Course Administration

- Exercise 2 Preliminary Exercise
- Project 1 Preliminary Exercise
DFAs as scanners (aka tokenizers)

- Alphabet = \{a, b, c, x, y, z, ∼\}
- Regular expression 1 (RE1)
  - a* (ab | bc) a+
- Regular expression 2 (RE2)
  - x+ (xy | yz | xz) z*
- Combined
  - (a* (ab | bc) a+) | (x+ (xy | yz | xz) z*)
(a* (ab | bc) a+) | (x+ (xy | yz | xz) z*)
Programming a DFA

Table

<table>
<thead>
<tr>
<th>ws</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Err1</td>
<td>7</td>
<td>Err1</td>
<td>Err1</td>
</tr>
<tr>
<td>2</td>
<td>Err2</td>
<td>2</td>
<td>4</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
</tr>
<tr>
<td>3</td>
<td>Err2</td>
<td>Err2</td>
<td>5</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
</tr>
<tr>
<td>4</td>
<td>Err2</td>
<td>6</td>
<td>5</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
</tr>
<tr>
<td>5</td>
<td>Err2</td>
<td>6</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
</tr>
<tr>
<td>6</td>
<td>RE1</td>
<td>6</td>
<td>RE1</td>
<td>RE1</td>
<td>RE1</td>
<td>RE1</td>
<td>RE1</td>
</tr>
<tr>
<td>7</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>8</td>
<td>9</td>
<td>Err2</td>
<td>Err2</td>
</tr>
<tr>
<td>8</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>Err2</td>
</tr>
<tr>
<td>9</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>Err2</td>
<td>11</td>
<td>Err2</td>
</tr>
<tr>
<td>10</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
</tr>
<tr>
<td>11</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
<td>RE2</td>
</tr>
</tbody>
</table>
Data Types

- Scalar
- Array
- Record
- Object
Why are we doing this?
What is “Currency”?

- A system of money in general use in a particular country.
- The tangible form of money that is paper bills and coins.
- Monetary amount → class Money
Class “Money”

- What attributes should it contain?
- What methods should it implement?
- What will it be used for?
  - A problem similar to the one posed for Exercise 1
Interface vs Implementation

- Interface “.h” file
- Implementation “.cpp” file
- Why separate interface and implementation
- Black box concept
  - Programmer needs to know what a method does
  - Programmer needs to know how to use a method
  - Programmer does not need to know the nitty-gritty details of how a method works.
  - For example: vector insert method
Separate compilation

- Money.h
- Money.cpp → Money.o
- makefile
  
  Money.o : Money.h Money.cpp
  g++ -c Money.cpp

  Exercise2.o : Exercise2.h Exercise2.cpp
  g++ -c Exercise2.cpp

  E2.out : Exercise2.o Money.o
  g++ -o E2.out Exercise2.o Money.o
#ifndef MONEY_H
#define MONEY_H

#include <iostream>
using namespace std;

class Money
{
    public:
        Money ();
        Money (const Money & M);
        ~Money ();
        Money & operator = (const Money & M);

    private:
        // int dollars, cents;
};

#endif
What else?

- Attributes
  - `int dollars, cents;`
- Methods
  - Constructors
  - Mutators (aka setters)
  - Accessors (aka getters)
  - Operators