CS100
Introduction to Programming

Tutorial 6: strings, stringsstreams, & templates

Note: Tutorial 6 contains only 2 problems. Please use the additional time to ask questions about the lecture material, as the mid-term exam is approaching

Problem 1

strings and stringstreams

Redefine column order

- Write a program that reads in data from a file, organized in table form. Each row simply contains the same number of elements, separated by a whitespace character, and terminated by an EOL character.
- The program should automatically identify the number of columns in the file, and prompt the user to indicate a new order of the columns.
- It should finish with writing a new file with the same content but the newly defined order of the columns.

Redefine column order

- Example file content:

```
 1   2   3   4
 5   6   7   8
 9  10  11  12
```

- Example interaction with the user:

```
The number of columns is 4
Please redefine the order of the columns:
3 2 1 0
The new column order that you defined is: 3 2 1 0
```

Redefine column order

- The generated output file would be:

```
  4   3   2   1
  8   7   6   5
 12  11  10   9
```

- For read-outs from the file, what would be the best format such that pretty much anything that has no white-space separations can be extracted as a single element?

Redefine column order

- Useful headers from the standard library:

```
#include <stdlib.h>
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
```

Problem 2
templates

Writing a matrix class

- Write an abstract class Matrix2 that defines a 2x2 Matrix for any type T
- The content of the matrix needs to be internally stored in some container. Use a std::vector!
- The constructor of the class should initialize the elements of Matrix2 with all zero elements

Writing a matrix class

- The Matrix2 class should provide the following functions:

```
//returns the element at row "row" and column "col"
T & at( int row, int col );
//performs in-place transposition of a matrix
void transpose();
//adds another matrix to this
void add( Matrix2<T> & matrix );
//removes another matrix from this
void remove( Matrix2<T> & matrix );
//computes and returns the determinant
T determinant();
//prints the matrix in proper format to the console
void print();
```

Test the matrix class

```
int main() {
  Matrix2<float> matrix;
  matrix.at(0,0) = 0.1f;
  matrix.at(1,0) = 1.2f;
  matrix.at(0,1) = 2.3f;
  matrix.at(1,1) = 3.4f;
  std::cout << "The original matrix is:\n";
  matrix.print();
  std::cout << "Transposing the matrix returns:\n";
  matrix.transpose();
  matrix.print();
  std::cout << "The determinant is: ”;
  std::cout << matrix.determinant() << "\n";
  std::cout << "The matrix times 2 is:\n";
  matrix.add(matrix);
  matrix.print();
  std::cout << "Subtracting the matrix from itself returns:\n";
  matrix.remove(matrix);
  matrix.print();
  return 0;
}
```

Test the matrix class

- The output should be

```
The original matrix is:
0.1 2.3
1.2 3.4

Transposing the matrix returns:
0.1 1.2
2.3 3.4

The determinant is: -2.42
The matrix times 2 is:
0.2 2.4
4.6 6.8

Subtracting the matrix from itself returns:
0 0
0 0
```

Observe what happens if you replace float by int!

```
int main() {
  Matrix2<float> matrix;
  matrix.at(0,0) = 0.1f;
  matrix.at(1,0) = 1.2f;
  matrix.at(0,1) = 2.3f;
  matrix.at(1,1) = 3.4f;
  std::cout << "The original matrix is:\n";
  matrix.print();
  std::cout << "Transposing the matrix returns:\n";
  matrix.transpose();
  matrix.print();
  std::cout << "The determinant is: ";
  std::cout << matrix.determinant() << "\n";
  std::cout << "The matrix times 2 is:\n";
  matrix.add(matrix);
  matrix.print();
  std::cout << "Subtracting the matrix from itself returns:\n";
  matrix.remove(matrix);
  matrix.print();
```