Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №12 «Компьютерные системы и технологии»

ОТЧЁТ

О выполнении лабораторной работы № 3

"Работа с массивами данных"

Студент: Гаврилов Д. А.

Группа: Б23-516

Преподаватель: Бабалова И. Ф.

Москва – 2023

1. Формулировка задания

Вариант № 69

Введение

Необходимо спроектировать и реализовать на языке C программу, осуществляющую по запросам пользователя ввод, обработку и вывод последовательности данных, которая представляется в виде массива. Программа должна реализовывать следующую функциональность:

1. Инициализация массива (заполнение массива в цикле).

2. Вставка нового элемента в массив по индексу.

3. Удаление элемента массива по индексу.

4. Обработка данных (в соответствии с индивидуальным заданием).

5. Вывод текущего состояния
массива. Примечания:
1. Взаимодействие программы с пользователем должно быть выстроено с помощью диалогового меню, имеющего приблизительно следующий вид:

(a) Инициализация массива.

(b) Вставка нового элемента.

(c) Удаление элемента.

(d) Индивидуальное задание.

(e) Вывод содержимого массива.

2. После выполнения любой из представленных операций программа должна автоматически выводить содержимое массива.

3. Программа должна штатным образом завершаться при выборе пользователем соответствующего пункта диалогового меню или при обнаружении EOF — конца файла (в UNIX-подобных ОС инициируется нажатием клавиш Ctrl + D, в Windows — Ctrl + Z).

4. Программа должна осуществлять проверку корректности вводимых данных и, в случае ошибок, выдавать соответствующие сообщения, после чего продолжать работу.

5. Обрабатываемые последовательности должны быть представлены в виде массива элементов, которые имеют соответствующий тип данных.

6. Память под массивы обрабатываемых данных должна выделяться и освобождаться динамически, с использованием функций calloc(), malloc(), realloc() и free() из состава стандартной библиотеки.

7. Использование VLA (массивов переменной длины) не допускается.

8. При вставке элемента в массив по указанному индексу, элементы массива с индексом, превышающим или совпадающим с заданным, должны «сдвигаться вправо».

9. При удалении элемента массива с указанным индексом, элементы массива с б*о*льшим индексом должны «сдвигаться влево».

10. При вставке в массив нового элемента по индексу, значение которого превышает максимально допустимое (оно соответствует длине массива), необходимо осуществлять вставку в конец массива.

11. При удалении элемента массива по индексу, значение которого превышает максимально допустимое (оно соответствует длине массива), необходимо вернуть ошибку.

12. Логически законченные части алгоритма решения задачи должны быть оформлены в виде отдельных функций с параметрами. Использование глобальных переменных не допускается.

13. Функции по обработке массивов не должны быть диалоговыми, т. е. они должны принимать все необходимые данные в качестве параметров и возвращать результат работы в виде соответствующих данных и кодов ошибок (исключения: функции инициализации и вывода массивов).

14. Исходные коды программы должны быть логичным образом разбиты на несколько файлов (необходимо использовать как *.c-файлы, так и *.h-файлы).

15. Согласно условиям индвидуального задания, может требоваться наличие нескольких исходных последовательностей. В таком случае, пользователь должен иметь возможность интерактивного взаимодействия с каждой из них.

16. Программа должна корректным образом работать с памятью, для проверки необходимо использовать соответствующие программные средства, например: valgrind (при тестировании и отладке программы необходимо запускать её командой вида valgrind ./lab3).

Индивидуальное задание

В исходной последовательности вещественных чисел найти те, старшая значащая цифра в которых равна 9. Сформировать из данных чисел новую последовательности, удалив их из исходной.

Правила изменения размера выделенной под массив области памяти

Размер выделенной под массив области памяти должен изменяться при выполнении операций, приводящих к изменению количества элементов в массиве.
Размер выделенной под массив области памяти должен быть всегда равен размеру реально используемой элементами массива области памяти.
В функциях вставки и удаления элементов массива должна быть предусмотрена соответствующая функциональность.

2. Описание использованных типов данных

Целочисленный тип данных – int (спецификатор формата %d)

Тип данных с плавающей точкой – float (спецификатор формата %d) , а также указатель на тип данных с плавающей точкой – float*.

3. Описание использованных алгоритмов

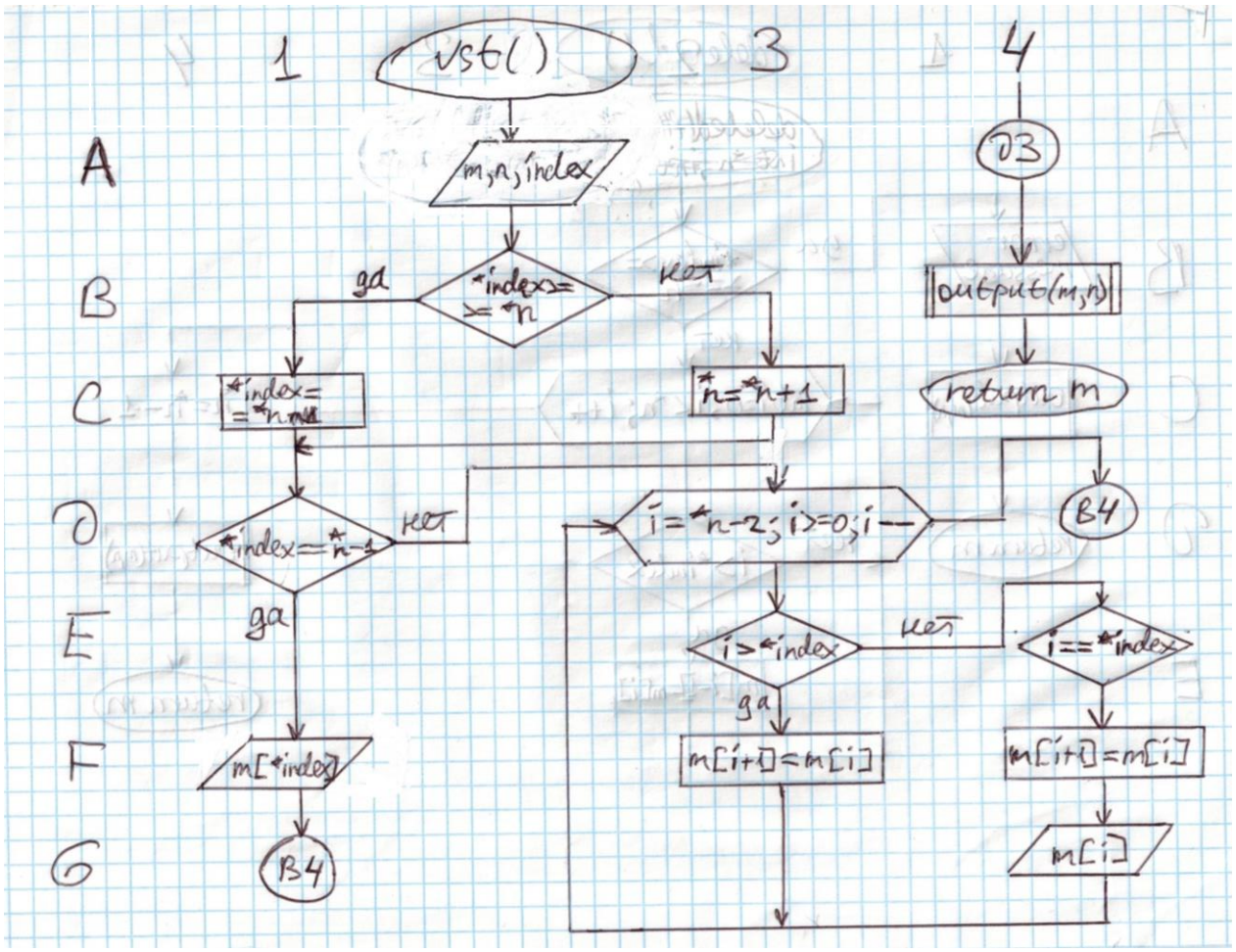

Рис. 1-2: Блок-схемы алгоритмов вставки vst() и удаления deleted() числа из массива.

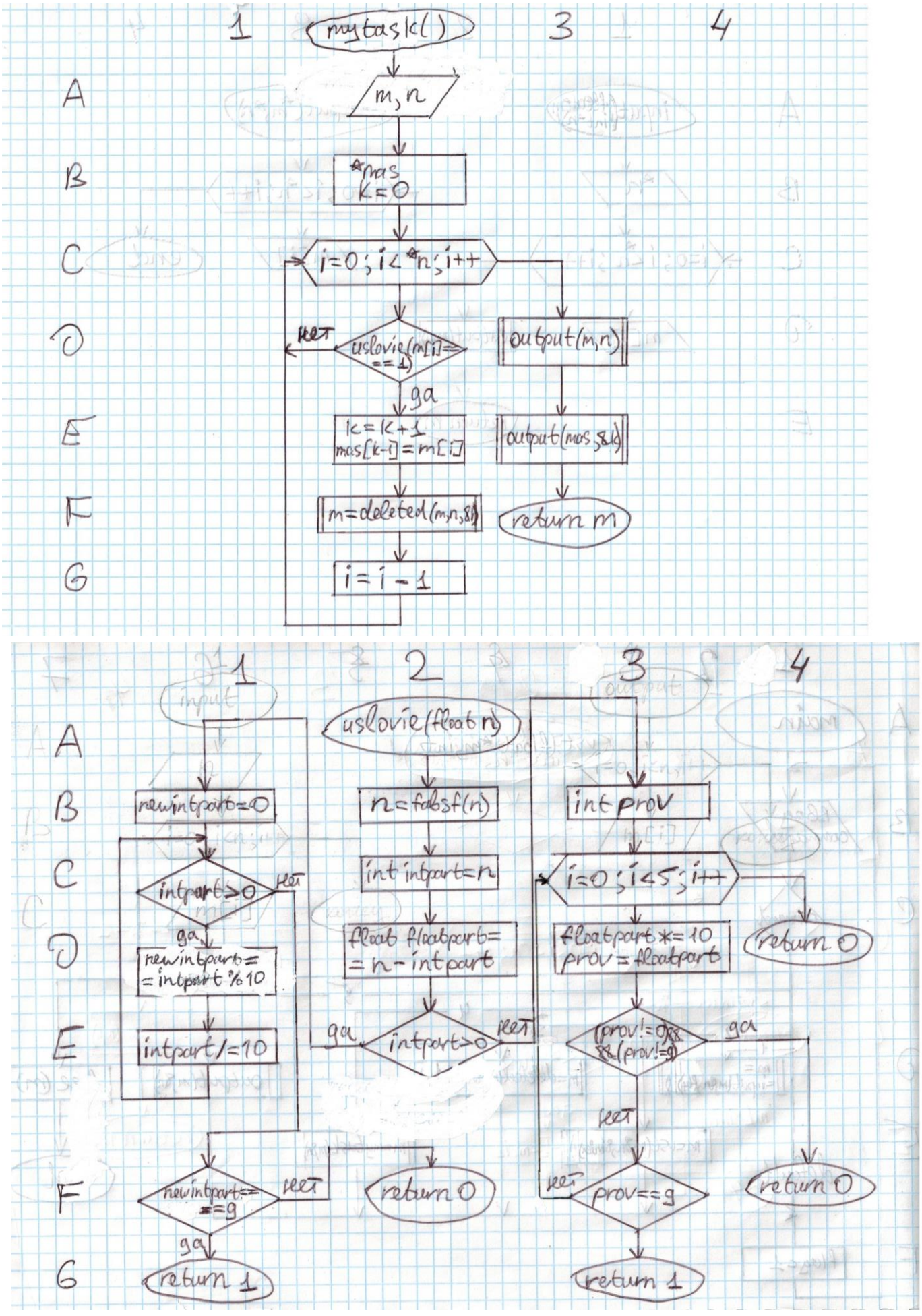

Рис. 3-4: Блок-схемы алгоритма индивидуального задания mytask() и вспомогательной функции uslovie().

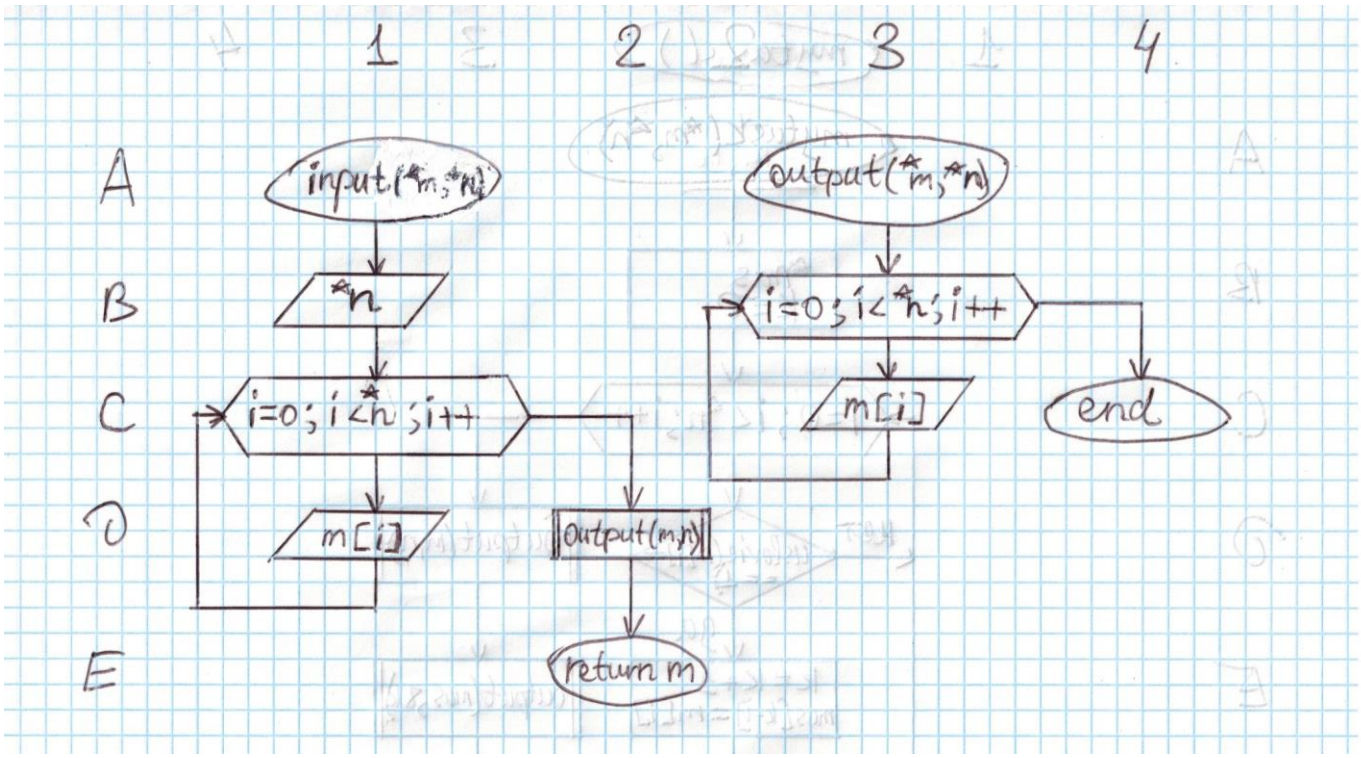

Рис. 5: Блок-схема алгоритма вывода output() и ввода input() массива.

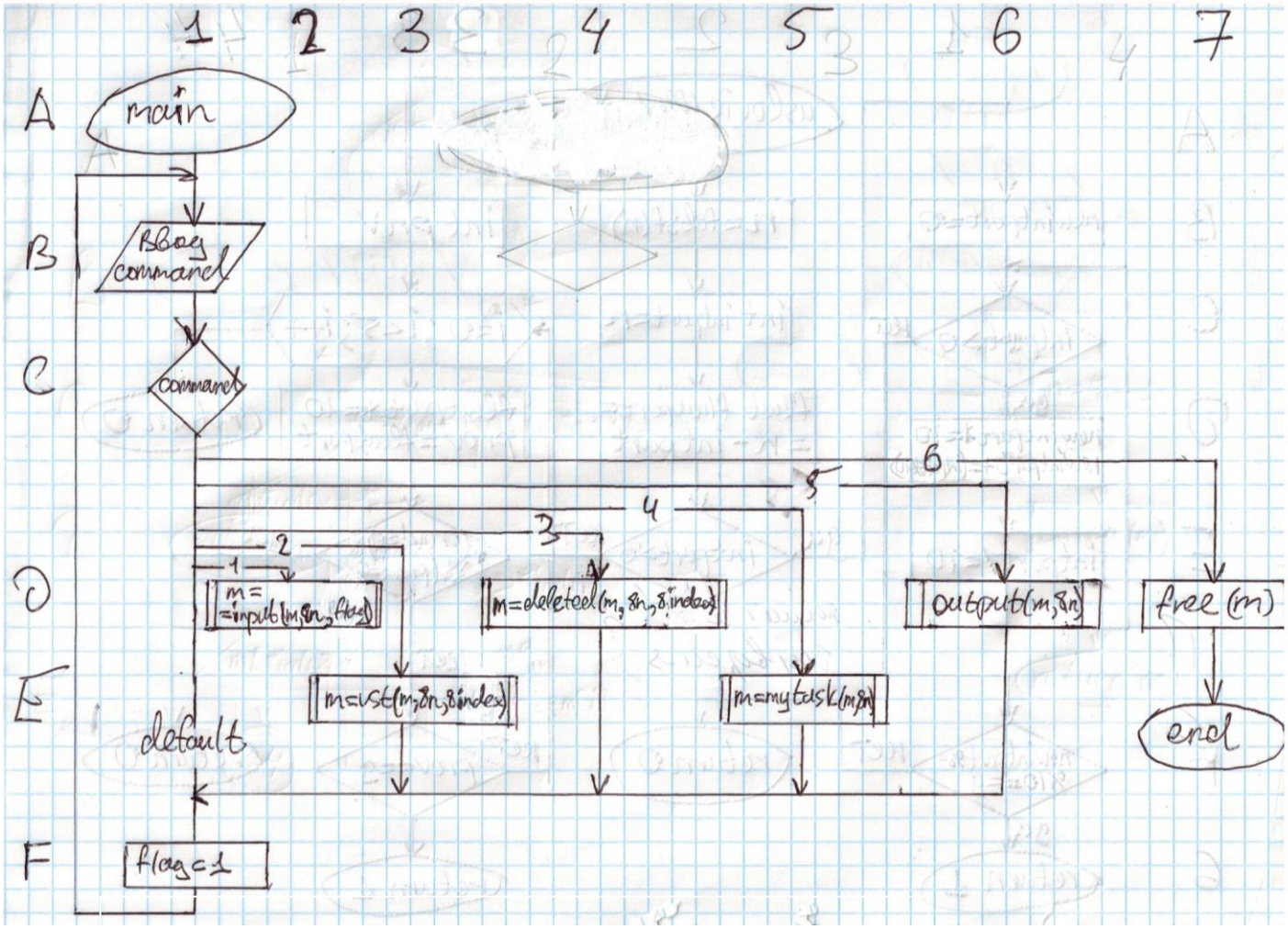

Рис. 6: Блок-схема алгоритма меню по работе с пользователем main().

4. Исходные коды разработанных программ

Структура проекта состоит из 4 файлов: заголовочного FILELAB.h и трех с исходным кодом – reslab3.c; file2.c; file3.c.

Листинг 1: Исходный код программы reslab3 (файл: reslab3.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "FILELAB.h"

int main(){
    int n, index, command, flag = 0;
    float* m = 0;
    do {
        if (intro(m,&command) == 0){return 0;}
        switch (command){
        case 1: m = input(m,&n,flag);break;
        case 2: m = vst(m,&n,&index);break;
        case 3: m = deleted(m,&n,&index,0);break;
        case 4: m = mytask(m,&n);break;
        case 5: output(m,&n);break;
        case 6: free(m);return 0;
        default: break;
        }
    flag = 1;
    } while(1);
}
```

Листинг 2: Исходный код программы reslab3 (файл:file2.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "FILELAB.h"


void output(float *m, int *n) { // функция вывода массива
    printf("Array: [");
    for(int i = 0; i < *n; i++){
        printf(" %f;", m[i]);
    }
    printf("]\n");
}


float* input(float *m, int *n, int code){// функция инициализации массива
    if (code == 1){free(m);}
    printf("Enter the number of members of array: ");
    n = prov(n, 1); // функция ввода и проверки n
    m = (float*)malloc(*n * sizeof(float));
    for(int i = 0; i < *n; i++){
        printf("Enter %d float member of array (<= 5 meaningful signs): ", i+1);
        int r = scanf("%f", &m[i]);
        while (r != 1){
            scanf("%*[^\n]");
            printf("Please, enter correct float number!\n");
            r = scanf("%f", &m[i]);
        }
    }
    output(m,n);
    return m;
}
```

```c


float* deleted(float *m, int *n, int *index, int code){ // функция удаления элемента из массива
    if (code == 0){ // "code" value нужен для использования функции в mytask()
        printf("Enter the index value (to delete): ");
        index = prov(index,2);
        if (*index >= *n){
        printf("Error, delete index value >= n\n");
        output(m,n);
        return m;
        }
    }

    for (int i = 0; i < *n; i++){
        if (i > *index){
            m[i-1] = m[i];
        }
    }
    *n = *n - 1;
    m = realloc(m, sizeof(float) * *n);
    if (code == 0){ // в mytask() нам не будет это требоваться
        output(m,n);
    }
    return m;
}
```

```
58
59
60 float* vst(float *m, int *n, int *index){ // функция вставки элемента в массив
61     printf("Enter the index value (to vst): ");
62     index = prov(index,2); // ввод индекса и проверка
63     if (*index >= *n){
64         (*index) = (*n); // если индекс >= n => index = n
65     }
66     *n = *n + 1;
67     m = realloc(m,sizeof(float) * *n);
68     if (*index == *n-1){ // если мы вставляем элемент в самый конец
69         floatvvod(m,*index);
70     }
71     else { // иначе
72         for (int i = *n-2; i >= 0; i--){
73             if (i > *index){
74                 m[i+1] = m[i];
75             }
76             else if (i == *index){
77                 m[i+1] = m[i];
78                 floatvvod(m,i);
79             break;
80             }
81         }
82     }
83     output(m,n);
84     return m;
85 }
86
87
```

```
88 float* mytask(float *m, int *n){ // индивидуальное задание
89     float* mas = 0;
90     int k = 0;
91     for (int i = 0; i < *n; i++){
92         if (uslovie(m[i]) == 1) {
93             mas = realloc(mas, sizeof(float) * (k+1));
94             k = k + 1;
95             mas[k-1] = m[i];
96             m = deleted(m,n,&i,1);
97             i = i - 1;
98         }
99     }
100    output(m,n);
101    printf("Task ");
102    output(mas,&k);
103    free(mas);
104    return m;
105 }
```

Листинг 3: Исходные код программы reslab3 (файл:file3.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "FILELAB.h"


int* prov(int *n, int code){ // функция ввода и проверки n, code для n и index
    if (code == 1) {
        int p = scanf("%d", &*n);
        while (p != 1 || *n <= 0){
            scanf("%*[^\n]");
            printf("Incorrect n value, enter again\n");
            p = scanf("%d", &*n);
        }
        return n;
    }
    if (code == 2){
        int p = scanf("%d", &*n);
        while (p != 1 || *n < 0){
            scanf("%*[^\n]");
            printf("Incorrect index value (<0 or not int), enter again\n");
            p = scanf("%d", &*n);
        }
        return n;
    }
}


void floatvvod(float *m, int index){ // функция ввода вещественного числа (+проверка)
    printf("Enter the vst float number: \n");
    int r = scanf("%f", &m[index]);
    while (r != 1){
        scanf("%*[^\n]");
        printf("Please, enter correct float number!\n");
        r = scanf("%f", &m[index]);
    }
}


int uslovie(float n){ // функция условия, которому должно соответствовать
    // число в mytask() - первая значимая цифра числа - 9
    n = fabsf(n);
    int intpart = n; // целая часть числа
    float floatpart = n - intpart; // дробная часть числа
    if (intpart > 0){ // если есть значимая целая часть, то...
        int newintpart = 0;
        while (intpart > 0){
            newintpart = (intpart%10);
            intpart /= 10;
        }
        if (newintpart == 9){return 1;}
        else return 0;
    }
    int prov;
    for (int i = 0; i < 5; i++){ // проверка дробной части
        floatpart *= 10;
        prov = floatpart;
        if ((prov != 0) && (prov != 9)){return 0;}
        else if (prov == 9){return 1;}
    }
}
```

```
}


int intro(float* m, int* command){ // функция main'a
    printf("\nCommands list:\n1) Enter mas\n2) Insert element\n3) Delete element\n4) Individual task\n5) Output mas\n6) Stop the program\n");
    printf("Enter the command\n");
    int r = scanf("%d",command);
    if (r == EOF){free(m); return 0;}
    while (r != 1 || *command <= 0 || *command >= 7){
        scanf("%*[^\n]");
        printf("This command doesn't exist, try again\n");
        r = scanf("%d",command);
    }
    return 1;
```

Заголовочные файлы:

Листинг 4: Исходные код программы reslab3 (файл: FILELAB.h)

```
#ifndef FILELAB
#define FILELAB
void output(float *m, int *n);
float* input(float *m, int *n, int code);
float* deleted(float *m, int *n, int *index, int code);
float* vst(float *m, int *n, int *index);
float* mytask(float *m, int *n);
int* prov(int *n, int code);
void floatvvod(float *m, int index);
int uslovie(float n);
int intro(float* m, int* command);
#endif

```

5. Текстовые примеры работы программы

Тесты алгоритма вставки

Входные данные (массив, длина, индекс вставки, новый элемент)	Вывод (длина, массив)
[-1.5, 3.02, 4.0], n = 3, i = 1 , x = 2.5	n = 4, [-1.5, 2.5, 3.02, 4.0]
[-5.0,90.0], n = 2, i = 50, x = 80.7	n = 3, [-5.0, 90.0, 80.7]
[0.009], n = 1, i = 0, x = 1.89	n = 2, [1.89, 0.009]
[322.21, 12.43, 7.99, -0.09, 3.5], n = 5, i = 3, x = 8.12	n = 6, [322.21, 12.43, 7.99, 8.12, -0.09, 3.5]
[1.5], n = 1, i = -100, x = 8.0	Ошибка – «Некорректное значение индекса, попробуйте снова»
[1.5], n = 1, i = 0, x = zcd	Ошибка – «Некорректное значение числа float, попробуйте снова»

Тесты алгоритма удаления

Входные данные (массив, длина, индекс удаления)	Вывод (длина, массив)
[1.5], n = 1, i = -100	Ошибка – «Некорректное значение индекса, попробуйте снова»
[1.5], n = 1, i = 10	Ошибка – «Индекс за пределами (>=n)», выход в меню
[8.3, 1.32, 0.08, -0.9], n = 4, i = 3	n = 3, [8.3, 1.32, 0.08]
[-0.023, 12.34, 34.2], n = 3, i = 1	n = 2, [-0.023, 34.2]

Тесты алгоритма выполнения индивидуального задания

Входные данные	Сформированная последовательность	Вывод измененного массива
[-0.009, 12.3, 4.5, 921.8], n = 4	[-0.009, 921.8]	[12.3, 4.5], n = 2
[12.3, 4.5], n = 2	[]	[12.3, 4.5], n = 2
[-9436.22, 0.009, 999.0, 91.8, -0.0091], n = 5	[-9436.22, 0.009, 999.0, 91.8, -0.0091]	[], n = 0
[], n = 0	[]	[], n = 0
[0.0, 0.0, 0.0], n = 3	[]	[0.0, 0.0, 0.0], n = 3
[12.3, 9.121, -23.2, -905.0, -12.21, 0.99, 48.0, 2222.2, 999.0], n = 9	[9.121, -905.0, 0.99, 999.0]	[12.3, -23.2, -12.21, 48.0, 2222.2], n = 5
[12.3, -23.2, -12.21, 48.0, 2222.2], n = 5	[]	[12.3, -23.2, -12.21, 48.0, 2222.2], n = 5

6. Скриншоты работы программы

Команда для сборки программы: cc -o reslab3 reslab3.c file2.c file3.c -lm

Запуск с valgrind: valgrind ./reslab3

Тесты алгоритма вставки:

```
[gavrilov.da@unix lab3]$ cc -o reslab3 reslab3.c file2.c file3.c -lm
[gavrilov.da@unix lab3]$ ./reslab3

Commands list:
1) Enter mas
2) Insert element
3) Delete element
4) Individual task
5) Output mas
6) Stop the program
Enter the command
1
Enter the number of members of array: 3
Enter 1 float member of array (<= 5 meaningful signs): -1.5
Enter 2 float member of array (<= 5 meaningful signs): 3.02
Enter 3 float member of array (<= 5 meaningful signs): 4.0
Array: [ -1.500000; 3.020000; 4.000000;]

Enter the command
2
Enter the index value (to vst): 1
Enter the vst float number: 2.5
Array: [ -1.500000; 2.500000; 3.020000; 4.000000;]

Enter the command
[gavrilov.da@unix lab3]$ ./reslab3

Commands list:
1) Enter mas
2) Insert element
3) Delete element
4) Individual task
5) Output mas
6) Stop the program
Enter the command
1
Enter the number of members of array: 2
Enter 1 float member of array (<= 5 meaningful signs): -5.0
Enter 2 float member of array (<= 5 meaningful signs): 90
Array: [ -5.000000; 90.000000;]

Enter the command
2
Enter the index value (to vst): 50
Enter the vst float number: 80.7
Array: [ -5.000000; 90.000000; 80.699997;]
```

```
Enter the command
1
Enter the number of members of array: 1
Enter 1 float member of array (<= 5 meaningful signs): 0.009
Array: [ 0.009000;]

Enter the command
2
Enter the index value (to vst): 0
Enter the vst float number: 1.89
Array: [ 1.890000; 0.009000;]

Enter the command
1
Enter the number of members of array: 5
Enter 1 float member of array (<= 5 meaningful signs): 322.21
Enter 2 float member of array (<= 5 meaningful signs): 12.43
Enter 3 float member of array (<= 5 meaningful signs): 7.99
Enter 4 float member of array (<= 5 meaningful signs): -0.09
Enter 5 float member of array (<= 5 meaningful signs): 3.5
Array: [ 322.209991; 12.430000; 7.990000; -0.090000; 3.500000;]

Enter the command
2
Enter the index value (to vst): 3
Enter the vst float number: 8.12
Array: [ 322.209991; 12.430000; 7.990000; 8.120000; -0.090000; 3.500000;]

Enter the command
1
Enter the number of members of array: 1
Enter 1 float member of array (<= 5 meaningful signs): 1.5
Array: [ 1.500000;]

Enter the command
2
Enter the index value (to vst): -100
Incorrect index value (<0 or not int), enter again
0
Enter the vst float number: 1.1
Array: [ 1.100000; 1.500000;]
```

Тесты алгоритма удаления:

```
[gavrilov.da@unix lab3]$ cc -o reslab3 reslab3.c file2.c file3.c -lm
[gavrilov.da@unix lab3]$ ./reslab3

Commands list:
1) Enter mas
2) Insert element
3) Delete element
4) Individual task
5) Output mas
6) Stop the program
Enter the command
1
Enter the number of members of array: 4
Enter 1 float member of array (<= 5 meaningful signs): 8.3
Enter 2 float member of array (<= 5 meaningful signs): 1.32
Enter 3 float member of array (<= 5 meaningful signs): 0.08
Enter 4 float member of array (<= 5 meaningful signs): -0.9
Array: [ 8.300000; 1.320000; 0.080000; -0.900000;]

Enter the command
3
Enter the index value (to delete): 3
Array: [ 8.300000; 1.320000; 0.080000;]

Enter the command
3
Enter the index value (to delete): 0
Array: [ 1.320000; 0.080000;]

Enter the command
3
Enter the index value (to delete): -100
Incorrect index value (<0 or not int), enter again
0
Array: [ 0.080000;]

Enter the command
3
Enter the index value (to delete): 10
Error, delete index value >= n
Array: [ 0.080000;]

Enter the command
```

Тесты программы целиком вместе с индивидуальным заданием (в соответствии с ручными тестами) и запуск с valgrind:

```
[gavrilov.da@unix lab3]$ cc -o reslab3 reslab3.c file2.c file3.c -lm
[gavrilov.da@unix lab3]$ valgrind ./reslab3
==27688== Memcheck, a memory error detector
==27688== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==27688== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==27688== Command: ./reslab3
==27688==

Commands list:
1) Enter mas
2) Insert element
3) Delete element
4) Individual task
5) Output mas
6) Stop the program
Enter the command
1
Enter the number of members of array: 4
Enter 1 float member of array (<= 5 meaningful signs): -0.009
Enter 2 float member of array (<= 5 meaningful signs): 12.3
Enter 3 float member of array (<= 5 meaningful signs): 4.5
Enter 4 float member of array (<= 5 meaningful signs): 921.8
Array: [ -0.009000; 12.300000; 4.500000; 921.799988;]

Enter the command
4
Array: [ 12.300000; 4.500000;]
Task Array: [ -0.009000; 921.799988;]

Enter the command
4
Array: [ 12.300000; 4.500000;]
Task Array: []

Enter the command
2
Enter the index value (to vst): 0
Enter the vst float number: -12
Array: [ -12.000000; 12.300000; 4.500000;]

Enter the command
3
Enter the index value (to delete): 0
Array: [ 12.300000; 4.500000;]

Enter the command
6
==27688==
==27688== HEAP SUMMARY:
==27688==     in use at exit: 0 bytes in 0 blocks
==27688==   total heap usage: 9 allocs, 9 frees, 2,116 bytes allocated
==27688==
==27688== All heap blocks were freed -- no leaks are possible
==27688==
==27688== For lists of detected and suppressed errors, rerun with: -s
==27688== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[gavrilov.da@unix lab3]$
```

```
[gavrilov.da@unix lab3]$ valgrind ./reslab3
==27789== Memcheck, a memory error detector
==27789== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==27789== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==27789== Command: ./reslab3
==27789==

Commands list:
1) Enter mas
2) Insert element
3) Delete element
4) Individual task
5) Output mas
6) Stop the program
Enter the command
1
Enter the number of members of array: 5
Enter 1 float member of array (<= 5 meaningful signs): -9436.22
Enter 2 float member of array (<= 5 meaningful signs): 0.009
Enter 3 float member of array (<= 5 meaningful signs): 999
Enter 4 float member of array (<= 5 meaningful signs): 91.8
Enter 5 float member of array (<= 5 meaningful signs): -0.0091
Array: [ -9436.219727; 0.009000; 999.000000; 91.800003; -0.009100;]

Enter the command
4
==27789== realloc() with size 0
==27789==    at 0x48489DC: realloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==27789==    by 0x10959B: deleted (in /home/students/g/gavrilov.da/lab3/reslab3)
==27789==    by 0x109817: mytask (in /home/students/g/gavrilov.da/lab3/reslab3)
==27789==    by 0x10927B: main (in /home/students/g/gavrilov.da/lab3/reslab3)
==27789==  Address 0x4b21b50 is 0 bytes inside a block of size 4 alloc'd
==27789==    at 0x48489DC: realloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==27789==    by 0x10959B: deleted (in /home/students/g/gavrilov.da/lab3/reslab3)
==27789==    by 0x109817: mytask (in /home/students/g/gavrilov.da/lab3/reslab3)
==27789==    by 0x10927B: main (in /home/students/g/gavrilov.da/lab3/reslab3)
==27789==
Array: []
Task Array: [ -9436.219727; 0.009000; 999.000000; 91.800003; -0.009100;]

Enter the command
4
Array: []
Task Array: []

Enter the command
6
==27789==
==27789== HEAP SUMMARY:
==27789==     in use at exit: 0 bytes in 0 blocks
==27789==   total heap usage: 12 allocs, 12 frees, 2,168 bytes allocated
==27789==
==27789== All heap blocks were freed -- no leaks are possible
==27789==
==27789== For lists of detected and suppressed errors, rerun with: -s
==27789== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
[gavrilov.da@unix lab3]$
```

Большой тест с использованием всех функций программы и valgrind:

```
[gavrilov.da@unix lab3]$ valgrind ./reslab3
==27875== Memcheck, a memory error detector
==27875== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==27875== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==27875== Command: ./reslab3
==27875==

Commands list:
1) Enter mas
2) Insert element
3) Delete element
4) Individual task
5) Output mas
6) Stop the program
Enter the command
1
Enter the number of members of array: 9
Enter 1 float member of array (<= 5 meaningful signs): 12.3
Enter 2 float member of array (<= 5 meaningful signs): 9.121
Enter 3 float member of array (<= 5 meaningful signs): -23.2
Enter 4 float member of array (<= 5 meaningful signs): -905
Enter 5 float member of array (<= 5 meaningful signs): -12.21
Enter 6 float member of array (<= 5 meaningful signs): 0.99
Enter 7 float member of array (<= 5 meaningful signs): 48
Enter 8 float member of array (<= 5 meaningful signs): 2222.2
Enter 9 float member of array (<= 5 meaningful signs): 999
Array: [ 12.300000; 9.121000; -23.200001; -905.000000; -12.210000; 0.990000; 48.000000; 2222.199951; 999.000000;]

Enter the command
4
Array: [ 12.300000; -23.200001; -12.210000; 48.000000; 2222.199951;]
Task Array: [ 9.121000; -905.000000; 0.990000; 999.000000;]

Enter the command
5
Array: [ 12.300000; -23.200001; -12.210000; 48.000000; 2222.199951;]

Enter the command
3
Enter the index value (to delete): -1
Incorrect index value (<0 or not int), enter again
3
Array: [ 12.300000; -23.200001; -12.210000; 2222.199951;]

Enter the command
2
Enter the index value (to vst): 1000
Enter the vst float number: 123456
Array: [ 12.300000; -23.200001; -12.210000; 2222.199951; 123456.000000;]

Enter the command
6
==27875==
==27875== HEAP SUMMARY:
==27875==     in use at exit: 0 bytes in 0 blocks
==27875==   total heap usage: 13 allocs, 13 frees, 2,264 bytes allocated
==27875==
==27875== All heap blocks were freed -- no leaks are possible
==27875==
==27875== For lists of detected and suppressed errors, rerun with: -s
==27875== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[gavrilov.da@unix lab3]$
```

Тест работы с новым массивом после повторной инициализации:

```
[gavrilov.da@unix lab3]$ valgrind ./reslab3
==28010== Memcheck, a memory error detector
==28010== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==28010== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==28010== Command: ./reslab3
==28010==

Commands list:
1) Enter mas
2) Insert element
3) Delete element
4) Individual task
5) Output mas
6) Stop the program
Enter the command
1
Enter the number of members of array: 5
Enter 1 float member of array (<= 5 meaningful signs): -0.0097
Enter 2 float member of array (<= 5 meaningful signs): 123241
Enter 3 float member of array (<= 5 meaningful signs): 9122
Enter 4 float member of array (<= 5 meaningful signs): -12
Enter 5 float member of array (<= 5 meaningful signs): 4.21
Array: [ -0.009700; 123241.000000; 9122.000000; -12.000000; 4.210000;]

Enter the command
4
Array: [ 123241.000000; -12.000000; 4.210000;]
Task Array: [ -0.009700; 9122.000000;]

Enter the command
3
Enter the index value (to delete): 0
Array: [ -12.000000; 4.210000;]

Enter the command
1
Enter the number of members of array: 3
Enter 1 float member of array (<= 5 meaningful signs): 0.002
Enter 2 float member of array (<= 5 meaningful signs): 12
Enter 3 float member of array (<= 5 meaningful signs): 42
Array: [ 0.002000; 12.000000; 42.000000;]

Enter the command
4
Array: [ 0.002000; 12.000000; 42.000000;]
Task Array: []

Enter the command
2
Enter the index value (to vst): 1
Enter the vst float number: 18.32
Array: [ 0.002000; 18.320000; 12.000000; 42.000000;]

Enter the command
6
==28010==
==28010== HEAP SUMMARY:
==28010==     in use at exit: 0 bytes in 0 blocks
==28010==   total heap usage: 10 allocs, 10 frees, 2,144 bytes allocated
==28010==
==28010== All heap blocks were freed -- no leaks are possible
==28010==
==28010== For lists of detected and suppressed errors, rerun with: -s
==28010== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[gavrilov.da@unix lab3]$
```

7. Ошибки и препятствия при выполнении работы

1. В ходе выполнения данной работы я столкнулся с проблемой возникновения утечек с памятью при работе с динамическими массивами данных при их инициализации. Проблема связана с тем, что после работы с массивом и последующей новой инициализации другого (команда 1), место, выделенное под предыдущий массив не отдавалось обратно, не происходила операция free(m), поэтому программа в дальнейшем работала некорректно. Решением проблемы послужило добавление в функцию input() free(m), память, выделенная под предыдущий массив освобождалась и потерь не стало. Так же было добавлено free(m) в команду 6) остановки программы. Также стоит отметить, что во время первой инициализации очистка не производится (реализовано через flag в файле reslab3.c).

2. Возникли небольшие трудности в реализации эффективного алгоритма вставки элемента в массив. После принятия решения о проходе (с конца) и сдвига только нужных нам элементов, программа стала работать эффективнее. Так же пришлось придумать отдельное разветвление алгоритма, если добавить элемент нужно в самый конец.

3. Также в ходе выполнения индивидуального задания, презентации алгоритма преподавателю, были учтены недочёты и было сделано упрощение алгоритма проверки вещественного числа на условие. (целую часть числа не обязательно было инвертировать)

8. Выводы

В ходе выполнения данной работы на примере программы, выполняющей базовые алгоритмы над массивами данных, были рассмотрены базовые принципы работы с динамическими массивами и управлением памяти в языке Си:

1. Основные алгоритмы по работе с массивами.
2. Выделение и освобождение нужного количества памяти посредством функций malloc(), realloc(), free().
3. Организация системы подпрограмм, находящихся в разных файлах.
4. Создание пользовательского интерфейса в виде меню с выбором действий.
5. Создание и работа с заголовочными файлами (.h) в языке Си.
6. Более подробная работа с указателями