Вторая программа

Во второй программе выполняется: чтение последовательности из файла и вывод на экран информации о том, является ли последовательность палиндромом. Для работы с последовательностью используется список.

Данная программа выполнялась на лабораторной работе в курсе "Программирование на языке C".

Листинг 1 – list.c

```
1  #include "../inc/list.h"
2
3  node_t *reverse(node_t *head)
4  {
5      if (!head)
6          return NULL;
7
8      node_t *result = NULL;
9      node_t *node;
10
11     while (head != NULL)
12     {
13         void *element = pop_front(&head);
14         node = create_node(element);
15         add_node(&result, node);
16     }
17
18     return result;
19 }
20
21
22 void sorted_insert(node_t **head, node_t *element,
23 int (*comparator)(const void *, const void *))
24 {
25     element->next = NULL;
26
27     if (!(*head))
28     {
29         element->next = *head;
30         *head = element;
```

```
31
32          return;
33      }
34
35      char *first, *second = element->data;
36
37      if ((*head)->next == NULL)
38      {
39          first = (*head)->data;
40
41          if (comparator(first, second) <= 0)
42          {
43              (*head)->next = element;
44              element->next = NULL;
45
46              return;
47          }
48
49          element->next = *head;
50          *head = element;
51
52          return;
53      }
54
55      node_t *node = *head, *add;
56      first = node->data;
57
58      if (comparator(first, second) > 0)
59      {
60          add = *head;
61          *head = element;
62          element->next = add;
63
64          return;
65      }
66
67      while (node->next)
68      {
69          first = node->next->data;
70
71          if (comparator(first, second) > 0)
```

```
72          {
73              add = node->next;
74              node->next = element;
75              element->next = add;
76
77              return;
78          }
79
80          node = node->next;
81      }
82
83      node->next = element;
84  }
85
86
87  node_t *sort(node_t *head, int (*comparator)(const void *, const
        void *))
88  {
89      if (!head)
90          return NULL;
91
92      if (!comparator)
93          return NULL;
94
95      node_t *result = NULL;
96
97      while (head != NULL)
98      {
99          node_t *node = head;
100         head = head->next;
101         sorted_insert(&result, node, comparator);
102     }
103
104     return result;
105 }
106
107
108 int compare_int(const void *first, const void *second)
109 {
110     return *(int *)first - *(int *)second;
111 }
```

Листинг 2 – node.c

```c
#include "../inc/node.h"

node_t *create_node(void *data)
{
    node_t *node = malloc(sizeof(node_t));

    if (node)
    {
        node->data = data;
        node->next = NULL;
    }

    return node;
}


void add_node(node_t **head, node_t *node)
{
    node->next = *head;
    *head = node;
}


void *pop_front(node_t **head)
{
    if (!head || !(*head))
        return NULL;

    node_t *node = (*head);
    void *data = node->data;
    (*head) = (*head)->next;

    free(node);

    return data;
}


void *pop_back(node_t **head)
{
```

```
41      if (!head || !(*head))
42          return NULL;
43
44      node_t *node = *head, *prev_node = NULL;
45
46      for (; node->next; node = node->next)
47          prev_node = node;
48
49      void *data = node->data;
50
51      if (prev_node)
52          prev_node->next = node->next;
53      else
54          *head = node->next;
55
56      free(node);
57
58      return data;
59  }
```

Листинг 3 – sequence.c

```
1 #include "../inc/sequence.h"
2
3 void init_sequence(sequence_t *sequence)
4 {
5     sequence->count = 0;
6     sequence->head = NULL;
7 }
8
9
10 int get_count_elements(FILE *in_file, int *count)
11 {
12     int element;
13
14     while (fscanf(in_file, "%d", &element) == 1)
15         (*count)++;
16
17     if (!*count)
18         return EMPTY_FILE;
19
20     return OK;
```

```
21 }
22
23
24 int read_sequence(const char *filename, sequence_t *sequence,
25 int **storage)
26 {
27     FILE *in_file = fopen(filename, "r");
28
29     if (!in_file)
30     {
31         LOG_ERROR("%s", "File open error");
32         return ERROR_FILE_OPEN;
33     }
34
35     init_sequence(sequence);
36
37     if (get_count_elements(in_file, &sequence->count))
38         return EMPTY_FILE;
39
40     rewind(in_file);
41
42     *storage = malloc(sequence->count * sizeof(int));
43
44     if (!*storage)
45         return ALLOCATE_ERROR;
46
47     int element;
48     int i = 0;
49
50     while (fscanf(in_file, "%d", &element) == 1)
51     {
52         (*storage)[i] = element;
53         i++;
54     }
55
56     create_sequence(sequence, *storage);
57
58     if (fclose(in_file) != OK)
59     {
60         free_sequence(sequence->head);
61         free(*storage);
```

```
62
63          LOG_ERROR("%s", "File close error");
64          return ERROR_FILE_CLOSE;
65      }
66
67      return OK;
68  }
69
70
71  void create_sequence(sequence_t *sequence, int *storage)
72  {
73      for (int i = 0; i < sequence->count; i++)
74      {
75          node_t *node = create_node(&storage[i]);
76          add_node(&(sequence->head), node);
77      }
78  }
79
80
81  void free_sequence(node_t *head)
82  {
83      node_t *node;
84
85      for (; head; head = node)
86      {
87          node = head->next;
88          free(head);
89      }
90  }
91
92
93  int is_palindrome(sequence_t *sequence)
94  {
95      int repeats = sequence->count / 2;
96      char *start, *end;
97
98      while (repeats > 0)
99      {
100         start = pop_front(&sequence->head);
101         end = pop_back(&sequence->head);
102
```

```
103             if (compare_int(start, end) != EQUAL)
104                 return NOT_PALINDROME;
105 
106             repeats--;
107         }
108 
109         return PALINDROME;
110 }
111 
112 
113 node_t* process_sequence(const char *filename, sequence_t
        *sequence,
114 int *storage, int (*comparator)(const void *, const void *))
115 {
116         int code = is_palindrome(sequence);
117 
118         sequence_t add_sequence;
119         init_sequence(&add_sequence);
120         add_sequence.count = sequence->count;
121 
122         free_sequence(sequence->head);
123 
124         create_sequence(&add_sequence, storage);
125         node_t *result;
126 
127         if (code == PALINDROME)
128             result = sort(add_sequence.head, comparator);
129         else
130             result = reverse(add_sequence.head);
131 
132         return result;
133 }
134 
135 
136 int write_sequence(const char *filename, sequence_t *sequence)
137 {
138         FILE *out_file = fopen(filename, "w");
139 
140         if (!out_file)
141         {
142             LOG_INFO("%s", "File open error");
```

```
143             return ERROR_FILE_OPEN;
144         }
145
146         int size = sequence->count;
147
148         while (size > 0)
149         {
150             int *number = (int *)pop_front(&sequence->head);
151             size--;
152             fprintf(out_file, "%d\n", *number);
153         }
154
155         if (fclose(out_file) != OK)
156         {
157             free_sequence(sequence->head);
158             return ERROR_FILE_CLOSE;
159         }
160
161         return OK;
162 }
163
164 void print_sequence(sequence_t *sequence)
165 {
166         int size = sequence->count;
167
168         while (size > 0)
169         {
170             int *number = (int *)pop_front(&sequence->head);
171             size--;
172             printf("%d ", *number);
173         }
174 }
```

Листинг 4 – main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "../inc/node.h"
#include "../inc/sequence.h"
#include "../inc/list.h"

int main(int argc, char **argv)
{
    if (argc != COUNT_ARGC)
        return ERROR_ARGC;

    sequence_t sequence, add_sequence;
    int *storage = NULL;

    int code = read_sequence(argv[1], &sequence, &storage);

    if (code)
        return code;

    init_sequence(&add_sequence);
    add_sequence.count = sequence.count;
    create_sequence(&add_sequence, storage);

    print_sequence(&sequence);

    if (is_palindrome(&add_sequence))
        printf("is palindrome\n");
    else
        printf("is not palindrome\n");

    free(storage);

    return OK;
}
```

Листинг 5 – list.h

```
#ifndef LIST_H
#define LIST_H

#include <stdio.h>
#include <stdlib.h>

#include "node.h"

node_t *reverse(node_t *head);

void sorted_insert(node_t **head, node_t *element,
int (*comparator)(const void *, const void *));

node_t *sort(node_t *head, int (*comparator)(const void *, const
    void *));

int compare_int(const void *first, const void *second);

#endif
```

Листинг 6 – node_t.h

```
#ifndef NODE_T_H
#define NODE_T_H

typedef struct node node_t;

struct node
{
    void *data;
    node_t *next;
};

#endif
```

Листинг 7 – node.h

```
#ifndef NODE_H
#define NODE_H

#include <stdio.h>
#include <stdlib.h>

#include "node_t.h"
#include "macrologger.h"

node_t *create_node(void *data);

void add_node(node_t **head, node_t *node);

void *pop_front(node_t **head);

void *pop_back(node_t **head);

#endif
```

Листинг 8 – sequence_t.h

```
#ifndef SEQUENCE_T_H
#define SEQUENCE_T_H

#include "../inc/node_t.h"

typedef struct sequence_t
{
    int count;
    node_t *head;
} sequence_t;

#endif
```

Листинг 9 – sequence.h

```
#ifndef SEQUENCE_H
#define SEQUENCE_H

#include <stdio.h>
#include <stdlib.h>

#include "list.h"
#include "return_codes.h"
#include "constants.h"
#include "macrologger.h"
#include "sequence_t.h"
#include "node_t.h"
#include "node.h"

void init_sequence(sequence_t *sequence);

int read_sequence(const char *filename, sequence_t *sequence,
int **storage);

void create_sequence(sequence_t *sequence, int *storage);

void free_sequence(node_t *head);

int is_palindrome(sequence_t *sequence);

node_t *process_sequence(const char *filename, sequence_t
   *sequence,
int *storage, int (*comparator)(const void *, const void *));

int write_sequence(const char *filename, sequence_t *sequence);

void print_sequence(sequence_t *sequence);

#endif
```

Листинг 10 – constants.h

```
#ifndef CONSTANTS_H
#define CONSTANTS_H

#define COUNT_ARGC      2

#define PALINDROME      1
#define NOT_PALINDROME  0
#define EQUAL           0

#endif
```

Листинг 11 – return_codes.h

```
#ifndef RETURN_CODES_H
#define RETURN_CODES_H

#define OK                  0
#define ERROR_ARGC          1
#define ERROR_FILE_OPEN     2
#define ERROR_DATA_TYPE     3
#define ERROR_FILE_CLOSE    4
#define ALLOCATE_ERROR      5
#define EMPTY_FILE          6

#endif
```