ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

имени В. И. ВЕРНАДСКОГО»

ТАВРИЧЕСКАЯ АКАДЕМИЯ

Факультет математики и информатики

ОТЧЕТ

об учебной практике

по направлению подготовки 01.03.02 Прикладная математика и информатика

образовательного уровня бакалавр

студента _2_ курса _____

(*Фамилия, имя, отчество*)

Симферополь, 2019 г.

Содержание

Общая информация

В соответствии с учебным планом подготовки бакалавров по направлению 01.03.02 Прикладная математика и информатика учебная практика проходила с 9 июля по 16 июля 2019 года в Центре компьютерных технологий Таврической академии (ТА КФУ) и Лаборатории программного обеспечения компьютерных систем факультета математики и информатики.

Целью учебной практики является закрепление теоретических знаний по программированию, приобретение профессиональных навыков и компетенций в области современных информационных технологий, получение опыта профессиональной деятельности в производственном коллективе.

Основные задачи учебной практики:

- выполнение индивидуального задания на разработку программы в среде C++.

Индивидуальное задание по программированию

Постановка задачи

Среднестатистическим назовём элемент массива, если для него модуль разности его значения и среднего арифметического элементов массива достигает минимума. Аналогично, *уникальным* будем называть элемент, для которого такой модуль разности достигает максимума. В заданном массиве $X(m)$ найти номера (индексы) среднестатистического и уникального элементов.

Описание алгоритма

Для решения задачи создаются следующие функции:

1. generateArray генерирует массив (динамический массив реализован классом vector из STL).
2. getAverage вычисляет среднее арифметическое массива.
3. getAverageIndex находит индекс среднестатистического элемента массива.
4. getUniqueIndex находит индекс уникального элемента массива.
5. printArray выводит сгенерированный массив на экран.

В основной программе выполняются следующие действия:

1. Создание генератора случайных чисел и инициализация начала последовательности случайных чисел текущим временем:

```
  mt19937
gen(chrono::steady_clock::now().time_since_epoch().count());
uniform_int_distribution<int> uid(1, 2e6);
```

2. Генерация динамического массива, реализованного классом vector из STL — generateArray.
3. Вывод сгенерированного массива на экран — printArray.
4. Вычисление среднего арифметического массива — getAverage.
5. Поиск индекса среднестатистического элемента массива — getAverageIndex.
6. Поиск индекса уникального элемента массива — getUniqueIndex.
7. Вывод индексов среднестатистического и уникального элементов массива.

Описание интерфейса пользователя программы

Ввод данных не требуется. Данные генерируются датчиком случайных чисел.

Программа

```
#include <iostream>
#include <vector>
#include <ctime>
#include <cmath>
#include <random>
#include <chrono>

using namespace std;

//eps - погрешность вычислений при работе с действительными числами
const double eps = 1e-9;

//Создание генератора случайных чисел и инициализация начала
последовательности случайных чисел текущим временем
```

```
mt19937 gen(chrono::steady_clock::now().time_since_epoch().count());
uniform_int_distribution<int> uid(1, 2e6);

//Функция generateArray генерирует массив (динамический массив
реализован классом vector из STL)
vector<int> generateArray() {
	int m = uid(gen); // Генерация размера массива
	vector<int> a(m);
	for (int i = 0; i < m; i++) a[i] = uid(gen) - 1e6;
	return a;
}

// Функция getAverage вычисляет среднее арифметическое массива
double getAverage(vector<int>& a) {
	int sum = 0;
	for (int i = 0; i < a.size(); i++) sum += a[i];
	return sum / (double)a.size();
}

// Функция getAverageIndex находит индекс среднестатистического
элемента массива
int getAverageIndex(vector<int>& a, double average) {
	double minAbs = abs(a[0] - average);
	int averageIndex = 0;
	for (int i = 1; i < a.size(); i++) {
		double currentAbs = abs(a[i] - average);
		if (currentAbs + eps < minAbs) {
			minAbs = currentAbs;
			averageIndex = i;
		}
	}
	return averageIndex;
}

// Функция getUniqueIndex находит индекс уникального элемента массива
int getUniqueIndex(vector<int>& a, double average) {
	double maxAbs = abs(a[0] - average);
	int uniqueIndex = 0;
	for (int i = 1; i < a.size(); i++) {
		double currentAbs = abs(a[i] - average);
		if (currentAbs - eps > maxAbs) {
			maxAbs = currentAbs;
			uniqueIndex = i;
		}
	}
	return uniqueIndex;
}

// Функция printArray выводит сгенерированный массив на экран
void printArray(vector<int>& a) {
	cout << "Размер массива: " << a.size() << endl;
	for (int i = 0; i < a.size(); i++) cout << a[i] << " ";
	cout << endl;
}

int main() {
```

```
	vector<int> X = generateArray();

	// Поддержка кириллицы
	setlocale(LC_ALL, "ru_RU");

	//printArray(X); Использовалась при тестировании
	double average = getAverage(X);
	int AverageIndex = getAverageIndex(X, average);
	int UniqueIndex = getUniqueIndex(X, average);

	// Вывод индексов среднестатистического и уникального элементов массива
	cout << "Индекс среднестатистического элемента: " << AverageIndex << endl;
	cout << "Индекс уникального элемента: " << UniqueIndex << endl;

	return 0;
}
```

Тестирование

Для визуализации результатов работы программы использовалась функция printArray, осуществляющая вывод элементов массива на экран.

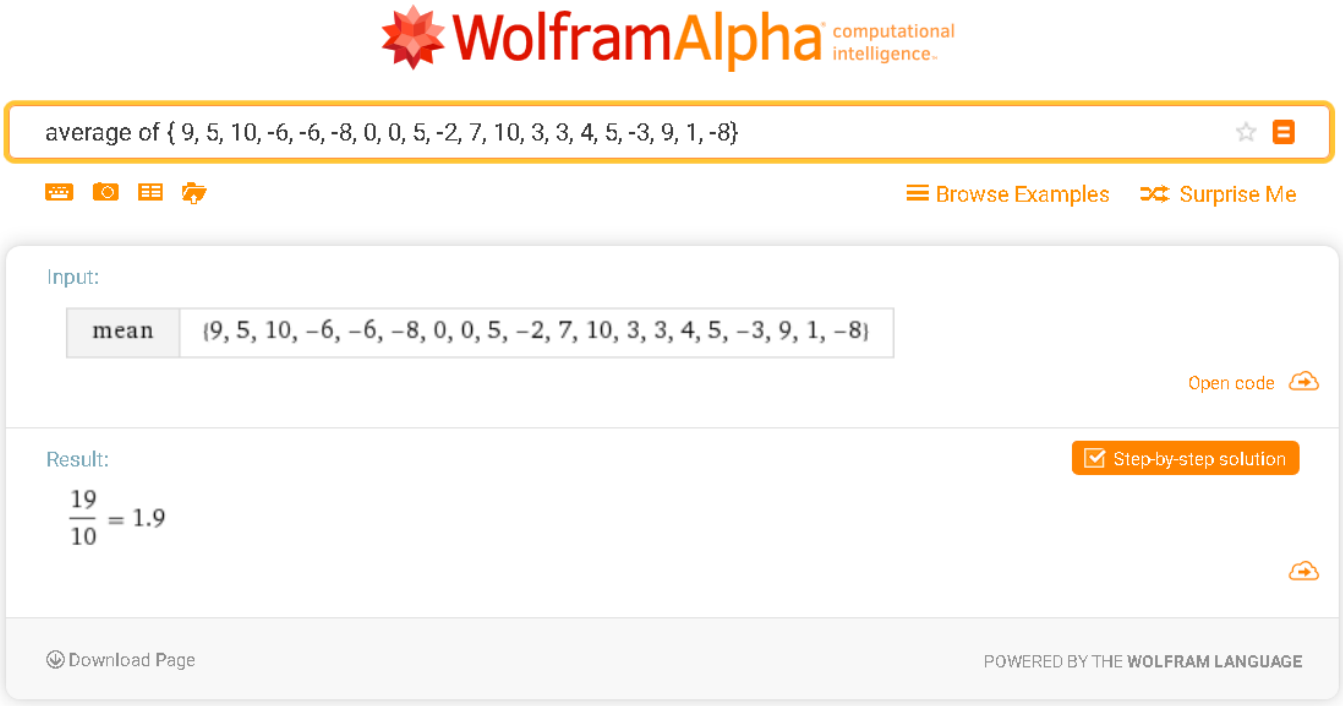
```
Размер массива: 20
9 5 10 -6 -6 -8 0 0 5 -2 7 10 3 3 4 5 -3 9 1 -8
Индекс среднестатистического элемента: 18
Индекс уникального элемента: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Анализ результатов

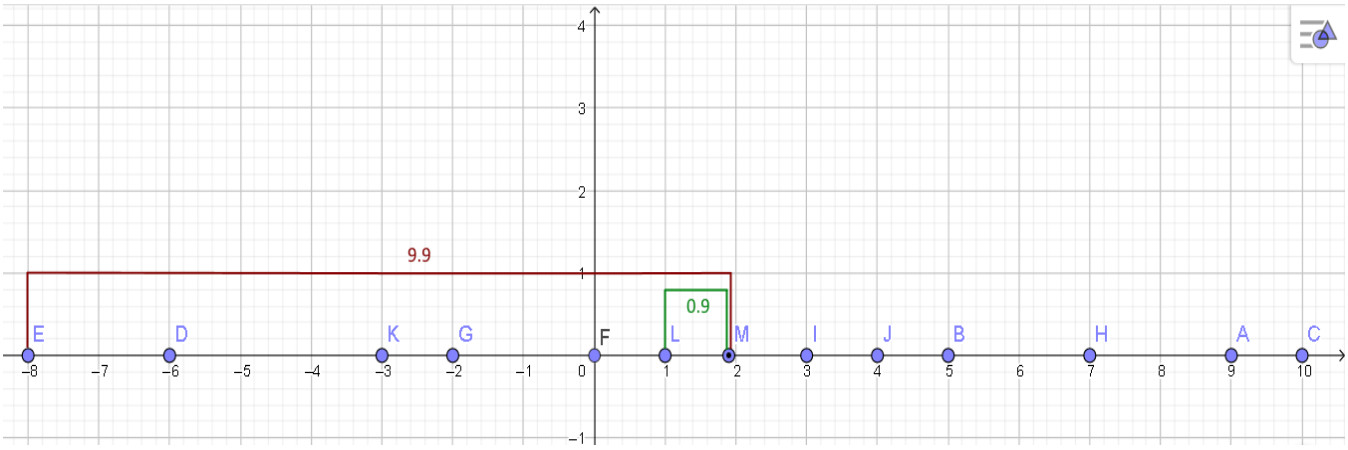
Представим сгенерированный массив X в виде таблицы:

9	**5**	**10**	**-6**	**-6**	**-8**	**0**	**0**	**5**	**-2**	**7**	**10**	**3**	**3**	**4**	**5**	**-3**	**9**	**1**	**-8**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*

Найдем среднее арифметическое элементов массива:



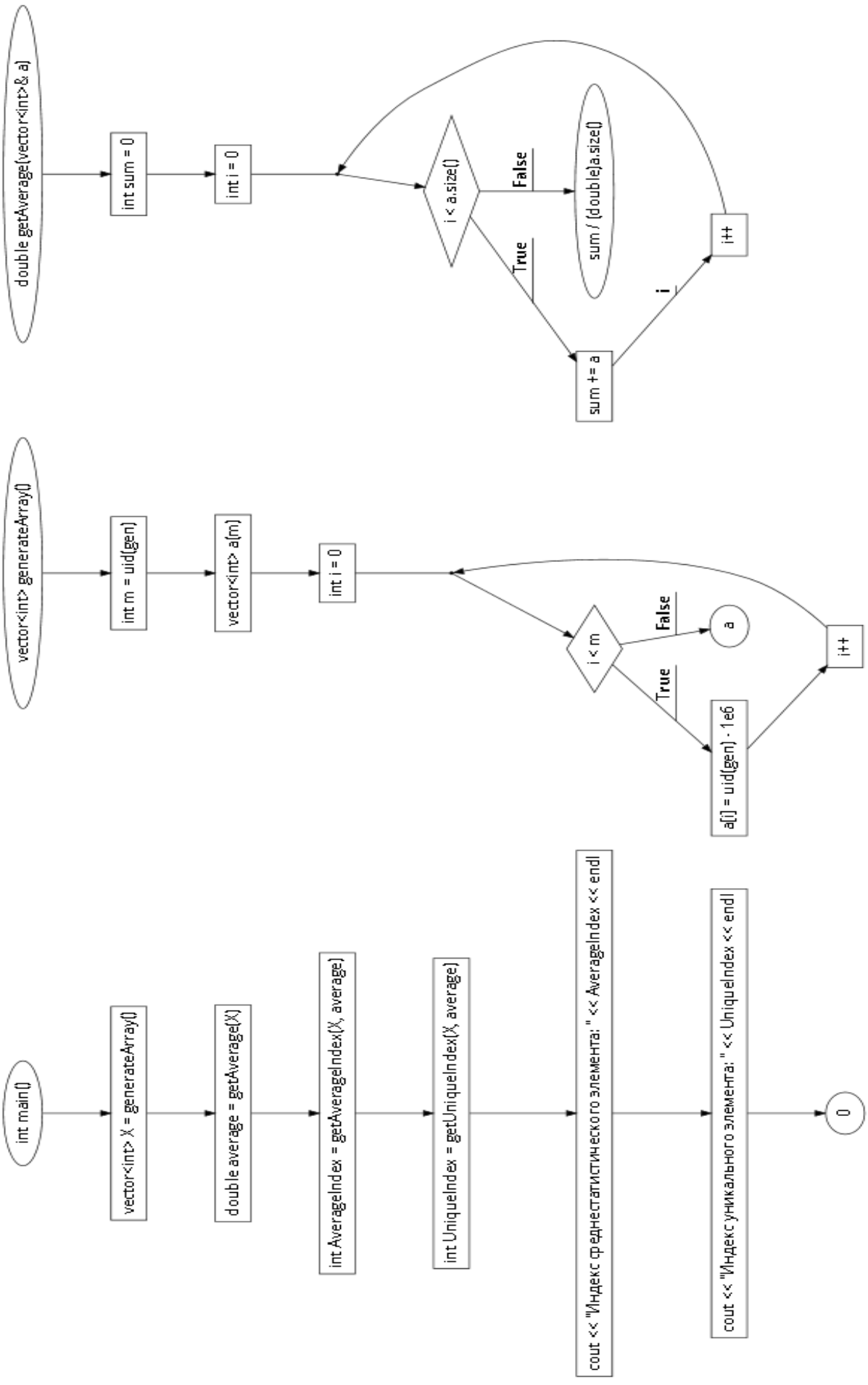
Найти модуль разности двух чисел означает найти расстояние между точками, координатами которых являются эти числа, на числовой оси. Отметим на оси абсцисс значения массива X и среднее арифметическое элементов массива, соответствующее точке M(1.9; 0):

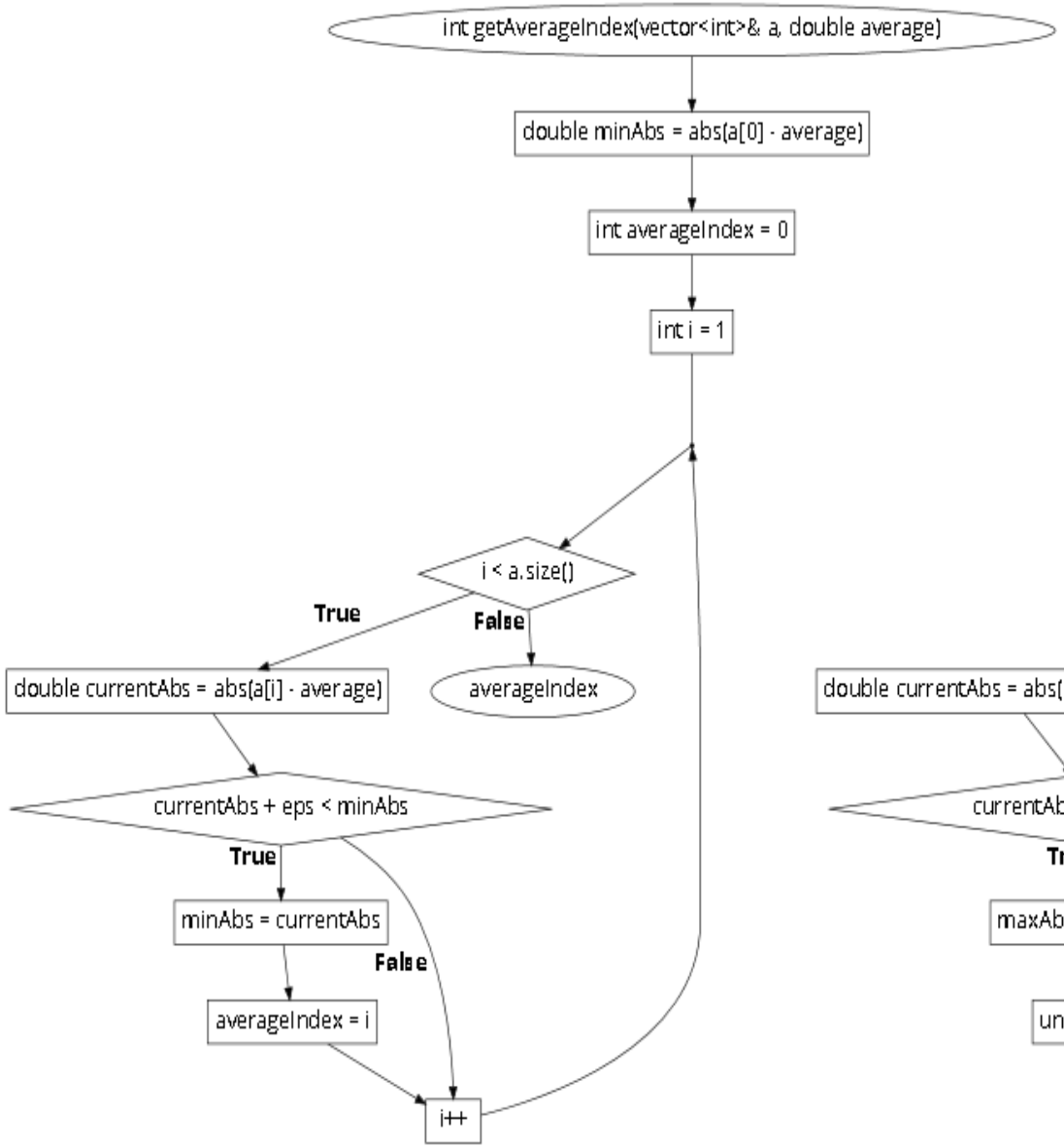


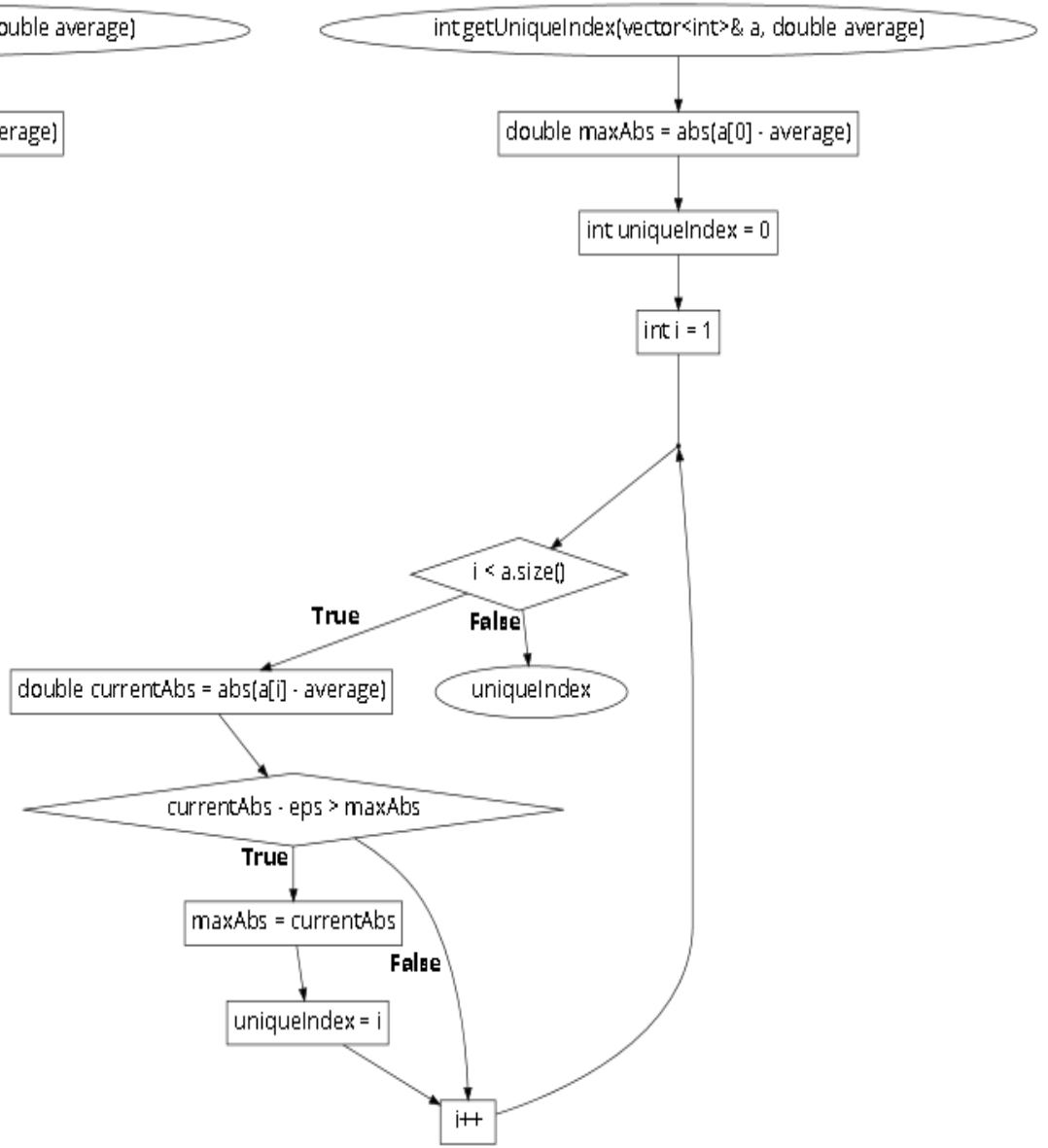
Для нахождения среднестатистического значения массива необходимо найти ближайшую точку к M из отмеченных на оси. На графике видно, что это L(1; 0). Элемент со значением 1 в массиве X единственный и находится в ячейке с индексом 18 (см. табл.).

Для нахождения уникального элемента массива необходимо найти точку, расстояние от которой до M максимально. На графике видно, что это E(-8; 0). В массиве X два элемента со значением -8. Данная программа находит элемент с наименьшим индексом, т. е. 5.

Блок-схема


int main()
vector<int> X = generateArray()
double average = getAverage(X)
int AverageIndex = getAverageIndex(X, average)
int UniqueIndex = getUniqueIndex(X, average)
cout << "Индекс среднестатистического элемента: " << AverageIndex << endl
cout << "Индекс уникального элемента: " << UniqueIndex << endl
0
vector<int> generateArray()
int m = uid(gen)
vector<int> a(m)
int i = 0
i < m
True
False
a[i] = uid(gen) - 1e6
i++
a
double getAverage(vector<int>& a)
int sum = 0
int i = 0
i < a.size()
True
False
sum += a
i
sum / (double)a.size()
i++


int getAverageIndex(vector<int>& a, double average)
double minAbs = abs(a[0] - average)
int averageIndex = 0
int i = 1
i < a.size()
True
False
double currentAbs = abs(a[i] - average)
averageIndex
currentAbs + eps < minAbs
True
minAbs = currentAbs
False
averageIndex = i
i++


int getUniqueIndex(vector<int>& a, double average)
double maxAbs = abs(a[0] - average)
int uniqueIndex = 0
int i = 1
i < a.size()
True
False
double currentAbs = abs(a[i] - average)
uniqueIndex
currentAbs - eps > maxAbs
True
maxAbs = currentAbs
False
uniqueIndex = i
i++

Информационное обеспечение

Интернет-ресурсы:

1. https://en.cppreference.com/w/cpp/numeric/random/rand
2. https://codeforces.com/blog/entry/61587
3. http://code2flow.com/
4. https://www.wolframalpha.com/
5. https://www.geogebra.org/classic
6. https://ru.cppreference.com/w/cpp/container/vector