Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №12 «Компьютерные системы и технологии»

|  |  |  |
| --- | --- | --- |
| Изображение выглядит как графическая вставка, Графика, графический дизайн, мультфильм  Автоматически созданное описание | ИИКС НИЯУ МИФИ — Институт интеллектуальных кибернетических систем | Изображение выглядит как снимок экрана, Графика, дизайн  Автоматически созданное описание |

**ОТЧЕТ**

**О выполнении лабораторной работы №3**

**«Работа с массивами данных»**

**Студент:** Рыженко Р.В.

**Группа:** Б23-506

**Преподаватель:** Курочкина М-А.А.

*Москва 2023*

1. **Формулировка индивидуального задания**

Вариант №56. В исходной последовательности вещественных чисел найти те, дробная часть которых, представленная в виде целого числа, превышает по модулю их целую часть. Сформировать из данных чисел новую последовательность, удалив их из исходной.

1. **Описание использованных типов данных**

При выполнении данной лабораторной работы использовался встроенные типы данных double и int, предназначенные для работы с вещественными и целыми числами, и указатели, предназначенные для работы с адресами в памяти.

1. Изображение выглядит как диаграмма, Технический чертеж, План, схематичный

   Автоматически созданное описание**Описание использованного алгоритма**

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

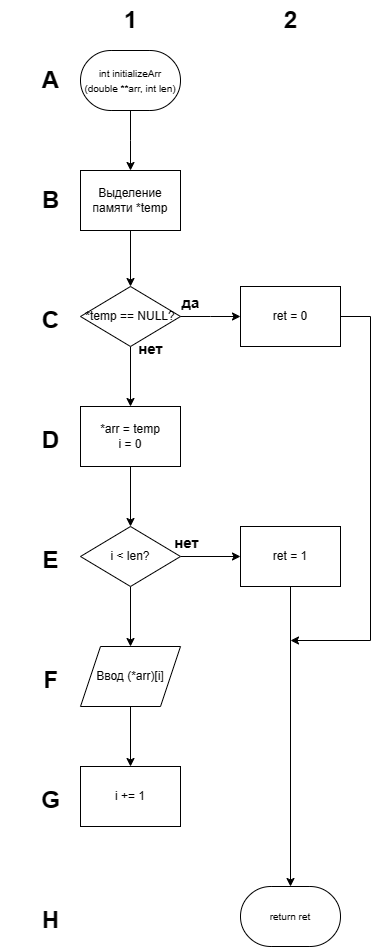
­­­

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

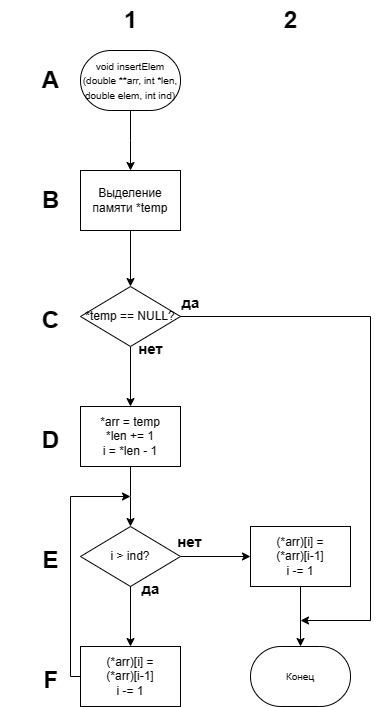


Рис. 3: Блок-схема алгоритма работы функции insertElem()

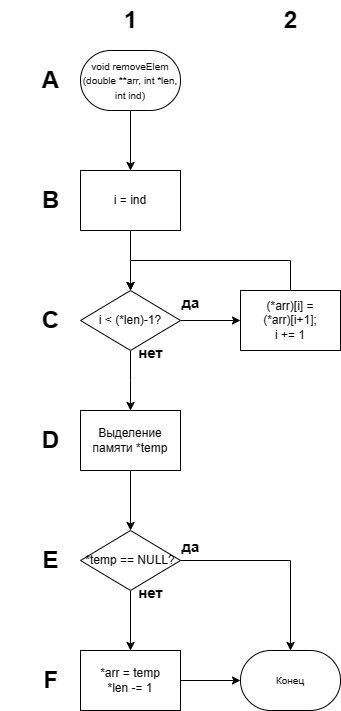


Рис. 4: Блок-схема алгоритма работы функции removeElem()

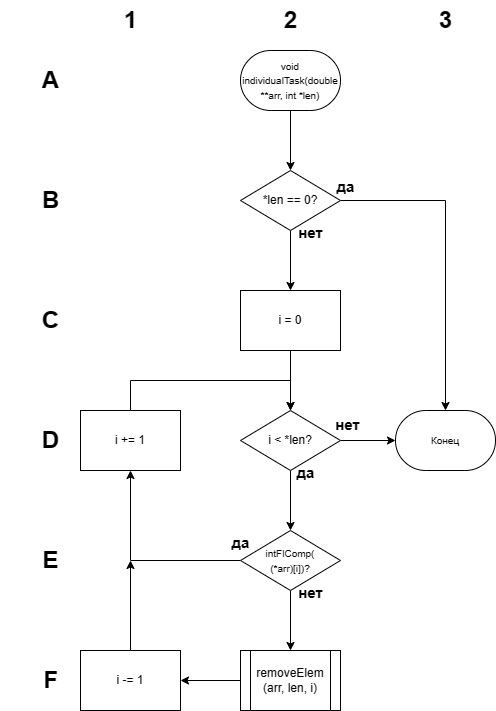


Рис. 5: Блок-схема алгоритма работы функции individualTask()

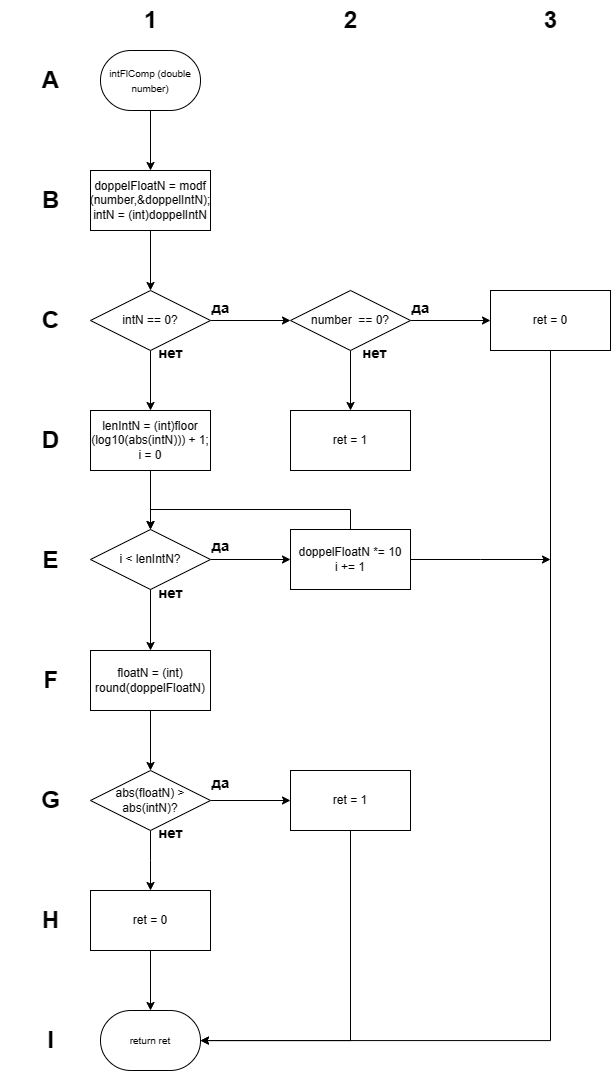


Рис. 6: Блок-схема алгоритма работы функции intFlComp()

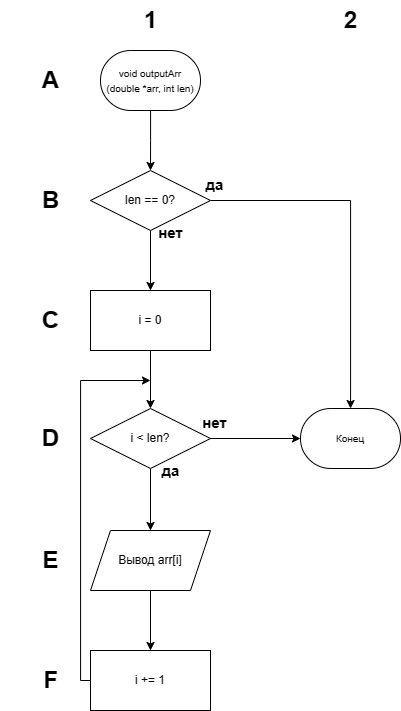


Рис. 7: Блок-схема алгоритма работы функции outputArr()

1. **Исходные коды разработанных программ**

Листинг 1: Исходный код функции main() (файл: main.c)

#include <stdio.h>

#include <stdlib.h>

#include "individualTask.h"

#include "initializeArr.h"

#include "insertElem.h"

#include "intFlComp.h"

#include "outputArr.h"

#include "removeElem.h"

#include "safeScanfDouble.h"

#include "safeScanfInt.h"

int main()

{

double \*arr = NULL;

int len = 0, lencopy;

int fl = 1;

int panel;

int returned;

double elem;

int ind;

while (fl == 1) {

panel = -1; ind = -1;

printf("Выберите одну из опций:\n\

(1) Инициализация массива\n\

(2) Вставка нового элемента\n\

(3) Удаление элемента\n\

(4) Индивидуальное задание\n\

(5) Вывод содержимого массива\n\

(6) Завершение программы\n");

while (panel < 0 || panel > 6) {

returned = safeScanfInt(&panel, 0);

if (returned == 0) {

free(arr);

return 0;

}

if (panel < 0 || panel > 6) {

printf("Введите число от 1 до 6!\n");

}

}

switch(panel) {

case 1:

lencopy = 0;

printf("Задайте длину массива\n");

while (lencopy < 1) {

returned = safeScanfInt(&lencopy, 1);

if (returned == 0) {

free(arr);

return 0;

}

if (lencopy < 1) {

printf("Введите целое число не меньше 0!\n");

}

}

returned = initializeArr (&arr, lencopy);

if (returned == -1) {

free(arr);

return 0;

}

else if (returned == 1) {

len = lencopy;

}

outputArr (arr, len);

break;

case 2:

printf("Введите добавляемый элемент\n");

returned = safeScanfDouble (&elem);

if (returned == 0) {

free(arr);

return 0;

}

printf("Введите индекс добавления\n");

while (ind < 0) {

returned = safeScanfInt (&ind, 22);

if (returned == 0) {

free(arr);

return 0;

}

if (ind < 0) {

printf("Введите целое число не меньше 0!\n");

}

}

if (ind >= len) {

ind = len - 1 + 1; // -1, т.к. индекса len нет, а +1, т.к. в этой ф.-и массив расширится

}

insertElem (&arr, &len, elem, ind);

outputArr (arr, len);

break;

case 3:

if (len == 0) {

printf("Массив не проинициализирован - удалять нечего.\n");

break;

}

printf("Введите индекс удаления\n");

while (ind < 0 || ind >= len) {

returned = safeScanfInt (&ind, 22);

if (returned == 0) {

free(arr);

return 0;

}

if (ind < 0 || ind >= len) {

printf("Введите целое число не меньше 0 и меньше длины массива!\n");

}

}

removeElem (&arr, &len, ind);

outputArr (arr, len);

break;

case 4:

individualTask (&arr, &len);

outputArr (arr, len);

break;

case 5:

outputArr (arr, len);

break;

case 6:

printf("Завершение программы.\n");

free(arr);

return 0;

}

}

return 0;

}

Листинг 2: Исходный код функции initializeArr() (файл: initializeArr.c)

#include <stdio.h>

#include <stdlib.h>

#include "initializeArr.h"

#include "safeScanfDouble.h"

int initializeArr (double \*\*arr, int len)

{

double \*temp;

if (\*arr == NULL) {

temp = (double\*)malloc(len \* sizeof(double));

}

else {

temp = (double\*)realloc(\*arr, len \* sizeof(double));

}

if (temp == NULL) {

printf("Не найдена свободная память для инициализации нового массива\n");

return 0;

}

\*arr = temp;

int i;

int returned;

printf("Введите элементы массива\n");

for (i = 0; i < len; ++i) {

returned = safeScanfDouble( &((\*arr)[i]) );

if (returned == 0) {

return -1;

}

}

return 1;

}

Листинг 3: Исходный код функции insertElem() (файл: insertElem.c)

#include <stdio.h>

#include <stdlib.h>

#include "insertElem.h"

void insertElem (double \*\*arr, int \*len, double elem, int ind)

{

double \*temp;

if (\*arr == NULL) {

temp = (double\*)malloc(((\*len) + 1) \* sizeof(double));

}

else {

temp = (double\*)realloc(\*arr, ((\*len) + 1) \* sizeof(double));

}

if (temp == NULL) {

printf("Не найдена свободная память для инициализации нового массива\n");

return;

// return 0;

}

\*arr = temp;

(\*len)++;

int i;

for (i = (\*len)-1; i > ind; --i) {

(\*arr)[i] = (\*arr)[i-1];

}

(\*arr)[ind] = elem;

// return 1;

}

Листинг 4: Исходный код функции removeElem() (файл: removeElem.c)

#include <stdio.h>

#include <stdlib.h>

#include "removeElem.h"

void removeElem (double \*\*arr, int \*len, int ind)

{

int i;

for (i = ind; i < (\*len)-1; ++i) {

(\*arr)[i] = (\*arr)[i+1];

}

if (\*len == 1) {

free(arr);

\*arr = NULL;

\*len = 0;

return;

}

double \*temp;

if (\*arr == NULL) {

temp = (double\*)malloc(((\*len) - 1) \* sizeof(double));

}

else {

temp = (double\*)realloc(\*arr, ((\*len) - 1) \* sizeof(double));

}

if (temp == NULL) {

printf("Не найдена свободная память для инициализации нового массива\n");

printf("Выбранный элемент удалён без уменьшения длины массива\n");

return;

// return 0;

}

\*arr = temp;

(\*len)--;

// return 1;

}

Листинг 5: Исходный код функции individualTask() (файл: individualTask.c)

#include <stdio.h>

#include <stdlib.h>

#include "individualTask.h"

#include "intFlComp.h"

#include "removeElem.h"

void individualTask(double \*\*arr, int \*len)

{

if (\*len == 0) {

printf("Нет цели для выполонения действия: массив не проинициализирован.\n");

return;

}

int i;

for (i = 0; i < \*len; ++i) {

if ( 1 - intFlComp((\*arr)[i]) ) {

removeElem(arr, len, i);

i--;

}

}

}

Листинг 6: Исходный код функции intFlComp() (файл: intFlComp.c)

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include "intFlComp.h"

int intFlComp (double number)

{

double doppelIntN;

double doppelFloatN;

doppelFloatN = modf(number,&doppelIntN);

int intN = (int)doppelIntN;

// printf("%lf\n%d %lf\n", number, intN, floatN);

if (intN == 0) {

if (number != 0) {

return 1;

}

else {

return 0;

}

}

int lenIntN = (int)floor(log10(abs(intN))) + 1;

// printf("%d\n",lenIntN);

int i;

for (i = 0; i < lenIntN; i++) {

doppelFloatN \*= 10;

}

int floatN = (int)round(doppelFloatN);

// printf("%d %lf %d\n", intN, doppelFloatN, floatN);

if (abs(floatN) > abs(intN)) {

return 1;

}

return 0;

}

Листинг 7: Исходный код функции outputArr() (файл: outputArr().c)

#include <stdio.h>

#include <stdlib.h>

#include "outputArr.h"

void outputArr (double \*arr, int len)

{

if (len == 0) {

printf("Массив ещё не проинициализирован\n");

return;

}

int i;

for (i = 0; i < len; ++i) {

printf("%lf ", arr[i]);

}

printf("\n");

}

Листинг 8: Исходный код функции safeScanfDouble (файл: safeScanfDouble.c)

#include <stdio.h>

#include <stdlib.h>

#include "safeScanfDouble.h"

int safeScanfDouble (double \*target)

{

int guard;

int flag = 1;

while (flag == 1) {

guard = scanf("%lf",target);

scanf("%\*[^\n]");

if (guard == EOF) {

printf("Завершение программы.\n");

return 0;

}

if (guard < 1) {

printf("Введите число!\n");

continue;

}

flag = 0;

}

return 1;

}

Листинг 9: Исходный код функции safeScanfInt (файл: safeScanfInt.c)

#include <stdio.h>

#include <stdlib.h>

#include "safeScanfInt.h"

int safeScanfInt (int \*target, int panel)

{

int guard;

int flag = 1;

while (flag == 1) {

guard = scanf("%d",target);

scanf("%\*[^\n]");

if (guard == EOF) {

printf("Завершение программы.\n");

return 0;

}

if (guard < 1) {

switch (panel) {

case 0:

printf("Введите целое число от 1 до 6!\n");

break;

case 1:

case 22:

printf("Введите целое число не меньше 0!\n");

break;

case 3:

printf("Введите целое число не меньше 0 и меньше длины массива!\n");

}

continue;

}

flag = 0;

}

return 1;

}

1. **Описание тестовых примеров**

Таблица 1: Тестовые примеры

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Операция | Название | Индекс | Элемент | Ожидаемый результат | Полученный результат |
| EOF | Завершение | - | - | Завершение работы | Завершение работы |
| 2 | Вставка нового элемента | 100 | 43 | 43 | 43.000000 |
| 1 | Инициализация массива | 4 | 15 100000 -3 0 | 15 100000 -3 0 | 15.000000 100000.000000  -3.000000 0.000000 |
| 2 | Вставка нового элемента | 2 | 2 | 15 100000 2 -3 0 | 15.000000 100000.000000  2.000000 -3.000000 0.000000 |
| 3 | Удаление элемента | a |  | Ошибка: не целое число | Ошибка: не целое число |
|  | Удаление элемента | 1 | - | 15 2 -3 0 | 15.000000 2.000000  -3.000000 0.000000 |
| 4 |  | - | - | Массив пуст | Массив пуст |
| 1 |  | 6 | 12.97 97.12  -134.14 321.1234 0 55.55 | 12.97 97.12 -134.14 321.1234 0 55.55 | 12.970000 97.120000 -134.140000 321.123400 0.000000 55.550000 |
| 4 |  | - | - | 12.97 -134.14 | 12.970000  -134.140000 |
| 5 |  | - | - | 12.97 -134.14 | 12.970000  -134.140000 |
| 6 |  | - | - | Завершение работы | Завершение работы |

1. **Скриншоты**

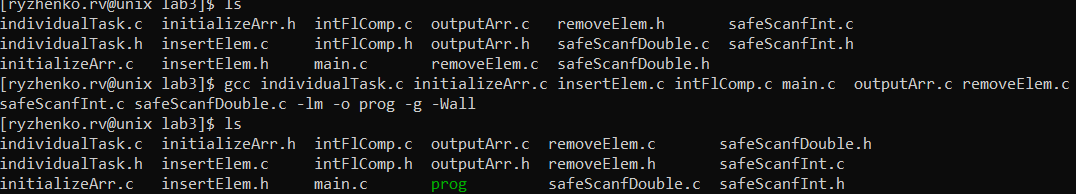


Рис. 8: Сборка программы prog

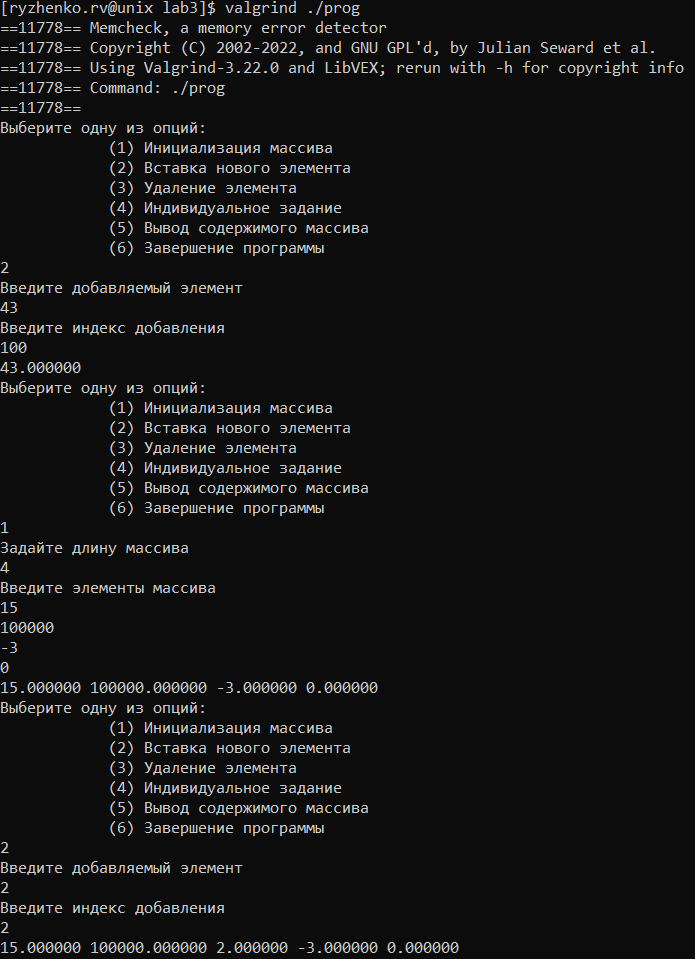


Рис. 9: Запуск программы prog

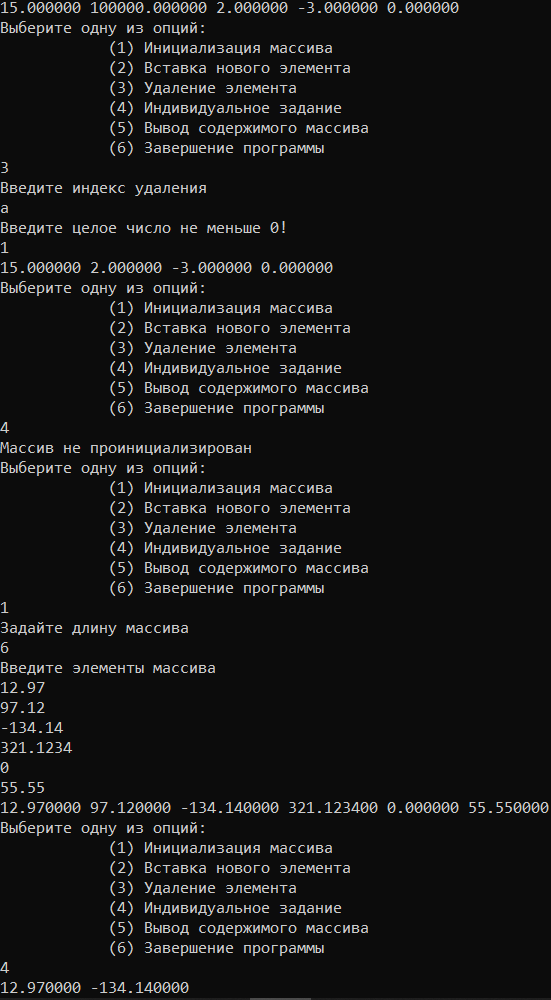


Рис. 10: Запуск программы prog

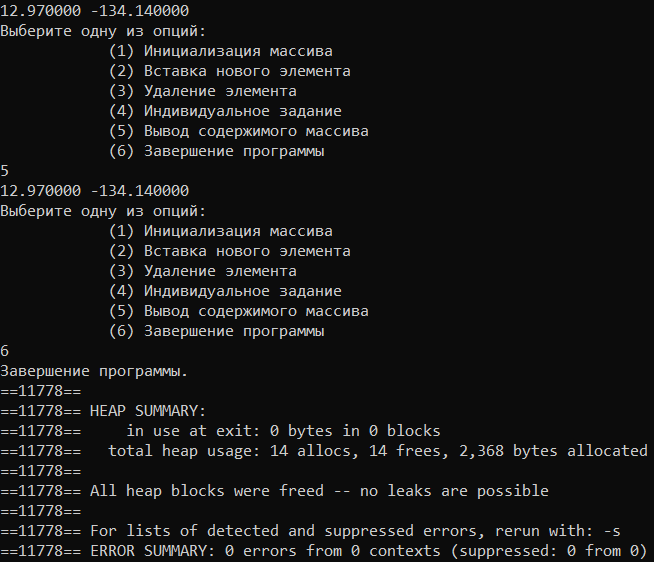


Рис. 11: Запуск программы prog

1. **Выводы**

В ходе выполнения данной работы на примере программы, выполняющей работу с массивом, были рассмотрены базовые принципы работы построения программ на языке C, обработки целых и вещественных чисел и массивов:

1. Организация ввода/вывода.
2. Разработка функций.
3. Объявление и использование переменных.
4. Выполнение простейших арифметических операций над целочисленными и вещественными операндами.
5. Реализация и операции над массивами.