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: Гатченко А.С.

Группа: Б22-525

Преподаватель: Половнева Ю. А.

Москва – 2022

1. Формулировка индивидуального задания

Из исходной последовательности целых чисел A сформировать новую последовательность B, значения элементов которой определяются по следующей формуле: $b_i = a_{3i} + a_{3i+1} + a_{3i+2}$. В исходной последовательности оставить только те числа, которые отсутствуют в сформированной последовательности.

Правила изменения размера выделенной под массив области памяти

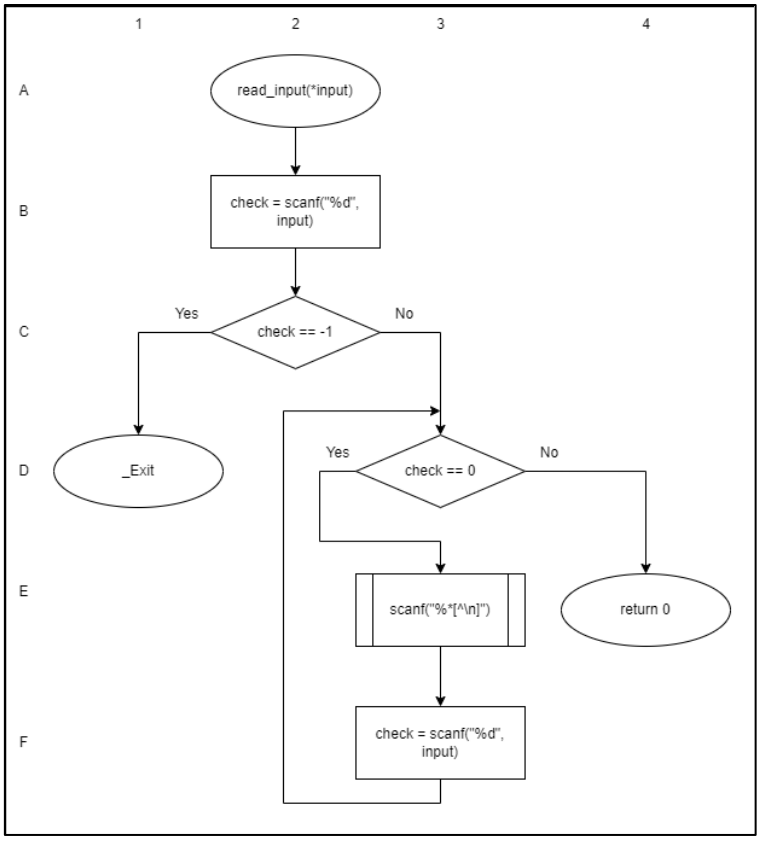
Размер выделенной под массив области памяти должен изменяться автоматически при выполнении операций, приводящих к изменению количества элементов в массиве.

При заполнении элементами массива всей выделенной под него области памяти её размер должен автоматически увеличиваться на объём, необходимый для размещения N дополнительных элементов массива. При наличии в выделенной под массив области памяти места для $N + 1$ новых элементов, её размер должен сократиться на объём, необходимый для хранения N элементов.

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовались встроенные типы данных int, предназначенные для работы с целыми числами, а также указатели на целые числа и на массивы целых чисел.

3. Описание использованного алгоритма



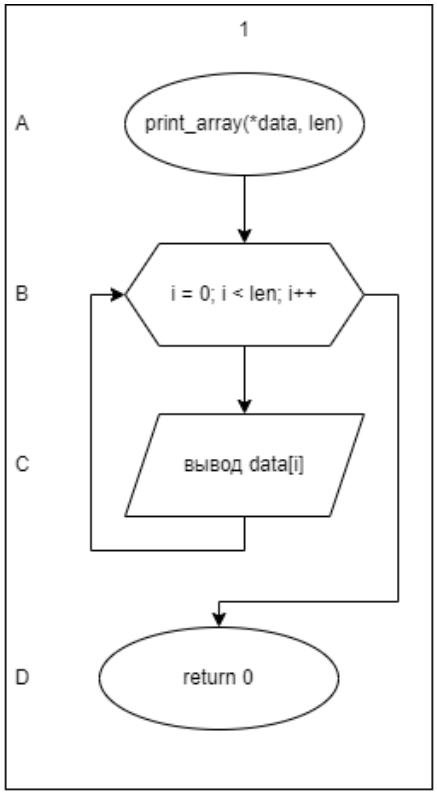
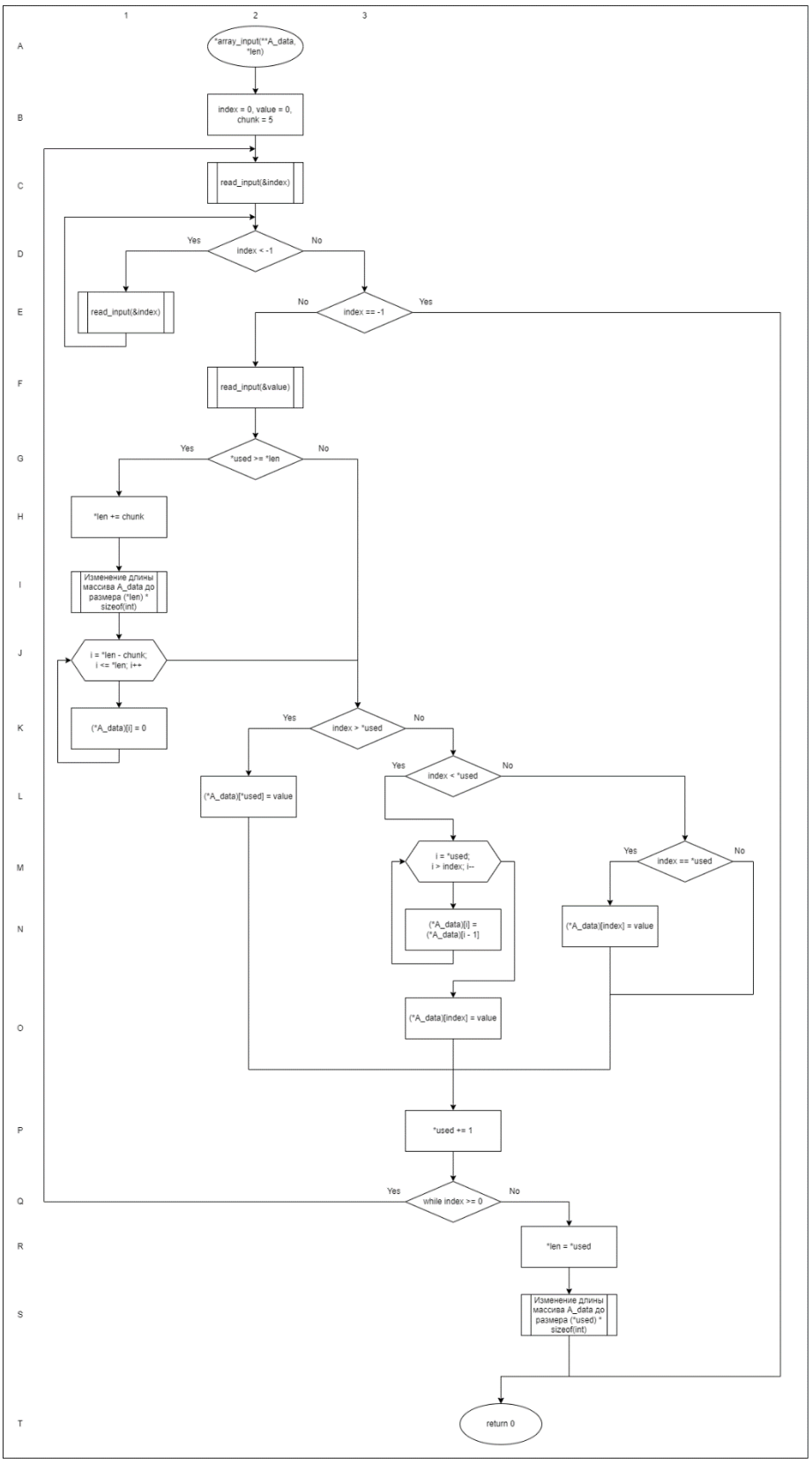


Рис. 1: Блок-схема алгоритма работы функции `read_input()`

Рис. 2: Блок-схема алгоритма работы функции `print_array()`



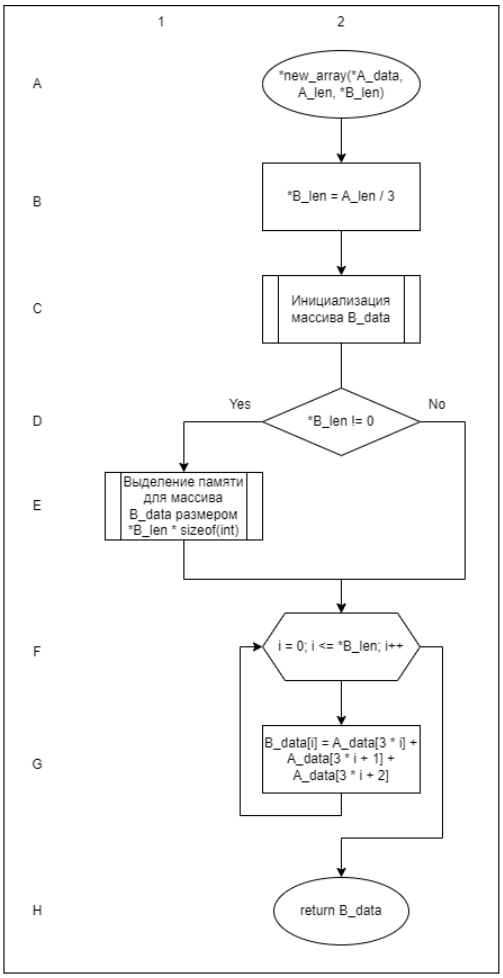


Рис. 3: Блок-схема алгоритма работы функции `array_input()`

Рис. 4: Блок-схема алгоритма работы функции `new_array()`

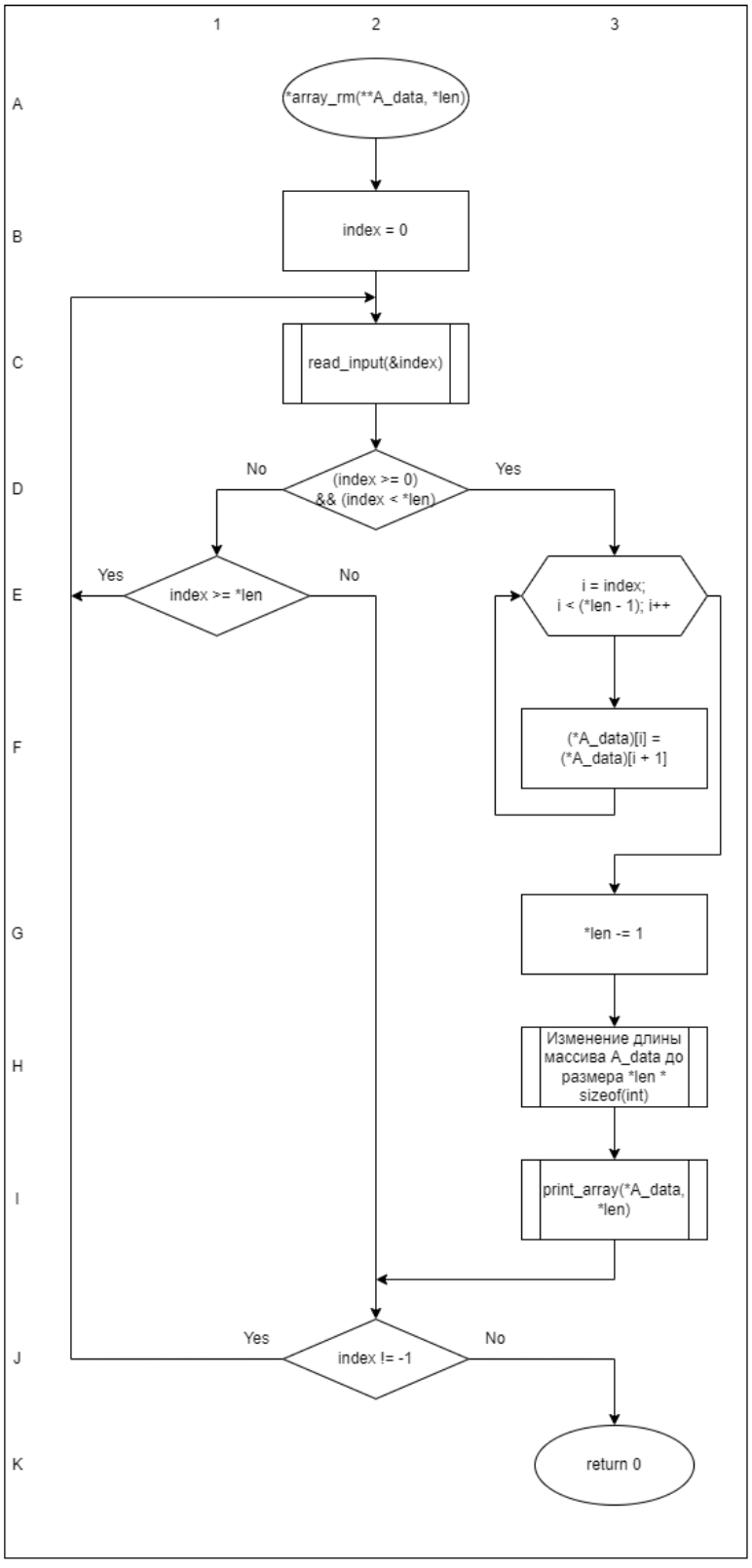


Рис. 5: Блок-схема алгоритма работы функции `array_rm()`

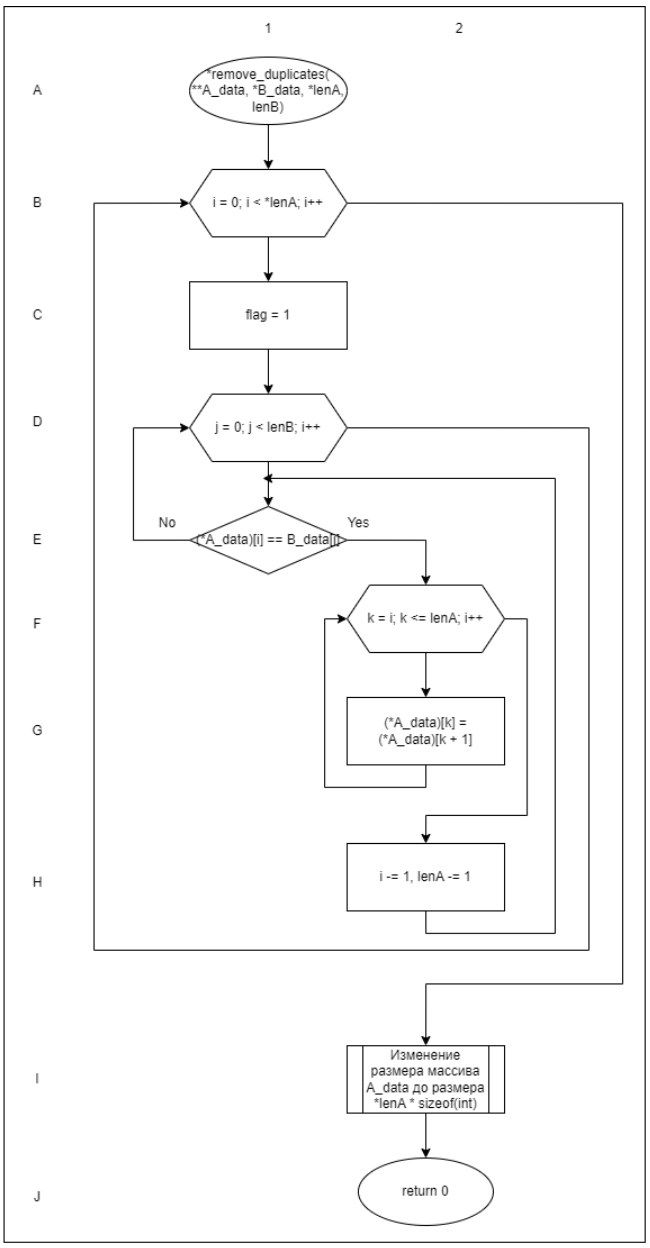
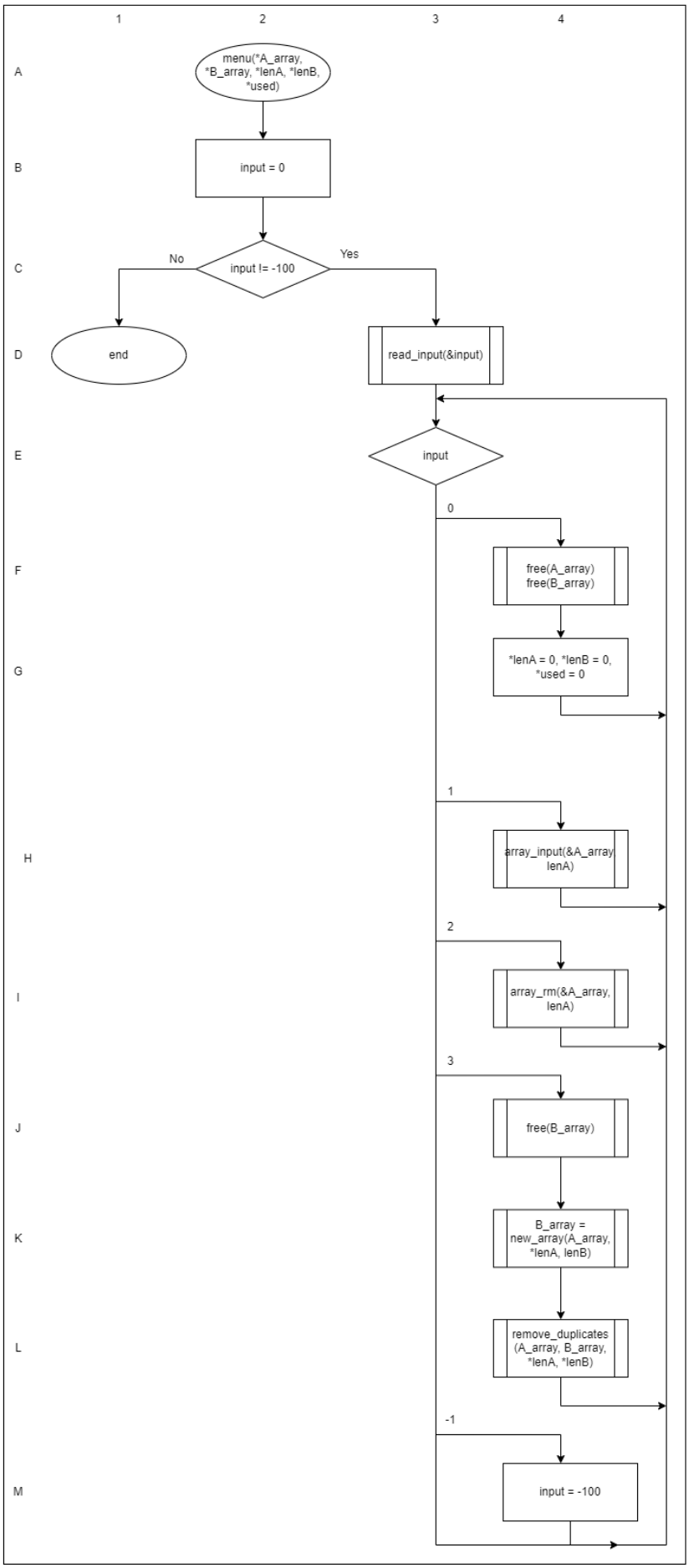


Рис. 6: Блок-схема алгоритма работы функции `remove_duplicates()`



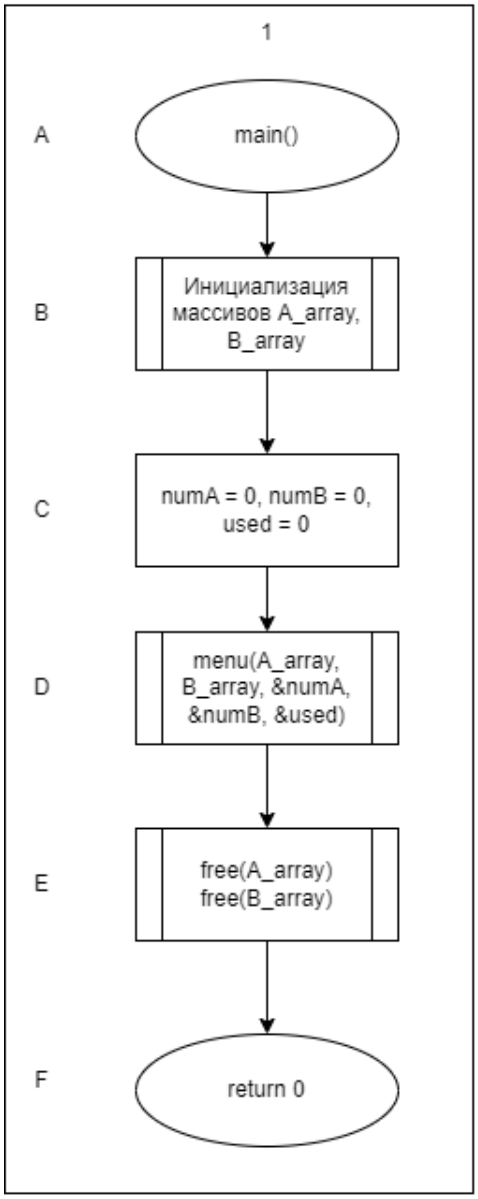


Рис. 7: Блок-схема алгоритма работы функции `menu()`

Рис. 8: Блок-схема алгоритма работы функции `main()`

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы lab3

1) Файл: main.c

```c
#include <stdio.h>
#include "menu.h"

int main(){
    int *A_array = NULL;
    int *B_array = NULL;
    int numA = 0;
    int numB = 0;
    int used = 0;
    menu(A_array, B_array, &numA, &numB, &used);
    free(A_array);
    free(B_array);
    return 0;
}
```

2) Файл: menu.c

```c
#include <stdio.h>
#include "ar_func.h"

void menu(int *A_array, int *B_array, int *lenA, int *lenB, int *used){
    int input = 0;

    while (input != -100){
        printf("Input 0 to initialize array from the beginning.\nInput 1 to
add some elements to the array.\nInput 2 to delete some elements from the
array.\nInput 3 to start the main function.\nInput -1 to stop program:\n");
        read_input(&input);
        switch (input){
           case 0:
              free(A_array);
              free(B_array);
              *lenA = 0;
              *lenB = 0;
              *used = 0;
              A_array = NULL;
              B_array = NULL;
              break;
            case 1:
                array_input(&A_array, lenA, used);
                break;
            case 2:
                array_rm(&A_array, lenA);
                break;
            case 3:
               free(B_array);
                B_array = new_array(A_array, *lenA, lenB);
                printf("B_array: ");
                print_array(B_array, *lenB);
                remove_duplicates(&A_array, B_array, lenA, *lenB);
                printf("Res_array: ");
                print_array(A_array, *lenA);
                break;
            case -1:
                printf("Program has been stopped");
                input = -100;
                free(A_array);
                free(B_array);
                break;
```

```
            default:
                printf("Incorrect value!\n");
        }
    }
}
```

3) Файл: ar_func.c

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

int read_input(int *input){
   int check = scanf("%d", input);
   if (check == -1){
      printf("Program has been stopped");
      _Exit (EXIT_SUCCESS);
   }
   while (check == 0){
      scanf("%*[^\n]");
      printf("You can't input letters.\n");
      check = scanf("%d", input);
   }
    return 0;
}

int print_array(int *data, int len){
    for (int i = 0; i < len; i++){
        printf("%d ", data[i]);
    }
    printf("\n");
    return 0;
}

int *array_input(int **A_data, int *len, int *used){
    int index = 0;
    int value = 0;
    int chunk = 5;
    printf("A_array: ");
    print_array(*A_data, *len);
    printf("Input an index of a new element, or -1 to stop this input:\n");

    do {
       read_input(&index);
        while (index < -1){
            printf("Please input an index in positive numbers:\n");
            read_input(&index);
        }
        if (index == -1){
            break;
        }

        read_input(&value);

        if (*used >= *len){
           *len += chunk;
           *A_data = realloc(*A_data, (*len) * sizeof(int));
           for (int i = (*len - chunk); i < *len; i++){
              (*A_data)[i] = 0;
           }
        }

        if (index > *used){
           printf("Adding element to the end of the real-used memory!\n");
           (*A_data)[*used] = value;
        }
```

```
        else if (index < *used){
            for (int i = *used; i > index; i--){
                (*A_data)[i] = (*A_data)[i - 1];
            }
            (*A_data)[index] = value;
        }
        else if (index == *used){
           (*A_data)[index] = value;
        }
        *used += 1;
        printf("A_array: ");
        print_array(*A_data, *len);
    }
    while (index >= 0);
    *len = *used;
    *A_data = realloc(*A_data, *used * sizeof(int));

    return 0;
}

int *array_rm(int **A_data, int *len){
    int index = 0;

    do {
        printf("Input index of an element you want to remove. Otherwise,
input -1.\n");
        read_input(&index);
        if ((index >= 0) && (index < *len)){
            for (int i = index; i < (*len - 1); i++){
                (*A_data)[i] = (*A_data)[i + 1];
            }
            *len = (*len) - 1;
            *A_data = realloc(*A_data, *len * sizeof(int));
            printf("A_array: ");
            print_array(*A_data, *len);
        }
        else if (index >= *len){
            printf("Index must be less than a length of an array!\n");
        }
    }
    while (index != -1);
    return 0;
}

int *new_array(int *A_data, int A_len, int *B_len){
   int *B_data = NULL;
    *B_len = A_len / 3;
    if (*B_len != 0){
       B_data = malloc((*B_len) * sizeof(int));
    }

    for (int i = 0; i < *B_len; i++){
        B_data[i] = A_data[3 * i] + A_data[3 * i + 1] + A_data[3 * i + 2];
    }

    return B_data;
}

int *remove_duplicates(int **A_data, int *B_data, int *lenA, int lenB){
    for (int i = 0; i < *lenA; i++){
        for (int j = 0; j < lenB; j++){
            if ((*A_data)[i] == B_data[j]){
               for (int k = i; k <= *lenA; k++){
```

```
                    (*A_data)[k] = (*A_data)[k + 1];
                }
                i -= 1;
                *lenA -= 1;
                break;
            }
        }
    }
    *A_data = realloc(*A_data, (*lenA) * sizeof(int));

    return 0;
}
```

4) Файл: menu.h

```
#ifndef MENU
#define MENU
#include <stdio.h>
#include <stdlib.h>

void menu(int *A_array, int *B_array, int *lenA, int *lenB, int *used);
#endif
```

5) Файл: ar_func.h

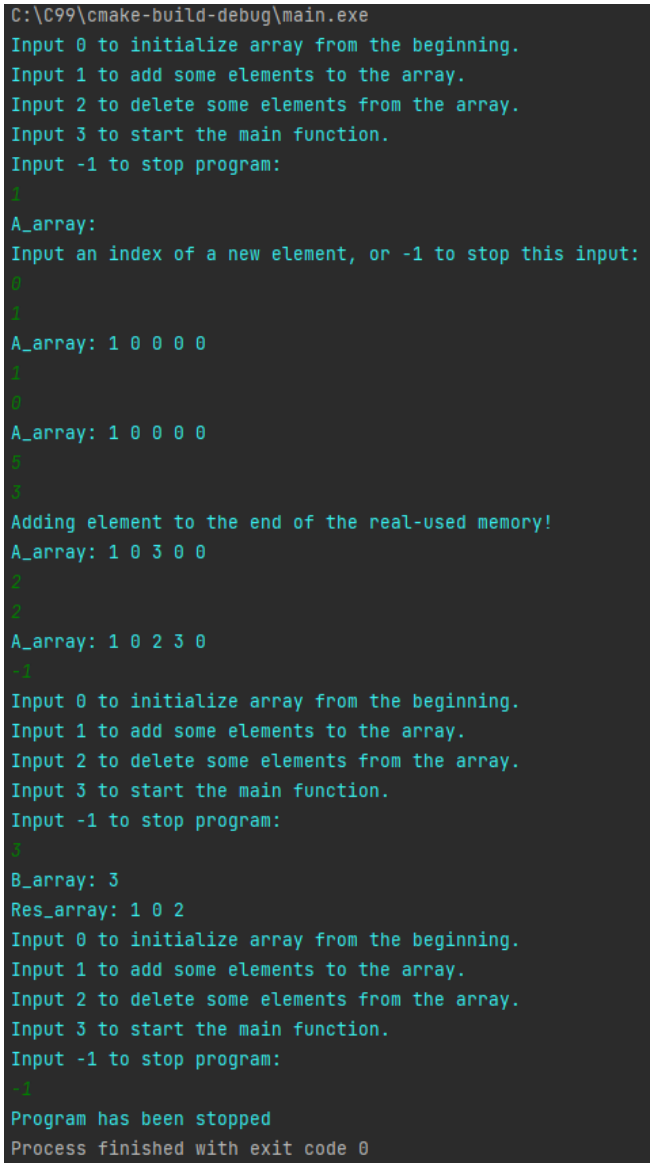
```
#ifndef AR_FUNC
#define AR_FUNC
#include <stdio.h>
#include <stdlib.h>

int read_input(int *input);
void print_array(int *data, int len);
int *array_input(int **A_data, int *len, int *used);
int *array_rm(int **A_data, int *len);
int *new_array(int *A_data, int A_len, int *B_len);
int *remove_duplicates(int **A_data, int *B_data, int *lenA, int lenB);

#endif
```

5. Описание тестовых примеров

Таблица 1: Тестовые примеры lab3.c

Изначальное значение массива A_array	Полученное значение массива B_array	Ожидаемое значение массива A_array после изменения	Полученное значение массива A_array после изменения
[1, 0, 2, 3]	[3]	[1, 0, 2]	[1, 0, 2]
[0, 5, 7, 5, 12, 1]	[12, 18]	[0, 5, 7, 5, 1]	[0, 5, 7, 5, 1]
[1, 0, 0]	[1]	[0, 0]	[0, 0]

6. Скриншоты



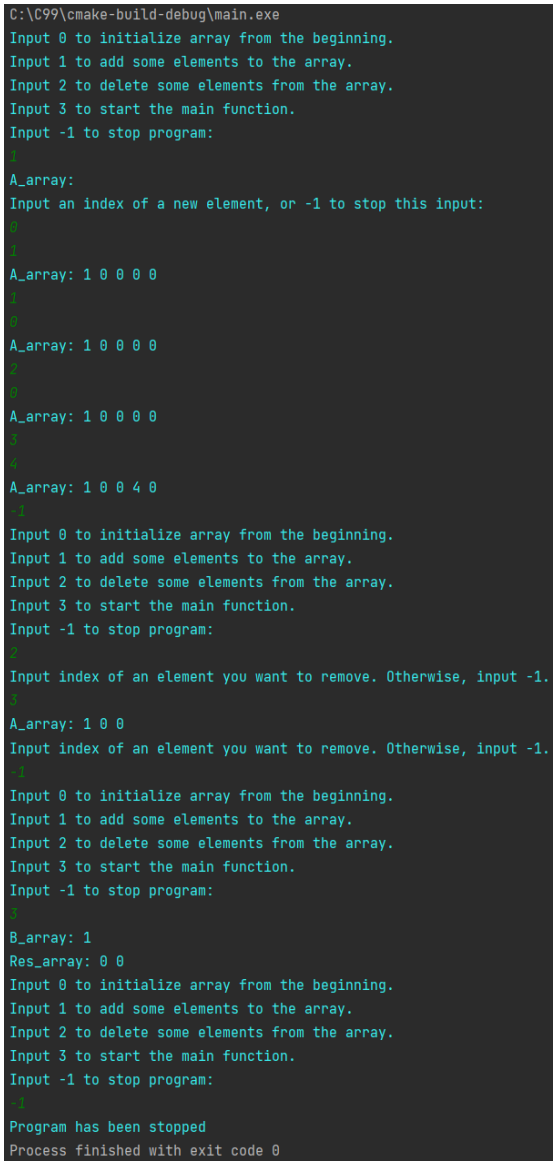


Рис. 9 и 10: Запуск программы lab3

7. Выводы

В ходе выполнения данной работы на примере программы, выполняющей вычисление значения функции в точке при помощи разложения в ряд, были рассмотрены базовые принципы работы построения программ на языке С и обработки чисел с плавающей запятой:

1. Организация ввода/вывода, а также проверка на ввод только целочисленных значений.
2. Разработка функций.
3. Объявление и использование переменных.
4. Выполнение простейших арифметических операций над целочисленными и дробными операндами.
5. Использование циклов и условий.

6. Использование указателей (параметров) на целые числа, а также на массивы целых чисел.
7. Разбиение программы на несколько файлов.