Класс Statistic – для заданного набора целых чисел считает гистограмму, среднее значение и среднее квадратичное отклонение (СКО). Есть возможность фильтрации значений (сигма = СКО):

* 1 сигма (~68,26% значений)
* 2 сигмы (~95,45% значений)
* 3 сигмы (~99,73% значений)

Интерфейс

Конструктор класса принимает три значения:

* <qint32> confidence\_interval– количество сигм, в которые уложатся отфильтрованные результаты; целое число [1..3]
* <qint32>num\_of\_interval – количество интервалов для построения гистограммы; целое число >1
* QList<qint32> \*input – указатель на массив входных данных. Указатель!=0 и массив не пуст

Для работы с классом предусмотрено 7 public методов и 1 public переменная:

* float average() – возвращает среднее значение исходного(неотфильтрованного) массива
* float filteredAverage() - возвращает среднее значение отфильтрованного массива
* float std\_deviation() – возвращает СКО исходного массива
* float filtered\_std\_deviation() – возвращает СКО отфильтрованного массива
* void make\_histogram() – рассчитывает гистограмму всего диапазона значений
* void filtered\_histogram() – рассчитывает гистограмму отфильтрованного диапазона значений
* float getBinWidth() – возвращает ширину интервала для текущей гистограммы. По умолчанию =0
* std::map<qint32,qint32> \*histogram – указатель на структуру, хранящую гистограмму. Гистограмма хранится в виде пар <key,value>, где key – порядковый номер интервала(начиная с 1), value – кол-во чисел, попавших в данный интервал. По умолчанию структура пуста

Комментарии к реализации:

* set- и get- методы отсутствуют для упрощения интерфейса
* расчет статистических параметров и гистограммы происходит в момент вызова соответствующей функции. Это позволяет менять исходные данные в ходе работы и рассчитывать новые значения для измененного диапазона
* контроль допустимости аргументов в данной версии производится с помощью assert()
* умные указатели не использовались в данной версии для упрощения реализации
* было реализовано 2 версии класса: в первой было минимизировано количество циклов и вызовов функций для повышения скорости работы, во второй акцент был сделан на читаемости кода и вынесении дублирующихся вычислений в функции. Результаты измерения скорости работы показали, что вторая функция работает на 5% медленнее первой. Был выбран второй вариант, т. к. его значительно проще использовать и поддерживать, а снижение скорости работы не является значимым. Измерения проводились для 775304 входящих значений, по 100 раз для каждого случая
* Итоговая скорость работы класса в мс (775304 значений) приведена в таблице, где № строки – кол-во сигм при расчете, столбец – тестируемая функция:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | гистограмма | ср.знач | СКО | фильтр. гист | фильтр. ср. знач | фильтр. СКО |
| 1 | 109.4 | 5.4 | 13.8 | 97.52 | 22.96 | 23.04 |
| 2 | 108.8 | 6.16 | 13.36 | 116.66 | 18.3 | 18.38 |
| 3 | 119.02 | 5.8 | 12.92 | 128.4 | 21.7 | 21.4 |