

Национальный исследовательский университет  
«Высшая школа экономики»  
**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2-3**

**«Анализ данных сенсора»**

**Выполнил:**  
Логвиненко Алексей Максимович

# Содержание

<b>1</b>	<b>Введение</b>	<b>2</b>
1.1	Цель проекта . . . . .	2
1.2	Задачи . . . . .	2
<b>2</b>	<b>Описание программы</b>	<b>2</b>
2.1	Структура данных . . . . .	2
2.2	Основные функции . . . . .	2
2.2.1	Ввод данных . . . . .	2
2.2.2	Вычисление среднего значения . . . . .	2
2.2.3	Нахождение минимального значения . . . . .	3
2.2.4	Нахождение максимального значения . . . . .	3
2.2.5	Фильтрация значений выше среднего . . . . .	3
<b>3</b>	<b>Алгоритмы и методы</b>	<b>4</b>
3.1	Обработка ввода пользователя . . . . .	4
3.2	Работа с динамической памятью . . . . .	4
3.3	Форматирование вывода . . . . .	4
<b>4</b>	<b>Инструкция по использованию</b>	<b>4</b>
4.1	Компиляция и запуск . . . . .	4
4.2	Меню программы . . . . .	4
<b>5</b>	<b>Пример работы</b>	<b>4</b>
5.1	Ввод данных . . . . .	4
5.2	Вывод статистики . . . . .	5
<b>6</b>	<b>Технические детали</b>	<b>5</b>
6.1	Ограничения . . . . .	5
6.2	Особенности реализации . . . . .	5
<b>7</b>	<b>Заключение</b>	<b>5</b>
<b>8</b>	<b>Приложения</b>	<b>5</b>
8.1	Приложение А. Полный код программы . . . . .	5

# 1 Введение

## 1.1 Цель проекта

Разработка консольного приложения на C++ для анализа данных с температурных датчиков с функциями статистического анализа и фильтрации данных.

## 1.2 Задачи

- Реализовать работу с динамическими массивами
- Освоить передачу массивов по указателю в функции
- Научиться обрабатывать массивы через арифметику указателей
- Реализовать функции статистического анализа данных
- Обеспечить корректное управление памятью

# 2 Описание программы

## 2.1 Структура данных

Для хранения данных измерений используется динамический массив типа float:

```
1 float* data = new float[n]; //  
2 float* filteredData; //
```

## 2.2 Основные функции

### 2.2.1 Ввод данных

```
1 void inputData(float* p, int n) {  
2     cout << "                               : ";  
3     for (int i = 0; i < n; i++) {  
4         cin >> *(p + i);  
5     }  
6 }
```

### 2.2.2 Вычисление среднего значения

```
1 float average(const float* p, int n) {  
2     float sum = 0.0;  
3     for (int i = 0; i < n; i++) {  
4         sum += *(p + i);  
5     }  
6     return sum / n;  
7 }
```

### 2.2.3 Нахождение минимального значения

```
1 float minValue(const float* p, int n) {
2     float minVal = *(p);
3     for (int i = 1; i < n; i++) {
4         if (*(p + i) < minVal) {
5             minVal = *(p + i);
6         }
7     }
8     return minVal;
9 }
```

### 2.2.4 Нахождение максимального значения

```
1 float maxValue(const float* p, int n) {
2     float maxVal = *(p);
3     for (int i = 1; i < n; i++) {
4         if (*(p + i) > maxVal) {
5             maxVal = *(p + i);
6         }
7     }
8     return maxVal;
9 }
```

### 2.2.5 Фильтрация значений выше среднего

```
1 float* filterAboveAverage(float* p, int n, int& newCount) {
2     float avg = average(p, n);
3
4     //
5
6     newCount = 0;
7     for (int i = 0; i < n; i++) {
8         if (*(p + i) > avg) {
9             newCount++;
10        }
11    }
12
13    //
14
15    float* filteredArray = new float[newCount];
16
17    //
18    int index = 0;
19    for (int i = 0; i < n; i++) {
20        if (*(p + i) > avg) {
21            *(filteredArray + index) = *(p + i);
22            index++;
23        }
24    }
25
26    return filteredArray;
27 }
```

## 3 Алгоритмы и методы

### 3.1 Обработка ввода пользователя

- Проверка корректности количества измерений
- Использование арифметики указателей для доступа к элементам массива
- Защита от некорректного ввода данных

### 3.2 Работа с динамической памятью

- Выделение памяти оператором `new[]`
- Освобождение памяти оператором `delete[]`
- Создание временных массивов для фильтрации данных

### 3.3 Форматирование вывода

- Использование `setprecision()` для форматирования чисел
- Вывод результатов в читаемом формате
- Отображение отфильтрованных данных

## 4 Инструкция по использованию

### 4.1 Компиляция и запуск

```
g++ Sensor_data_analysis.cpp -o sensor_analysis
```

### 4.2 Меню программы

Программа работает в интерактивном режиме:

1. Ввод количества измерений
2. Ввод значений измерений
3. Автоматический расчет статистики
4. Вывод результатов анализа

## 5 Пример работы

### 5.1 Ввод данных

```
=== Анализ сенсора ===
```

```
Введите количество измерений: 6
```

```
Введите значения: 10 15 8 -2 18 20
```

## 5.2 Вывод статистики

Среднее значение: 11.5

Минимум: -2.0

Максимум: 20.0

Значения выше среднего:

15.0 18.0 20.0

## 6 Технические детали

### 6.1 Ограничения

- Количество измерений ограничено доступной памятью
- Значения температур предполагаются в диапазоне от -50 до +50°C
- Используется тип данных `float` для хранения значений

### 6.2 Особенности реализации

- Использование исключительно арифметики указателей
- Ручное управление памятью
- Модульная структура программы

## 7 Заключение

В ходе выполнения лабораторной работы было разработано консольное приложение на C++ для анализа данных сенсора. Программа демонстрирует:

- Работу с динамическими массивами и указателями
- Реализацию статистических функций
- Фильтрацию данных по заданному критерию
- Корректное управление памятью
- Создание удобного пользовательского интерфейса

Программа успешно решает поставленные задачи и может быть расширена дополнительным функционалом для более сложного анализа данных.

## 8 Приложения

### 8.1 Приложение А. Полный код программы

Полный код программы доступен в файле: `Sensor_data_analysis.cpp`