More Classes (Lab10)

cse.msu.edu/~cse232/Labs/lab10.html

Compiler Options

g++ is a very complicated (but powerful) program. Technically, it is just a alias (alternate name) for gcc, which is the Gnu C Compiler. g++ (and gcc) have a great many options for determining how it should compile your program.

The full list of options can be found here, but I'll be pointing out the most important ones.

Here are the flags you should already be comfortable with:

- `-c` This flag instructs the compiler to compile all your cpp files, but not link them together (see Week01).
- `-o filename` This flag instructs the compiler to make a file named `filename` with the executable compiled program (instead of the default `a.out`).
- `-Wall` This flag instructs the compiler to warn about as many potential erroneous code elements as possible (useful for beginners).
- `-std=c++11` or `-std=c++14` These flags instruct the compiler to use a particular version of the C++ language.
- `-g` This flag instructs your compiler to include debugging information in the compilered program for use by `gdb`.

And here are some new ones:

- `-O1` This flag enables various code optimizations that allow your program to run faster, without a large increase in time to compile.
- `-O2` This flag enables various code optimizations that allow your program to run faster, but may increase the time needed to compile.
- `-O3` This flag is similar to O2, but with even more extreme optimizations and possibly very long compilations.
- `-Ofast` This flag is allows optimizations that aren't necessary allowed by the standards set forth by the C++ language committee. This is where experimental optimations are used by those who want speed at all costs.
- `-Os` This flag instructs the compiler to optimize for size instead of speed. It is often useful if you need to run your program on embedded computers with limited memory.
- `-Wextra` This flag instructs the compiler to add additional warnings for bad code (even more than covered by `-Wall`).
- `-Wpedantic` This flag instructs the compiler to add additional warnings for code that doesn't comply with the strict C++ language definition (useful if you want your code to be compiled by other compilers).

 Download the file warnings.cpp. Show your TA the output when you compile it with more warnings than `-Wall` checks for.

Programmming (Table Class)

The Problem

We are going to work on making our own classes with private data members and accessors. We are going to build a `Table` class, a 2D vector class.

Table Class

The header for the Table class has the following private elements:

```
private:
  vector<vector<long>> t_;    // 2D vector of long
  long width_;                // how wide is t_ (how many columns)
  long height_;               // how high is t_ (how many rows)
```

The methods are as follows:

- `Table(long width, long height, long val=0)` constructor. Makes a `Table` that is rectangle shaped, `width` x `height`. Each element is set to `val`, which defaults to 0. Remember that `t_` is a `vector<vector<long>>` and that what you can `push_back` onto `t_` is a `vector<long>` which constitutes a row of `t_`
- `void fill_random(long lo, long hi, unsigned int seed=0)`. Method to set every `t_` element to a random number of `long` between `lo` and `hi` inclusive. `seed` sets the random number seed, defaults to 0. Use the technique described here (https://diego.assencio.com/?index=6890b8c50169ef45b74db135063c227c) with a uniform distribution drawn from the MT19937 random number generator.

   Show your TA when your `Table` can perform fill_random.

- `bool set_value(unsigned int row_num, unsigned int col_num, long val)`. A method to set a particular element, indicated by `row_num` and `col_num`, to the provided `val`.
  - If `row_num` and `col_num` are indicies that exist in `t_`, sets that element to `val` and returns `true`.
  - Otherwise it does not set the element and then it returns `false`.

- `long get_value (unsigned int row_num, unsigned int col_num) const`. Method to provide the value at (`row_num` , `col_num`) of `t_` if those two indicies exist.
    - If the two indicies are valid, return the `t_` element.
    - If not, well we have a decision to make. Any long we return might actually be a legit long in the table, even though our intention was to indicate "not there" somehow. So instead we throw an out_of_range error
- `void print_table(ostream&)` a member function to print the contents of `t_` in a "nice way" (as a square with rows and columns) to `ostream` reference provided.

Assignment

You are provided with table.h and main-table.cpp. Create "table.cpp". Results should look like the below.

```
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,
0,0,0,0,0,

2,10,1,6,6,
1,7,5,9,10,
5,7,10,4,3,
5,8,7,6,3,
2,6,6,2,2,

Result:false

6
Correct!

100,10,1,6,6,
1,100,5,9,10,
5,7,100,4,3,
5,8,7,100,3,
2,6,6,2,100,
```

 Show your TA when you complete the lab.