Бюджетное учреждение высшего образования

Ханты-Мансийского автономного округа

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 4 «Одномерные массивы»

по дисциплине «Программирование и основы алгоритмизации»

Выполнил: Скорба Р.В.

студент группы 609-11

Проверил: Гришмановский П. В.

доцент кафедры автоматики и

компьютерных систем

Сургут

2022 г.

**Цель работы:**

Целью работы является закрепление теоретических знаний об одномерных массивах, приобретение практических навыков использования массивов, а также закрепление знаний и навыков использования условных и циклических операторов.

**Задание:**

Решить задачу в соответствии с индивидуальным заданием, выданным преподавателем, с использованием в реализации алгоритма одномерных массивов.

**Вариант №23:**

Задан размер массива действительных чисел и значения его элементов. Заменить недостоверные значения результатом линейной интерполяции. Недостоверными считать значения, не входящие в интервал [0,5; 1,5], где – среднее арифметическое значение всех элементов массива.

**Вариант 8.**

**Формальное описание задачи.**

Вначале пользователь вводит размер массива, в условии не указан максимальный размер массива, но в «практикуме» сказано, что если в условии не оговорен максимальный размер массива, значит, он может быть равен максимум 20.

После чего запускается цикл for от 0 до size (размер указанный пользователем), и пользователь вводит поочередно значения каждого элемента массива. После программа считает среднеарифметическое значение всех элементов массива по формуле: , где m – сумма всех элементов массива.

Программа проверяет первое и последнее значение массива, если оно не находится в интервале [0,5; 1,5], то присваивает этому элементу значение .

Потом программа начинает по порядку проверять находится ли каждый элемент массива в интервале [0,5; 1,5], если нет, то оно берет среднее значение между предыдущим элементом массива и следующим элементом который находится в интервале [0,5; 1,5]. В случае когда следующий элемент массива не находится в нужном интервале, оно проверяет следующий элемент массива, если он подходит то методом линейной интерполяции присваивается элементом не входящим в интервал соответствующие значения. Для этого нам нужно найти число шага чтобы правильно заменить элементы массива.

- разность между элементом недостоверным и первым достоверным, где - недостоверный элемент, – первый достоверный элемент.

- кол-во недостоверных чисел, где - номер недостоверного элемента, - номер первого достоверного элемента.

- число шага.

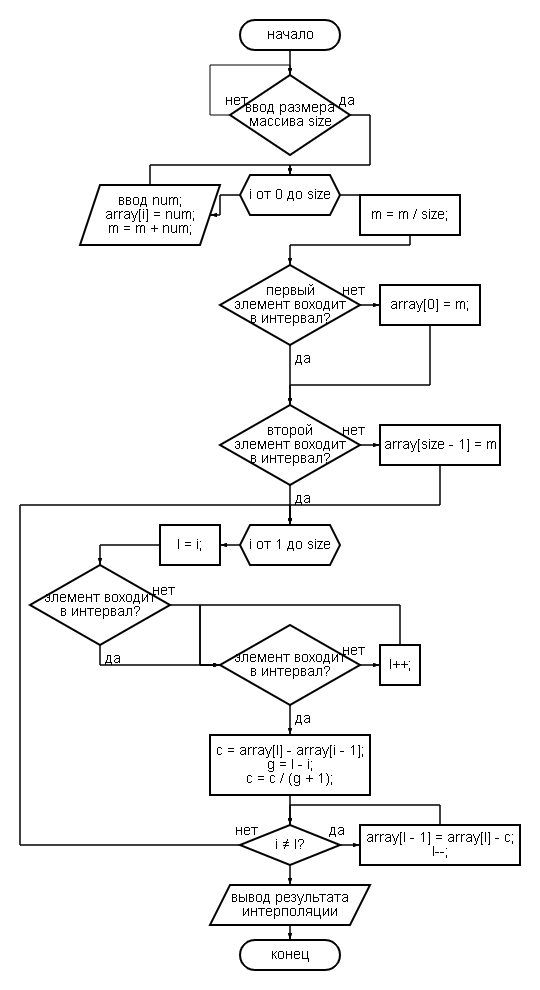
После чего цикл while от первого достоверного до недостоверного элемента меняет значение элемента массива, используя данную формулу:

В конце программа выводит результат линейной интерполяции.

**Алгоритм программы.**

1. Пользователь вводит размер массива.
2. Пользователь по очереди вводит значение каждого элемента массива.
3. Считается среднеарифметическое всех элементов массива.
4. Если первый или последний элемент массива не входит в указанный интервал, то происходит замена на среднеарифметическое значение всех элементов массива.
5. Идет проверка каждого элемента массива на то, входит ли он в интервал, если нет, то считается число шага и методом линейной интерполяции происходит замена недостоверных элементов массива.
6. Вывод результата линейной интерполяции.

**Блок-схема алгоритма программы.**



**Листинг программы.**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main () {

system ("chcp 1251 > null");

int size, i, I, count;

double num, array[20]={0}, m, c, g;

while (printf("Введите размер массива: "), fflush(stdin), scanf("%d", &size), size > 20 || size < 0)

printf("Wrong input!\a\nЗначение массива может быть от 1 до 20\n");

printf("Введите значения элементов массива:\n");

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

printf("Array[%d] = ", i+1);

scanf("%lf", &num);

array[i] = num;

m = m + num;

}

m = m / size; // m - Среднее арифметическое всех элементов массива.

if (array[0] > (1.5 \* m) || array[0] < (0.5 \* m)){

array[0] = m;

}

if (array[size - 1] > (1.5 \* m) || array[size - 1] < (0.5 \* m)){

array[size - 1] = m;

}

for (i = 1; i < size; i++) {

I = i;

if (array[i] > (1.5 \* m) || array[i] < (0.5 \* m)) {

while (array[I] > (1.5 \* m) || array[I] < (0.5 \* m)) {

I++;

}

c = array[I] - array[i - 1]; //Разность между элементом недостоверным и первым достоверным

g = I - i; //Кол-во недостоверных чисел

c = c / (g + 1); //Число шага

while (i != I) {

array[I - 1] = array[I] - c;

I--;

}

}

}

printf("\n\nРезультаты линейной интерполяции:\n");

for (count = 0; count < size; count++){

printf("Array[%d] = %.3lg\n", count+1, array[count]);

}

system("pause");

return 0;

}

**Пояснения к программе.**

Подключены 3 библиотеки: <stdio.h> - для функций printf, scanf, fflush; <stdlib.h> - для использования функции system(); <math.h> - для математических вычислений.

Для того чтобы пользователь понимал что от него требуется, используется ‘printf();’, в котором описано условие. В программе используется тип int для счетчиков, номера элемента массива и размера массива, так как все величины после всех вычислений должны быть целочисленные. Тип double используется для одномерного массива, написания элементов массива и остальных операций, так как все величины могут быть не целыми числами, а тип double используется по умолчанию для вещественных вычислений, имеет достаточную точность и используется стандартными математическими функциями.

В самом начале ввод пользователя осуществляется через цикл while, он осуществляет проверку на правильность ввода, чтобы пользователь не мог ввести значения, которые не соответствуют условию. Когда пользователь вводит значение, которое вводить нельзя, выходит сообщение об ошибке и цикл повторяется до тех пор, пока значение не будет соответствовать требованиям.

Для ввода всех элементов массива используется цикл for, с помощью которого пользователь поочередно вводит значение для всех элементов массива.

Оператор выбора if используется для определения находится ли элемент в заданном интервале.

**Вывод:** Благодаря тому, что все проверки которые возможно поместить в один цикл находятся в одном цикле, я уменьшил количество итераций до минимума.