Chris Culling

Lab report for **assignment 4**

7 Nov 2024

Exercise 1: Sorting whole numbers

Exercise 1a: Sorting `vector`

To create the randomized vector:

```
int size = 12;
std::vector<int> numbers;

for (int i = 0; i < size; i++)
{
    numbers.push_back(rand());
}
```

The problem here is that the numbers are random in a sense, but they will be the same every time the program runs because `rand()` isn't really random. But it works for our purposes, so let's move on.

SortRandomlyGeneratedNumbersInVector()

```
std::vector<int> randomNumbers = GetRandomNumbers();
cout << "Random ";
PrintNumbers(randomNumbers);

std::sort(randomNumbers.begin(), randomNumbers.end());
cout << "Sorted ";
PrintNumbers(randomNumbers);
```

Output ✅

```
Random Numbers: 41 18467 6334 26500 19169 15724 11478 29358 26962 24464 5705 28145
Sorted Numbers: 41 5705 6334 11478 15724 18467 19169 24464 26500 26962 28145 29358
```

Exercise 1b: Sorting `int[]`

Instead of `vector.push_back()`, I use pointer iteration to initialize the int array with random values.

```
int* numbers = new int[size];

for(int* pointer = numbers; pointer != numbers + size; ++pointer)
{
    *pointer = rand();
}
```

Thank you stackoverflow for teaching me how to use `std::sort` with an `int array`. Although I couldn't get `std::begin()` and `std::end()` to work, my actual implementation, inspired by the same thread, seems simpler.

```
std::sort(randomNumbers, randomNumbers + size);
```

SortRandomlyGeneratedNumbersInIntArray()

```
// 1.
int* randomNumbers = GetRandomIntArray();

// 2.
cout << "Random c-array ";
PrintNumbersInIntArray(randomNumbers);

// 3.
std::sort(randomNumbers, randomNumbers + size);

// 4.
cout << "Sorted c-array ";
PrintNumbersInIntArray(randomNumbers);

delete randomNumbers;
```

Output ☑

```
Random c-array Numbers: 23281 16827 9961 491 2995 11942 4827 5436 32391 14604 3902
153
Sorted c-array Numbers: 153 491 2995 3902 4827 5436 9961 11942 14604 16827 23281
32391
```

Exercise 1c: Sorting `vector` in descending order

I added a `bool descending` parameter to my `SortRandomlyGeneratedNumbersInVector()` method, which I set to false by default (`void SortRandomlyGeneratedNumbersInVector(bool descending = false);`). Then, I simply added a simple if-statement in the method. This way, I didn't have to write a new method!

```
if(descending)
    std::sort(randomNumbers.rbegin(), randomNumbers.rend());
else
    std::sort(randomNumbers.begin(), randomNumbers.end());

cout << "Sorted "; if(descending) cout << "DESCENDING ";
PrintNumbers(randomNumbers);
```

Output ✅

```
Random Numbers: 292 12382 17421 18716 19718 19895 5447 21726 14771 11538 1869
19912
Sorted DESCENDING Numbers: 21726 19912 19895 19718 18716 17421 14771 12382 11538
5447 1869 292
```

Exercise 1d: Sorting `int[]` in descending order

Apparently, just swapping the arguments when `std::sort`ing the `int[]` crashes the program 👍

```
if(descending)
    // this crashes the program
    std::sort(randomNumbers + size, randomNumbers); // <-- woops
else
    std::sort(randomNumbers, randomNumbers + size);
```

Despite my unaltered efforts, `std::begin()` and `std::end()` still don't seem to exist.

Giving a function as a third parameter as the assignment instructions recommended did the trick.

```
if(descending)
{
    auto greater = [](int x, int y){ return x>y; };

    std::sort(randomNumbers, randomNumbers + size, greater);
}
else
    std::sort(randomNumbers, randomNumbers + size);
```

Output ✅

```
Random c-array Numbers: 25667 26299 17035 9894 28703 23811 31322 30333 17673 4664
15141 7711
```

```
Sorted c-array DESCENDING Numbers: 31322 30333 28703 26299 25667 23811 17673 17035
15141 9894 7711 4664
```

Exercise 2: Sorting the person register

Exercise 2a: Sorting persons

I added my **`cpplab3`** repository as a submodule to this one (**`cpplab4`**), and configured my `tasks.json` so that my program looks for header and `.cpp` files inside the `cpplab3/` subdirectory. This took about an hour of troubleshooting to get right.

Remember that issue we had in exercise `1a`? Yeah we're gonna have to fix that now...

The issue in question was the non-functioning `random_shuffle()` *which was deprecated in C++14*. To address the issue, I used `shuffle()` instead, which needed a random generator as a parameter.

```
std::random_device rd;
std::mt19937 generator(rd());
std::shuffle(personRegister.begin(), personRegister.end(), generator);
```

Using `sort()` was pretty straight forward now that we have `PersonRegister::begin()` and `PersonRegister::end()`.

```
std::sort(personRegister.begin(), personRegister.end()); // Easy peasy!
```

Output ☑

```
SHUFFLED REGISTER
----------------------------------
Tess Addison, Englandismycity 1, 018 12 FACTS
Rick Astley, Im Thinking Of, 012 98 YOU
Rick Astley, And So Do I, 012 98 DOI
Flossie Firmin, Fortenite 5, 999 99 GRIND
Rick Astley, From Any Other, 012 98 GUY
Kathlyn Simons, Simonsaid Lane 2, 012 34 PLACE
Rick Astley, Were No Strangers, 012 98 TOLOVE
Rick Astley, A Full Commitments, 012 98 WHAT
Rick Astley, Wouldnt Get, 012 98 THIS
Arline Ansel, Jolene road 12, 000 00 PLEASE
Oz Janson, Wizard Tower 4, 012 35 PLACE
Rick Astley, You Know The, 012 98 RULES
----------------------------------
SORTED REGISTER
----------------------------------
Arline Ansel, Jolene road 12, 000 00 PLEASE
```

```
Flossie Firmin, Fortenite 5, 999 99 GRIND
Kathlyn Simons, Simonsaid Lane 2, 012 34 PLACE
Oz Janson, Wizard Tower 4, 012 35 PLACE
Rick Astley, Im Thinking Of, 012 98 YOU
Rick Astley, And So Do I, 012 98 DOI
Rick Astley, From Any Other, 012 98 GUY
Rick Astley, Were No Strangers, 012 98 TOLOVE
Rick Astley, A Full Commitments, 012 98 WHAT
Rick Astley, Wouldnt Get, 012 98 THIS
Rick Astley, You Know The, 012 98 RULES
Tess Addison, Englandismycity 1, 018 12 FACTS
----------------------------------
```

Exercise 2b: Sorting backwards by address

Interesting how we're influencing the behaviour of sort by adjusting our operator overload code!

Output ✅

```
SHUFFLED REGISTER
----------------------------------
Flossie Firmin, Fortenite 5, 999 99 GRIND
Rick Astley, You Know The, 012 98 RULES
Tess Addison, Englandismycity 1, 018 12 FACTS
Rick Astley, Im Thinking Of, 012 98 YOU
Rick Astley, From Any Other, 012 98 GUY
Rick Astley, Wouldnt Get, 012 98 THIS
Rick Astley, A Full Commitments, 012 98 WHAT
Kathlyn Simons, Simonsaid Lane 2, 012 34 PLACE
Oz Janson, Wizard Tower 4, 012 35 PLACE
Rick Astley, Were No Strangers, 012 98 TOLOVE
Arline Ansel, Jolene road 12, 000 00 PLEASE
Rick Astley, And So Do I, 012 98 DOI
----------------------------------
SORTED REGISTER
----------------------------------
Rick Astley, You Know The, 012 98 RULES
Rick Astley, Wouldnt Get, 012 98 THIS
Oz Janson, Wizard Tower 4, 012 35 PLACE
Rick Astley, Were No Strangers, 012 98 TOLOVE
Kathlyn Simons, Simonsaid Lane 2, 012 34 PLACE
Arline Ansel, Jolene road 12, 000 00 PLEASE
Rick Astley, Im Thinking Of, 012 98 YOU
Rick Astley, From Any Other, 012 98 GUY
Flossie Firmin, Fortenite 5, 999 99 GRIND
Tess Addison, Englandismycity 1, 018 12 FACTS
Rick Astley, And So Do I, 012 98 DOI
Rick Astley, A Full Commitments, 012 98 WHAT
----------------------------------
```

Exercise 3

I heard on the grapevine that `erase()` and `erase_if()` are preferrable (C++20). Anyway,

RemoveEvenNumbersFromRandomNumbers()

```
// bool Even(int i) { return i%2==0; }

//...

// 3. How I removed the even numbers
auto result = std::remove_if(randomNumbers.begin(), randomNumbers.end(), Even);
randomNumbers.erase(result, randomNumbers.end());

// ...
```

Output ✓

```
Random Numbers: 28253 6868 25547 27644 32662 32757 20037 12859 8723 9741 27529 778
Removed EVEN Numbers: 28253 25547 32757 20037 12859 8723 9741 27529
```