COL 100 Lecture 14

Note Title 13-09-2018

Review:

- Strings
  - member functions
  - general functions
  - operators
  - C strings

Today:

Vectors

Abstract Data Types (ADTs)

Also called "collections"

Collection: an object that stores data

also called a "data structure"

eg. Vector is a data structure for representing lists

use vector in C++ STL (standard template library)

Vector

```
#include "vector.h"

Vector<int> a = {3, -2, 0, 1};
```

type

```
Vector<string> b = {"hello", "world", ""};

int m = 10;
Vector<int> aa = {3, m, 2};

Vector<int> x = {};
Vector<int> y;
```

```
Vector<string> names;
names.add("abc"); // { "abc" }
names.add("xyz"); // { "abc", "xyz" }
```

add ↓

member function of Vector

string

vs

Vector<char>

Vector names; ✗

Vector<String> names;

```
names.add("abc");  // {"abc"}
names.add("ghi");  // {"abc", "ghi"}
                   //   0      1
names.insert(0, "def"); // {"def", "abc", "ghi"}
```

new member function

Vectors vs. arrays

What are arrays?

Also used to represent lists

Different syntax

More restrictive

```
int a[100]; // array of 100 integers
string s[100]; // array of 100 strings
char c[100]; // array of 100 characters
```

c[10] s[11]

Member function	
v.add(value) v += value v += value1, value2, ...valueN	append value(s) to v
v.clear()	removes all elements
v[i] or v.get(i)	get the value at index i
v.insert(i, value)	inserts a given value just before the given index shifting subsequent values to right
v.isEmpty()	returns true if vector contains no elements
e = v.remove(i)	remove value at index i shifting subsequent values to left. returns the removed value

v[i] = value v.set(i, value)	replaces value at given index
v.subList(start, length)	returns new vector of elements in subrange [start .. start + length - 1]
v.size()	number of elements in v
v.toString()	returns string representation. "{ 1, 30, -2, 7 }" or "{ "hello", "world" }"
cout << v (<< : operator)	print the string representation to cout

```
cout << v
```

same as

```
cout << v.toString()
```

✓

```
Vector<string> names = {"ab", "cd", "ef"};

for (int i = 0; i < names.size(); i++)
{
    cout << names[i] << endl;
}

for (int i = names.size()-1; i >= 0; i--)
{
    cout << names[i] << endl;
}
```

for-each loop over collections

```
for ( string name : names)
{
        cout << name << endl;

}

// "ab
//  cd
//  ef
```

```
string a;
for (a : names)
{
    // changing "a" has no effect on "names":
    cout << a << endl;
}
```

Cannot change names in the body of the for_each loop

value is copied from names to a

v=

3	8	9	7	5
0	1	2	3	4

v.insert (2, 42);

v=

3	8	42	9	7	5
0	1	2	3	4	5

v.remove(1);

or

int i = v.remove(1);

cout << i; // 8

v = {3, 42, 9, 7, 5}

Efficiency: slower as more elements need to shift

Grids

"grid.h"

2-D organizations

e.g. board games like chess

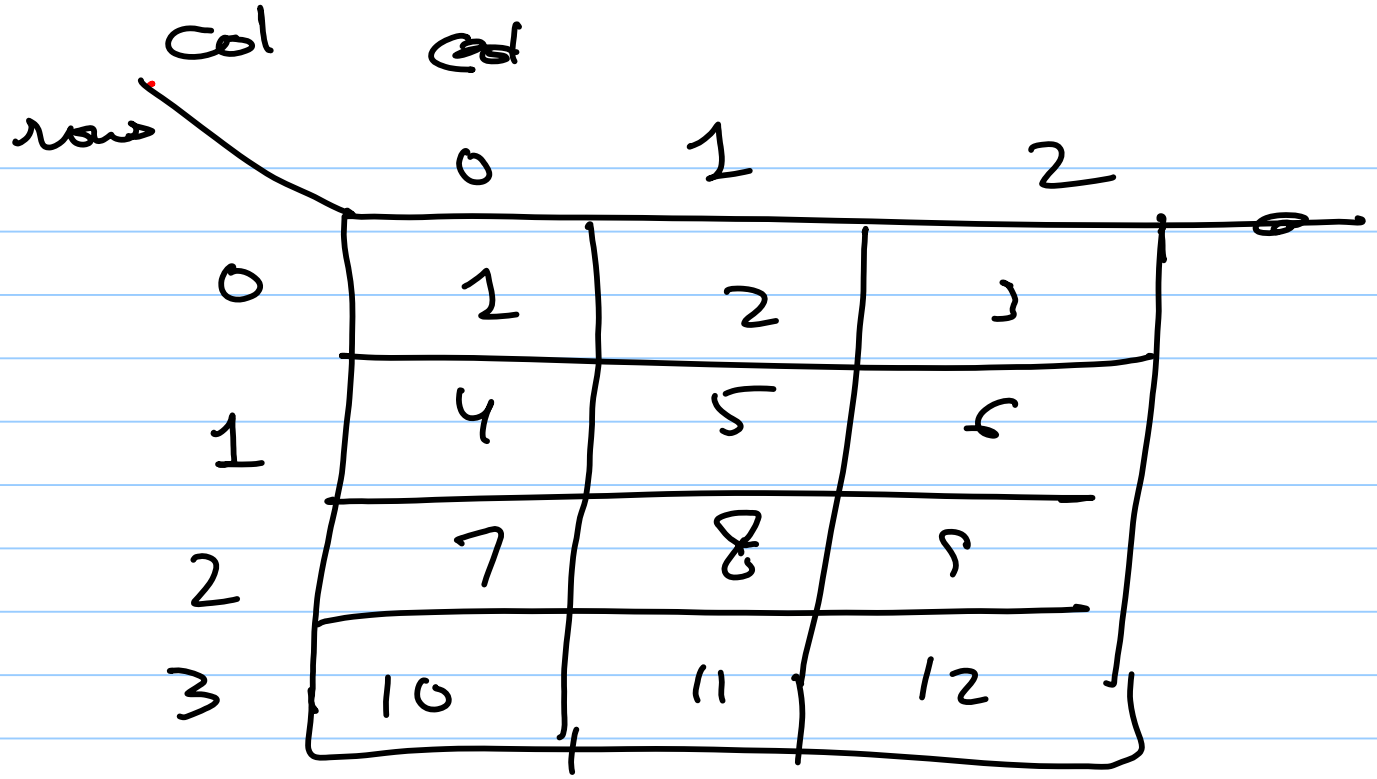
model matrices

specify element type using < >

type parameter or template parameter

template parameter or type parameter

```
Grid<int> matrix
    = {
        { 1, 2, 3},
        { 4, 5, 6 },
        { 7, 8, 9},
        { 10, 11, 12}
    };
```

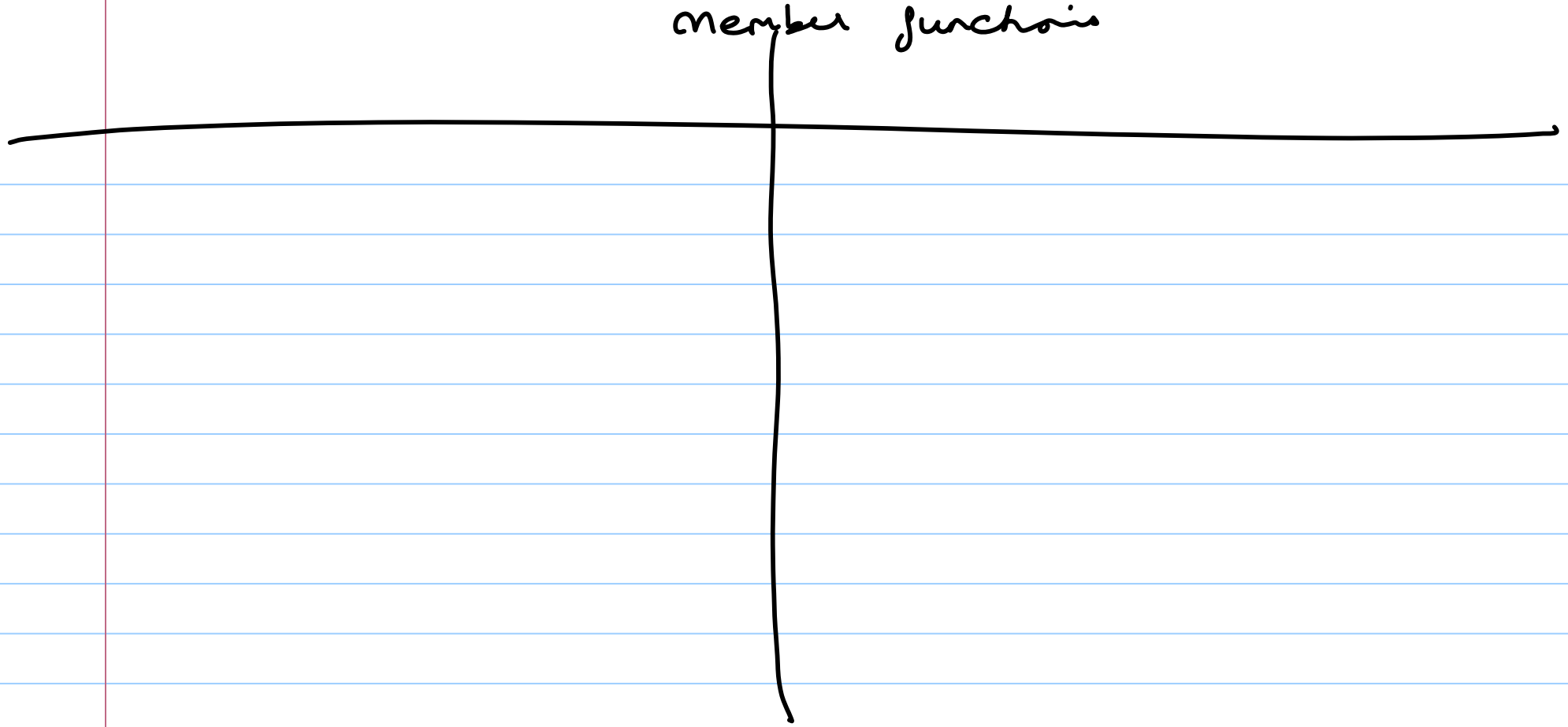

Col
Col
row
0
1
2
0
1
2
3
1
2
3
4
5
6
7
8
9
10
11
12

number of rows

number of columns

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

```
Grid<int> a; // 0 x 0 grid
```


member functions