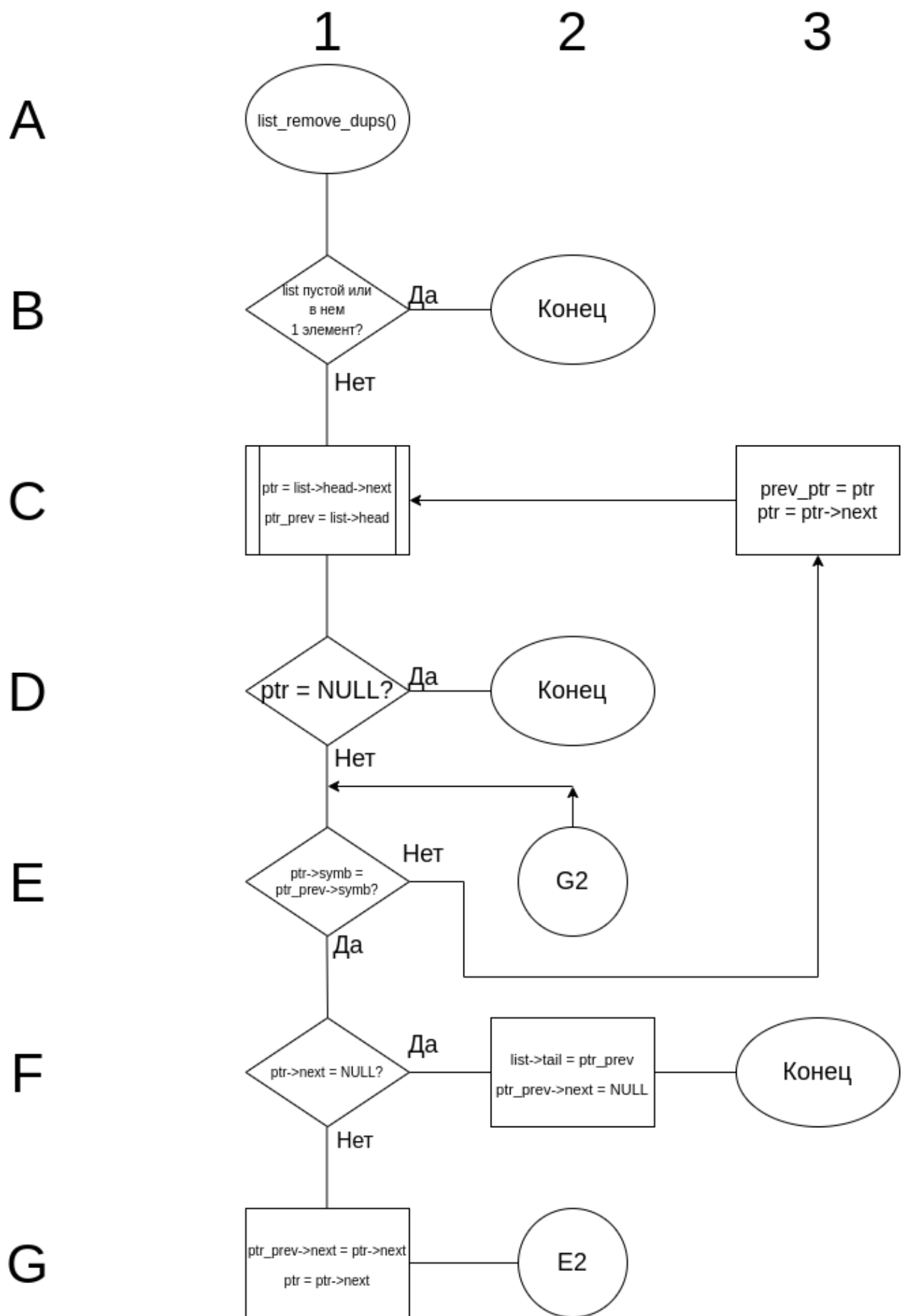


Рис. 3: Блок- схема алгоритма работы функции `list_remove_dups`

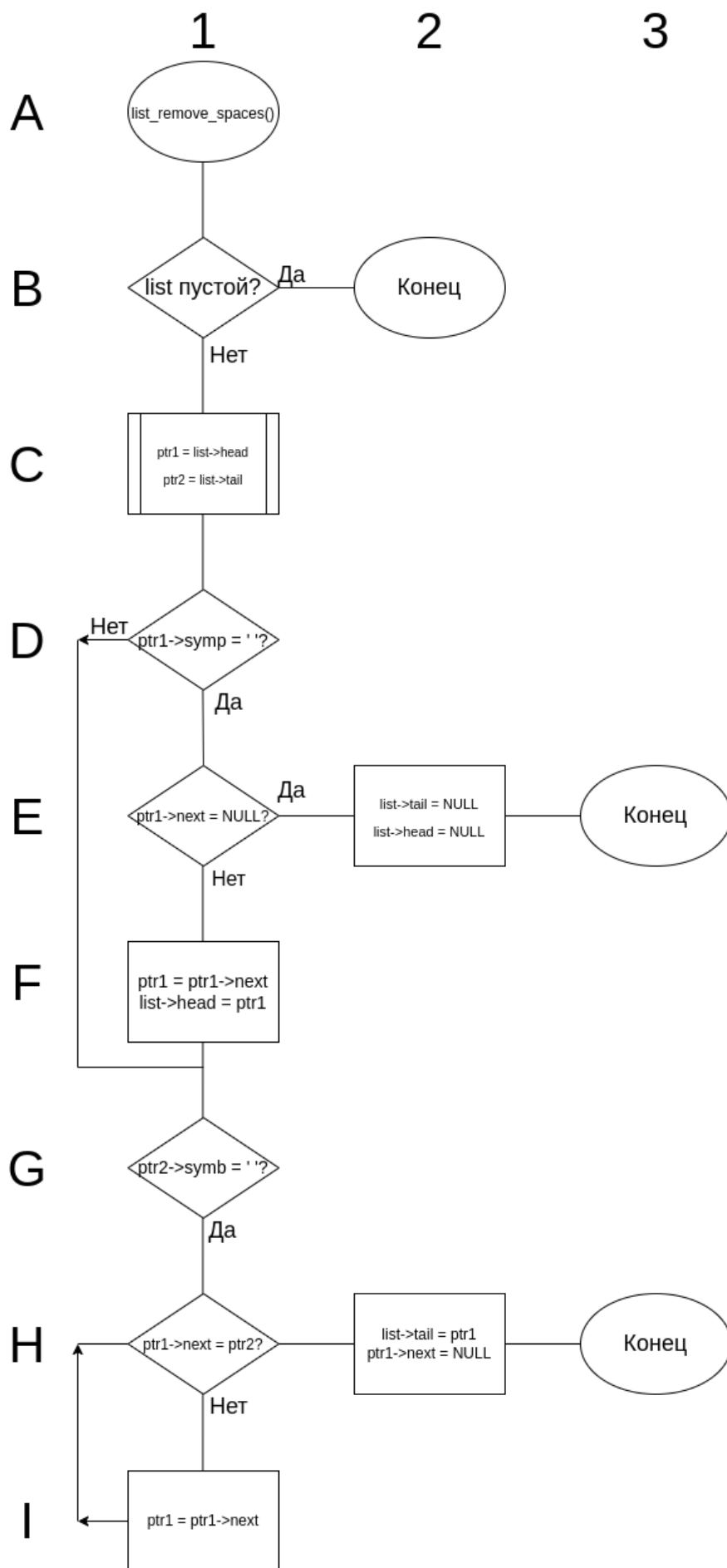


Рис. 4: Блок- схема алгоритма работы функции `list_remove_spaces`

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы `prog1` (файл: `main.c`)

```
1. #include <stdio.h>
2. #include "list.h"
3. int main ()
4. {
5.     while(1)
6.     {
7.         list *list = list_new();
8.         if(NULL == list) goto A;
9.
10.        printf("Введите строку: ");
11.
12.        char c;
13.        while((c = (char)getchar()) != '\n')
14.        {
15.            if (c == EOF)
16.            {
17.                list_delete(list);
18.                goto A;
19.            }
20.            if(list_push_back(list, c) == 1)
21.            {
22.                list_delete(list);
23.                goto B;
24.            }
25.        }
26.        printf("Начальная строка: \n");
27.        list_print(list);
28.        list_remove_dups(list);
29.        list_remove_spaces(list);
30.        printf("Конечная строка: \n");
31.        list_print(list);
32.        printf("\n");
33.        list_delete(list);
34.    }
35.    A:
36.    return 0;
37.    B:
38.    fprintf(stderr, "Failed to allocate some piece of list\n");
39.    return 1;
40. }
```

```
1. #ifndef LAB6_LIST_H
2. #define LAB6_LIST_H
3.
4. typedef struct item {
5.     char symb;
6.     struct item *next;
7. } item;
8.
9. typedef struct list {
10.     item *head;
11.     item *tail;
12. } list;
13.
14. list *list_new();
15.
16. void list_delete(list *list);
17.
18. void list_print(const list *list);
19.
20. int list_push_back(list *list, char symb);
21.
22. void list_remove_dups(list *list);
23.
24. void list_remove_spaces(list *list);
25.
26. #endif //LAB6_LIST_H
```



```

1. #include "list.h"
2. #include <stdlib.h>
3. #include <stdio.h>
4.
5. list *list_new()
6. {
7.     list *tmp = (list *) calloc(1, sizeof(list));
8.     return (tmp)? tmp:NULL;
9. }
10.
11. void list_delete(list *list)
12. {
13.     item *ptr = list->head, *ptr_prev;
14.     while (ptr)
15.     {
16.         ptr_prev = ptr;
17.         ptr = ptr->next;
18.         free(ptr_prev);
19.     }
20.     free(list);
21. }
22.
23. void list_print(const list *list)
24. {
25.     item *ptr = list->head;
26.
27.     while (ptr)
28.     {
29.         printf("%c", ptr->symb);
30.         ptr = ptr->next;
31.     }
32.     printf("\n");
33. }
34.

```

```

35. int list_push_back(list *list, char symb)
36. {
37.     item *ptr = (item *) malloc(sizeof(item));
38.     if (!ptr) return 1;
39.
40.     ptr->symb = symb;
41.     ptr->next = NULL;
42.     if (!list->head)
43.     {
44.         list->head = ptr;
45.         list->tail = ptr;
46.     }
47.     else
48.     {
49.         list->tail->next = ptr;
50.         list->tail = ptr;
51.     }
52.     return 0;
53. }
54.
55. void list_remove_dups(list *list)
56. {
57.     if(list->head == NULL || list->tail == NULL
58.         || list->head == list->tail) return;
59.
60.     item *ptr = list->head->next, *ptr_prev = list->head;
61.     while(ptr)
62.     {
63.         while(ptr->symb == ptr_prev->symb)
64.         {
65.             if(!ptr->next)
66.             {
67.                 list->tail = ptr_prev;
68.                 ptr_prev->next = NULL;
69.                 free(ptr);
70.                 return;
71.             }

```

```

72.     else
73.     {
74.         ptr_prev->next = ptr->next;
75.         item *tmp = ptr;
76.         ptr = ptr->next;
77.         free(tmp);
78.     }
79. }
80. ptr_prev = ptr;
81. ptr = ptr->next;
82. }
83. }
84. void list_remove_spaces(list *list)
85. {
86.     if(list->head == NULL || list->tail == NULL) return;
87.     item *ptr1 = list->head, *ptr2 = list->tail;
88.     if(ptr1->symb == ' ')
89.     {
90.         if(!ptr1->next)
91.         {
92.             list->tail = NULL;
93.             list->head = NULL;
94.             free(ptr1);
95.             return;
96.         }
97.     else
98.     {
99.         item *tmp = ptr1;
100.         ptr1 = ptr1->next;
101.         list->head = ptr1;
102.         free(tmp);
103.     }
104. }
105. if(ptr2->symb == ' ')
106. {
107.     while(ptr1->next != ptr2)
108.         ptr1 = ptr1->next;
109.     list->tail = ptr1;
110.     ptr1->next = NULL;
111.     free(ptr2);
112. }}

```

5. Описание тестовых примеров

Таблица 1: Тестовые наборы для программы prog1

№ теста	Ввод	Ожидаемое значение	Полученное значение
1	" "	«»	«»
2	" "	«»	«»
3	" Daaaaa zdraaa per ma"	«Da zdra per ma»	«Da zdra per ma»
4	"0 "	«0»	«0»

6. Скриншоты с результатами тестов

```

retrobanner@retrozephyrus:~ — ssh baranov.at@samos.dozen.mephi.ru
[baranov.at@unix:~/home/baranov/informatics/lab6]$ ls
list.c list.h main.c
[baranov.at@unix:~/home/baranov/informatics/lab6]$ gcc main.c list.c -o prog1
[baranov.at@unix:~/home/baranov/informatics/lab6]$ valgrind --leak-check=full ./prog1
==13328== Memcheck, a memory error detector
==13328== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==13328== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==13328== Command: ./prog1
==13328==
Введите строку:
Начальная строка: ""
Конечная строка: ""

Введите строку:
Начальная строка: " "
Конечная строка: ""

Введите строку: Daaaaa zdraaa per ma
Начальная строка: " Daaaaa zdraaa per ma"
Конечная строка: "Da zdra per ma"

Введите строку: 0
Начальная строка: "0 "
Конечная строка: "0"

Введите строку: v traavaaaaa siiiideeell kuznechik
Начальная строка: " v traavaaaaa siiiideeell kuznechik "
Конечная строка: "v trave sidel kuznechik"

Введите строку: ==13328==
==13328== HEAP SUMMARY:
==13328== in use at exit: 0 bytes in 0 blocks
==13328== total heap usage: 85 allocs, 85 frees, 3,376 bytes allocated
==13328==
==13328== All heap blocks were freed -- no leaks are possible
==13328==
==13328== For lists of detected and suppressed errors, rerun with: -s
==13328== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[baranov.at@unix:~/home/baranov/informatics/lab6]$

```

Рис. 5: Сборка, запуск и тест программы prog1

8. Выводы

В ходе выполнения данной работы на примере программы, обрабатывающей строки, были рассмотрены базовые принципы работы построения программ на языке C:

1. Разработка функций для работы со списками: удаление подряд стоящих одинаковых символов.
2. Объявление и использование переменных.
3. Работа с динамическим выделением и освобождением памяти.
4. Объявления структур данных.
5. Запуск программы с обязательными и необязательными опциями с помощью `getopt`.