

# NumberSeries

Ane Søgaaard Jørgensen

February 25, 2022

Listing 1: NumberSeries.h

```
1  //
2  // Created by ane on 18/02/2022.
3  //
4
5  #ifndef SESSION_5_NUMBERSERIES_H
6  #define SESSION_5_NUMBERSERIES_H
7
8
9  #include <vector>
10 #include <random>
11 #include <functional>
12 #include <iostream>
13
14 namespace series {
15
16     class NumberSeries {
17     public:
18         std::vector<int> series{};
19         int maximum{std::numeric_limits<int>::min()};
20         int minimum{std::numeric_limits<int>::max()};
21
22         int GetMaximum() {
23             return maximum;
24         }
25
26         int GetMimimum() {
27             return minimum;
28         }
29
30         void AddNumber(int);
31
32         void RemoveNumbers(int);
33
34         static NumberSeries MakeRandom(int);
35
36         NumberSeries operator+(const NumberSeries&);
37
38         NumberSeries& operator+=(const NumberSeries&);
39
40         bool operator<(const NumberSeries&) const;
41
42         friend std::ostream& operator<<(std::ostream&, const NumberSeries&);
43     };
44
45 }
46
47
48 #endif //SESSION_5_NUMBERSERIES_H
```

```

1  //
2  // Created by ane on 18/02/2022.
3  //
4
5  #include <chrono>
6  #include "NumberSeries.h"
7
8  namespace series {
9
10     void NumberSeries::AddNumber(int num) {
11         if (num < minimum) {
12             minimum = num;
13         }
14         else if (num > maximum) {
15             maximum = num;
16         }
17
18         series.push_back(num);
19     }
20
21     void NumberSeries::RemoveNumbers(int num) {
22         auto it = std::remove_if(series.begin(), series.end(), [num](int x) -> bool {
23             return x == num;
24         });
25
26         series.erase(it, std::end(series));
27     }
28
29     NumberSeries NumberSeries::MakeRandom(int length) {
30         NumberSeries result{};
31
32         std::random_device dev{};
33         std::default_random_engine random_engine {dev()};
34         std::uniform_int_distribution<> dist {0, 10000};
35         auto rng = std::bind(dist, random_engine);
36
37         for (int i = 0; i < length; i++) {
38             // result.AddNumber(dist(random_engine));
39             result.AddNumber(rng());
40         }
41
42         return result;
43     }
44
45     NumberSeries NumberSeries::operator+(const NumberSeries &other) {
46         NumberSeries result{};
47
48         int a = series.size();
49         int b = other.series.size();
50
51         int min_length = a < b ? a : b;
52
53         int i = 0;
54         for (; i < min_length; i++) {
55             result.AddNumber(series[i] + other.series[i]);
56         }
57
58         for (; i < series.size(); i++) {
59             result.AddNumber(series[i]);
60         }

```

```

61         for (; i < other.series.size(); i++) {
62             result.AddNumber(other.series[i]);
63         }
64     }
65
66     return result;
67 }
68
69 NumberSeries &NumberSeries::operator+=(const NumberSeries &other) {
70     int i = 0;
71     for (; i < series.size() && i < other.series.size(); i++) {
72         series[i] += other.series[i];
73         if (series[i] > maximum) {
74             maximum = series[i];
75         }
76         else if (series[i] < minimum) {
77             minimum = series[i];
78         }
79     }
80
81     for (; i < other.series.size(); i++) {
82         AddNumber(other.series[i]);
83     }
84
85     return *this;
86 }
87
88 bool NumberSeries::operator<(const NumberSeries &other) const {
89     int my_amplitude = maximum - minimum;
90     int other_amplitude = other.maximum - other.minimum;
91
92     return my_amplitude < other_amplitude;
93 }
94
95 std::ostream& operator<<(std::ostream& os, const NumberSeries& numberSeries) {
96     if (numberSeries.series.empty()) {
97         os << "(empty series)";
98     }
99     else {
100         for (const auto& i : numberSeries.series) {
101             os << i << " ";
102         }
103     }
104
105     return os;
106 }
107 }

```

Listing 3: NumberSeriesWrapper.cpp

```

1  //
2  // Created by ane on 18/02/2022.
3  //
4
5  #include "NumberSeriesWrapper.h"
6
7  namespace series {
8      int NumberSeriesWrapper::GetMaximum() {
9          return ptr->GetMaximum();
10     }
11
12     int NumberSeriesWrapper::GetMinimum() {

```

```

13     return ptr->GetMimumum();
14 }
15
16 NumberSeriesWrapper NumberSeriesWrapper::MakeRandom(int length) {
17     return NumberSeriesWrapper(NumberSeries::MakeRandom(length));
18 }
19
20 NumberSeriesWrapper NumberSeriesWrapper::operator+(const NumberSeriesWrapper& other) {
21     return NumberSeriesWrapper(*ptr + *other.ptr);
22 }
23
24 NumberSeriesWrapper &NumberSeriesWrapper::operator+=(const NumberSeriesWrapper& other) {
25     *ptr += *other.ptr;
26     return *this;
27 }
28 }

```

Listing 4: NumberSeriesWrapper.h

```

1  //
2  // Created by ane on 18/02/2022.
3  //
4
5  #ifndef SESSION_5_NUMBERSERIESWRAPPER_H
6  #define SESSION_5_NUMBERSERIESWRAPPER_H
7
8
9  #include <memory>
10 #include "NumberSeries.h"
11
12 namespace series {
13     class NumberSeriesWrapper {
14     public:
15         std::unique_ptr<NumberSeries> ptr{};
16
17         explicit NumberSeriesWrapper(const NumberSeries& series) {
18             ptr = std::make_unique<NumberSeries>(series);
19         }
20
21         NumberSeriesWrapper() = default;
22
23         ~NumberSeriesWrapper() noexcept = default;
24
25         NumberSeriesWrapper(const NumberSeriesWrapper& other) = delete;
26
27         NumberSeriesWrapper& operator=(const NumberSeriesWrapper& other) = delete;
28
29         NumberSeriesWrapper(NumberSeriesWrapper&& other) noexcept {
30             *this = std::move(other);
31         }
32
33         NumberSeriesWrapper& operator=(NumberSeriesWrapper&& other) noexcept {
34             if (this == &other) {
35                 return *this;
36             }
37
38             // std::swap(ptr, other.ptr);
39             ptr = std::move(other.ptr);
40
41             return *this;
42         }
43

```

```

44     int GetMaximum();
45
46     int GetMinimum();
47
48     static NumberSeriesWrapper MakeRandom(int);
49
50     NumberSeriesWrapper operator+(const NumberSeriesWrapper&);
51
52     NumberSeriesWrapper& operator+=(const NumberSeriesWrapper&);
53
54     bool operator<(const NumberSeriesWrapper& other) const {
55         return *ptr < *other.ptr;
56     }
57 };
58 }
59
60
61
62
63 #endif //SESSION_5_NUMBERSERIESWRAPPER_H

```

Listing 5: main.cpp

```

1  #include <iostream>
2  #include <vector>
3  #include <chrono>
4  #include "NumberSeries.h"
5  #include "NumberSeriesWrapper.h"
6  #include <numeric>
7
8  #define NUM_SERIES 100000
9  #define NUM_ELEMENTS 100
10
11 using namespace std;
12 using namespace series;
13
14 vector<NumberSeries> numberSeries(NUM_SERIES);
15 vector<NumberSeriesWrapper> numberSeriesWrappers(NUM_SERIES);
16
17 void testProgram() {
18
19     cout << "Series: " << endl;
20
21     auto first = std::chrono::high_resolution_clock::now();
22     for (auto& series : numberSeries) {
23         series = NumberSeries::MakeRandom(NUM_ELEMENTS);
24     }
25     auto last = std::chrono::high_resolution_clock::now();
26
27     cout << "Building: " << chrono::duration<double, milli>(last - first).count() << endl;
28
29     first = std::chrono::high_resolution_clock::now();
30     std::sort(numberSeries.begin(), numberSeries.end());
31     last = std::chrono::high_resolution_clock::now();
32
33     cout << "Time: " << chrono::duration<double, milli>(last - first).count() << endl;
34     // auto sum = std::accumulate(numberSeries.begin(), numberSeries.end(), NumberSeries());
35     // cout << sum.GetMaximum() << endl;
36
37     cout << endl;
38
39     cout << "Wrappers: " << endl;

```

```

40
41     first = std::chrono::high_resolution_clock::now();
42     for (auto& series : numberSeriesWrappers) {
43         series = NumberSeriesWrapper{NumberSeriesWrapper::MakeRandom(NUM_ELEMENTS)};
44     }
45     last = std::chrono::high_resolution_clock::now();
46
47     cout << "Building: " << chrono::duration<double, milli>(last - first).count() << endl;
48
49     first = std::chrono::high_resolution_clock::now();
50     std::sort(numberSeriesWrappers.begin(), numberSeriesWrappers.end());
51     last = std::chrono::high_resolution_clock::now();
52
53
54     cout << "Time: " << chrono::duration<double, milli>(last - first).count() << endl;
55 }
56
57 int main() {
58     testProgram();
59
60     return 0;
61 }

```

Listing 6: CMakeLists.txt

```

1  cmake_minimum_required(VERSION 3.21)
2  project(Session_5)
3
4  set(CMAKE_CXX_STANDARD 20)
5
6  add_executable(Session_5 main.cpp NumberSeries.cpp NumberSeries.h NumberSeriesWrapper.cpp ↗
↪NumberSeriesWrapper.h)

```