

# CS 106X

## Lecture 6: Sets and Maps

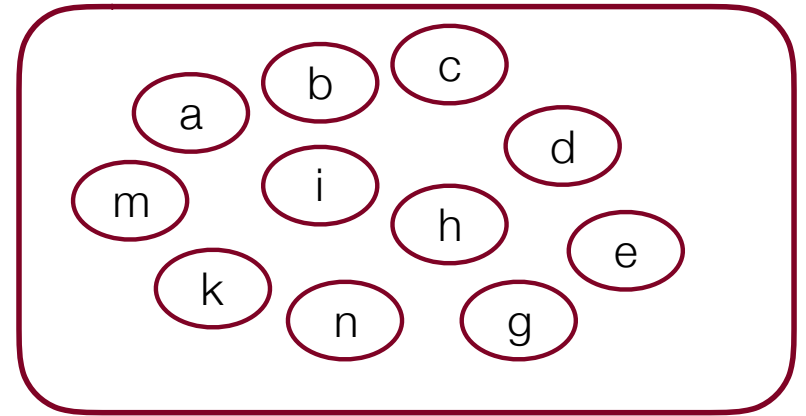
Monday, January 23, 2017

Programming Abstractions (Accelerated)  
Winter 2017  
Stanford University  
Computer Science Department

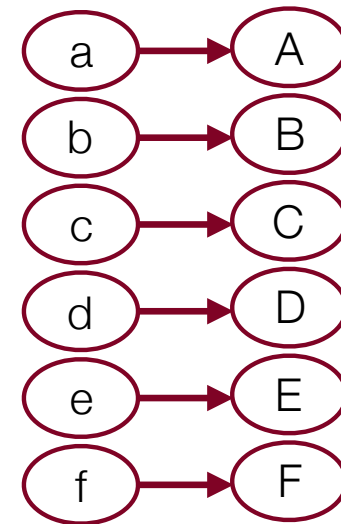
Lecturer: Chris Gregg

reading:  
Programming Abstractions in C++, Chapter 5.4-5.6

Set:



Map:



# Today's Topics

- Logistics:
  - Tiny Feedback: a few responses!
  - Better references to Stanford Library material ✓
  - Special cases of Big O: we will get there, and feel free to ask on Piazza, in section, in class, etc.
  - More audience participation: will do!
- ADTs Due Friday, January 27th, noon
- One submission of three files (wordLadder, Ngrams, and TranspositionCipher)
- Postfix refresher
- Structs (details will come later!)
- Sets
- Maps



# Assignment 2: ADTs

- How is it going?



# Postfix (RPN) Refresher

What does the following postfix (RPN) computation equal?

**10 3 5 \* 9 4 - / +**

Feel free to use our stack algorithm:

**Read the input and push numbers onto a stack until you reach an operator.  
When you see an operator, apply the operator to the two numbers that are popped from the stack.  
Push the resulting value back onto the stack.  
When the input is complete, the value left on the stack is the result.**

Answer: **13**

How would our stack-based RPN know that we had made an error, e.g.,

**10 3 5 \* - + 9 4 -**

Answer: the stack is empty when we try to pop two operands



# Brief Introduction to Structs

Recall that in C++, we can only return one value from a function. We have overcome this in the past by using references:

```
/*  
 * Solves a quadratic equation  $ax^2 + bx + c = 0$ ,  
 * storing the results in output parameters root1 and root2.  
 * Assumes that the given equation has two real roots.  
 */  
void quadratic(double a, double b, double c,  
               double& root1, double& root2) {  
    double d = sqrt(b * b - 4 * a * c);  
    root1 = (-b + d) / (2 * a);  
    root2 = (-b - d) / (2 * a);  
}
```

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



# Brief Introduction to Structs

- There is another way we can return variables by packaging them up in a type called a "struct"
- Structs are a way to *define a new type* for us to use.
- Once we define a struct, we can use that type anywhere we would normally use another type (e.g., an `int`, `double`, `string`, etc.)

```
struct Roots {  
    double root1;  
    double root2;  
};
```

new type name

struct variables,  
referred to with dot  
notation

don't forget the semicolon



# Brief Introduction to Structs

- Let's re-write our quadratic equation solver to use the Roots struct.



# Brief Introduction to Structs

- Let's re-write our quadratic equation solver to use the Roots struct.

```
struct Roots {
    double root1;
    double root2;
};

/*
 * Solves a quadratic equation  $ax^2 + bx + c = 0$ ,
 * storing the results in output parameters root1 and root2.
 * Assumes that the given equation has two real roots.
 */
Roots quadratic(double a, double b, double c) {
    Roots roots;
    double d = sqrt(b * b - 4 * a * c);
    roots.root1 = (-b + d) / (2 * a);
    roots.root2 = (-b - d) / (2 * a);
    return roots;
}
```

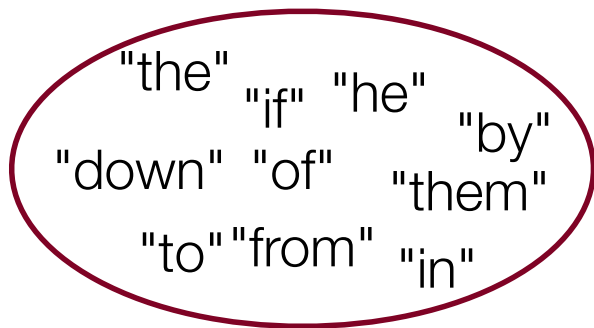




# Sets and Maps

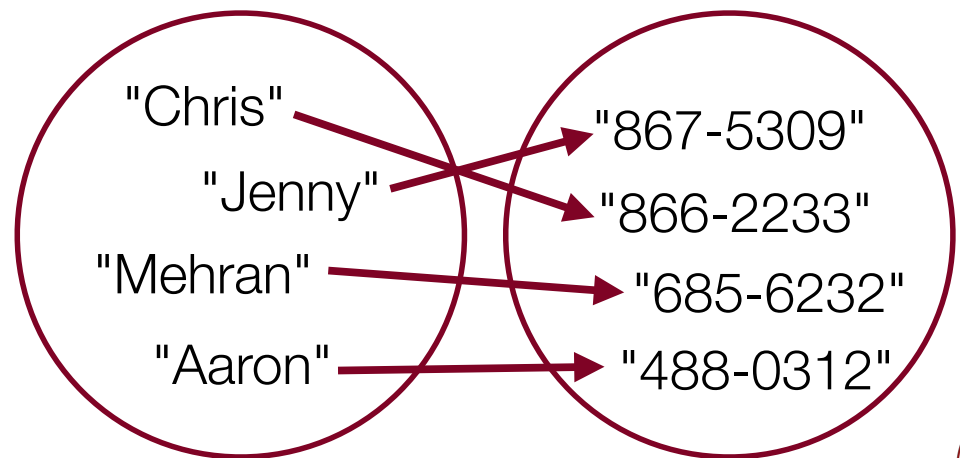
## Sets

- Collection of elements with *no duplicates*.



## Maps

- Collection of key/value pairs
- The key is used to find its associated value.

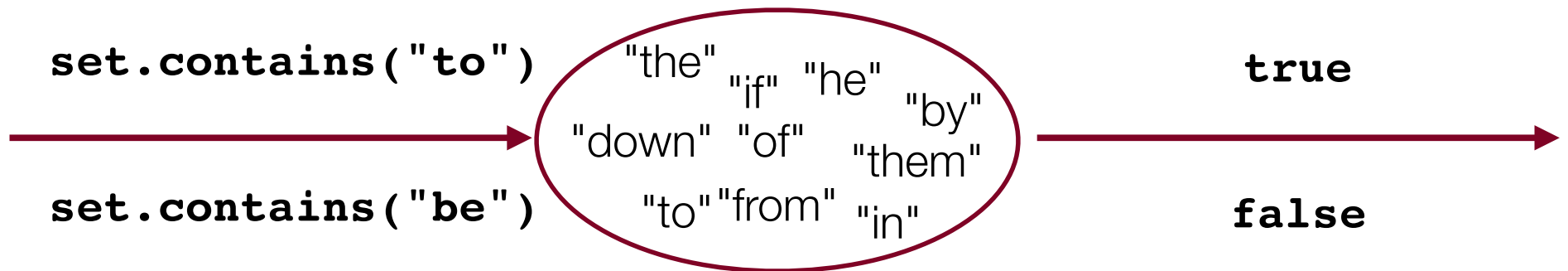


# Sets

- **set**: a collection of elements with no duplicates.

Add, contains, remove operation are all fast

We don't think of sets as having indices



## Sets: Simple Example

```
Set<string> friends;  
friends.add("chris");  
friends.add("aaron");  
cout << friends.contains("voldemort") << endl;  
for(string person : friends) {  
    cout << person << endl;  
}
```



# Set Essentials

**int set.size()**

Returns the number of elements in the set.

**void set.add(value)**

Adds the new value to the set (ignores it if the value is already in the set)

**bool set.contains(value)**

Returns `true` if the value is in the set, `false` otherwise.

**void set.remove(value)**

Removes the value if present in the set. Does not return the value.

**bool set.isEmpty()**

Returns `true` if the set is empty, `false` otherwise.

Sets also have other helpful methods. See the online docs for more.



# Looping Over a Set

```
for(type currElem : set) {  
    // process elements one at a time  
}
```

can't use a normal for loop and get each element [i]

```
for(int i=0; i < set.size(); i++) {  
    // does not work, no index!  
    cout << set[i];  
}
```



# Types of Sets

## Set

Iterate over elements in  
*sorted* order

**REALLY FAST!**

$O(\log n)$  per  
retrieval

Implemented using a  
"binary search tree"

## HashSet

Iterate over elements in  
*unsorted (jumbled)* order

**REALLY,  
RIDICULOUSLY FAST!**

$O(1)$  per retrieval

Implemented using a  
"hash table"



# Set Operands

Sets can be compared, combined, etc.

**s1 == s2**

true if the sets contain exactly the same elements

**s1 != s2**

true if the sets don't contain the same elements

**s1 + s2**

returns the union of s1 and s2 (all elements in both)

**s1 \* s2**

returns intersection of s1 and s2 (elements must be in both)

**s1 - s2**

returns difference of s1, s2 (elements in s1 but not s2)



# Set Operands

Sets can be compared, combined, etc.

**s1 == s2**

true if the sets contain exactly the same elements

**s1 != s2**

true if the sets don't contain the same elements

**s1 + s2**

returns the union of s1 and s2 (all elements in both)

**s1 \* s2**

returns intersection of s1 and s2 (elements must be in both)

**s1 - s2**

returns difference of s1, s2 (elements in s1 but not s2)





# Count Unique Words

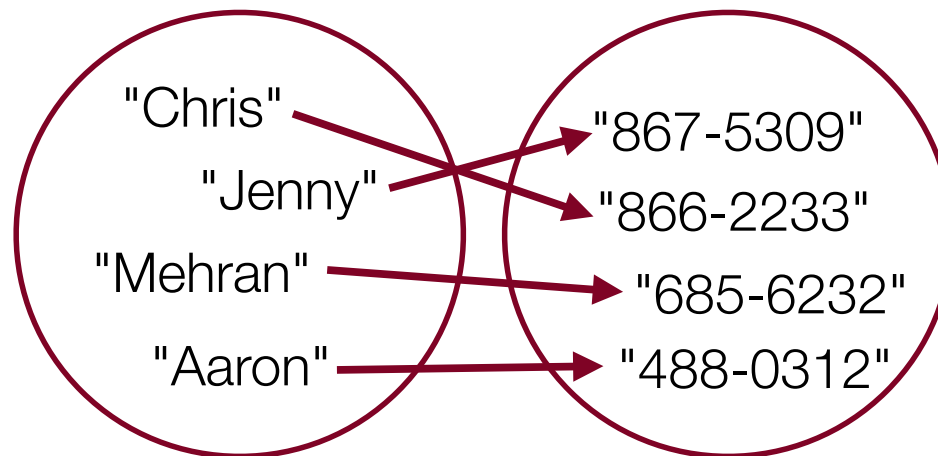


# Maps

**map:** A collection of pairs  $(k, v)$ , sometimes called key/value pairs, where  $v$  can be found quickly if you know  $k$ .

a.k.a. dictionary, associative array, hash

a generalization of an array, where the "indexes" need not be ints.



# Using Maps

A map allows you to get from one half of a pair to the other.

store an association from "Jenny" to "867-5309"

```
//      key      value  
// m["Jenny"] = "867-5309"; or  
m.put("Jenny", "867-5309");
```



Map

What is Jenny's number?

```
// string ph = m["Suzy"] or  
string ph = m.get("Suzy")  
"206-685-2181"
```



Map



# Maps are Everywhere

key = title, value = article

key:  
"Yosemite National Park"

value:

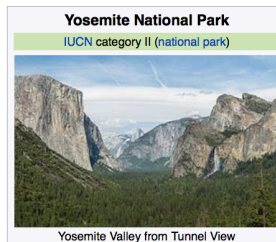
## Yosemite National Park

From Wikipedia, the free encyclopedia

Coordinates: 37°51′N 119°33′W﻿ / ﻿37.85°N 119.55°W﻿ / 37.85; -119.55<sup>[1]</sup>

*"Yosemite" redirects here. For other uses, see Yosemite (disambiguation).*

**Yosemite National Park** (/joʊˈseɪmɪti/ *yoh-SEM-it-ee*<sup>[4]</sup>) is a national park spanning portions of Tuolumne, Mariposa and Madera counties in Northern California.<sup>[5][6]</sup> The park, which is managed by the National Park Service, covers an area of 747,956 acres (1,168,681 sq mi; 302,687 ha; 3,026.87 km<sup>2</sup>)<sup>[2]</sup> and reaches across the western slopes of the Sierra Nevada mountain range.<sup>[7]</sup> About 4 million people visit Yosemite each year;<sup>[3]</sup> most spend the majority of their time in the seven square miles (18 km<sup>2</sup>) of Yosemite Valley.<sup>[8]</sup> Designated a World Heritage Site in 1984, Yosemite is internationally recognized for its granite



key:  
"Mariana Trench"

value:

## Mariana Trench

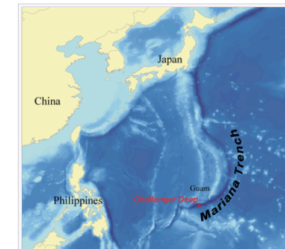
From Wikipedia, the free encyclopedia

Coordinates: 11°21′N 142°12′E﻿ / ﻿11.35°N 142.2°E﻿ / 11.35; 142.2

*This article is about the deep sea trench. For the Canadian band named after the geological formation, see Marianas Trench (band).*

The **Mariana Trench** or **Marianas Trench**<sup>[1]</sup> is the deepest known part of the world's oceans. It is located in the western Pacific Ocean, to the east of the Mariana Islands. The trench is about 2,550 kilometres (1,580 mi) long with an average width of 69 kilometres (43 mi). It reaches a maximum-known depth of 10,994 metres (36,070 ft) (± 40 metres [130 ft]) at a small slot-shaped valley in its floor known as the Challenger Deep, at its southern end,<sup>[2]</sup> although some unrepeatable measurements place the deepest portion at 11,034 metres (36,201 ft).<sup>[3]</sup>

At the bottom of the trench the water column shows



# Creating Maps

Requires 2 type parameters: one for keys, one for values.

```
// maps from string keys to integer values
```

```
Map<string, int> votes;
```

```
// maps from double keys to Vector<int> values
```

```
Map<string, Vector<string>> friendMap;
```



# Map Methods

<code>m.clear();</code>	removes all key/value pairs from the map
<code>m.containsKey(<i>key</i>)</code>	returns true if the map contains a mapping for the given key
<code>m[<i>key</i>]</code> or <code>m.get(<i>key</i>)</code>	returns the value mapped to the given key; if key not found, <b>adds</b> it with a default value (e.g. 0, "")
<code>m.isEmpty()</code>	returns true if the map contains no k/v pairs (size 0)
<code>m.keys()</code>	returns a Vector copy of all keys in the map
<code>m[<i>key</i>] = <i>value</i>;</code> or <code>m.put(<i>key</i>, <i>value</i>);</code>	adds a mapping from the given key to the given value; if the key already exists, <b>replaces</b> its value with the given one
<code>m.remove(<i>key</i>);</code>	removes any existing mapping for the given key
<code>m.size()</code>	returns the number of key/value pairs in the map
<code>m.toString()</code>	returns a string such as "{a:90, d:60, c:70}"
<code>m.values()</code>	returns a Vector copy of all values in the map



# Map Example

```
Map<string, string> wiki;
```

```
// adds name / text pair to dataset
```

```
wiki.put("Neopalpa donaldtrumpi", articleHTML);
```

SCIENCE JAN 17 2017, 9:43 PM ET

## Tiny Moth Named for President-Elect Donald Trump

by MINDY WEISBERGER, LIVE SCIENCE

SHARE



Moth species *Neopalpa donaldtrumpi*. © Vazrick Nazari



# Map Example

```
Map<string, string> wiki;
```

```
// adds name / text pair to dataset  
wiki.put("Neopalpa donaldtrumpi", articleHTML);
```

```
// returns corresponding articleHTML  
cout << wiki.get("Yosemite National Park");
```

## Yosemite National Park

From Wikipedia, the free encyclopedia

Coordinates: 37°51′N 119°33′W﻿•﻿37°51′N 119°33′W﻿•﻿37°51′N 119°33′W

*"Yosemite" redirects here. For other uses, see [Yosemite \(disambiguation\)](#).*

**Yosemite National Park** (/joʊˈseɪmɪti/ *yoh-SEM-it-ee*<sup>[d]</sup>) is a *national park* spanning portions of Tuolumne, Mariposa and Madera counties in Northern California.<sup>[5][6]</sup> The park, which is managed by the *National Park Service*, covers an area of 747,956 acres (1,168.681 sq mi; 302,687 ha; 3,026.87 km<sup>2</sup>)<sup>[2]</sup> and reaches across the western slopes of the *Sierra Nevada* mountain range.<sup>[7]</sup> About 4 million people visit Yosemite each year.<sup>[3]</sup> most spend the majority of their time in the seven square miles (18 km<sup>2</sup>) of *Yosemite Valley*.<sup>[8]</sup> Designated a *World Heritage Site* in 1984, Yosemite is internationally recognized for its *granite*





# Map Example

```
Map<string, string> wiki;  
  
// adds name / text pair to dataset  
wiki.put("Neopalpa donaldtrumpi", articleHTML);  
  
// returns corresponding articleHTML  
cout << wiki.get("Yosemite National Park");  
  
// removes the article  
wiki.remove("Britain in the E.U.");
```



# Types of Maps

## Map

Iterate over elements in  
*sorted* order

**REALLY FAST!**

$O(\log n)$  per  
retrieval

Implemented using a  
"binary search tree"

## HashMap

Iterate over elements in  
*unsorted (jumbled)* order

**REALLY,  
RIDICULOUSLY FAST!**

$O(1)$  per retrieval

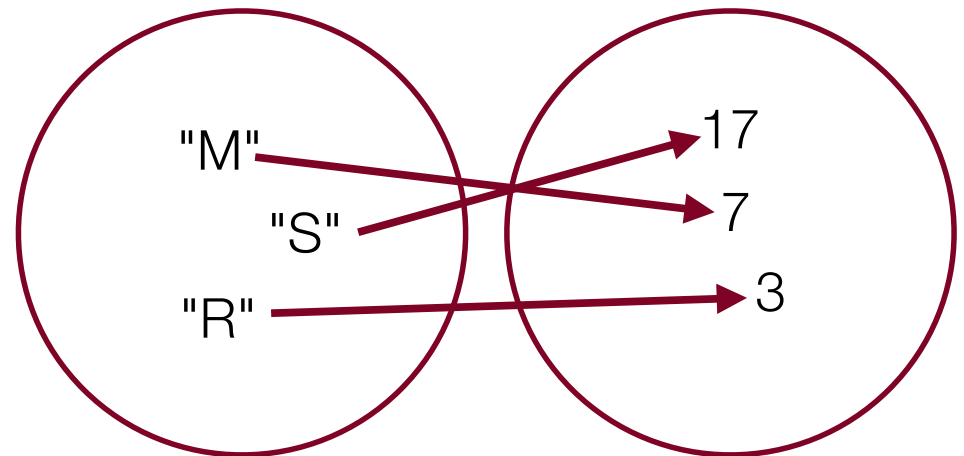
Implemented using a  
"hash table"



# Map Example: Tallying Votes

count votes:  
// (M)ilk, (S)tokes, (R)ogers  
"MMRMSSMSSMMMMRRMMMMRRRMMM"

key:	"M"	"S"	"R"
value:	17	7	3



\*In 1976 Harvey Milk became the first openly gay elected official in the US



# Tallying Words



## Looping Over a Map

```
Map<string, double> gpa = load();  
for (string name : gpa) {  
    cout << name << "'s GPA is ";  
    cout << gpa[name] << endl;  
}
```

\*The order is unpredictable in a HashMap



# Recap

## •Structs

- Used to define a type that holds multiple other types.
- Useful for returning more than one value, or keeping things together (e.g., a coordinate could be an x,y and it is nice to keep them together:

```
struct coordinate {  
    double x,y;  
}
```

- Uses dot notation to access elements.

## •Sets:

- Container that holds non-duplicate elements
- $O(\log n)$  behavior per element access (**HashSet**:  $O(1)$ , but unordered)

## •Map:

- Container that relates keys to values.
- Needs two types when defining: `Map<keyType, valueType>`
- $O(\log n)$  behavior per element access (**HashMap**:  $O(1)$ , but unordered)



# References and Advanced Reading

- **References:**

- Stanford Set reference: <http://stanford.edu/~stepp/cppdoc/Set-class.html>
- Stanford Map reference: [stanford.edu/~stepp/cppdoc/Map-class.html](http://stanford.edu/~stepp/cppdoc/Map-class.html)

- **Advanced Reading:**

- Hashing: [https://en.wikipedia.org/wiki/Hash\\_table](https://en.wikipedia.org/wiki/Hash_table)
- Relational Databases: [https://en.wikipedia.org/wiki/Relational\\_database](https://en.wikipedia.org/wiki/Relational_database) (especially idencies)



# Extra Slides

