COL100 Lecture 23

Note Title 27-10-2018

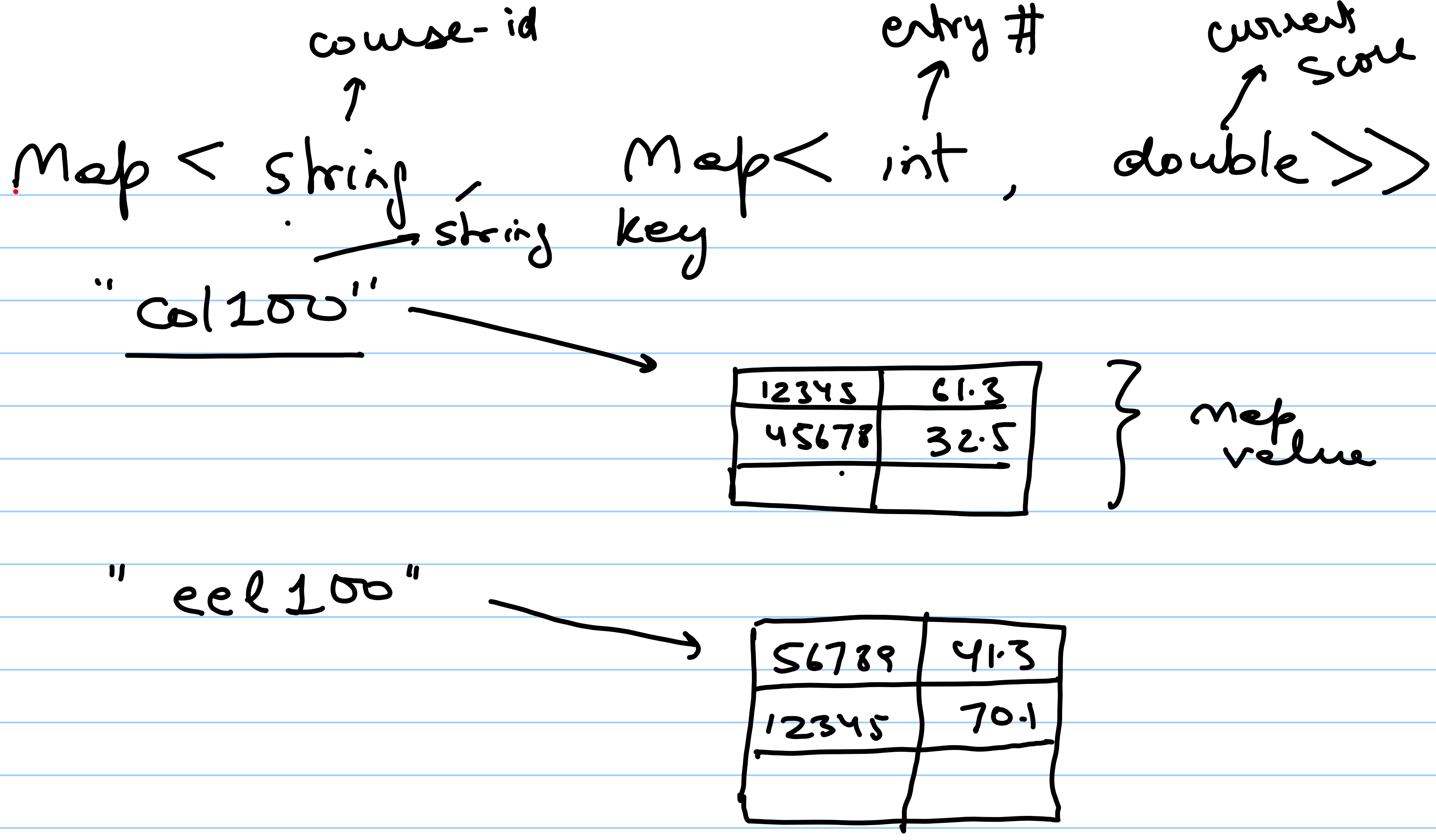
Review:

maps

```
m.put(key, value)
 m[key] = value
  m.get (key)      — error if key not present
   m[key]          — adds default value if key not present & returns it
    m.remove (key)
     m.containsKey(key)
```

Dictionary example:

```
map<string, string>   dict;
     ↓        ↓
    word    definition
```


course-id
entry #
current score
Map < string , Map< int , double>>
string key
"col100"
12345 | 61.3
45678 | 32.5
map value
"eel100"
56789 | 41.3
12345 | 70.1

Dictionary example

// first populate the map from file
using dict.add(word, meaning)

// now repeatedly give the
user a prompt to
enter a word
and query its meaning
from dict

Reverse dictionary

~~one~~ ~~one~~

multiple words can have same meaning

- Given a meaning (query by the user), print all the words ~~the~~ with that meaning.

for-each loop

name

phone number

```
Map <string , int>     phonebook ;

phonebook [" Director "] =  1234;
 phonebook [" Dean"] =  3456;
              :
string   name ;
for  ( name  :  phonebook)
{
         int   phonenumber =  phonebook[name];
         cout << name << " : " << phonenumber
                                   << endl;

}
```

Stanford
Library
Syntax

STL syntax

```
pair <string , int> name_phone;
for (name_phone : phonebook)
{
        string name = name_phone.first;
        int phonenumber = name_phone.second;

        cout << *name << ":" << phonenumbe
                                   << endl;

}
```

Find all words with given meaning

```
string query;
cin >> query;
string word;
for ( word : dict)
{
    string meaning = dict [word];
    if (meaning == query)
    {
        cout << word << endl;
    }
}
```

O(N logN)

O(logN) → string meaning = dict [word];

Word count

Given a file, how many times does each word occur in the file

in.txt:

```
to be or not to be
hello world
the world is not enough
i think therefore i am
```

Output:

```
am : 1                 or : 1
be : 2                 the : 1
enough : 1             therefore : 1
hello : 1              think : 1
i : 2                  to : 2
is : 1                 world : 2
not : 2
```

```
int     main()
{
            ifstream    in("int.txt");
            string   word;
             Map<string, int>    wc;
            while   ( in   >>   word)
            {
                   if   (! wc. containsKey(word))
                   {
                        wc[word]= 1
                   }  else  {
wc[word]++;  <-  { int    c  =  wc[word];    | -> easier
                    c++;
                    wc[word] =   c;
                   }
            }
            for  ( word : wc )
            {       int    c=    wc[word] ;
                     cout   <<  word  <<   ":" << c
                                                 << endl;
    }           return  0 ;
```

Ex. : Output [illegible] in increasing order of word frequency

```
am: 1
enough : 1
hello : 1
is : 1
or : 1
the : 1
therefore : 1
think : 1
be : 2
i : 2
not : 2
to : 2
world : 2
```

Forward map

"am" $\longrightarrow$ 1

"be" $\longrightarrow$ 2

"enough" $\longrightarrow$ 1

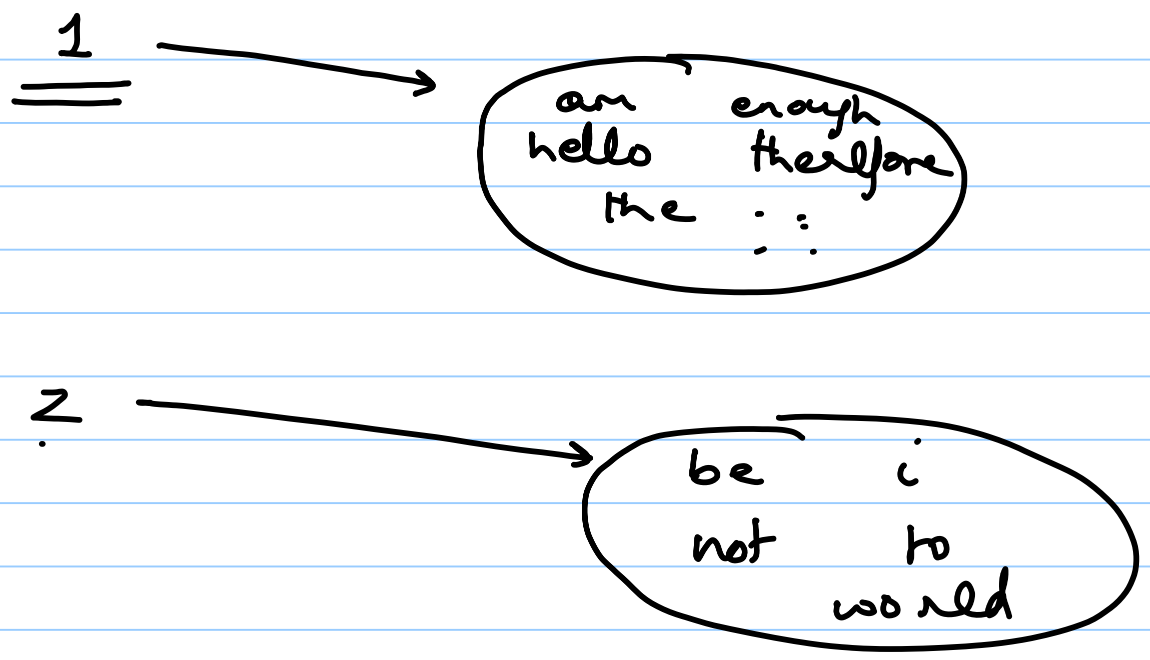
$\vdots$

Reverse map

$1 \longrightarrow \boxed{\begin{matrix}\text{"am"} \\ \text{"enough"}\end{matrix}}$

$2 \longrightarrow \boxed{\text{"be"}}$

If I had a map: Map<int, Set<string>> rwc;



```
map <string, int>  wc; //populated  already
Map < int ,  Set <string >>   rwc;
string   word;
for  ( word  :  wc )
{
        int   f =   wc[word];
        if   (!rwc . containsKey (f))
        {
              Set<string> v;;
             v. add (word);
        } else rwc[f] = v ;
        {
              Set <string> v ;
              v = rwc. get (f);
              v. add (word);
              rwc [f] = v;
        }
}
}
```

```
//print rwc
 int f;
for ( f : rwc)
{
        Set<string> words = rwc.get(f);
        string word;
        for (word : words)
        {
              cout << word << " : "
                                   << f <<endl;
         }

  }
```

```
Map <int, Vector <string >> awc;
```

is this a good idea ??

Dinner with Friends

Friends have preferences on where to go

choose venue to Maximize satisfaction of everyone:

E.g.

ram
Staff canteen
Rainbows

geeta
Govardhan

shyam
Rajdhani Chatkare
Govardhan
Rainbows

ADTs

$\text{Map} < \overset{\text{friend name}}{\text{string}}, \quad \text{Set} < \overset{\text{restaurants}}{\text{string}} >> \quad \text{pref};$

$\text{Map} < \overset{\text{restaurant name}}{\text{string}}, \quad \overset{\text{\# of votes}}{\text{int}} > \quad \text{rvotes};$

<u>Homework</u>