Лабораторная работа №2.

Создать класс для хранения набора чисел с дополнительной возможностью получения статистики:

- Количество чисел в наборе.
- Максимальное, минимальное число в наборе.
- Среднее арифметическое набора.
- Количество чисел, которые больше заданного числа.
- Количество чисел, которые меньше заданного числа.

Задание следует выполнять пошагово. Но это необязательно. Ниже приведены шаги выполнения в которых содержаться **примеры** интерфейса класса. Вы можете предложить и другие решения.

Шаг 1. Реализовать класс, имеющий примерно такой интерфейс.

```
class StatisticMultiset {
public:
    // Добавляет число в набор.
    void  AddNum( int num );

    // Максимальное число в наборе.
    int   GetMax();

    // Минимальное число в наборе.
    int   GetMin();

    // Среднее арифметическое всего набора.
    float GetAvg();

    // Кол-во чисел в наборе меньше заданного порога.
    int GetCountUnder( float threshold );

    // Кол-во чисел в наборе больше заданного порога.
    int GetCountAbove( float threshold );
};
```

Документация:

- std::multiset
  - http://www.cplusplus.com/reference/set/multiset/
  - http://www.cplusplus.com/reference/set/multiset/insert/
  - http://www.cplusplus.com/reference/set/multiset/size/
- std::vector
- Любой другой контейнер стандартной библиотеки, который кажется вам удобным для решения.
- **Загляните обязательно:** http://en.cppreference.com/w/cpp/types/numeric_limits

Шаг 2. Сделать соответствующие функции класса константными.

```
class StatisticMultiset {
public:
    StatisticMultiset();
    ~StatisticMultiset();
    void  AddNum( int num );
    int   GetMax() const;
    int   GetMin() const;
    float GetAvg() const;
    int GetCountUnder( float threshold ) const;
    int GetCountAbove( float threshold ) const;
};
```

Документация:

- https://msdn.microsoft.com/en-us/library/07x6b05d.aspx (раздел const member functions)

Шаг 3. Добавить возможность загрузки данных из файла, из vector/multiset и т.п. Выбирайте сами для своего удобства. Формат файла также на ваше усмотрение.
Ответить на вопрос, зачем в этих функциях используется константная ссылка: **`const`** `std::vector<int>&` Почему `const`? Почему ссылка? Почему не просто `AddNum(std::vector<int>)` ?

```
class StatisticMultiset {
public:
    StatisticMultiset();
    ~StatisticMultiset();

    void  AddNum( int num );
    void  AddNum( const std::multiset<int>& numbers );
    void  AddNum( const std::vector<int>& numbers );
    void  AddNum( const std::list<int>& numbers );
    void  AddNumsFromFile( const char* filename );

    int   GetMax() const;
    int   GetMin() const;
    float GetAvg() const;
    int GetCountUnder( float threshold ) const;
    int GetCountAbove( float threshold ) const;
};
```

Шаг 4. Добавить возможность добавлять в набор данные из другого набора. Обратите внимание, что в функции `AddNums(const StatisticMultiset& a_stat_set)` возможен доступ к закрытым (private) полям экземпляра `a_stat_set`.

```
class StatisticMultiset {
public:
    StatisticMultiset();
    ~StatisticMultiset();

    void  AddNum( int num );
    void  AddNum( const std::multiset<int>& numbers );
    void  AddNum( const std::vector<int>& numbers );
    void  AddNum( const std::list<int>& numbers );
    void  AddNums( const StatisticMultiset& a_stat_set );
    void  AddNumsFromFile( const char* filename );

    int   GetMax() const;
    int   GetMin() const;
    float GetAvg() const;
    int GetCountUnder( float threshold ) const;
    int GetCountAbove( float threshold ) const;
};
```

Документация:

- https://msdn.microsoft.com/en-us/library/07x6b05d.aspx (раздел const member functions)

Шаг 5. Кешировать внутри класса результаты вычислений статистических значений: минимум, максимум, среднее и, возможно, даже результаты функций GetCountUnder, GetCountAbove. Это необходимо в случае, когда набор изменяется редко, а функции получения статистических данных вызываются часто. Без кеширования результатов вычислений программа будет каждый раз заново вычислять то, что уже было рассчитано.

Для min/max/avg всё просто, вот пример алгоритма:

```
int StatisticMultiset::GetMax() const {
  // Если набор не изменялся с прошлого расчета максимума,
       // то вернуть уже расчитанное значние.
  // иначе
      // 1. Расчитать новое значение максимума.
      // 2. Сохранить это значение (закешировать).
      // 3. Вернуть это значение.
}
```

Для GetCountUnder сложнее, т.к. функция имеет входной аргумент. Скорее всего, частые вызовы будут происходить от одного и того же значения, так что предлагаю кешировать результаты вычислений для последнего вызванного параметра. Либо от нескольких таких параметров.

Функции объявлены константными (и они по сути своей константные), однако технически в их реализации придется изменять значения полей класса. Для этого существует специальный механизм. См. документацию на ключевое слово mutable

- https://msdn.microsoft.com/ru-ru/library/4h2h0ktk.aspx
- https://msdn.microsoft.com/en-us/library/4h2h0ktk.aspx

Шаг 6. Сделать тип хранимых значений произвольным.

```
template<class T>
class StatisticMultiset {
public:
    StatisticMultiset();
    ~StatisticMultiset();

    void  AddNum( const T& );
    void  AddNum( const std::multiset<T>& numbers );
    void  AddNum( const std::vector<T>& numbers );
    void  AddNum( const std::list<T>& numbers );
    void  AddNums( const StatisticMultiset& a_stat_set );
    void  AddNumsFromFile( const char* filename );

    T GetMax() const;
    T GetMin() const;
    float GetAvg() const;
    int GetCountUnder( float threshold ) const;
    int GetCountAbove( float threshold ) const;
};
```

Документация:

- http://www.cplusplus.com/doc/tutorial/templates/ (раздел Class templates)
- http://en.cppreference.com/w/cpp/language/class_template
- http://www.cplusplus.com/doc/tutorial/templates/

Пример использования

```
int main(int argc, char *argv[])
{
    cout << "Lab 02" << endl;
    StatisticMultiset<int> ms1;
    ms1.AddNum( 89 );
    ms1.AddNum( 54 );
    ms1.AddNum( 54 );
    ms1.AddNum( 24 );

    StatisticMultiset<int> ms2;
    std::vector<int> somedata = { 10, 40, 6, 87 };
    ms.AddNum( somedata );

    StatisticMultiset<int> ms3;
    ms3.AddNums( ms1 );
    ms3.AddNums( ms2 );

    cout << " Min: " << ms3.GetMin()
         << " Avg: " << ms3.GetAvg()
```

```
        << " Max: " << ms3.GetMax()
        << endl;

    return 0;
}
```

Примечания

- Вы *можете* использовать возможности C++11, если ваш компилятор их поддерживает. Рассматривать эти возможности мы будем позднее, поэтому вы можете изучить некоторые из них самостоятельно:
  - http://en.cppreference.com/w/cpp/language/range-for (советую сначала просмотреть раздел Examples).
  - https://msdn.microsoft.com/ru-ru/library/jj203382.aspx
  - Обратите внимание на существенную разницу между двумя записями:
    - `for (auto& x : data) { /* ... */ }`
    - `for (auto  x : data) { /* ... */ }`

- Данная программа будет вам необходима в дальнейшем для выполнения других заданий, поэтому обязательно сохраните исходники.
- Вариант у работы всего один, цель работы – наработка опыта. Списывая работу, вы обрекаете себя на получения неуда на экзамене, так как перестанете успевать.
- В случае невозможности (как вам кажется) выполнения этого задания, обратитесь к преподавателю и согласуйте с ним ослабление условий задачи (например, выполнение не всех шагов).