Note Title 17-09-2018

COL 100

Sep 17, 2018

Announcements:

Mentors for COL100

Next graded Assignment → strings & vectors

```
vector<string> vecStrings = {"ab", "cd", "ef"};
```

① → ffcdab

② → fedcba

→ vector<int> nums = {1, -5, 6, 7, 8, -8, 100}

→ ⑬ → Compute the sum of even integers in nums

Vector

Grids: → (collections)

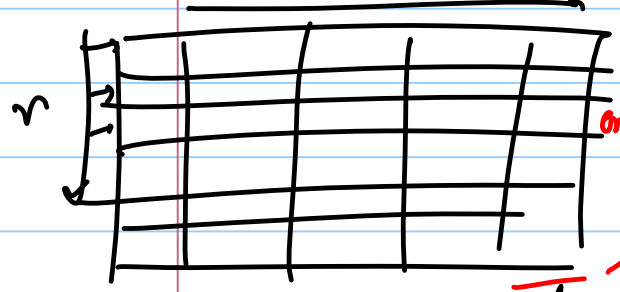
int a = "abc";

#include "grid.h" → // Stanford lib

① Grid<int> matrix; // 0 rows 0 cols

② Grid<int> matrix(3,4);

Grid size nxm



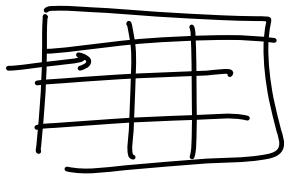
③ Grid<int> matrix

= {{1, 2, 4, 5},
{3, 5, 6, 7}, {8, 9, -10, 15}};

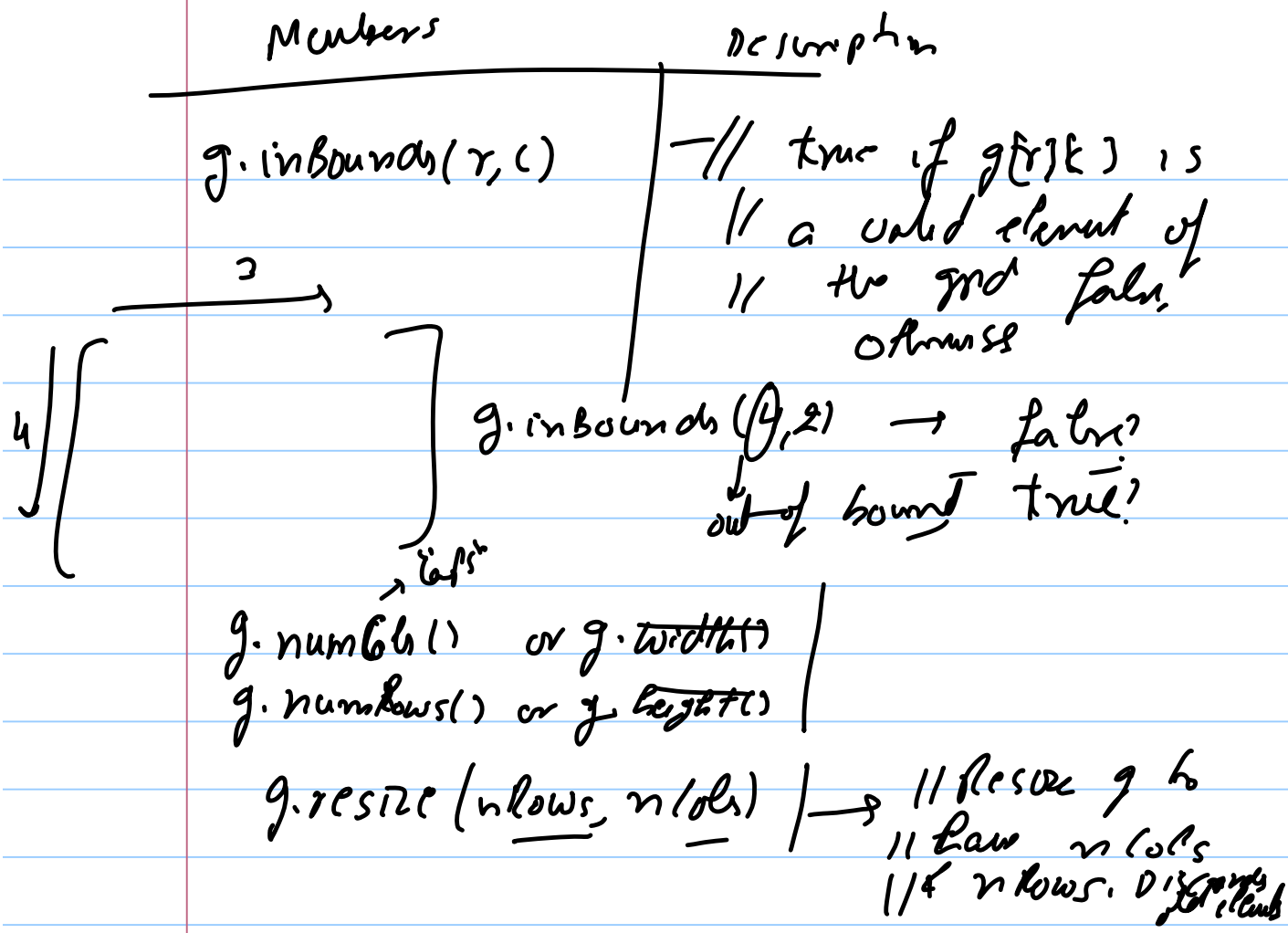
Vector<string> names;

Vector<string> names = {"al" "cl" "el" ... }

g[i][j]



Members	Description
`Grid<type> name(r, c);`	// initializes a grid of size r x c with elements of type 'type'
`Grid<type> name;`	// initializes a grid of size 0x0 // empty for now with elements of type 'type'
`g[r][c]` or `g.get(r, c)`	// element at rth row & cth col. Note - 0 based indexing
`g.fill(value)`	// sets every element of the grid to value 'value'


Members
Description
g.inBounds(r,c)
// true if g[r][c] is
// a valid element of
// the grid false
otherwise
3
4
g.inBounds(0,2) → false?
out of bound true?
"left"
g.numCols() or g.width()
g.numRows() or g.height()
g.resize(nRows, nCols)
// Resize g to
// have nCols
// & nRows. Discard

g[r][c] = value;

~~g[r][c].set(value);~~

g.set(r, c, value);

// set the value at (r,c) position to 'value'

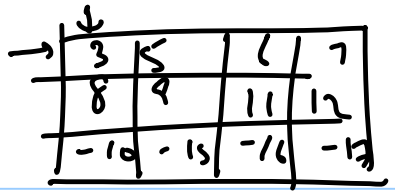
g.toString()

// { {1,2,4,5}, {3,5,6,7},
//   { 8,9,-10,15 } };

cout << g;

1	2	4	5
3	5	6	7
8	9	-10	15

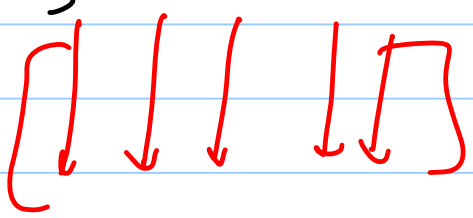
Looping over a grid:-




```
Grid<int> matrix = { {1, 2, 3},
                     {5, 6, 7},
                     {-10, -15, -17} };

int nrows = matrix.numRows();
int ncols = matrix.numCols();
// row-major
for (int i = 0; i < nrows; i++) {
    for (int j = 0; j < ncols; j++) {
        cout << matrix[i][j];
    }
}
```

```
Vector<int> nums
        = {1, 2, 5, 4, -16};
for (int i = 0; i < nums.
                    size(); i++) {
    cout << nums[i];
}
```



Grid<int> matrix = {{1,2,3,4}, {-10, -15, -16, -17},
{5, 9, 18, 0}};

```
for (int
// row-major fashion
for (int value: matrix) {
    cout << value << " ";
}
// goes in row-major order
```

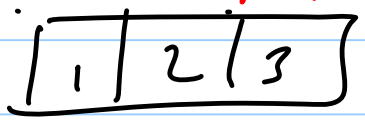
1 2 3 4 -10 -15 -16
-17 5 9 18 0

No access to corresponding position in matrix → you can't change matrix

```
for (int i=0; i<nRows; i++){
    for (int j=0; j<nCols; j++)
        cout << matrix[i][j] << " ";
}
```

```
Vector<int>
nums: {1,2,3};
for (int value: nums)
{   cout << value;
}
```

→ can't change nums



computeSum({{1, 2, 3}, {4, 5, 6}});

Grid as a parameter.

```
int computeSum(Grid<int> g) { ... }
```
inefficient

```
int computeSum(Grid<const int> &g) { }
```

```
int computeSum(const Grid<int> g) { }
```
best

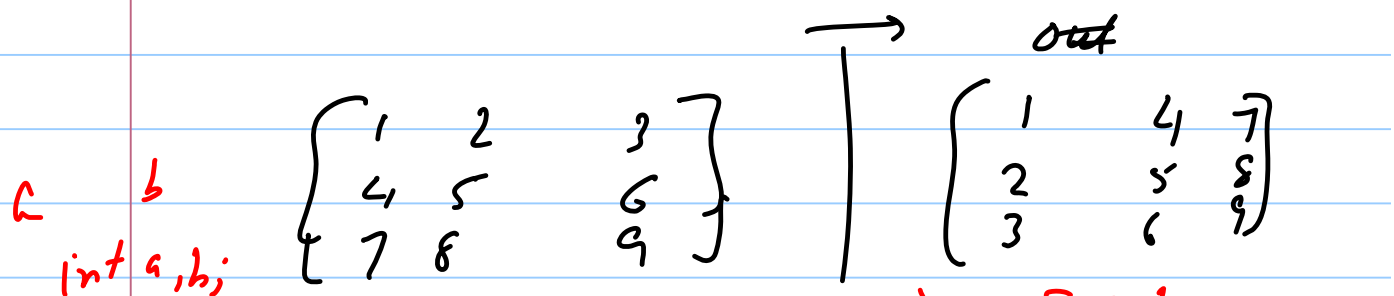
```
int computeSum(const Grid<int> &g) { }
```

Better programming practice → avoid unintended errors

Will this work?

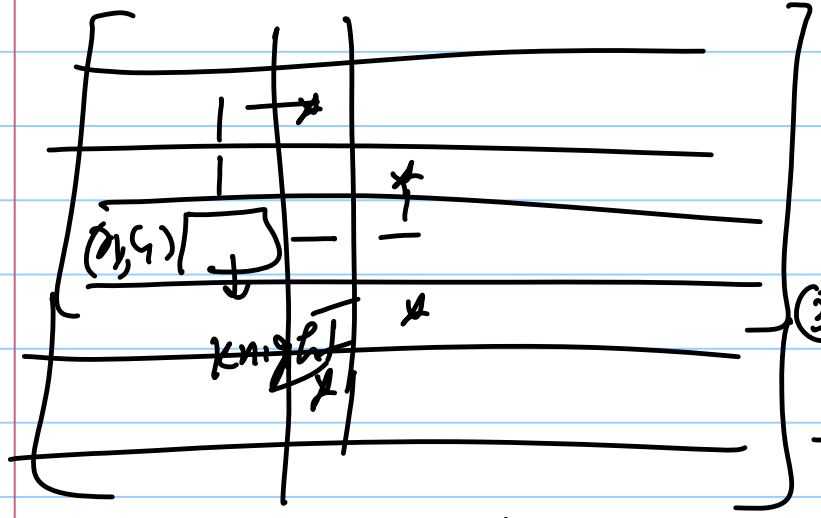
Copy Does not work

```
void transpose(Grid<int> g) { }
void transpose(Grid<int> &g) { }
void transpose(const Grid<int> g) { }
void transpose(const Grid<int> &g) { }
```




```
int a, b;
int tmp = a;
a = b; b = tmp;
```

g[i][j] ✗ g[j][i]



board.

① inBounds(r1, c1);

② board.inBounds(r2, c2);

③ r = abs(r1 - r2);
c = abs(c1 - c2);
→ if((r == 2 && c == 1) || (r == 1 && c == 2)

boot

bool knightCanMove(Grid<string> const &board,
int r1,
int c1,
int r2,
int c2) {