Первая программа

В первой программе выполняется: чтение массива из файла, сортировка массива по возрастанию и вывод массива на экран.

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

Листинг 1 – array.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "../inc/array.h"


int get_array_len(FILE *in_file, array_t *const array)
{
    if (array == NULL)
    {
        LOG_ERROR("%s", "Invalid pointer");
        return ERROR_POINTER;
    }

    int array_len = 0, number;

    if (fscanf(in_file, "%d", &number) == 1)
    {
        array_len++;

        while (fscanf(in_file, "%d", &number) == 1)
            array_len++;

        array->len = array_len;

        return OK;
    }

    LOG_ERROR("%s", "Data read error");
    return DATA_ERROR;
}

```

```
32
33 int fill_array(FILE *in_file, int *const first, int *const last)
34 {
35     if (first == NULL || last == NULL)
36     {
37         LOG_ERROR("%s", "Invalid pointer");
38         return ERROR_POINTER;
39     }
40
41     if (first == last)
42     {
43         LOG_ERROR("%s", "Empty array");
44         return EMPTY_ARRAY;
45     }
46
47     int number;
48
49     for (int *array_pointer = first; array_pointer != last;
          array_pointer++)
50     {
51         if (fscanf(in_file, "%d", &number))
52             *array_pointer = number;
53     }
54
55     return OK;
56 }
57
58
59 void even_passage(char *start, char *end, size_t size, int
       (*compar)(const void *, const void *))
60 {
61     for (char *array_pointer = end; array_pointer > start;
          array_pointer -= size)
62         if (compar(array_pointer, array_pointer - size) <= 0)
63             swap(size, array_pointer, array_pointer - size);
64 }
65
66
67 void odd_passage(char *start, char *end, size_t size, int
       (*compar)(const void *, const void *))
68 {
```

```
69      for (char *array_pointer = start; array_pointer < end;
            array_pointer += size)
70          if (compar(array_pointer, array_pointer + size) >= 0)
71              swap(size, array_pointer, array_pointer + size);
72  }
73
74
75  void mysort(void *base, size_t nmemb, size_t size, int
        (*compar)(const void *, const void *))
76  {
77      if (base == NULL || nmemb <= 0)
78          return;
79
80      char *start = base;
81      char *end = start + size * (nmemb - 1);
82
83      for (size_t i = 0; i < nmemb; i++)
84      {
85          if (!(i % 2))
86              even_passage(start, end, size, compar);
87          else
88              odd_passage(start, end, size, compar);
89      }
90  }
91
92
93  int key(const int *pb_src, const int *pe_src, int **pb_dst, int
        **pe_dst)
94  {
95      if (pb_src == NULL || pe_src == NULL)
96      {
97          LOG_ERROR("%s", "Invalid pointer");
98          return ERROR_POINTER;
99      }
100
101     if (pb_src == pe_src)
102     {
103         LOG_ERROR("%s", "Empty array");
104         return EMPTY_ARRAY;
105     }
106
```

```
107
108     int filter_len = 0;
109
110     for (const int *array_pointer = pb_src; array_pointer <
            pe_src - 1; array_pointer++)
111     {
112         if (sum_numbers(array_pointer + 1, pe_src) <
                *array_pointer)
113             filter_len++;
114     }
115
116     if (!filter_len)
117     {
118         LOG_ERROR("%s", "There are no such elements");
119         return ERROR_FILTER;
120     }
121     LOG_INFO("Count such elements is %d", filter_len);
122
123     *pb_dst = malloc(filter_len * sizeof(int));
124
125     if (pb_dst == NULL)
126     {
127         LOG_ERROR("%s", "Memory allocation error");
128         return MEMORY_ERROR;
129     }
130     LOG_INFO("%s", "New array created");
131
132     *pe_dst = *pb_dst + filter_len;
133
134     int *new_array_pointer = *pb_dst;
135     for (const int *array_pointer = pb_src; array_pointer <
            pe_src - 1; array_pointer++)
136     {
137         if (sum_numbers(array_pointer + 1, pe_src) <
                *array_pointer)
138         {
139             *new_array_pointer = *array_pointer;
140             new_array_pointer++;
141         }
142     }
143     LOG_INFO("%s", "Numbers written");
```

```
144
145     return OK;
146 }
147
148
149 void write_array_file(FILE *out_file, const int *const first,
        const int *const second)
150 {
151     for (const int *pa = first; pa != second; pa++)
152         fprintf(out_file, "%d ", *pa);
153 }
154
155 void print_array(const int *const first, const int *const second)
156 {
157     for (const int *pa = first; pa != second; pa++)
158         printf("%d ", *pa);
159
160     printf("\n");
161 }
```

Листинг 2 – numbers.c

```
1 #include <stdio.h>
2 #include "../inc/numbers.h"
3
4
5 int compare_int(const void *first, const void *second)
6 {
7     if (first == NULL || second == NULL)
8     {
9         LOG_ERROR("%s", "Invalid pointer");
10        return ERROR_POINTER;
11    }
12
13    return *(int *)first - *(int *)second;
14 }
15
16
17 void swap(size_t size, char *number_one, char *number_two)
18 {
19     char add_number;
20
```

```
21      while (size > 0)
22      {
23          add_number = *number_one;
24          *number_one = *number_two;
25          *number_two = add_number;
26
27          number_one++;
28          number_two++;
29          size--;
30      }
31  }
32
33
34  int sum_numbers(const int *first, const int *last)
35  {
36      if (first == NULL || last == NULL)
37      {
38          LOG_ERROR("%s", "Invalid pointer");
39          return ERROR_POINTER;
40      }
41
42      if (first == last)
43      {
44          LOG_ERROR("%s", "Empty array");
45          return EMPTY_ARRAY;
46      }
47
48      int sum = 0;
49
50      for (const int *array_pointer = first; array_pointer < last;
            array_pointer++)
51          sum += *array_pointer;
52
53      return sum;
54  }
```

Листинг 3 – main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../inc/array_t.h"
#include "../inc/array.h"
#include "../inc/numbers.h"
#include "../inc/errors.h"
#include "../inc/macrologger.h"

int main(int argc, char **argv)
{
    if (argc < 2 || argc > 3)
    {
        LOG_ERROR("%s", "Invalid number of arguments");
        return ERROR_ARGC;
    }


    FILE *in_file = fopen(argv[1], "r");

    if (in_file == NULL)
    {
        LOG_ERROR("%s", "File open error");
        return FILE_OPEN_ERROR;
    }
    LOG_INFO("%s", "Input file open");


    array_t array;
    int code = get_array_len(in_file, &array);

    if (code != OK)
        return code;
    LOG_INFO("Array len is %d", array.len);

    array.start = malloc(array.len * sizeof(int));

    if (array.start == NULL)
    {
        LOG_ERROR("%s", "Memory allocation error");
```

```
41          return MEMORY_ERROR;
42      }
43      LOG_INFO("%s", "Array created");
44
45      rewind(in_file);
46      code = fill_array(in_file, array.start, array.start +
            array.len);
47
48      if (code != OK)
49      {
50          free(array.start);
51          return code;
52      }
53      LOG_INFO("%s", "Array filled");
54
55
56      if (fclose(in_file))
57      {
58          free(array.start);
59
60          LOG_ERROR("%s", "File close error");
61          return FILE_CLOSE_ERROR;
62      }
63      LOG_INFO("%s", "Input file close");
64
65
66      if (argc == 2)
67      {
68          printf("Before sort: ");
69          print_array(array.start, array.start + array.len);
70          mysort(array.start, array.len, sizeof(int), compare_int);
71          LOG_INFO("%s", "Array is sorted");
72
73          printf("After sort: ");
74          print_array(array.start, array.start + array.len);
75          free(array.start);
76      }
77
78
79      if (argc == 3)
80      {
```

```
81          if (strcmp("f", argv[2]) != OK)
82          {
83              free(array.start);
84              LOG_ERROR("%s", "Invalid command");
85              return ERROR_COMMAND;
86          }
87
88
89          array_t new_array;
90          int *end;
91
92          code = key(array.start, array.start + array.len,
                &new_array.start, &end);
93          free(array.start);
94
95          if (code != OK)
96              return code;
97          LOG_INFO("%s", "Array filtered");
98
99          new_array.len = end - new_array.start;
100         LOG_INFO("New array len is %d", new_array.len);
101
102         mysort(new_array.start, new_array.len, sizeof(int),
                compare_int);
103         LOG_INFO("%s", "Array is sorted");
104
105         print_array(new_array.start, new_array.start +
                new_array.len);
106         free(new_array.start);
107     }
108     LOG_INFO("%s", "Numbers written to file");
109
110     return OK;
111 }
```

Листинг 4 – array_t.h

```
#ifndef ARRAY_T_H
#define ARRAY_T_H

typedef struct
{
    int *start;
    int len;
} array_t;

#endif
```

Листинг 5 – array.h

```
#ifndef ARRAY_H
#define ARRAY_H

#include "array_t.h"
#include "numbers.h"
#include "errors.h"
#include "macrologger.h"


int get_array_len(FILE *in_file, array_t *const array);

int fill_array(FILE *in_file, int *const first, int *const last);

void mysort(void *base, size_t nmemb, size_t size, int
    (*compar)(const void *, const void *));

int key(const int *pb_src, const int *pe_src, int **pb_dst, int
    **pe_dst);

void write_array_file(FILE *out_file, const int *const first,
    const int *const second);

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

#endif
```

Листинг 6 – numbers.h

```
#ifndef NUMBERS_H
#define NUMBERS_H

#include "errors.h"
#include "macrologger.h"

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

void swap(size_t size, char *number_one, char *number_two);

int sum_numbers(const int *first, const int *last);

#endif
```

Листинг 7 – errors.h

```
#ifndef ERRORS_H
#define ERRORS_H

#define ERROR_ARGC 1
#define FILE_OPEN_ERROR 2
#define MEMORY_ERROR 3
#define FILE_CLOSE_ERROR 4
#define DATA_ERROR 5
#define ERROR_FILTER 6
#define ERROR_COMMAND 7
#define ERROR_POINTER 8
#define EMPTY_ARRAY 9

#define OK 0

#endif
```