Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №12 «Компьютерные системы и технологии»

**ОТЧЕТ**

**О выполнении лабораторной работы №3**

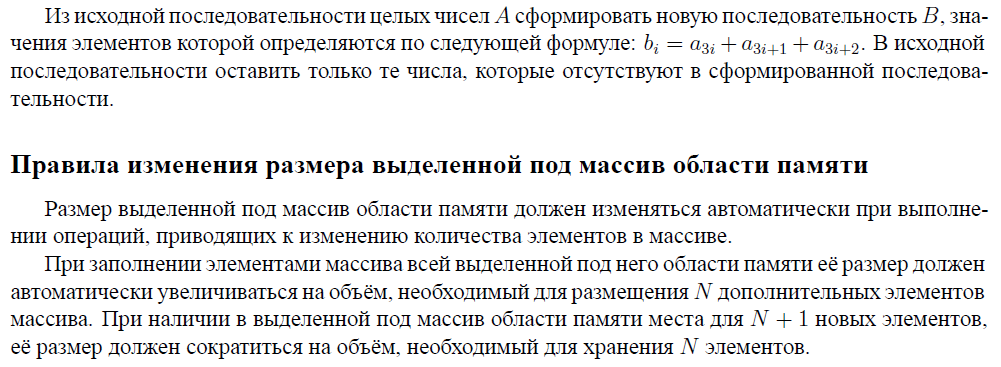
**«Работа с массивами данных»**

**Cтудент**: Гатченко А.С.

**Группа**: Б22-525

**Преподаватель**: Половнева Ю. А.

**1. Формулировка индивидуального задания**



**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**

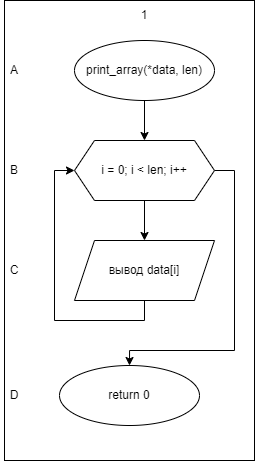
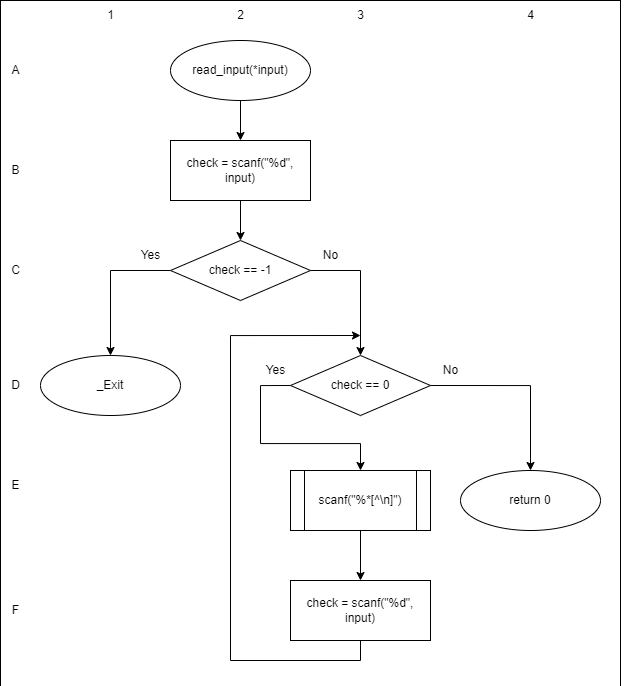


Рис. 1: Блок­схема алгоритма работы функции read\_input()

Рис. 2: Блок-схема алгоритма работы функции print\_array()

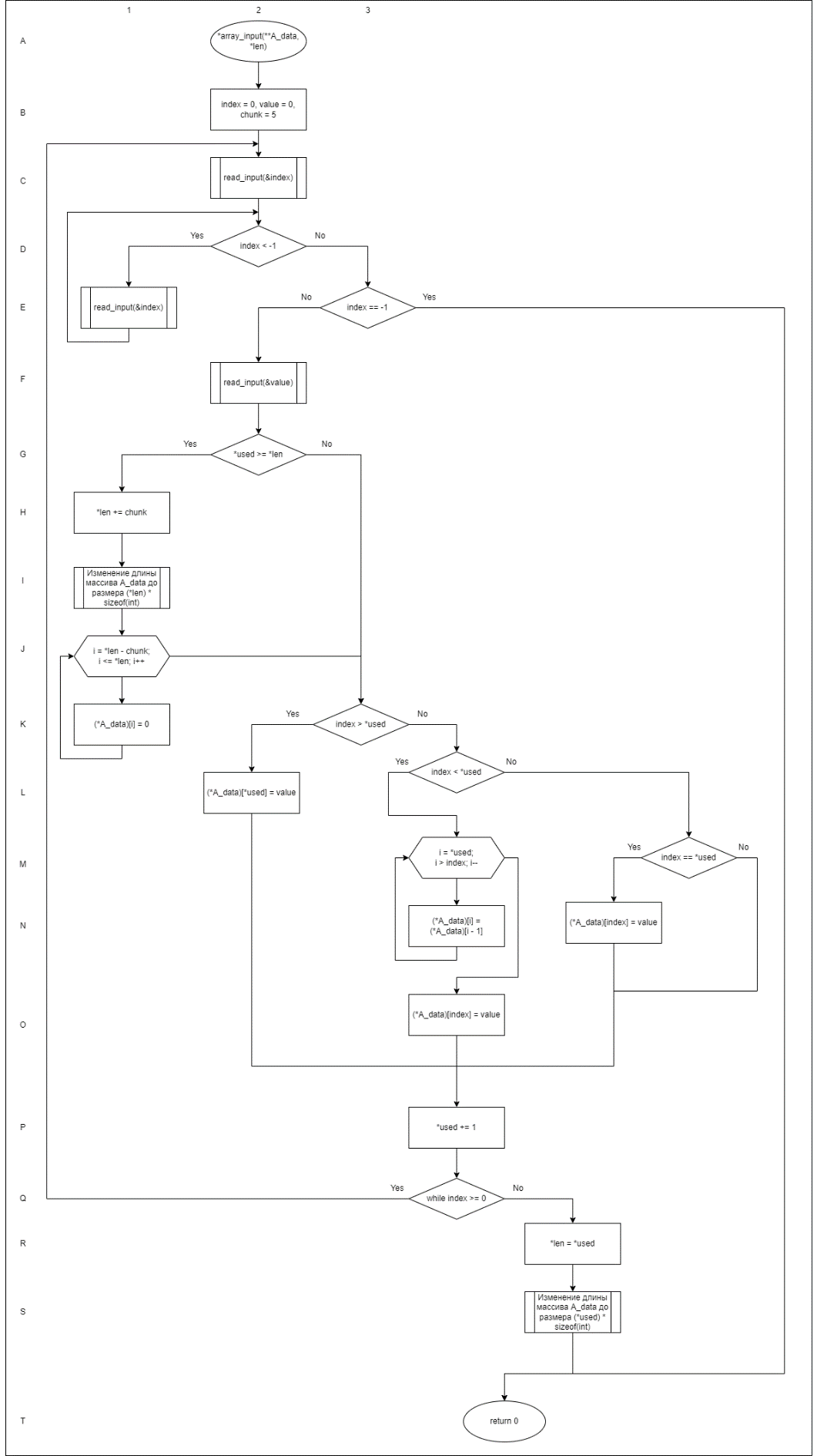
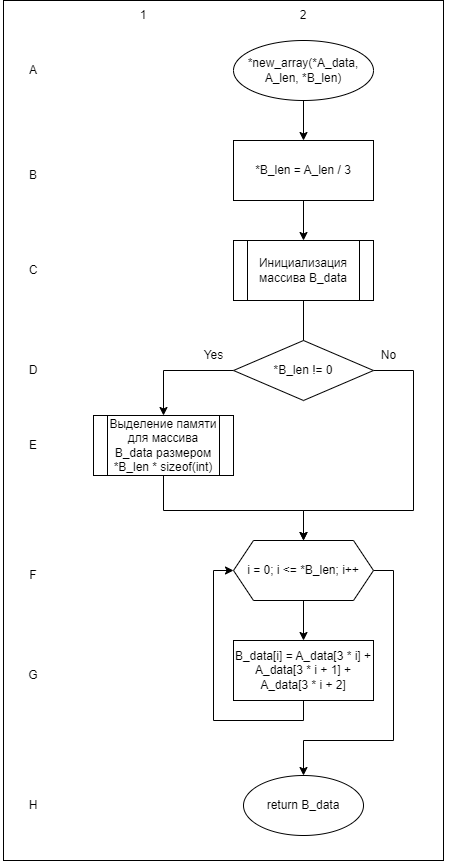


Рис. 3: Блок-схема алгоритма работы функции array\_input()

Рис. 4: Блок-схема алгоритма работы функции new\_array()

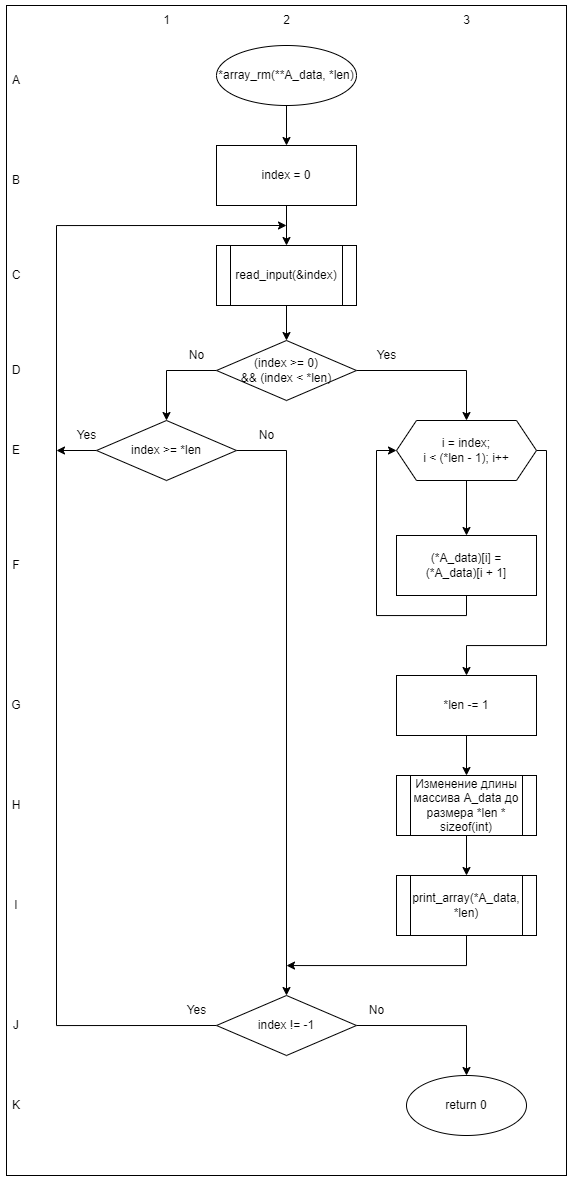
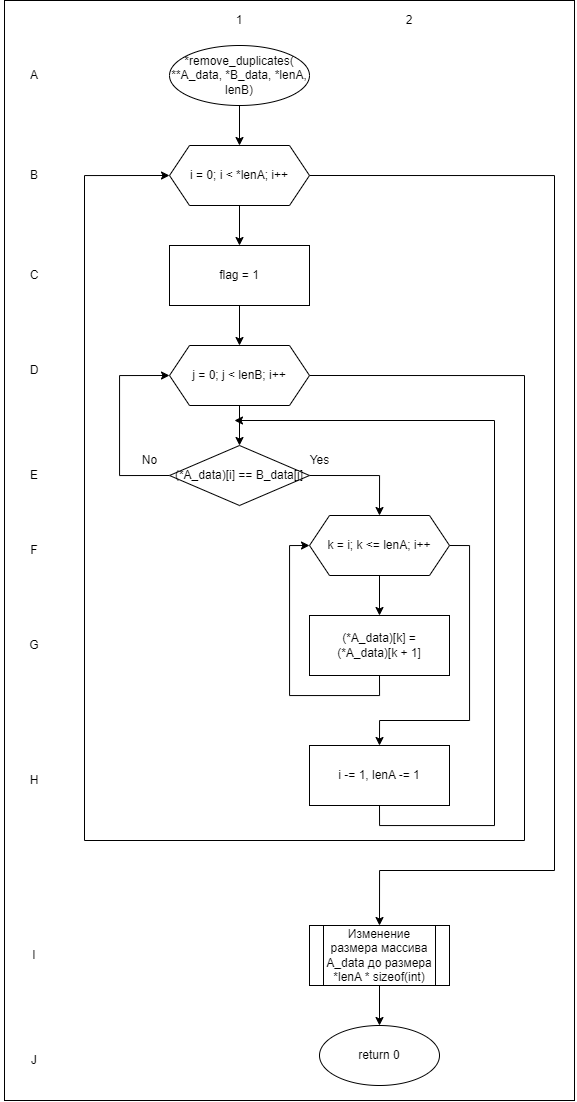


Рис. 5: Блок-схема алгоритма работы функции array\_rm()

Рис. 6: Блок-схема алгоритма работы функции remove\_duplicates()

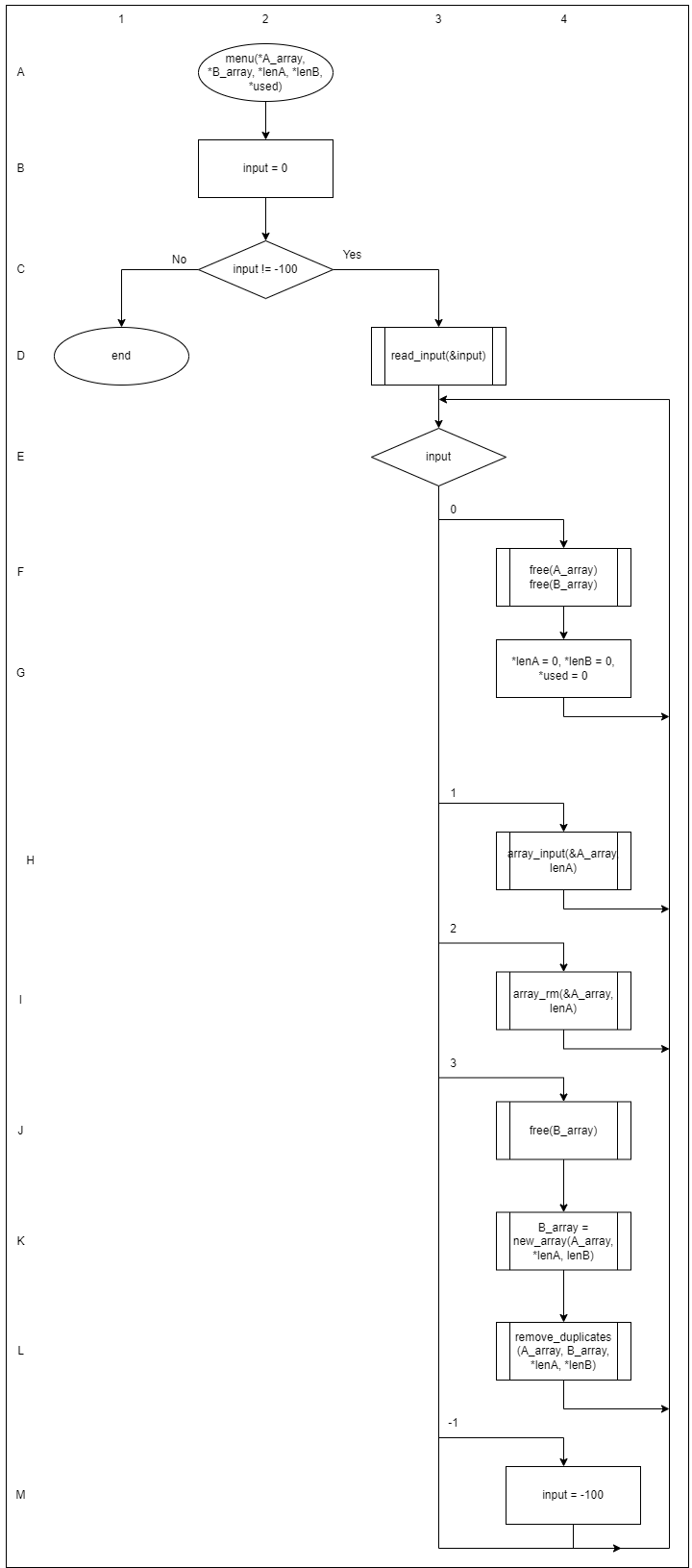
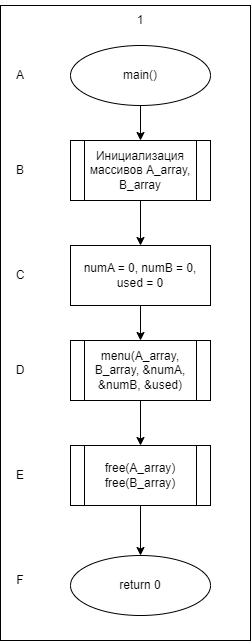


Рис. 7: Блок-схема алгоритма работы функции menu()

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

**4. Исходные коды разработанных программ**

Листинг 1: Исходные коды программы lab3

1) Файл: main.c

#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

#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**

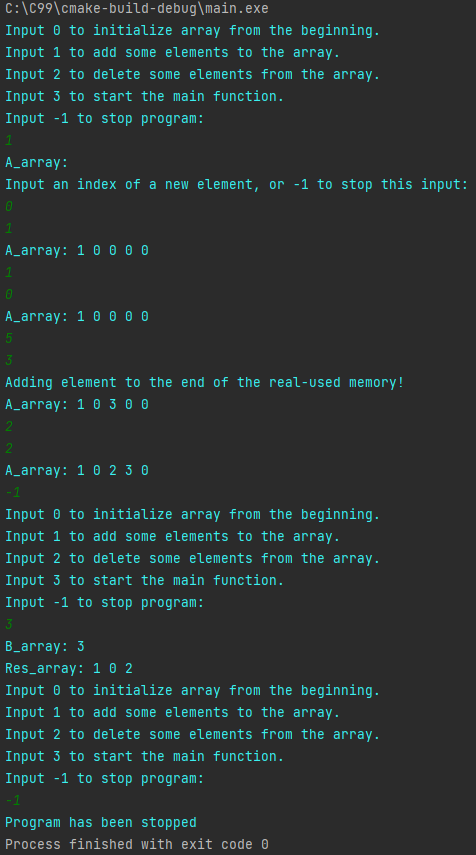
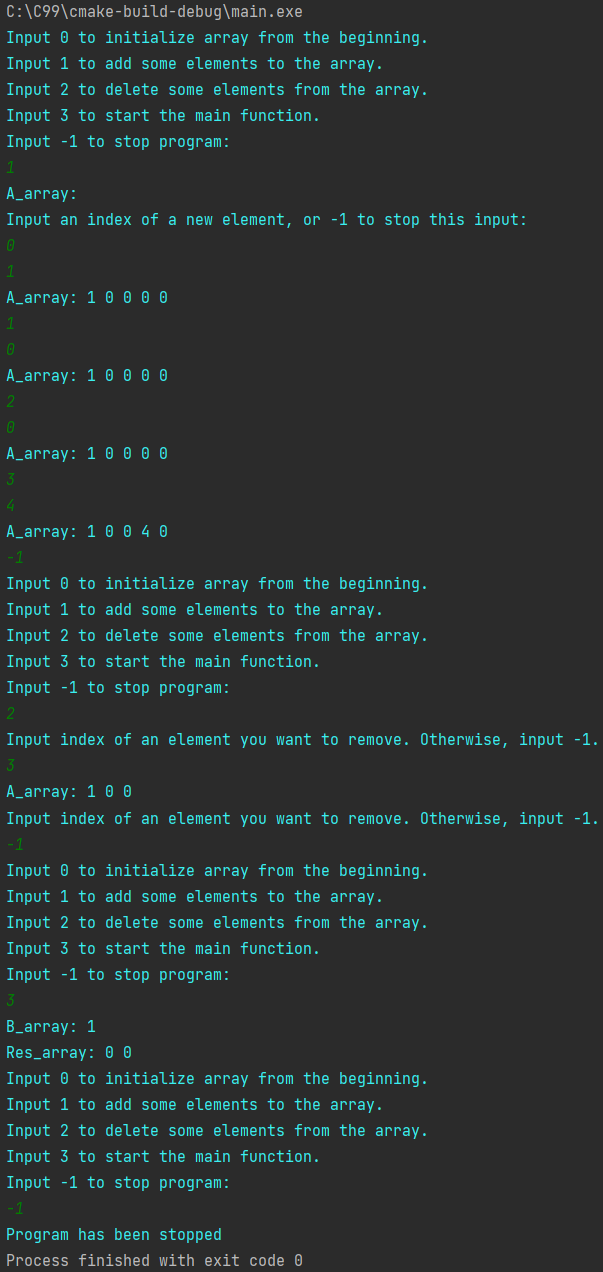


Рис. 9 и 10: Запуск программы lab3

**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 C 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.