COL100 Lecture 15

Note Title 17-09-2018

Review:

template parameter

Vector<int>
Vector<char>
Vector<string>

member functions
eg. at, add, get, set, insert, remove, sublist

operators
eg. +=, [ ], <<

Grids eg. Grid<int> matrix(3, 4);

Grid <int> a = ⌜ ⌝

Vector <Vector <int>> b

Vector <int>

member functions

g.get(r,c) g[r][c] (would this work for vector of vectors? Yes!)	returns value at given row/col
g.fill(v)	set every cell to store given value
bool g.inBounds(r,c)	returns true if given position is in the grid
g.numCols() g.width()	# of columns
g.numRows() g.height()	# of rows
g.resize(nRows, nCols)	resizes grid to new size discarding old contents

g.set(r, c, value) g[r][c] = value	stores value at given row/col
g.toString()	returns string representation. "{ {1, 3}, {-3, 2} }"

cout << g

↓

stream operator

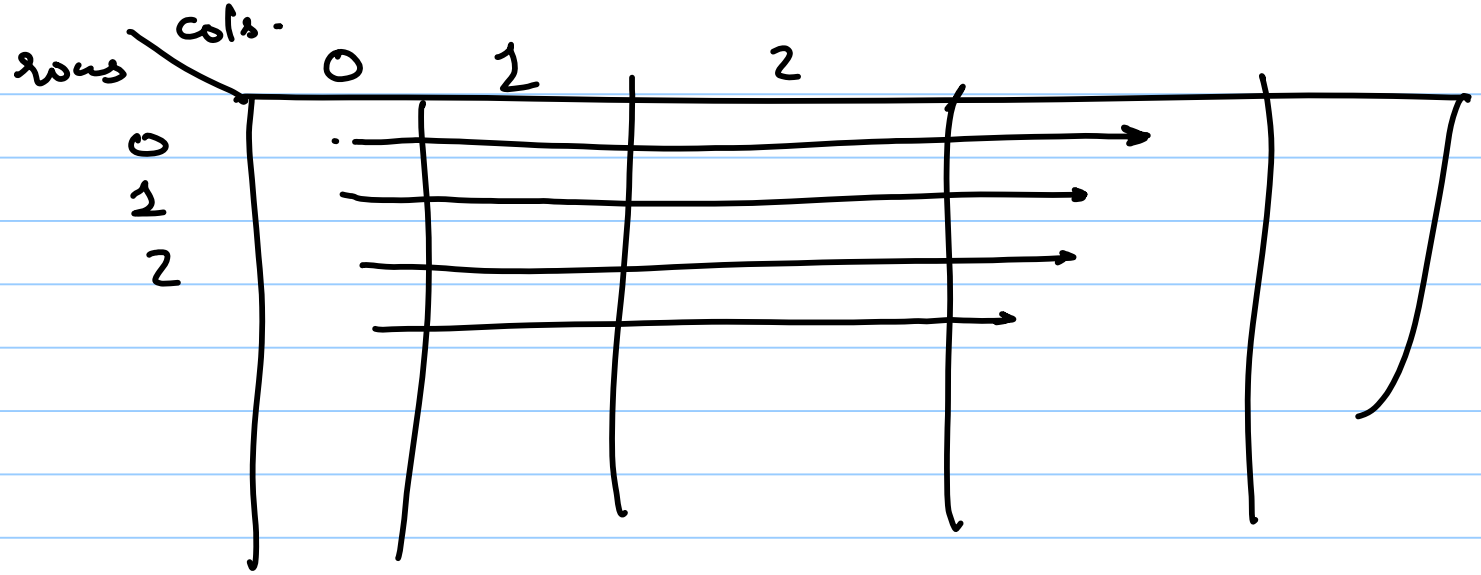
↓

cout << g.toString()

```
Grid<int> g; // 0 x 0
g.resize(5, 6); // 5 x 6
g.fill(0);
```

Looping over grids

```
for (int r=0; r < g.numRows(); r++)
{
    for (int c=0; c < g.numCols(); c++)
    {
        // do something with g[r][c]
        cout << g[r][c] << endl;
        g[r][c] = r+c;
    }
}
```

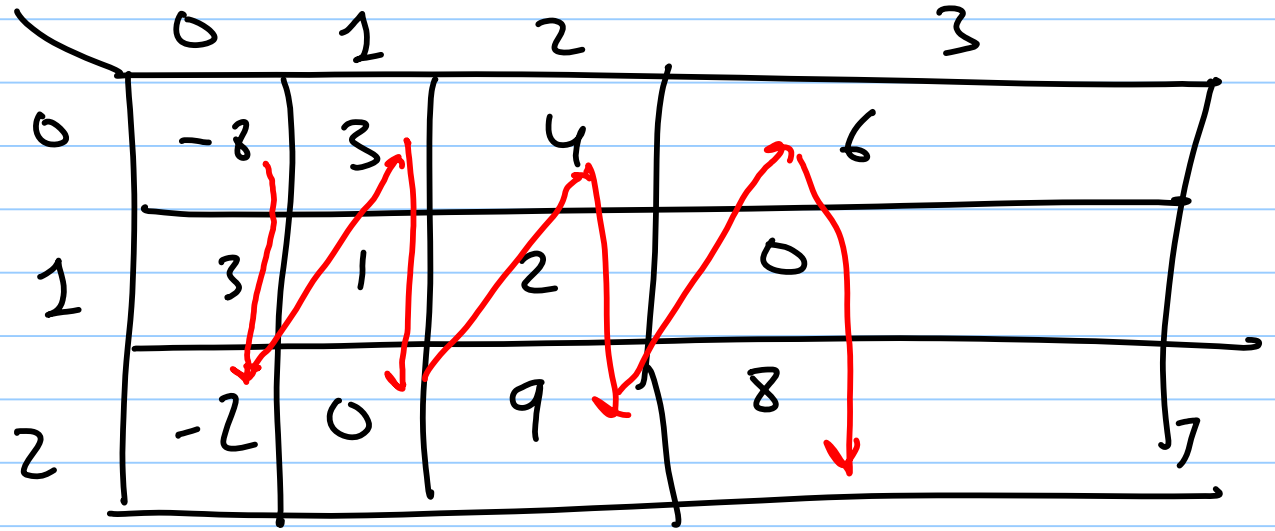


row-major order

Column-major

```
for (int c = 0; c < g.numCols(); c++)
{
    for (int r=0; r < g.numRows(); r++)
    {
        // do something with g[r][c]
        cout << g[r][c] << endl;

    }
}
```



8
3
-2
3
10

for-each loop

```
for ( int value : g )
{
        // do something with value
        // order is row-major
        cout << value << endl;

}
```

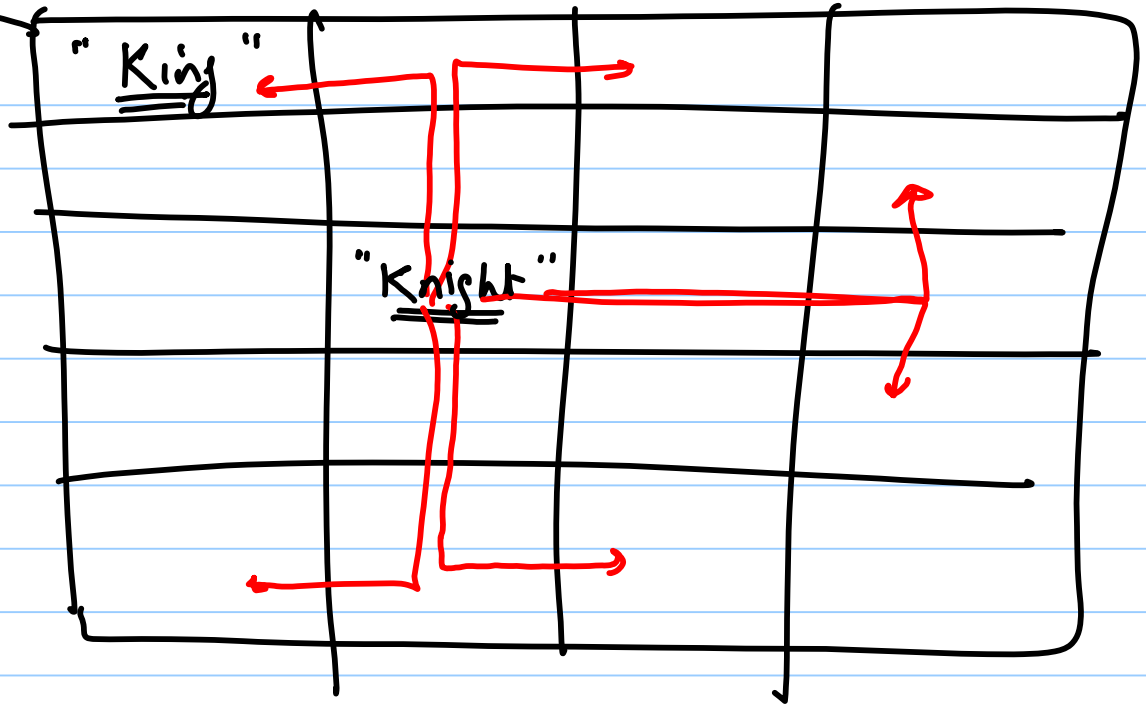
1. `int computeSum(Grid<int> g) { ... }`
2. `int computeSum(Grid<int> &g) { ... }`
3. `int computeSum(const Grid<int> &g) { ... }`

X 1. void invert (Grid<int> g)

✓ 2. void invert (Grid<int> &g)

X 3. void invert (Grid<int> const &g)

8x8



```
bool knightCanMove(Grid<string> const & board,
                   int r1,
                   int c1,
                   int r2,
                   int c2)
{
    int dr, dc;
    dr = absdiff(r1, r2);
    dc = absdiff(c1, c2);
    return board.inBounds(r1, c1)
        && board.inBounds(r2, c2)
        && board[r1][c1] == "knight"
        && board[r2][c2] == ""
        && (dr==2 && dc==1 || dr==1 && dc==2);
}
```

```
int absdiff(int a, int b)
{
    if (a < b)
    {
        return b-a;
    } else {
        return a-b;
    }
}
```

Grids as return values

```
Grid<int> invert(Grid<int> const &in)
{

    . . .

    return inverse;
}

Grid<int> inverted_matrix = invert(matrix);
```

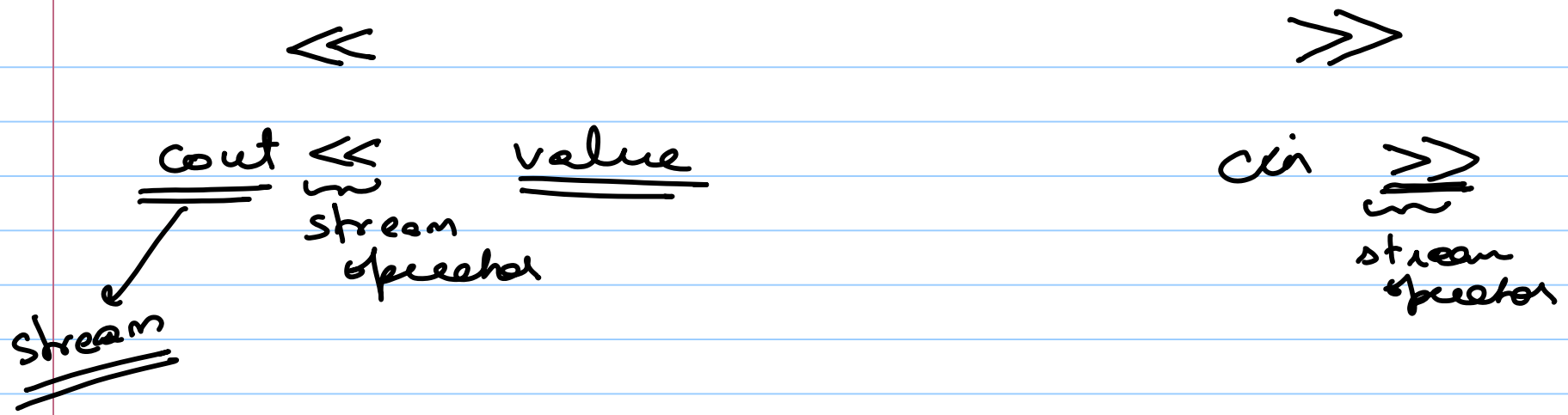
Streams

```
cout << 10
```

```
void addFive_1 (Grid<int> &g)
{
    for (int r=0; r< g.numRows(); r++)
    {
        for (int c=0; c<g.numCols(); c++)
        {
            g[r][c]+= 5 ;
        }
    }
}
```

```
Grid<int> addFive_2
    (const Grid<int> &g)
{
    Grid<int> ig(g.numRows(),
                 g.numCols());
    for (int r=0; r< g.numRows(); r++)
    {
        for (int c=0; c<g.numCols(); c++)
        {
            ig[r][c]= g[r][c]+5;
        }
    }
    return ig;
}
```

Streams



String stream

#include <sstream>

istringstream lets you read "tokens" from a string

ostringstream lets you write output tokens to a string buffer

```
istringstream input("abc def 123 4567");
                                  age  phone
string first, last, phone;
int age, ~~o phone~~;
int x;

input >> first;   // read the first token from
                  // input and store it in "first".
                  // first ← "abc"
input >> last;    // last ← "def"

input >> age;     // age ← 123
input >> phone;   // phone ← "4567"
(input >> x);     // no value gets written to x
```

tokenizing a string

```
istringstream input("abc def 123 456 hello world");

string word;

while ( input >> word )
{       // do something with word.
        cout << word << endl;
}
```