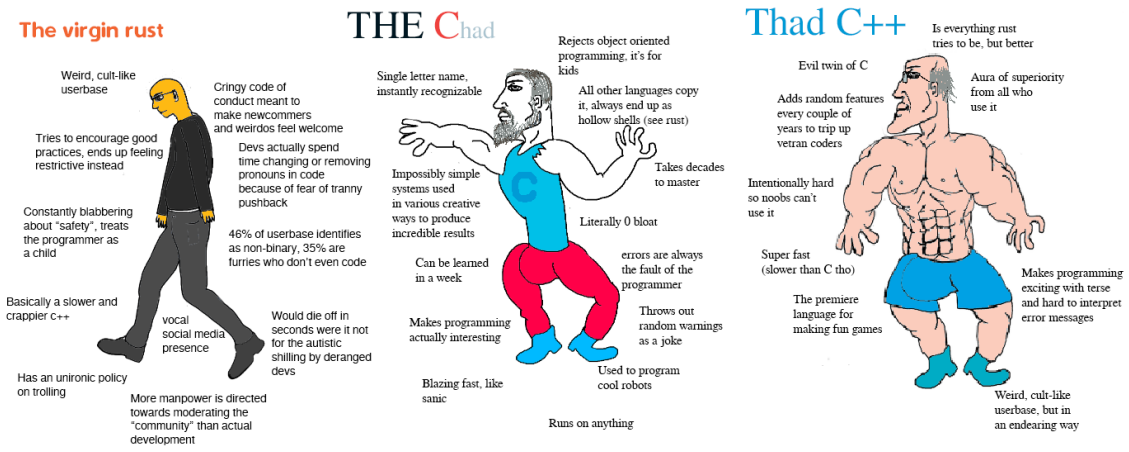
Homework02 for Architecture.



Files:

- **main.cpp(3KB)**: main function which reads, writes, etc...
- **container.cpp(2KB)/.h(654B)**: container with all the functions.
- **baseMatrix.cpp(395B)/.h(649B)**: basic matrix structure with all the functions.
- **matrix.cpp(1KB)/.h(612B)**: usual matrix structure with all the functions.
- **diagonalMatrix.cpp(1KB)/.h(651B)**: diagonal matrix structure with all the functions.
- **lowerTriangularMatrix.cpp(2KB)/.h(775B)**: lower-triangular matrix structure with all the functions.

Used hardware and software:

- OS: MacOS Big Sur 11.6
- RAM: 16gb
- Architecture: x86-x64
- IDE: CLion
- Assembly: CMake

Command line input guide:

1) Write ***./ArchitectureHW2 -f [inputFileName].txt [outputFileName].txt*** for the file input. 2) Write ***./ArchitectureHW2 -n [number] [outputFileName].txt*** for the random input.

File input guide:

You need to input a couple of matrices according to this template:

1) Input type: ***0*** for usual matrix, ***1*** for diagonal matrix, ***2*** for lower-triangular matrix. 2) Input size: ***N*** for ***NxN*** matrix. 3) Input N^2 real numbers for usual matrix, ***N*** real numbers for diagonal matrix and ***N * (N + 1) / 2*** real numbers for lower-triangular matrix.

Tests:

For your conviniece I created a set of 11 tests via python *FillingScript.py* with input00.txt as an empty file.

Comparing to procedure programming:

- most of the files are bigger now except for base class files, because most functions are virtual now.
- Architecture is much more convenient in object-oriented programming.

Average time:

- for 1 matrix: 2ms
- for 10 matrices: 17ms
- for 100 matrices: 246ms
- for 1000 matrices: 3338ms

Conclusion:

- Object-oriented programming is clearer for developer to see and write, comparing to procedure programming.
- According to time, this program has O(n) complexity, where n is a number of matrices.