Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 6 «Функции»

по дисциплине «Программирование и основы алгоритмизации»

Выполнил: Комбаев И. Д.

студент группы 609-01

Проверил: Гришмановский П. В.

доцент кафедры автоматики и компьютерных систем

Сургут

2021 г.

Цель работы:

* закрепление теоретических знаний
* приобретение практических навыков объявления и реализации функций языка C, передачи параметров и возвращаемого значения при их вызове

Задание: требуется выполнить реализацию задачи в соответствии с индивидуальным заданием, выданным преподавателем по теме «Одномерные массивы», с использованием функций для обработки массивов.

**Формальное описание задачи.**

Мой код программы можно разбить на 4 функции, а именно:  
 1) Функция, которая отвечает за ввод и проверку значения кол-во элементов

2) Функция, которая отвечает за заполнение массива

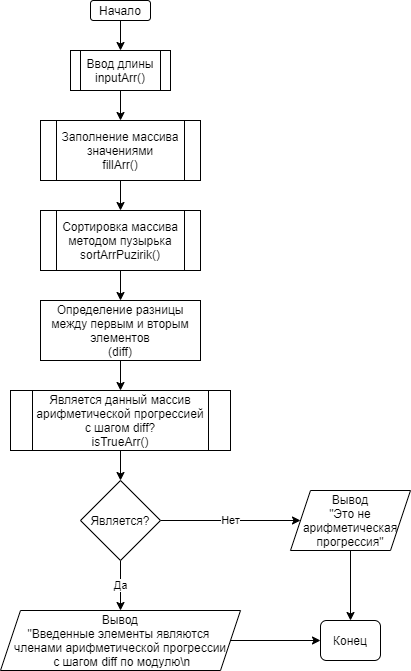
3) Функция, которая отвечает за сортировку массива (Например методом пузырька)

4) Функция, которая отвечает за проверку массива на соответствие арифметической прогрессии с шагом d

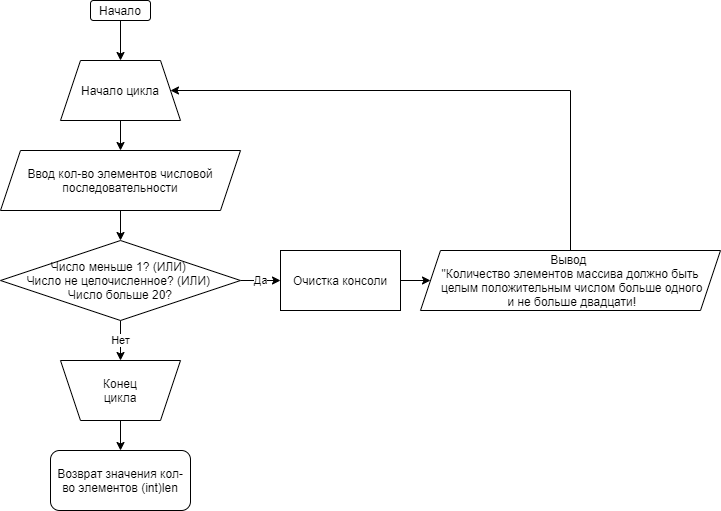
**Алгоритм решения:**

1. Создать 4 функции
2. Используя эти 4 функции создать целостную, работающую программу

**Блок схема программы**



**Блок схема функции inputArr()**



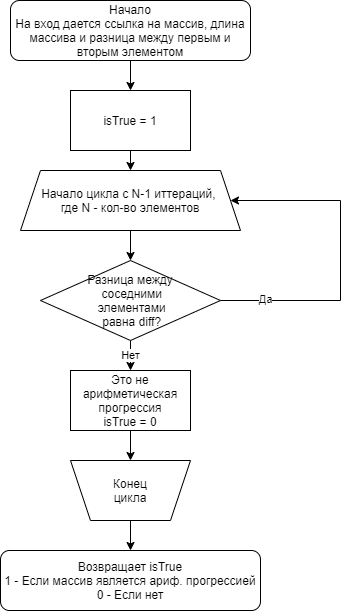
**Блок схема функции fillArr(double \*numbers, int len)**



**Блок схема функции inputArr(double \*numbers, int len)**



**Блок схема функции inputArr(double \*numbers, int len, double diff)**



**Листинг программы.**

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <math.h>
5. //Ввод значений
6. **int** inputArr();
7. //Заполнение массива с помощью ввода значений
8. **void** fillArr(**double** \*numbers, **int** len);
9. //Сортировка массива с помощью сортировки пузырьком
10. **void** sortArrPuzirik(**double** \*numbers, **int** len);
11. // Является ли данный массив арифметической прогрессией с шагом diff
12. **int** isTrueArr(**double** \*numbers, **int** len, **double** diff);
14. main() {
15. **int** isTrue = 1, len;
16. **double** numbers[20], diff;
17. system("chcp 1251 > nul");
19. len = inputArr();
21. printf("Введите значения массива: ");
22. fillArr(numbers, len);
23. sortArrPuzirik(numbers, len);
24. diff = numbers[1] - numbers[0];
25. isTrue = isTrueArr(numbers, len, diff);
27. **if**(isTrue) {
28. printf("Введенные элементы являются членами арифметической прогрессии с шагом %lg по модулю\n", diff);
29. } **else** {
30. printf("Это не арифметическая прогрессия\n");
31. }
33. system("pause");
34. **return** 0;
35. }
37. **int** inputArr() {
38. **double** len;
39. **while**(1) {
40. printf("Введите размер массива: ");
41. scanf("%lg", &len);
42. **if** (len <= 1 || len > 20 || round(len) != len) {
43. system("cls");
44. printf("Количество элементов массива должно быть целым положительным числом больше одного и не больше двадцати!\n");
45. **continue**;
46. }
47. **break**;
48. }
49. **return** (**int**)len;
50. }
52. **void** fillArr(**double** \*numbers, **int** len) {
53. **int** i;
54. **for**(i = 0; i < len; i++) {
55. scanf("%lg", &numbers[i]);
56. }
57. }
59. **void** sortArrPuzirik(**double** \*numbers, **int** len) {
60. **int** i, j;
61. **double** diff;
62. // сортировка методом пузырька
63. **for**(i = 0; i < len; i++) {
64. **for**(j = 0; j < len - i - 1; j++) {
65. **if** (numbers[j] > numbers[j + 1]) {
66. diff = numbers[j + 1];
67. numbers[j+1] = numbers[j];
68. numbers[j] = diff;
69. }
70. }
71. }
72. }
74. **int** isTrueArr(**double** \*numbers, **int** len, **double** diff) {
75. **int** isTrue = 1, i;
76. // Определение разницы между двумя соседними числами. В арифметической прогрессии
77. // эта разница будет одинакова при любых разностях соседних чисел ариф. прогрес.
78. **for**(i = 1; i < len - 1; i++) {
79. **if** (fabs((numbers[i + 1] - numbers[i]) - diff) > 0.00001) { // Так как нельзя использовать == с double числами, я применил альтернитиву
80. isTrue = 0; //isTrue - истина, если это ариф. прогрес., ложь, если нет
81. **break**;
82. }
83. }
84. **return** isTrue;
85. }

**Пояснения к программе.**

Названия функций и их назначение:

**int inputArr() –** Функция просит пользователя ввести значение – кол-во элементов в массиве. Также функция проверяет введенное число на соответствие условиям задачи (Число больше нуля, меньше или равно 20ти, число целочисленное). Если число не проходит проверку – функция очищает консоль, выводит ошибку и вновь просит ввести значение. Возвращает число – кол-во элементов. Выбран тип int потому что кол-во элементов это целочисленное число.

**void fillArr(double \*numbers, int len)** – Функция просит пользователя ввести элементы. Заполняет массив введенными элементами. На вход принимает ссылку на массив, который заполняет, и длину такового.

**void sortArrPuzirik(double \*number, int len)** – Функция сортирует массив в порядке возрастания с помощью метода Пузырька. На вход принимает ссылку на массив, который сортирует, и длину такового. Ничего не возвращает.

**int isTrueArr(double \*numbers, int len, double diff) –** Функция проверяет полученный массив на соответствии такового определению арифметической прогрессии с шагом **diff.** На вход принимает ссылку на массив, которой будет проверен, длину такового и размер шага diff. Возвращает число 1 или 0. 1 – если массив соответствует арифметической прогрессии, 0 – если нет.

Для организации дружественного интерфейса пользователя перед вводом данных выводится сообщение для пользователя. Ввод и вывод организован при помощи стандартных функций scanf() и printf(), для этого подключен заголовочный файл stdio.h.

Заголовочный файл stdlib.h необходим для использования функции system(), чтобы приостановить выполнение программы перед завершением (выполняется команда «pause» операционной системы).

Заголовочный файл math.h необходим для функции fabs.

**Выводы:**

Я провёл рефакторинг программы, разбив старый код на функции. Это определенно улучшило читаемость кода. Также это скорее всего затронуло оптимизацию, потому что все временные переменный в функциях благополучно было удалено после их использования, что не происходит в случае отсутствия функций. Я научился передавать функциям ссылки на массив. Я научился делать обособленные функции.