

Московский авиационный институт  
(Национальный исследовательский университет)  
Факультет "Информационные технологии и прикладная математика"  
Кафедра "Вычислительная математика и программирование"

**Лабораторная работа №5 по курсу  
“Операционные системы”**

*Студент:* Живалев Е.А.

*Группа:* М8О-206Б

*Преподаватель:* Соколов А.А.

*Вариант:* 7

*Оценка:* \_\_\_\_\_

*Дата:* \_\_\_\_\_

*Подпись:* \_\_\_\_\_

Москва, 2019

# 1 Задание

Требуется создать динамическую библиотеку, которая реализует определенный Функционал - работу с массивом, содержащим целые 32-битные числа.

## 2 Описание работы программы

В файле `arrayAPI.c` реализованы следующие функции для работы с массивом: `arrayCreate`, `arrayInsert`, `arrayGet`, `arrayDelete`, `arrayResize`, `arrayDestroy`, `arrayPrint`. В файле `main_dynamic.c` эти функции загружаются в память, выделенную для программы

### 3 Исходный код

#### main\_dynamic.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <dlfcn.h>
4
5 #include "arrayAPI.h"
6
7 int main() {
8     array* (*arrayCreate)(size_t size);
9     void (*arrayInsert)(array* a, size_t pos, int val);
10    int (*arrayGet)(array* a, size_t pos);
11    void (*arrayDelete)(array* a, size_t pos);
12    void (*arrayResize)(array* a, size_t size);
13    void (*arrayDestroy)(array* a);
14    void (*arrayPrint)(array* a);
15    char* error;
16    void* libHandle;
17    libHandle = dlopen("./libarr.so", RTLD_LAZY);
18    if(!libHandle) {
19        printf("%s\n", dlerror());
20        return -1;
21    }
22    arrayCreate = dlsym(libHandle, "arrayCreate");
23    arrayInsert = dlsym(libHandle, "arrayInsert");
24    arrayGet = dlsym(libHandle, "arrayGet");
25    arrayDelete = dlsym(libHandle, "arrayDelete");
26    arrayResize = dlsym(libHandle, "arrayResize");
27    arrayDestroy = dlsym(libHandle, "arrayDestroy");
28    arrayPrint = dlsym(libHandle, "arrayPrint");
29    printf("1 - create array with given size\n");
30    printf("2 - insert element to array at given position\n");
31    printf("3 - get element value on given position\n");
32    printf("4 - delete element from given position\n");
33    printf("5 - resize array to given size\n");
34    printf("6 - print array\n");
35    printf("0 - exit\n");
36    int command = 0;
37    array* a;
38    int value = 0;
39    int pos = 0;
40    while(scanf("%d", &command) && command) {
41        if(command == 1) {
42            printf("Enter value\n");
43            scanf("%d", &value);
44            a = (*arrayCreate)(value);
45        } else if(command == 2) {
46            printf("Enter position\n");
47            scanf("%d %d", &pos, &value);
48            (*arrayInsert)(a, pos, value);
49        } else if(command == 3) {
50            printf("Enter position\n");
51            scanf("%d", &pos);
52            printf("%d\n", (*arrayGet)(a, pos));
53        } else if(command == 4) {
54            printf("Enter position\n");
55            scanf("%d", &pos);
```

```

56         (*arrayDelete)(a, pos);
57     } else if(command == 5) {
58         printf("Enter value\n");
59         scanf("%d", &value);
60         (*arrayResize)(a, value);
61     } else if(command == 6) {
62         (*arrayPrint)(a);
63     }
64 }
65 (*arrayDestroy)(a);
66 return 0;
67 }

```

## main\_static.c

```

1  #include <stdio.h>
2
3  #include "arrayAPI.h"
4
5  int main() {
6      int command = 0;
7      array* a;
8      int value = 0;
9      int pos = 0;
10     printf("1 - create array with given size\n");
11     printf("2 - insert element to array at given position\n");
12     printf("3 - get element value on given position\n");
13     printf("4 - delete element from given position\n");
14     printf("5 - resize array to given size\n");
15     printf("6 - print array\n");
16     printf("0 - exit\n");
17     while(scanf("%d", &command) && command) {
18         if(command == 1) {
19             printf("Enter value\n");
20             scanf("%d", &value);
21             a = arrayCreate(value);
22         } else if(command == 2) {
23             printf("Enter position and value\n");
24             scanf("%d %d", &pos, &value);
25             arrayInsert(a, pos, value);
26         } else if(command == 3) {
27             printf("Enter position\n");
28             scanf("%d", &pos);
29             printf("%d\n", arrayGet(a, pos));
30         } else if(command == 4) {
31             printf("Enter position\n");
32             scanf("%d", &pos);
33             arrayDelete(a, pos);
34         } else if(command == 5) {
35             printf("Enter value\n");
36             scanf("%d", &value);
37             arrayResize(a, value);
38         } else if(command == 6) {
39             arrayPrint(a);
40         }
41     }
42     arrayDestroy(a);
43     return 0;
44 }

```

## arrayAPI.h

```

1 #ifndef _ARRAY_API_H
2 #define _ARRAY_API_H
3
4 #include <stdlib.h>
5
6 typedef struct array array;
7
8 struct array{
9     int* data;
10    size_t size;
11 };
12
13 array* arrayCreate(size_t size);
14 void arrayInsert(array* a, size_t pos, int val);
15 int arrayGet(array* a, size_t pos);
16 void arrayDelete(array* a, size_t pos);
17 void arrayResize(array* a, size_t size);
18 void arrayDestroy(array* a);
19 void arrayPrint(array* a);
20 #endif // _ARRAY_API_H

```

## arrayAPI.h

```

1 #include <stdlib.h>
2 #include <stdio.h>
3
4 #include "arrayAPI.h"
5
6 array* arrayCreate(size_t size) {
7     if(size <= 0) {
8         array* a = (array*)malloc(sizeof(array));
9         a->data = (int*)malloc(sizeof(int));
10        a->size = 1;
11    }
12    array* a = (array*)malloc(sizeof(array));
13    a->data = (int*)malloc(sizeof(int) * size);
14    a->size = size;
15    return a;
16 }
17
18 void arrayInsert(array* a, size_t pos, int val) {
19     if(pos >= a->size) {
20         printf("Out of bounds\n");
21         exit(-1);
22     }
23     a->data[pos] = val;
24 }
25
26 int arrayGet(array* a, size_t pos) {
27     if(pos >= a->size) {
28         printf("Out of bounds\n");
29         exit(-1);
30     }
31     return a->data[pos];
32 }
33
34 void arrayDelete(array* a, size_t pos) {
35     if(pos >= a->size) {
36         printf("Out of bounds\n");
37         exit(-1);

```

```

38     }
39     int* newData = (int*)malloc(sizeof(int) * (a->size - 1));
40     for(unsigned i = 0; i < a->size; ++i) {
41         if(i < pos) {
42             newData[i] = a->data[i];
43         } else if(i > pos) {
44             newData[i - 1] = a->data[i];
45         }
46     }
47     free(a->data);
48     a->data = newData;
49 }
50
51 void arrayResize(array* a, size_t size) {
52     int* newData = (int*)malloc(sizeof(int) * size);
53     if(size <= a->size) {
54         for(unsigned i = 0; i < size; ++i) {
55             newData[i] = a->data[i];
56         }
57     } else {
58         for(unsigned i = 0; i < a->size; ++i) {
59             newData[i] = a->data[i];
60         }
61     }
62     free(a->data);
63     a->data = newData;
64     a->size = size;
65 }
66
67 void arrayDestroy(array* a) {
68     free(a->data);
69     a->size = 0;
70     free(a);
71 }
72
73 void arrayPrint(array* a) {
74     for(unsigned i = 0; i < a->size; ++i) {
75         printf("%d ", a->data[i]);
76     }
77     printf("\n");
78 }

```

## 4 Консоль

```
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_5/src$ ./dynamic
1 - create array with given size
2 - insert element to array at given position
3 - get element value on given position
4 - delete element from given position
5 - resize array to given size
6 - print array
0 - exit
1
Enter value
3
2
Enter position
0 1
2
Enter position
1 2
2
Enter position
2 3
6
1 2 3
3
Enter position
0
1
0
```

## 5 Выводы

В ходе выполнения лабораторной работы я познакомился с такой полезной вещью, как использование динамических библиотек, которые позволяют ускорить работу со сторонними библиотеками, поскольку позволяют включать в программу только необходимые функции, а не все, реализованные в библиотеке.