Matrix and File IO

EECE 3326 Optimization Methods

Instructor: Ningfang Mi

Matrix - Two-Dimensional Array

```
int mat[3][4] = {{8,1,7,-2}, {0,-3,4,6}, {10,-14,1,0}};
mat[1][2] :
mat[0][3] :
```

- Declare a matrix:
- Access a matrix:
- C++ stores a matrix in memory by rows
  - Limitation: the compiler needs to know the number of columns in advance

Matrix Container

- Using `vector.h`

```
vector<vector <T> > mat;
mat[0]: the vector of column entries in row 0;
mat[1]: the vector of column entries in row 1;
```

- Matrix class uses matrix container as private data member.

Matrix class

```
template <typename T>
class matrix
{
   public:
      matrix(int numRows = 1, int numCols = 1,
                                    const T& initVal = T());
      vector<T>& operator[] (int i);
      const vector<T>& operator[](int i) const;
      int rows() const;
      int cols() const;
      void resize(int numRows, int numCols);

   private:
      int nRows, nCols;
      vector<vector<T> > mat;
}
```

Using Matrix class

```
matrix<int> intMat(3,4);

matrix<time24> timeMat(2,5,time24(8,30));


intMat.resize(2,7);

intMat[1][5] = 7;
```

File I/O

- Standard input / output streams
  - and
- Attaching Streams to External Files
  - ifstream and ofstream ➔ fstream class
  - ifstream:

```
ifstream fin;
string fileName = "input";
fin.open(fileName.c_str());
if(!fin)
{
   //error-handling;
}
fin >> x;
fin.close();
```

File I/O

- Attaching Streams to External Files
  - ifstream and ofstream ➔ fstream class
  - ofstream:

```
ofstream fout;
string fileName = "output";
fout.open(fileName.c_str());
if(!fout)
{
...//error-handling;
}
fout << x;
fout.close();
```