

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: Мартыненко Д. М.

Группа: С22-501

Преподаватель: Овчаренко Е. С.

Москва — 2022

1. Формулировка индивидуального задания

Вариант 35: продублировать в строке все слова чётной длины.

2. Описание использованных типов данных

При выполнении лабораторной работы были использованны следующие типы данных:

1. Структура `LinkedList` - список по заданию, имеющий следующие поля: `.base` - указатель (`Node*`) на начальный элемент списка.
2. Структура `Node` - элемент списка, имеющий следующие поля: `.value` - значение (`char`) элемента списка, `.next` - указатель (`Node*`) на следующий элемент списка.

3. Описание использованного алгоритма

3.1. Основной алгоритм программы

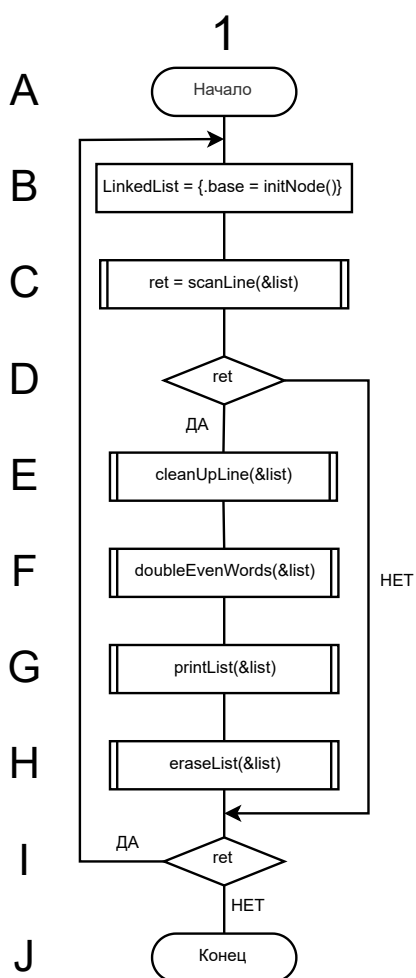


Рис. 1: Блок-схема алгоритма работы функции `main()`

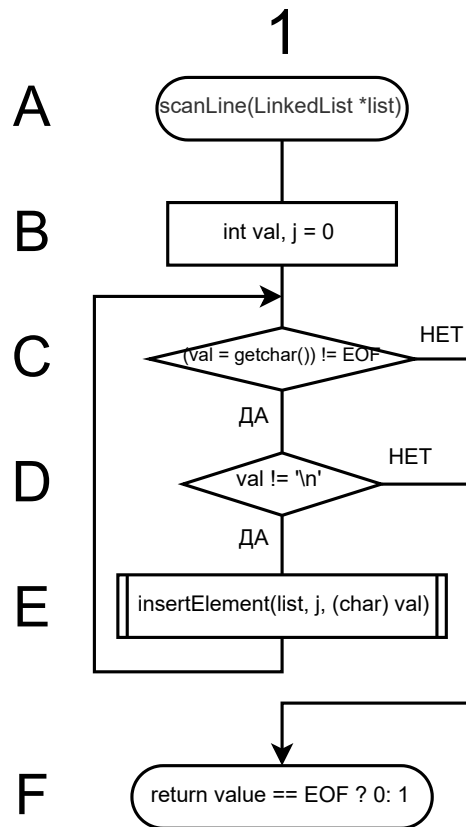


Рис. 2: Блок-схема алгоритма работы функции `scanLine()`

3.2. Описание алгоритма работы со списками

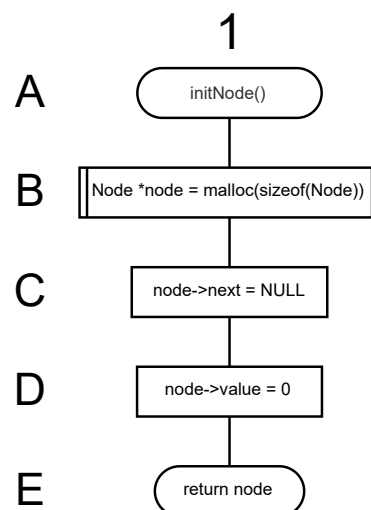


Рис. 3: Блок-схема алгоритма работы функции `initNode()`

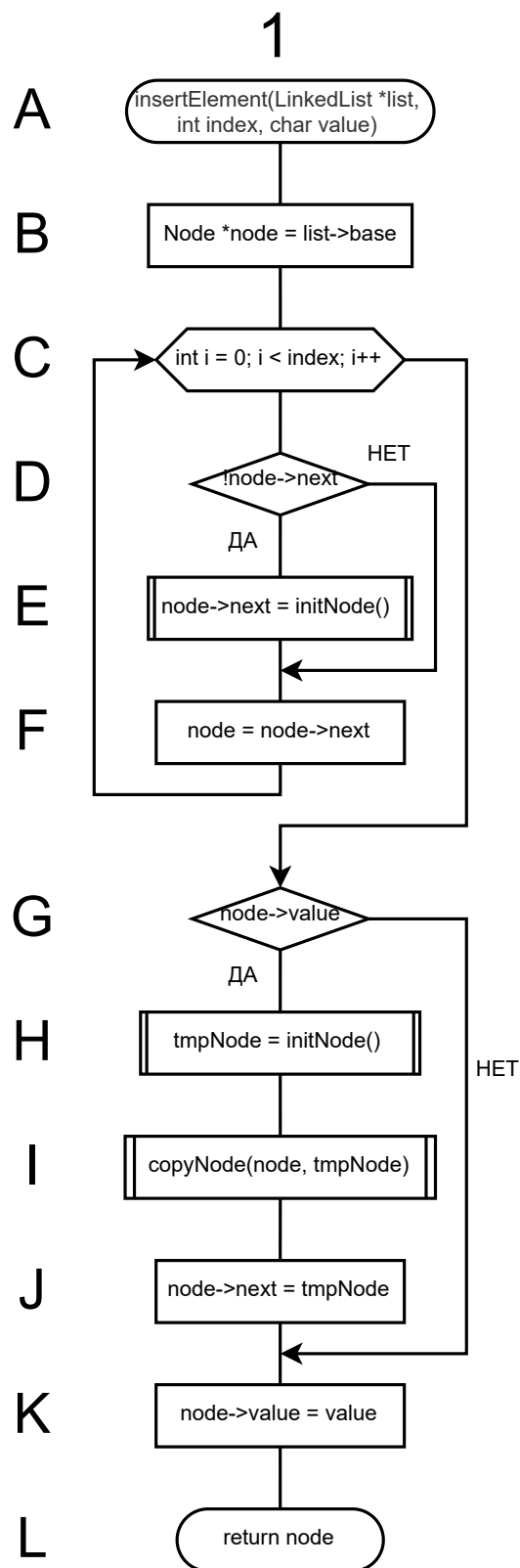


Рис. 4: Блок-схема алгоритма работы функции `insertElement()`

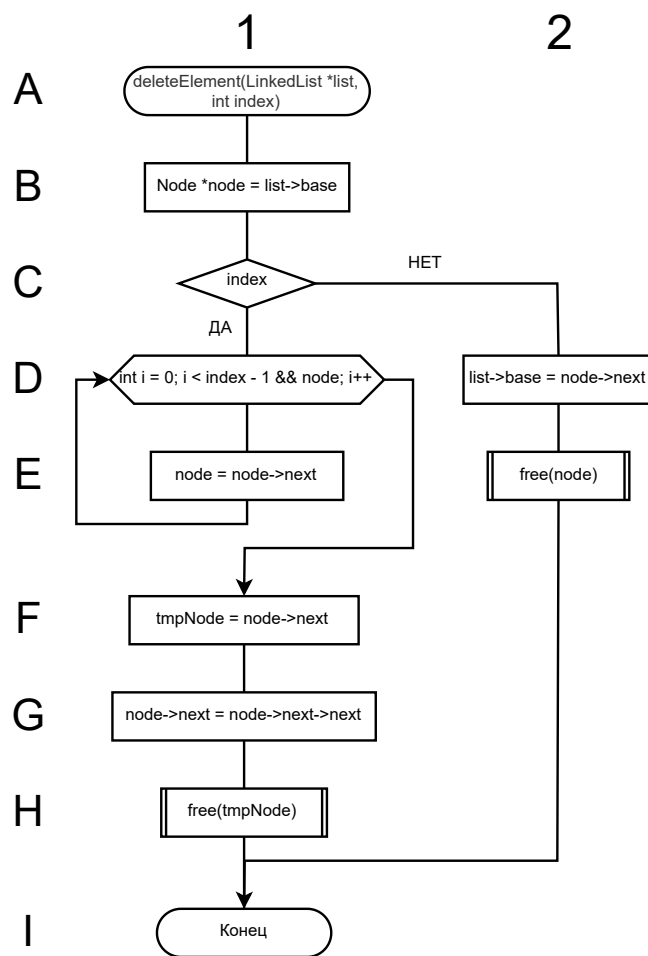


Рис. 5: Блок-схема алгоритма работы функции deleteElement ()

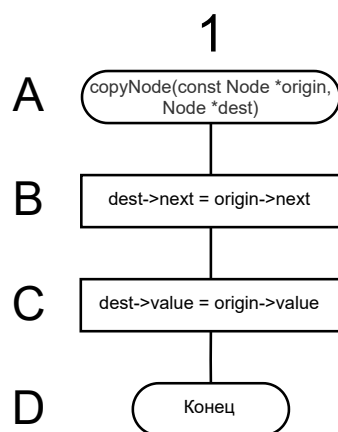


Рис. 6: Блок-схема алгоритма работы функции copyNode ()

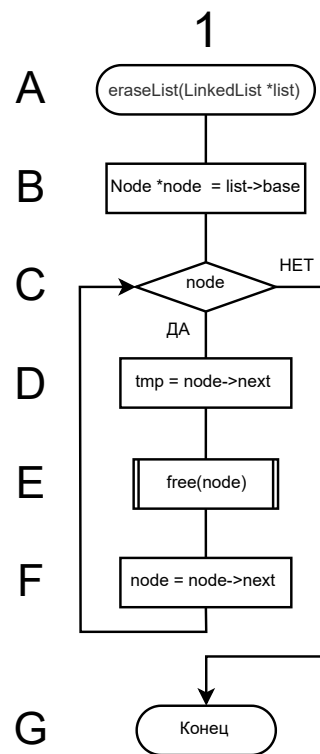


Рис. 7: Блок-схема алгоритма работы функции `eraseList()`

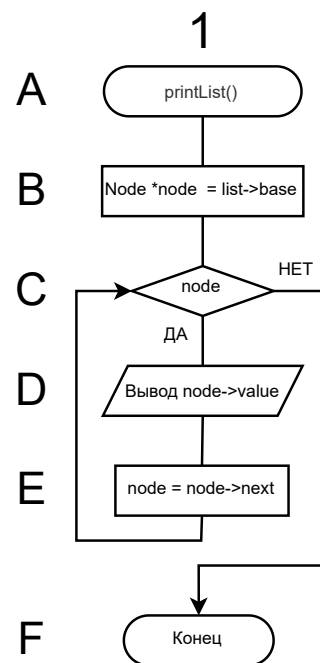


Рис. 8: Блок-схема алгоритма работы функции `printList()`

3.3. Описание алгоритма обработки списка

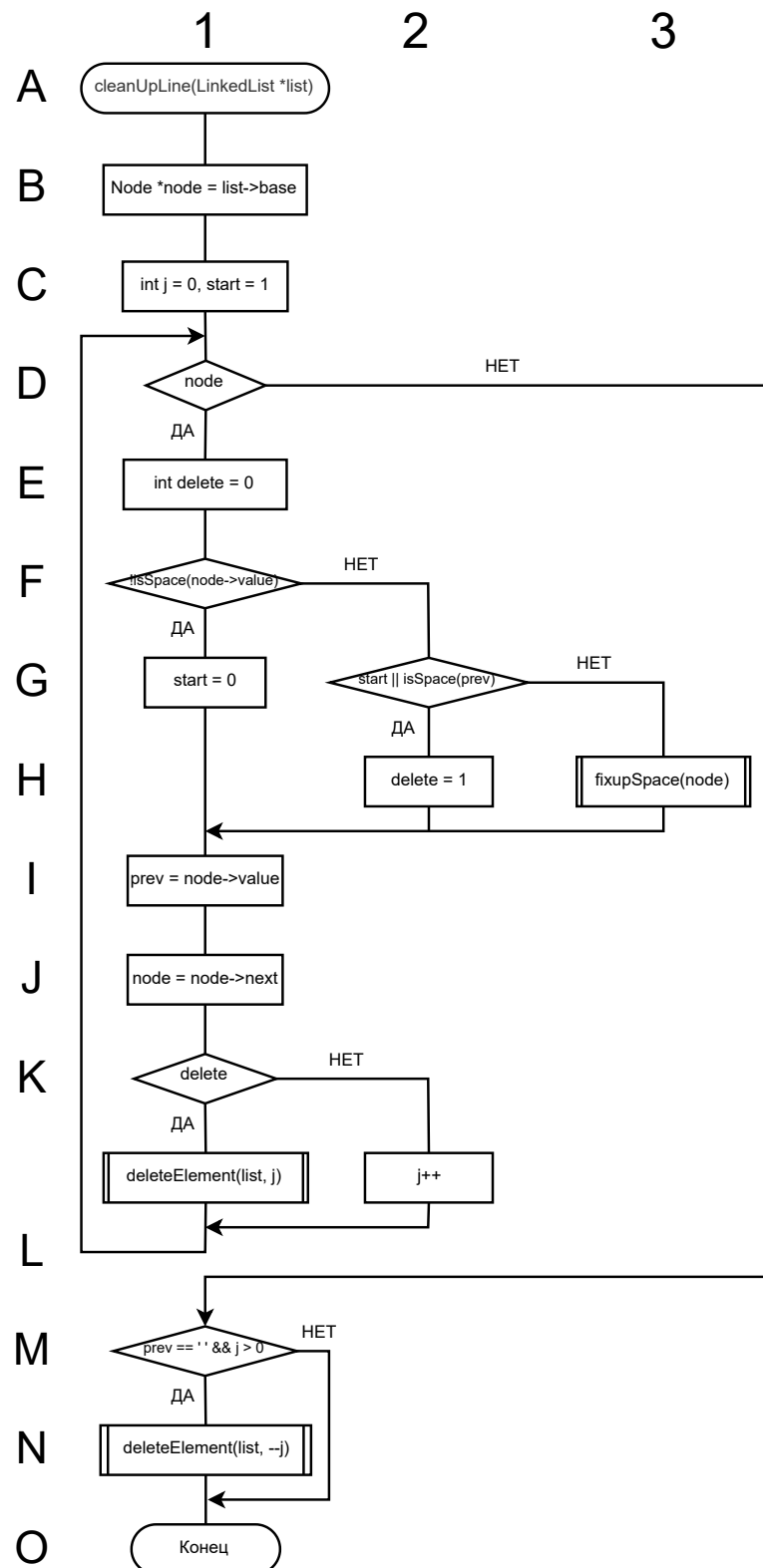


Рис. 9: Блок-схема алгоритма работы функции cleanUpLine()

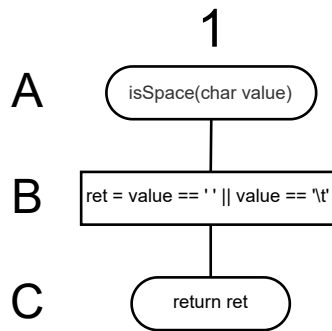


Рис. 11: Блок-схема алгоритма работы функции `isSpace()`

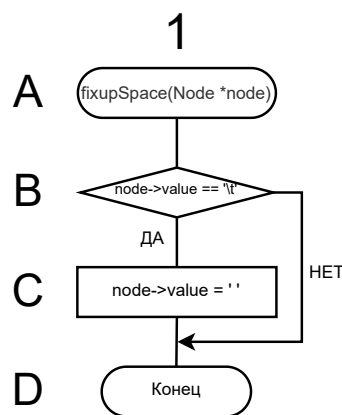


Рис. 12: Блок-схема алгоритма работы функции `fixupSpace()`

4. Исходные коды разработанных программ

4.1. Исходный код основной программы

Исходный код 1: библиотека `main.c`

```

1  #include <stdio.h>
2
3  #include "include/object.h"
4  #include "include/parse.h"
5  #include "include/utils.h"
6
7  int scanLine(LinkedList *list);
8
9  int main() {
10     int ret;
11
12     do {
13         LinkedList list = {.base = initNode()};
14         ret = scanLine(&list);
15         if (ret) {
16             cleanUpLine(&list);
17             doubleEvenWords(&list);
18             printList(&list);
19             printDelimiterMsg();

```

```

20         } else {
21             printExitMsg();
22         }
23         eraseList(&list);
24     } while (ret);
25
26     return 0;
27 }
28
29 int scanLine(LinkedList *list) {
30     int value, j = 0;
31
32     printf("Enter any string: ");
33     while ((value = getchar()) != EOF) {
34         if (value == '\n') break;
35         insertElement(list, j, (char) value);
36         j++;
37     }
38
39     return value == EOF ? 0 : 1;
40 }

```

4.2. Исходный код списка и его функций

Исходный код 2: библиотека object.c.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "include/object.h"
5  #include "include/utils.h"
6
7  Node *insertElement(LinkedList *list, int index, char value) {
8      if (index < 0) {
9          printErrorMsg("Index (%d) is out of bounds!\n", index);
10         return NULL;
11     }
12
13     if (!list->base) {
14         printErrorMsg("Linked list is NOT initialized!\n");
15         return NULL;
16     }
17
18     Node *node = list->base, *tmpNode;
19
20
21     for (int i = 0; i < index; i++) {
22         if (!node->next) {
23             node->next = initNode();
24         }
25         node = node->next;
26     }
27
28     if (node->value) {
29         tmpNode = initNode();
30         copyNode(node, tmpNode);
31         node->next = tmpNode;
32     }
33     node->value = value;
34
35     return node;

```

```

36 }
37
38 void deleteElement(LinkedList *list, int index) {
39     if (index < 0) {
40         printErrorMsg("Index (%d) is out of bounds!\n", index);
41         return;
42     }
43
44     if (!list->base) {
45         printErrorMsg("Linked list is NOT initialized!\n");
46         return;
47     }
48
49     Node *node = list->base, *tmpNode = NULL;
50
51     if (index) {
52         for (int i = 0; i < index - 1 && node; i++) {
53             node = node->next;
54         }
55
56         if (!node || !node->next) {
57             printErrorMsg("Index (%d) is out of bounds!\n", index);
58         } else {
59             tmpNode = node->next;
60             node->next = node->next->next;
61             free(tmpNode);
62         }
63     } else {
64         list->base = node->next;
65         free(node);
66     }
67 }
68
69 void eraseList(LinkedList *list) {
70     Node *node = list->base;
71
72     while (node) {
73         Node *tmp = node->next;
74         free(node);
75         node = tmp;
76     }
77 }
78
79 void printList(LinkedList *list) {
80     Node *node = list->base;
81
82     printf("Result: ");
83     while (node) {
84         if (node->value) printf("%c", node->value);
85         node = node->next;
86     }
87     printf("\n");
88 }
89
90 Node *initNode() {
91     Node *node = malloc(sizeof(Node));
92     node->next = NULL;
93     node->value = 0;
94     return node;
95 }

```

```

96
97 void copyNode(const Node *origin, Node *dest) {
98     dest->next = origin->next;
99     dest->value = origin->value;
100 }

```

Исходный код 3: заголовочный файл include/object.h.

```

1 #ifndef INCLUDE_OBJECT_H
2 #define INCLUDE_OBJECT_H
3
4 typedef struct Node {
5     char value;
6     struct Node *next;
7 } Node;
8
9 typedef struct LinkedList {
10     Node *base;
11 } LinkedList;
12
13 Node *insertElement(LinkedList *list, int index, char value);
14 void deleteElement(LinkedList *list, int index);
15 void eraseList(LinkedList *list);
16 void printList(LinkedList *list);
17 Node *initNode();
18
19 static void copyNode(const Node *origin, Node *dest);
20
21 #endif //INCLUDE_OBJECT_H

```

4.3. Исходный код обработки списка

Исходный код 4: библиотека parse.c.

```

1 #include <stdarg.h>
2 #include <stdio.h>
3
4 #include "include/utils.h"
5
6 void printErrorMsg(const char *msg, ...) {
7     va_list args;
8     va_start(args, msg);
9
10    printf("\033[1;31m");
11    printMsg(msg, args);
12    printf("\033[0m");
13 }
14
15 void printSuccessMsg(const char *msg, ...) {
16     va_list args;
17     va_start(args, msg);
18
19    printf("\033[1;32m");
20    printMsg(msg, args);
21    printf("\033[0m");
22 }
23
24 void printMsg(const char *msg, va_list args) {
25     while (*msg != '\0') {
26         if (*msg != '%') {
27             putchar(*msg);

```

```

28         } else {
29             msg++;
30             if (*msg != '\0') {
31                 switch (*msg) {
32                     case 's':
33                         printf("%s", va_arg(args, char*));
34                         break;
35                     case 'c':
36                         printf("%c", va_arg(args, int));
37                         break;
38                     case 'd':
39                         printf("%d", va_arg(args, int));
40                     default:
41                         break;
42                 }
43             }
44         }
45         msg++;
46     }
47 }
48
49 void printDelimiterMsg() {
50     printSuccessMsg("*****\n");
51 }
52
53 void printExitMsg() {
54     printSuccessMsg("\nGoodbye...\n\n");
55 }

```

Исходный код 5: заголовочный файл include/parse.h.

```

1  #ifndef INCLUDE_PARSE_H
2  #define INCLUDE_PARSE_H
3
4  #include "object.h"
5
6  void cleanUpLine(LinkedList *list);
7  void doubleEvenWords(LinkedList *list);
8
9  static int isSpace(char value);
10 static void fixupSpace(Node *node);
11
12 #endif //INCLUDE_PARSE_H

```

4.4. Исходный код вспомогательной библиотеки

Исходный код 6: библиотека utils.c.

```

1  #include <stdarg.h>
2  #include <stdio.h>
3
4  #include "include/utils.h"
5
6  void printErrorMsg(const char *msg, ...) {
7      va_list args;
8      va_start(args, msg);
9
10     printf("\033[1;31m");
11     printMsg(msg, args);
12     printf("\033[0m");
13 }

```

```

14
15 void printSuccessMsg(const char *msg, ...) {
16     va_list args;
17     va_start(args, msg);
18
19     printf("\033[1;32m");
20     printMsg(msg, args);
21     printf("\033[0m");
22 }
23
24 void printMsg(const char *msg, va_list args) {
25     while (*msg != '\0') {
26         if (*msg != '%') {
27             putchar(*msg);
28         } else {
29             msg++;
30             if (*msg != '\0') {
31                 switch (*msg) {
32                     case 's':
33                         printf("%s", va_arg(args, char*));
34                         break;
35                     case 'c':
36                         printf("%c", va_arg(args, int));
37                         break;
38                     case 'd':
39                         printf("%d", va_arg(args, int));
40                     default:
41                         break;
42                 }
43             }
44         }
45         msg++;
46     }
47 }
48
49 void printDelimiterMsg() {
50     printSuccessMsg("*****\n");
51 }
52
53 void printExitMsg() {
54     printSuccessMsg("\nGoodbye...\n\n");
55 }

```

Исходный код 7: заголовочный файл include/utils.h.

```

1 #ifndef INCLUDE_UTILS_H
2 #define INCLUDE_UTILS_H
3
4 void printErrorMsg(const char *, ...);
5 void printSuccessMsg(const char *msg, ...);
6 void printMsg(const char *msg, va_list args);
7 void printDelimiterMsg();
8 void printExitMsg();
9
10 #endif //INCLUDE_UTILS_H

```

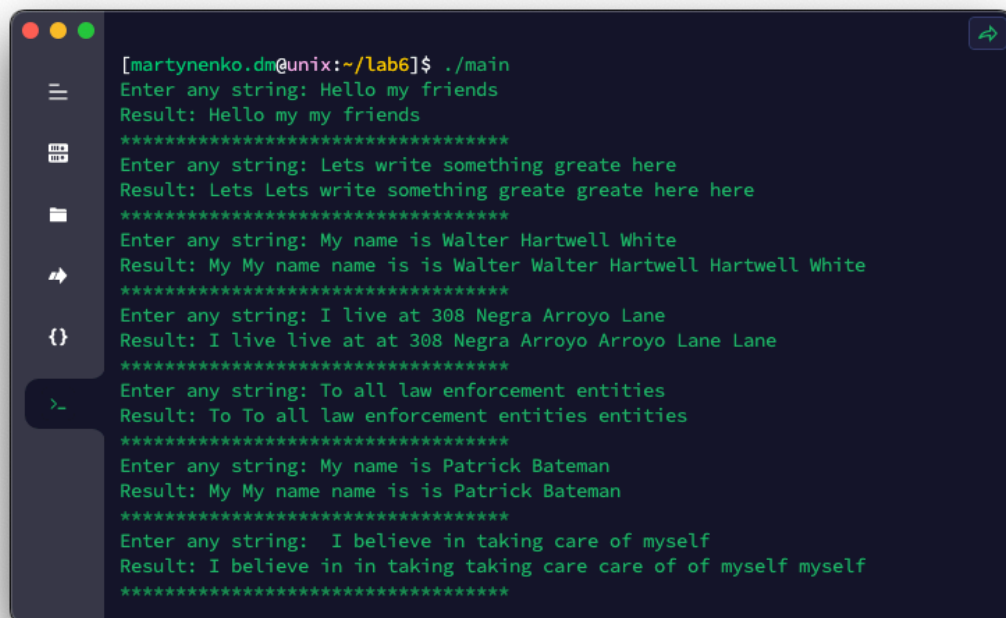
5. Описание тестовых примеров

Таблица 1: Тестовые примеры

Входная строка	Ожидаемое значение	Полученное значение
The memory we used to share is no longer coherent	The memory memory we we used used to to share is is no no longer longer coherent coherent	The memory memory we we used used to to share is is no no longer longer coherent coherent
It was the scarcity that fueled his creativity	It It was the scarcity scarcity that that fueled fueled his creativity creativity	It It was the scarcity scarcity that that fueled fueled his creativity creativity
This is a test string	This This is is a test test string string	This This is is a test test string string
Riddle me this, Riddle me that	Riddle Riddle me me this, Riddle Riddle me me that that	Riddle Riddle me me this, Riddle Riddle me me that that
Whos afraid of the Big Black	Whos Whos afraid afraid of of the Big Black	Whos Whos afraid afraid of of the Big Black

6. Скриншоты

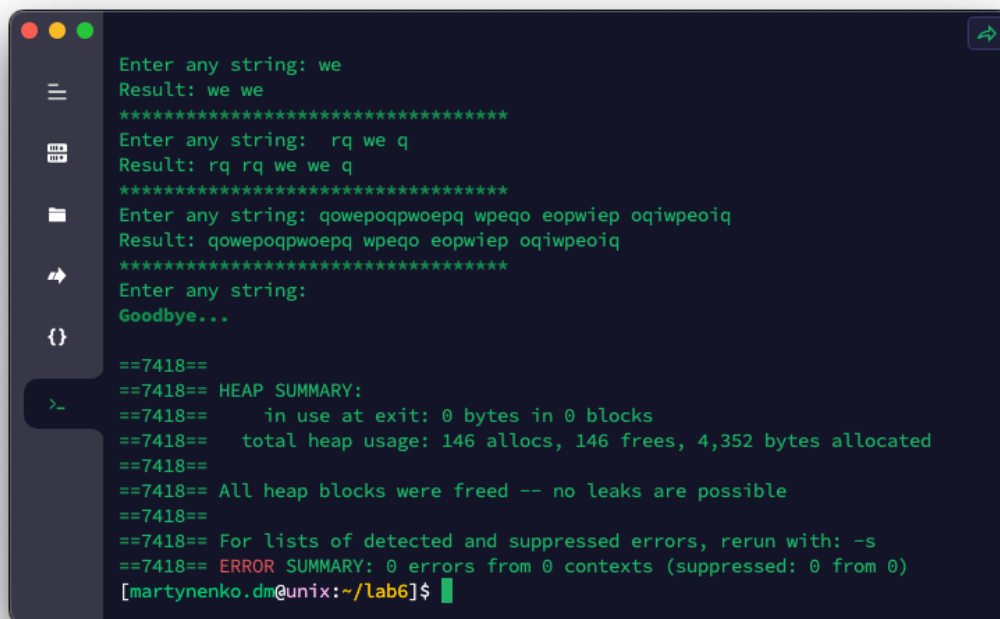
6.1. Программа



```
[martynenko.dn@unix:~/lab6]$ ./main
Enter any string: Hello my friends
Result: Hello my my friends
*****
Enter any string: Lets write something greate here
Result: Lets Lets write something greate greate here here
*****
Enter any string: My name is Walter Hartwell White
Result: My My name name is is Walter Walter Hartwell Hartwell White
*****
Enter any string: I live at 308 Negra Arroyo Lane
Result: I live live at at 308 Negra Arroyo Arroyo Lane Lane
*****
Enter any string: To all law enforcement entities
Result: To To all law enforcement entities entities
*****
Enter any string: My name is Patrick Bateman
Result: My My name name is is Patrick Bateman
*****
Enter any string: I believe in taking care of myself
Result: I believe in in taking taking care care of of myself myself
*****
```

Рис. 13: Работа программы.

6.2. Использование памяти



```
Enter any string: we
Result: we we
*****
Enter any string:  rq we q
Result: rq rq we we q
*****
Enter any string: qoweroqrwoerpq wrepo eorwier oqiwrpeoiq
Result: qoweroqrwoerpq wrepo eorwier oqiwrpeoiq
*****
Enter any string:
Goodbye...

==7418==
==7418== HEAP SUMMARY:
==7418==      in use at exit: 0 bytes in 0 blocks
==7418==    total heap usage: 146 allocs, 146 frees, 4,352 bytes allocated
==7418==
==7418== All heap blocks were freed -- no leaks are possible
==7418==
==7418== For lists of detected and suppressed errors, rerun with: -s
==7418== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[martynenko.dm@unix:~/lab6]$
```

Рис. 14: Успешная проверка использования памяти при помощи valgrind.

7. Выводы

В ходе выполнения данной работы были:

1. Изучены основные принципы работы со списками.
2. Создана своя реализация списков на языке программирования Си.
3. Изучен организации ввода текста при помощи функции `getchar()`.