Financial Programming in C++: Homework Assignment 8

Fall 2024,

MSQF, Fordham University

Due: November 8th, 2024

Problem 1

Define a class `VectorInt` in a header `VectorInt.h` for a *container* of integers. The members of the class will be as follows

```
VectorInt {
public:
 //  public interface defined here
private:
        size_t m_size;
        int *m_data;
};
```

In the header file for the class declare the following public member functions:

1. a **constructor** taking as argument the `size` (number of elements) of the conatiner. It should default to an empty container with zero elements.
2. a **destructor**.
3. a **copy constructor**.
4. a **copy assignment operator**.
5. a **move constructor**.
6. a **move assigment operator**.

Problem 2

In a separate file `VectorInt.cpp` implement the methods defined in Problem 1.

Problem 3

Declare (in `VectorInt.h`) an implement (in `VectorInt.cpp`) the public indexing operations of the class `VectorInt`:

1. a `size` function taking no arguments and returning `size_t` with the size of the container.
2. an indexing operator taking a `size_t index` argument and returning a *reference* to the integer at position `index` in the container.
3. a constant indexing operator taking a `size_t index` argument and returning a *constant reference* to the integer at position `index` in the container.

Problem 4

Declare and implement a *friend* `operator<<` taking as input a `std::ostream` and a `VectorInt`. The operator should output the contents of the vector in the stream.

Problem 5

Create a new source file `test_int.cpp` with a `main` function: The `main` function must:

1. Take as input from the command line a single argument `size` of type int with the size of the container (you can see example of how to do this on the file `test_double.cpp` shared in class).
2. Initialize a `VectorInt` object (named `vec1`) with `size` elements. Set element at index i of vector to the value $(i+1)^2$. Display the contents of `vec1` to the output stream (`std::cout`)
3. Initialize a new vector (named `vec2`) to a copy of `vec1`. Display the contents of `vec2`.
4. Set all elements of `vec2` to their negative values. Display `vec1` and `vec2` to demonstrate that `vec2` has changed but `vec1` has not.
5. Set `vec1` to a copy of `vec2`. Show that the contents of the two vectors are now equal.

Problem 6

Using operator overloading define the following mathematical operations on the `VectorInt` class:

1. `VectorInt` addition: given two vectors $v1$ and $v2$ it should return a new vector v so that each component of v is the sum of the components of $v1$ and $v2$. $v1$ and $v2$ must have the same size (if you have trouble writing the error checking code, you can just assume that this is true and do not need to check).

2. scalar multiplication: given a vector $v1$ and integer k create a new vector v where each component of v is k times the same component of $v1$. [HINT] this is slightly nicer if implemented as a friend function of the class.

Problem 7

In the `main` function of `test_int.cpp`, define vectors v_1, v_2 and v_3 with initial values

$$v1 = \{0, 1, 2, 3, 4\};$$
$$v2 = \{0, 1, 4, 9, 16\};$$
$$v3 = \{1, -1, 2, -2, 3\};$$

Use the `VectorInt` class to compute the expression:

$$v = 3(v_1 + 2v_2) - 7v_3$$

Display the final value of v.

[HINT] you *do not need* to define a substraction operator.

Problem 8

1. Compile file `test_matrix0.cpp`. Verify that the program crashed. Can you explain why?

2. Copy `Matrix0.h`, `Matrix0.cpp` and `test_matrix0.cpp` into `Matrix.h`, `Matrix.cpp` and `test_matrix.cpp`. Implement in class `Matrix` the special member functions required to fix the issues you just demonstrated. The only change you can make to `test_matrix.cpp` is to include `Matrix.h` instead of `Matrix0.h`.

3. Copy again `Matrix0.h`, `Matrix0.cpp` and `test_matrix.cpp` into `Matrix1.h`, `Matrix1.cpp` and `test_matrix1.cpp`. Implement class `Matrix` using a `VectorDouble` container instead of a `double *` data member. Demonstrate that then you do not need to explicitly implement the special member functions.

You must submit:

1. `test_matrix0.txt` with the output of `test_matrix0.cpp`.

2. `Matrix.h`, `Matrix.cpp`, `test_matrix.cpp` and the output of running the program as `test_matrix.txt`.

3. `Matrix1.h`, `Matrix1.cpp`, `test_matrix1.cpp` and the output of running the program as `test_matrix1.txt`.

Problem 9

Consider the two asset portfolio replication problem with correlation matrix:

$$C = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

Write a program `test_condition_number.cpp` that uses Cholesky decomposition to solve the linear system

$$Cw = b$$

with

$$b = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

and also solves the slightly modified linear system

$$Cw' = b'$$

with the modified right hand side

$$b' = \begin{pmatrix} 0.5 + \epsilon \\ 0.5 \end{pmatrix}$$

where $\epsilon = 0.01$.

for the correlations $\rho = 0.9, 0.99, 0.999, 0.9999$ the program must output (in a comma separated file), one line per correlation, the following values:

- the correlation ρ.
- the condition number κ of the matrix C.
- The maximum error between the solution w obtained with right hand side b and the solution w' obtained with right hand side b'.
- $$\text{max\_error} = \kappa \cdot \epsilon$$
- the actual error between the solutions w and w' obtained with right hand sides b and b' respectively:

$$\text{error} = ||w - w'|| = \sqrt{\sum_{i=1}^{2} (w_i - w'_i)^2}$$

You must summit:

- the source code of the program `test_condition_number.cpp`
- the output of the program `test_condition_number.csv`