Национальный исследовательский ядерный университет «МИФИ»
Институт интеллектуальных кибернетических систем
Кафедра №12 «Компьютерные системы и технологии»

ОТЧЕТ

О выполнении лабораторной работы №3

«Работа с массивами данных»

Студент: Кругликова М. В.
Группа: Б22-504
Преподаватель: Комаров Т. И.

Москва – 2022

1. **Формулировка индивидуального задания**

Вариант №8

Индивидуальное задание

В исходной последовательности целых чисел найти все уникальные элементы (те, которые встречаются только один раз) и добавить их в новую последовательность, удалив из исходной.

Правила изменения размера выделенной под массив области памяти

Размер выделенной под массив области памяти должен изменяться автоматически при выполнении операций, приводящих к изменению количества элементов в массиве.

При заполнении элементами массива всей выделенной под него области памяти её размер должен автоматически удваиваться. При заполнении элементами массива выделенной под него области памяти менее, чем наполовину, её размер должен автоматически сокращаться вдвое

2. **Описание использованных типов данных**

При выполнении данной лабораторной работы использовались такие типы данных, как:

- встроенный тип данных int, предназначенный для работы с целыми числами
- встроенный тип данных int *, предназначенный для работы с указателями на целые числа и на массивы из них

3. Описание использованного алгоритма

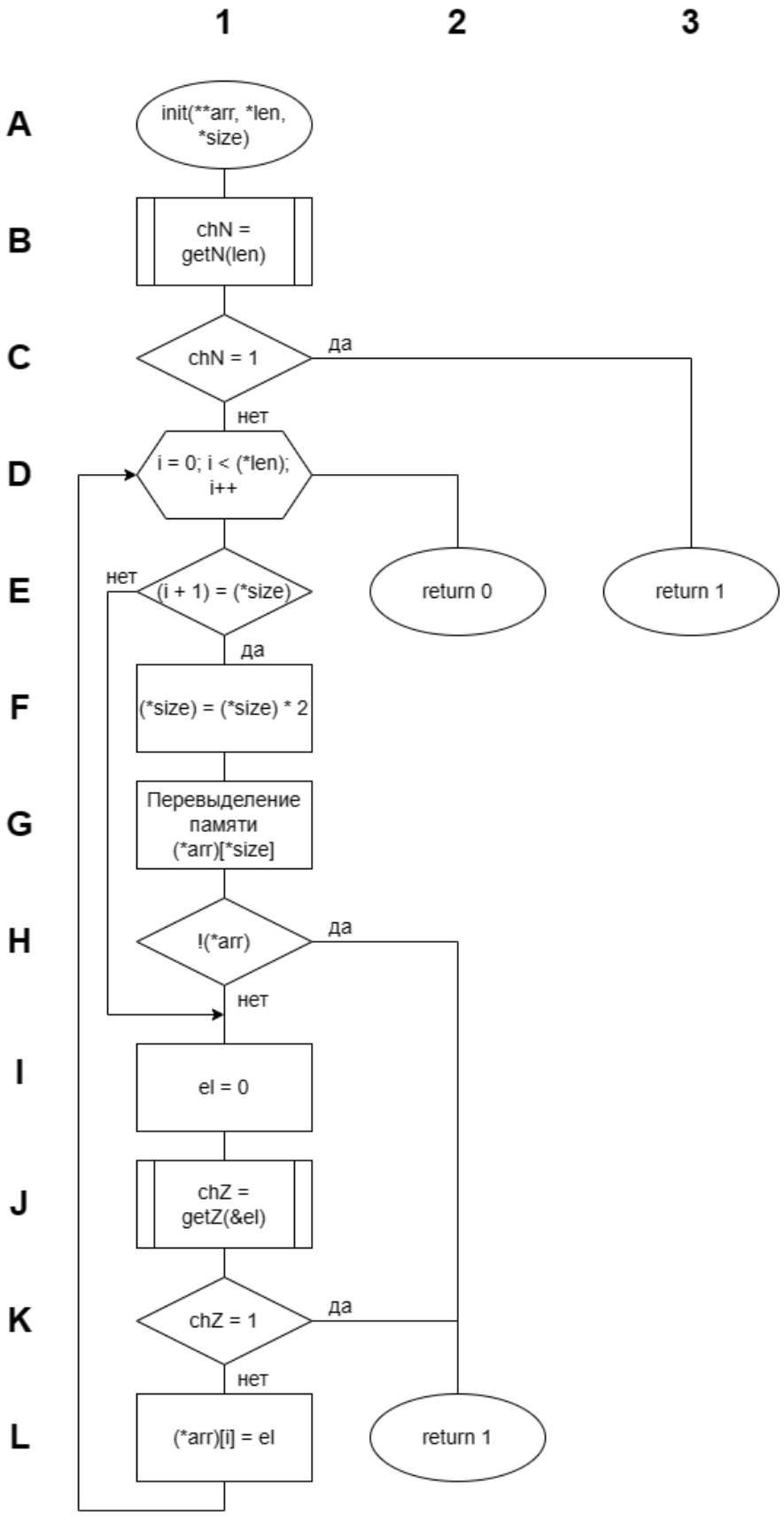

Рисунок 1. Блок-схема алгоритма работы функции init (файл arrayLib.c)

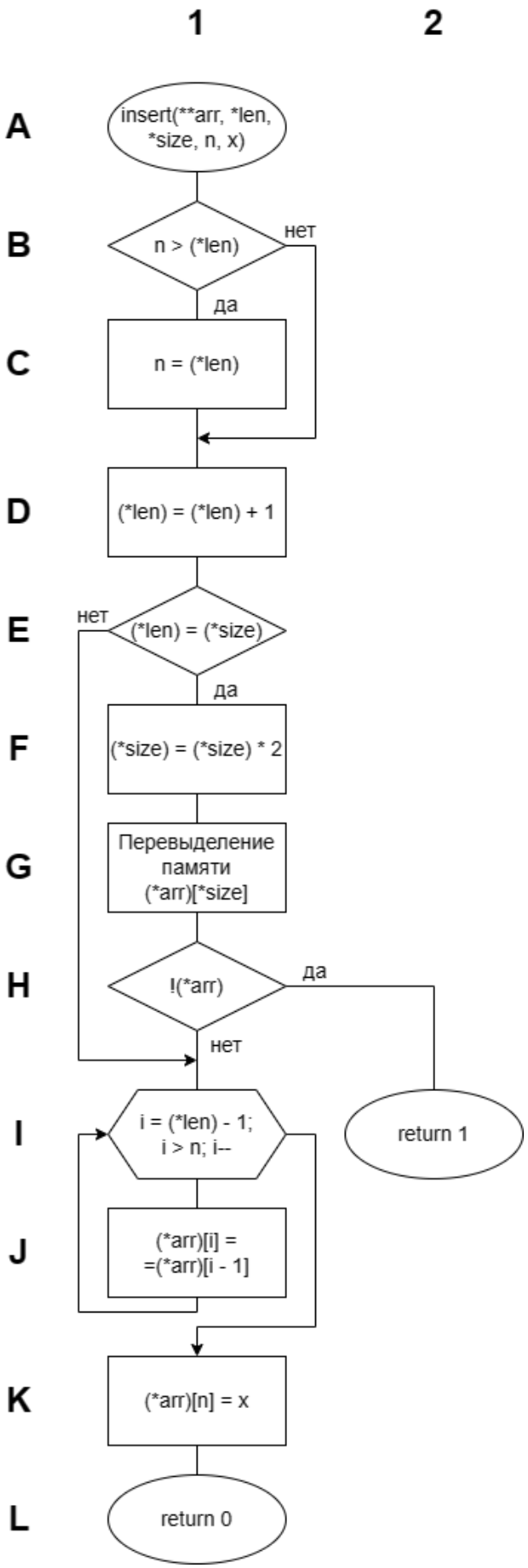

Рисунок 2. Блок-схема алгоритма работы функции insert (файл arrayLib.c)

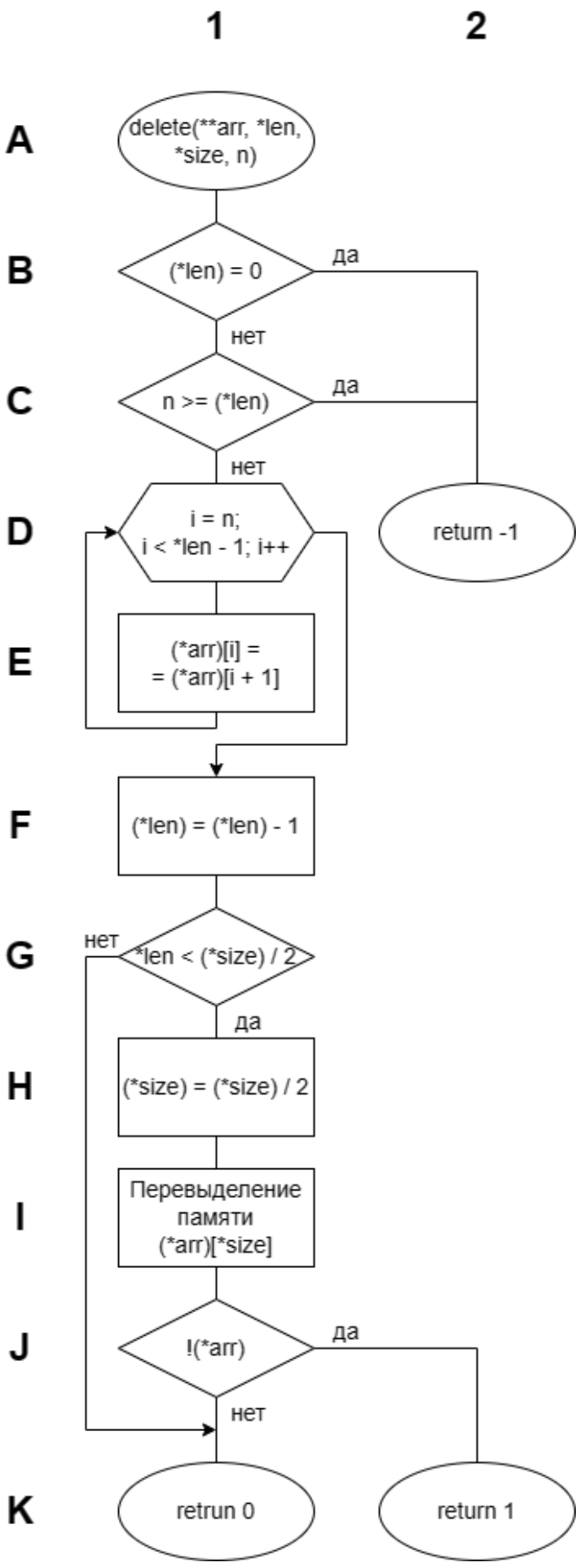

Рисунок 3. Блок-схема работы функции delete (файл arrayLib.c)

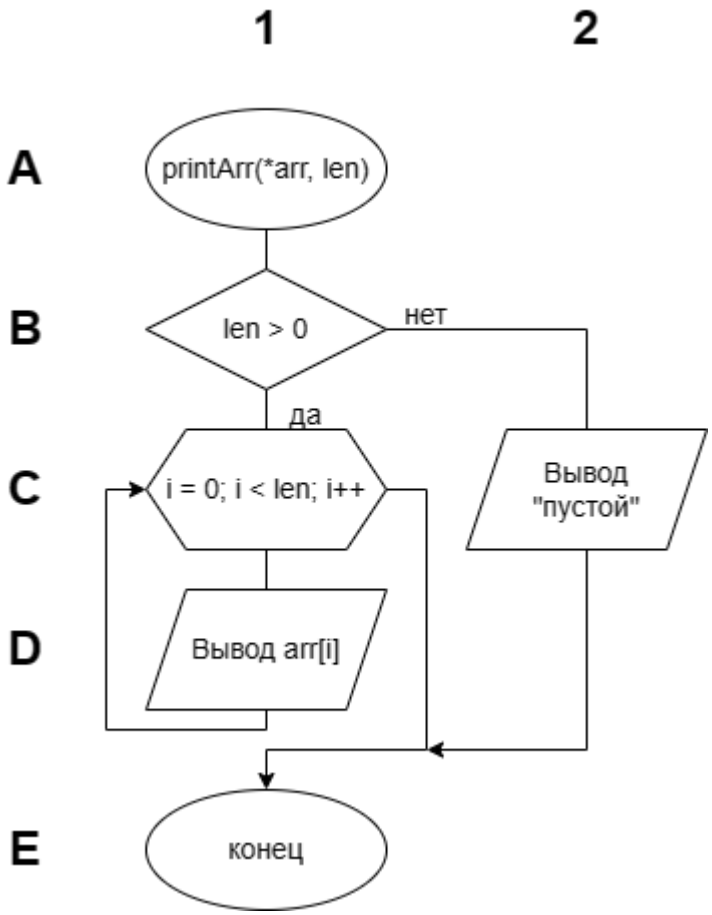

Рисунок 4. Блок-схема работы функции printArr (файл arrayLib.c)

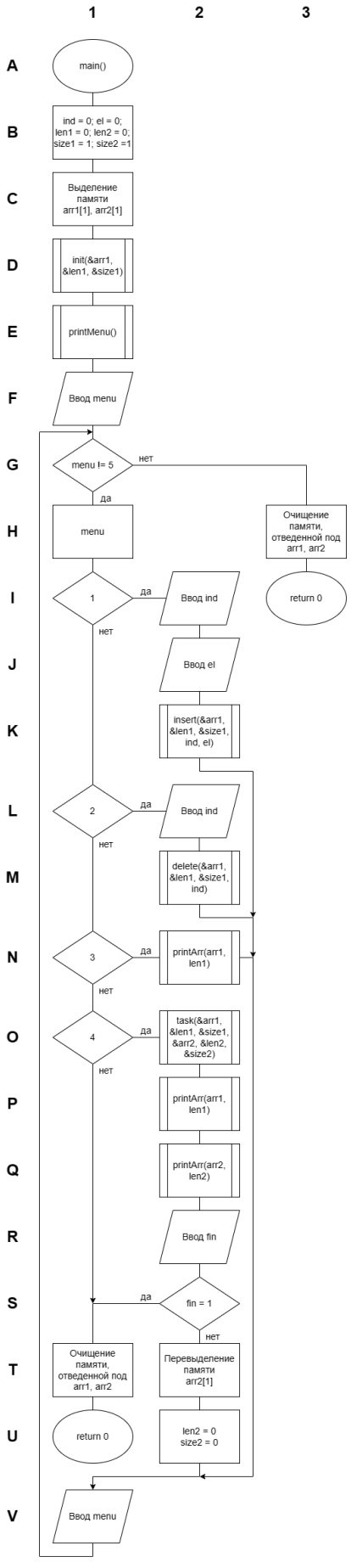

Рисунок 5. Блок-схема работы функции main (файл main3.c)

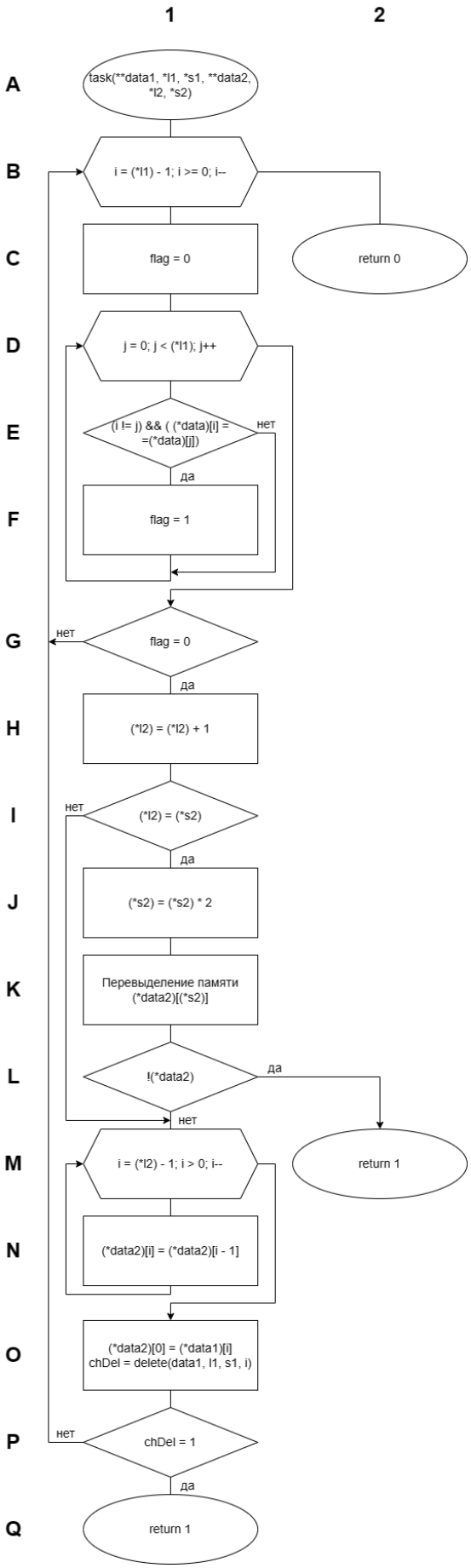

Рисунок 6. Блок-схема работы функции task (файл task.c)

4. Исходные коды разработанных программ

Листинг 1. Исходные коды программы arrayLib (файл arrayLib.c)

```
#include <stdio.h>
#include <stdlib.h>
#include "checkInput.h"

int init(int **arr, int *len, int *size) {
    printf("=== Инициализация массива === \n");
    printf("Введите длину массива: ");
    int chN = getN(len);
    if (chN == 1) {
        return 1;
    }
    for (int i = 0; i < (*len); i++) {
        if ((i + 1) == (*size)) {
            (*size) *= 2;
            (*arr) = realloc((*arr), (*size) *
sizeof(int));
            if (!(*arr)) {
                printf("Ошибка выделения памяти
\n");
                return 1;
            }
        }
        int el = 0;
        printf("Введите %d элемент массива: ", i);
        int chZ = getZ(&el);
        if (chZ == 1) {
            return 1;
        }
        (*arr)[i] = el;
    }
    return 0;
}

int insert(int **arr, int *len, int *size, int n,
int x) {
    if (n > (*len)) {
        n = (*len);
    }
    (*len)++;
    if ((*len) == (*size)) {
```

```
        (*size) *= 2;
        (*arr) = realloc((*arr), (*size) *
sizeof(int));
        if (!(*arr)) {
            printf("Ошибка выделения памяти \n");
            return 1;
        }
    }
    for (int i = (*len) - 1; i > (n); i--) {
        (*arr)[i] = (*arr)[i - 1];
    }
    (*arr)[n] = x;
    return 0;
}

int delete(int **arr, int *len, int *size, int n) {
    if ((*len) == 0) {
        printf("Массив пустой. Невозможно удалить
элемент \n");
        return -1;
    }
    if (n >= (*len)) {
        printf("Введенный индекс больше длины
массива. Невозможно удалить элемент \n");
        return -1;
    }
    for (int i = n; i < *len - 1; i++) {
        (*arr)[i] = (*arr)[i + 1];
    }
    (*len)--;
    if (*len < (*size) / 2) {
        *size /= 2;
        (*arr) = realloc((*arr), (*size) *
sizeof(int));
        if (!(*arr)) {
            printf("Ошибка выделения памяти \n");
            return 1;
        }
    }
    return 0;
}
```

```
void printArr(int *arr, int len) {
    if (len > 0) {
        for (int i = 0; i < len; i++) {
            printf("arr[%d] = %d \n", i, arr[i]);
        }
    } else {
        printf("пустой \n");
    }
}
```

Листинг 2. Исходные коды программы checkInput (файл checkInput.c)

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

int getN(int *n) {
    int ch = 0;
    while ((ch != 1) || ((*n) < 0)) {
        ch = scanf("%d", n);
        if (ch == EOF) {
            return 1;
        }
        if ((ch != 1) || (*n < 0)) {
            printf("Некорректное значение, введите
снова: ");
            scanf("%*[^\n]");
        }
    }
    return 0;
}

int getZ(int *z) {
    int ch = 0;
    while (ch != 1) {
        ch = scanf("%d", z);
        if (ch == EOF) {
            return 1;
        }
        if (ch != 1) {
            printf("Некорректное значение, введите
снова: ");
            scanf("%*[^\n]");
        }
```

```
    }
    return 0;
}

int getFin(int *f) {
    int ch = 0;
    while ((ch != 1) || (((*f) != 0) && ((*f) !=
1))) {
        ch = scanf("%d", f);
        if (ch == EOF) {
            return 1;
        }
        if ((ch != 1) || (((*f) != 0) && ((*f) !=
1))) {
            printf("Некорректное значение, введите
снова: ");
            scanf("%*[^\n]");
        }
    }
    return 0;
}
```

Листинг 3. Исходные коды программы main3 (файл main3.c)

```
#include <stdio.h>
#include <stdlib.h>
#include "checkInput.h"
#include "arrayLib.h"
#include "menu.h"
#include "task.h"

int main() {
    int ind = 0;
    int el = 0;
    int len1 = 0;
    int len2 = 0;
    int size1 = 1;
    int size2 = 1;
    int *arr1 = malloc(sizeof(int));
    int *arr2 = malloc(sizeof(int));
    int chInit = init(&arr1, &len1, &size1);
    if (chInit == 1) {
        free(arr1);
```

```
        free(arr2);
        return 1;
    }
    printMenu();
    int menu = 0;
    int chM = readMenu(&menu);
    if (chM == 1) {
        free(arr1);
        free(arr2);
        return 1;
    }
    while (menu != 5) {
        switch(menu) {
            case 1:
                printf("=== Добавление элемента по
индексу === \n");
                printf("Введите индекс: ");
                int chN = getN(&ind);
                if (chN == 1) {
                    free(arr1);
                    free(arr2);
                    return 1;
                }
                printf("Введите элемент: ");
                int chZ = getZ(&el);
                if (chZ == 1) {
                    free(arr1);
                    free(arr2);
                    return 1;
                }
                int chIns = insert(&arr1, &len1,
&size1, ind, el);
                if (chIns == 1) {
                    free(arr1);
                    free(arr2);
                    return 1;
                }
                break;
            case 2:
                printf("=== Удаление элемента по
индексу === \n");
                chN = getN(&ind);
```

```
                if (chN == 1) {
                    free(arr1);
                    free(arr2);
                    return 1;
                }
                int chDel = delete(&arr1, &len1,
&size1, ind);
                if (chDel == 1) {
                    free(arr1);
                    free(arr2);
                    return 1;
                }
                break;
            case 3:
                printf("=== Вывод текущего
состояния массива === \n");
                printf("Введенный массив: \n");
                printArr(arr1, len1);
                break;
            case 4:
                printf("=== Выполнение операции с
массивом === \n");
                int chT = task(&arr1, &len1,
&size1, &arr2, &len2, &size2);
                if (chT == 1) {
                    free(arr1);
                    free(arr2);
                    return 1;
                }
                printf("Введенный массив: \n");
                printArr(arr1, len1);
                printf("Новый массив: \n");
                printArr(arr2, len2);
                printf("Завершить программу? 1 -
да, 0 - нет \n");
                printf("> ");
                int fin = 0;
                int chFin = getFin(&fin);
                if (chFin == 1) {
                    free(arr1);
                    free(arr2);
                    return 1;
```

```
                }
                if (fin == 1) {
                    free(arr1);
                    free(arr2);
                    return 0;
                } else {
                    arr2 = realloc(arr2,
sizeof(int));
                    len2 = 0;
                    size2 = 1;
                }
                break;
            }
            menu = 0;
            chM = readMenu(&menu);
            if (chM == 1) {
                free(arr1);
                free(arr2);
                return 1;
            }
        }
        free(arr1);
        free(arr2);
        return 0;
}
```

Листинг 4. Исходные коды программы menu (файл menu.c)

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

void printMenu() {
    printf("Меню команд: \n");
    printf("1 - добавить элемент \n");
    printf("2 - удалить элемент \n");
    printf("3 - вывести текущее состояние массива
\n");
    printf("4 - выполнить операцию с массивом \n");
    printf("5 - выйти из программы \n");
}

int readMenu(int *m) {
    int ch = 0;
```

```
    printf("Введите команду: ");
    while ((*m != 1) && (*m != 2) && (*m != 3) &&
(*m != 4) && (*m != 5)) {
        ch = scanf("%d", m);
        if (ch == EOF) {
            return 1;
        }
        if ((*m != 1) && (*m != 2) && (*m != 3) &&
(*m != 4) && (*m != 5)) {
            printf("Некорректное значение, введите
снова: ");
            scanf("%*[^\n]");
        }
    }
    return 0;
}
```

Листинг 5. Исходные коды программы task (файл task.c)

```
#include <stdio.h>
#include <stdlib.h>
#include "arrayLib.h"

int task(int **data1, int *l1, int *s1, int
**data2, int *l2, int *s2) {
    for (int i = (*l1) - 1; i >= 0; i--) {
        int flag = 0;
        for (int j = 0; j < (*l1); j++) {
            if ((i != j) && ((*data1)[i] ==
(*data1)[j])) {
                flag = 1;
            }
        }
        if (flag == 0) {
            (*l2)++;
            if ((*l2) == (*s2)) {
                (*s2) *= 2;
                (*data2) = realloc((*data2), (*s2)
* sizeof(int));
                if (!(*data2)) {
                    printf("Ошибка выделения
памяти \n");
                    return 1;
```

```
                    }
                }
                for (int i = (*l2) - 1; i > 0; i--) {
                    (*data2)[i] = (*data2)[i - 1];
                }
                (*data2)[0] = (*data1)[i];
                int chDel = delete(data1, l1, s1, i);
                if (chDel == 1) {
                    return 1;
                }
            }
        }
        return 0;
}
```

5. Описание тестовых примеров

Введенный массив до обработки	Введенный массив после обработки (ожидание)	Новый массив (ожидание)	Введенный массив после обработки (реальность)	Новый массив (реальность)
2; 5; 1; 6; 9; 4; 5; 9	5; 9; 5; 9	2; 1; 6; 4	5; 9; 5; 9	2; 1; 6; 4
5; 5; 9	5; 5	9	5; 5	9
5	пустой	5	пустой	5

6. Скриншоты

```
[kruglikova.mv@unix:~/inf/lab3]$ gcc arrayLib.c checkInput.c main3.c menu.c task.c -o lab3
[kruglikova.mv@unix:~/inf/lab3]$ valgrind ./lab3
==4942== Memcheck, a memory error detector
==4942== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==4942== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==4942== Command: ./lab3
==4942==
=== Инициализация массива ===
Введите длину массива: 8
Введите 0 элемент массива: 2
Введите 1 элемент массива: 5
Введите 2 элемент массива: 2
Введите 3 элемент массива: 6
Введите 4 элемент массива: 9
Введите 5 элемент массива: 4
Введите 6 элемент массива: 5
Введите 7 элемент массива: 9
Меню команд:
1 - добавить элемент
2 - удалить элемент
3 - вывести текущее состояние массива
4 - выполнить операцию с массивом
5 - выйти из программы
Введите команду: 3
=== Вывод текущего состояния массива ===
Введенный массив:
arr[0] = 2
arr[1] = 5
arr[2] = 2
arr[3] = 6
arr[4] = 9
arr[5] = 4
arr[6] = 5
arr[7] = 9
Введите команду: 1
=== Добавление элемента по индексу ===
Введите индекс: 2
Введите элемент: 1
```

Рисунок 7. Запуск программы lab3 (скриншот №1)

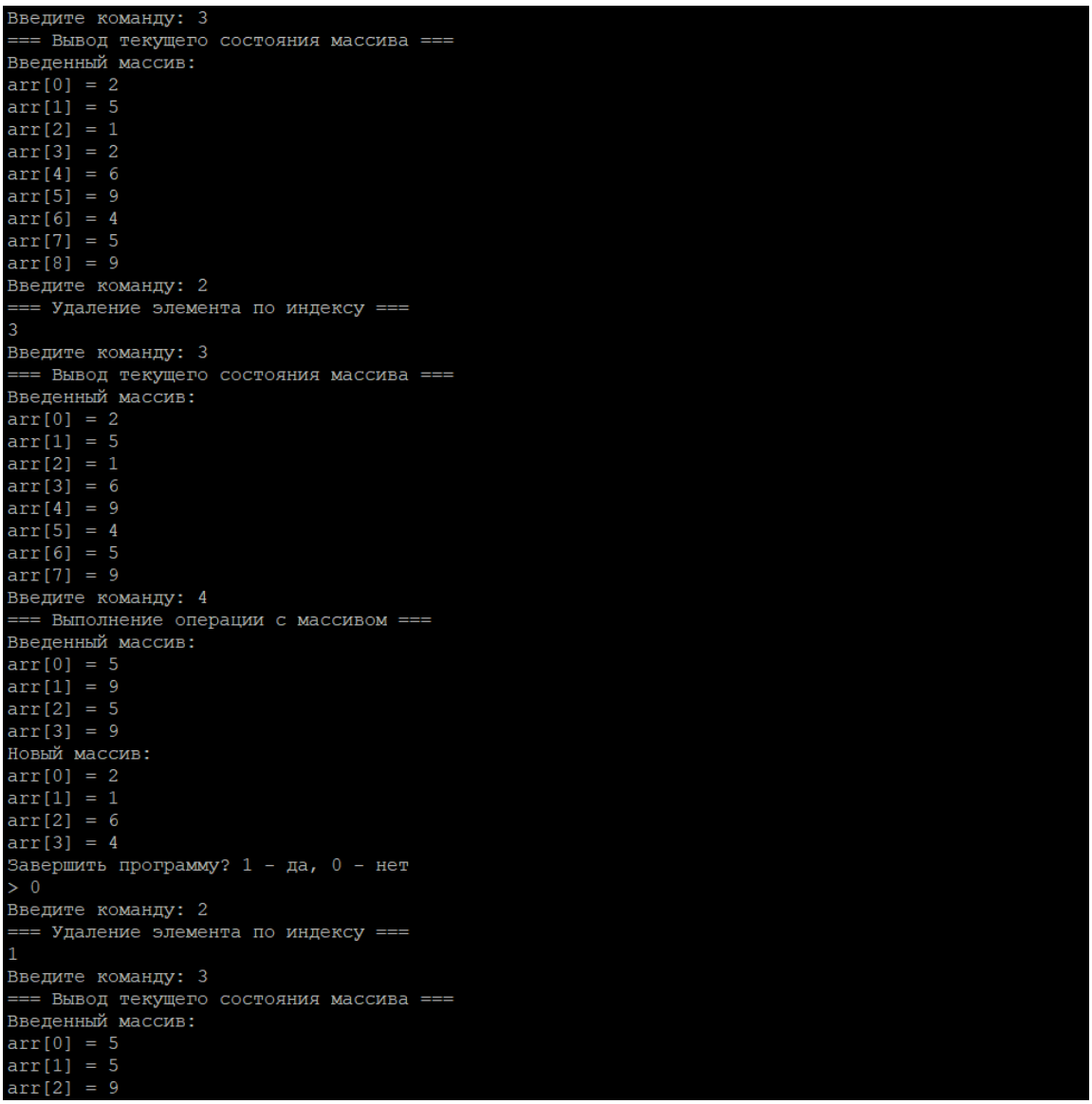

Рисунок 8. Запуск программы lab3 (скриншот №2)

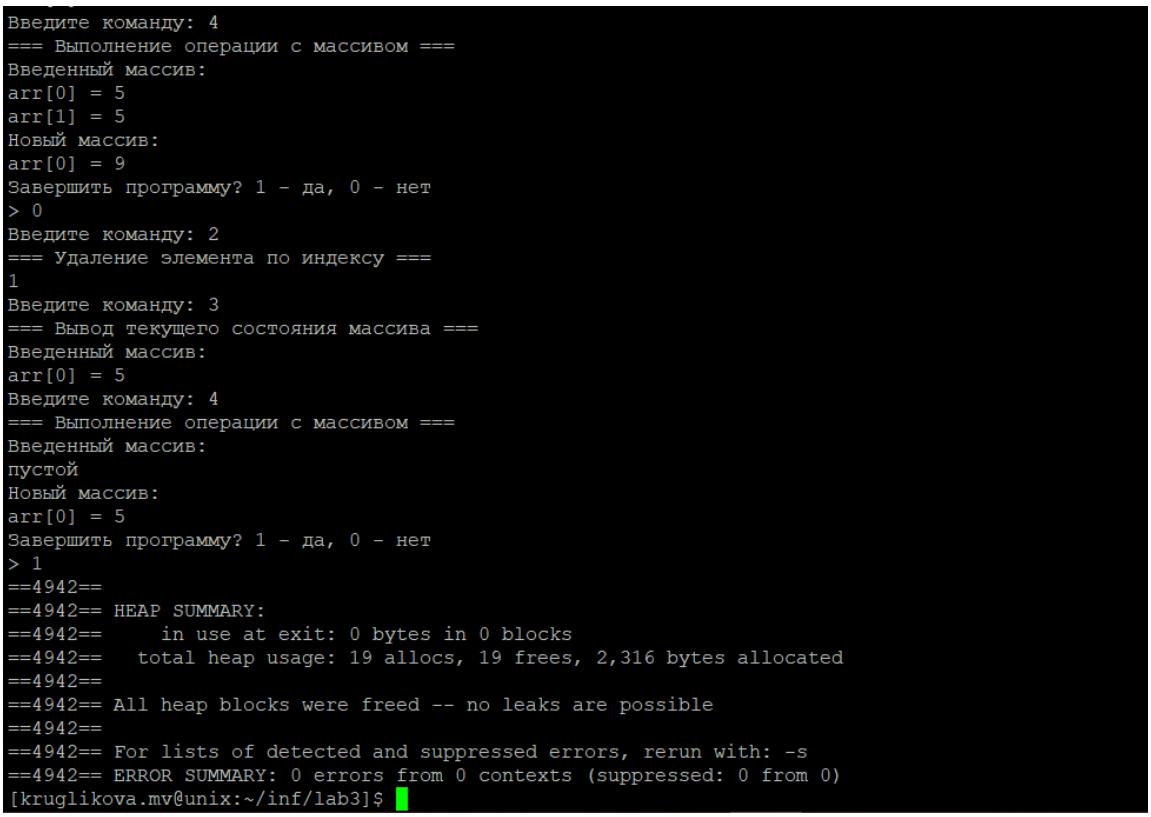

Рисунок 9. Запуск программы lab3 (скриншот №3)

7. Выводы

В ходе выполнения данной работы были изучены:

1. Принципы отладки программы при помощи valgrind
2. Способы динамического выделения и очищения памяти на куче при помощи функций стандартной библиотеки языка С
3. Принципы работы с массивами
4. Принципы сборки проекта, состоящего из нескольких файлов