Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session : Template Class “map”

Quick Recap of Relevant Topics

IIT Bombay

- Object-oriented programming with structures and classes
- Template classes and functions
- C++ Standard Library
  - The “string” class
  - The “vector” class

Overview of This Lecture

IIT Bombay

- The template class “map”

Acknowledgment

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

The “map” class

IIT Bombay

- For representing **associative one-dimensional arrays/vectors**
  - Arrays in which index is not necessarily an integer
  - Indices are objects of specified types, e.g. string, V3, …
  - Example usage: marks[“Shyam”], position[“BA207”]

Array of double values Indexed by strings

Array of V3 values (3-dimensional vectors) Indexed by strings

- Notation: **key** (or **index**) and **value**

  Key type and value type are completely independent

  Key values must be ordered by (overloaded) < operator

The “map” class

IIT Bombay

- For representing associative one-dimensional arrays/vectors
  - Template class : Can be instantiated with key type and value type
  - Internally, elements stored as (key, value) pairs and sorted by key
  - Internal representation: binary search tree
  - Dynamic memory management built in
- “**map**” objects are container objects
- Must use **#include <map>** at start of program
- Large collection of member functions
  - We’ll see only a small subset

The “pair” Template Class in Maps

IIT Bombay

- C++ Standard Library provides a template class

  pair<T1, T2> with two data members named

  first (of type T1) and **second (of type T2)**

- Every **map<key_type, value_type>** object is really a collection of (key, value) pairs stored as **pair<key_type, value_type>** objects

Simple Programming using "map"

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  string stName; double stMarks;
  while (true) {
      cout << "Give name of student: ";  cin >> stName;
      if (stName == "end") {cout << "Bye!!!" << endl; break;}
      else { cout << "Give marks: "; cin >> stMarks;   marks[stName] = stMarks; }
   }
   … Some other code …
}
```

Key type: string
Value type: double

Name of map

Simple Programming using "map"

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  string stName; double stMarks;
  while (true) {
      cout << "Give name of student: ";  cin >> stName;
      if (stName == "end") {cout << "Bye!!!" << endl; break;}
      else { cout << "Give marks: "; cin >> stMarks;   marks[stName] = stMarks; }
   }
   … Some other code …
}
```

Create an empty map of (string, double) pairs

Writing/Inserting "map" Elements

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  string stName; double stMarks;
  while (true) {
      cout << "Give name of student: ";  cin >> stName;
      if (stName == "end") {cout << "Bye!!!" << endl; break;}
      else { cout << "Give marks: "; cin >> stMarks;   marks[stName] = stMarks; }
   }
   … Some other code …
}
```

Accessing an element indexed by string

Writing/Inserting "map" Elements

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  string stName; double stMarks;
  while (true) {
      cout << "Give name of student: ";  cin >> stName;
      if (stName == "end") {cout << "Bye!!!" << endl; break;}
      else { cout << "Give marks: "; cin >> stMarks;   marks[stName] = stMarks; }
   }
   … Some other code …
}
```

If (key, value) pair with key matching stName does not exist in marks, create a new (stName, stMarks) pair and add to marks

Writing/Inserting "map" Elements

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  string stName; double stMarks;
  while (true) {
      cout << "Give name of student: ";  cin >> stName;
      if (stName == "end") {cout << "Bye!!!" << endl; break;}
      else { cout << "Give marks: "; cin >> stMarks;   marks[stName] = stMarks; }
   }
   … Some other code …
}
```

If (key, value) pair with key matching stName already exists in marks, update the value of this pair to stMarks and erase the previous value

Over-writing "map" Elements

IIT Bombay

- What happens if we execute the following code ?

```
marks[stName] = stMarks;
```

Insert/update (key, value) pair

```
marks[stName] = stMarks + 10;
```

Update value in (key, value) pair

Reading "map" Elements

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  string stName; double stMarks;
  … Some other code …
  while (true) {
      cout << "Give name of student: ";  cin >> stName;
      if (stName == "end") {cout << "Bye!!!" << endl; break;}
      else { cout << "Marks of " << stName << " is: " << marks[stName] << endl; }
   }
  return 0;}
```

Accessing an element indexed by string

Gives value associated with key stName

Reading "map" Elements

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  string stName; double stMarks;
  … Some other code …
  while (true) {
      cout << "Give name of student: ";  cin >> stName;
      if (stName == "end") {cout << "Bye!!!" << endl; break;}
      else { cout << "Marks of " << stName << " is: " << marks[stName] << endl; }
   }
  return 0;}
```

What if stName is "Abdul" but no (key, value) pair with key matching "Abdul" exists in marks?

Reading "map" Elements

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
      else { cout << "Marks of " << stName << " is: " << marks[stName] << endl; }
  }
 return 0;}
```

What if stName is "Abdul" but no (key, value) pair with key matching "Abdul" exists in marks?

A new (key, value) pair is created with key set to "Abdul" and value obtained from default constructor of value type (here, double)

Reading "map" Elements

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
```

What if stName is "Abdul" but no (key, value) pair with key matching "Abdul" exists in marks?

A new (key, value) pair is created with key = "Abdul" and value = value returned by default constructor of double

```
    else { cout << "Marks of " << stName << " is: " << marks[stName] << endl; }
  }
ret
```

Garbage: value returned by default constructor of double

Accessing Elements using at

- Like vectors, we can use

  marks.at(stName) instead of **marks[stName]**

If stName doesn't match the key of any (key, value) pair in marks, an out_of_range exception is thrown

Comparing Key Values

IIT Bombay

- (key, value) pairs are stored **sorted** by key values

  Requires a comparison operator for key values

  Preferable: operator< defined for key type

  map<string, double> marks;

operator< already defined in string class:
Lexicographic (dictionary) order
"Abdul" < "Ajanta" < "Bobby"

Comparing Key Values

IIT Bombay

- (key, value) pairs are stored **sorted** by key values

  Preferable: operator< defined for key type

What if operator< is not pre-defined for key type?

- **Custom-define operator< function (operator overloading)**

  bool operator<(key_type &a, key_type &b) { … }

  Must ensure that < is transitive and anti-symmetric

a < b and b < c implies a < c

a < b implies b ≮ a

Comparing Key Values

IIT Bombay

- (key, value) pairs are stored **sorted** by key values

  Preferable: operator< defined for key type

What if operator< is not pre-defined for key type?

- **Custom-define operator< function (operator overloading)**

  bool operator<(key_type &a, key_type &b) { … }

  Must ensure that < is transitive and anti-symmetric

  Must ensure that every pair of distinct keys are ordered

a and b distinct implies either a < b or b < a

Comparing Key Values

IIT Bombay

- (key, value) pairs are stored **sorted** by key values

  Preferable: operator< defined for key type

What if operator< is not pre-defined for key type?

- Custom-define operator< function (operator overloading)

  bool operator<(key_type &a, key_type &b) { … }

  Must ensure that < is transitive and anti-symmetric

  Must ensure that every pair of distinct keys are ordered
- **Custom-define separate comparison function and indicate in map declaration**

  Won't cover in our discussions ….

Iterator Related Functions in "map" Class

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  marks["Ajanta"] = 10.0;  marks["Bobby"] = 15.0;  marks["Abdul"] = 25.0;
  map<string, double>::iterator it;
  for (it = marks.begin(); it != marks.end(); it++)  {
      cout << it->first << " : " << it->second << endl;
  }
  return 0;
}
```

begin(), end()
member functions

Recall: elements of map are (key, value) pairs

Abdul: 25.0
Ajanta : 10.0
Bobby : 15.0

Iterator Related Functions in "map" Class

IIT Bombay

```
#include <iostream>
#include <map>
using namespace std;
int main() {
  map<string, double> marks;
  marks["Ajanta"] = 10.0;  marks["Bobby"] = 15.0;  marks["Abdul"] = 25.0;
  map<string, double>::reverse_iterator rit;
  for (rit = marks.rbegin(); rit != marks.rend(); rit++)  {
      cout << rit->first << " : " << rit->second << endl;
  }
  return 0;
}
```

rbegin(), rend()
member functions

Bobby: 15.0
Ajanta : 10.0
Abdul: 25.0

Finding if an Element Exists in a Map

IIT Bombay

```
int main() {
  map<string, double> marks;  string stName;
  … Some other code …
  while (true) {
    cout << “Give name of student: “;  cin >> stName;
    if (stName == “end”) {cout << “Bye!!!” << endl; break;}
    else { if (marks.count(stName) > 0) {cout << “Marks: “ << marks[stName] << endl;}
            else { cout << “No student with name: “ << stName << endl;}
    }
 }
  return 0;}
```

Finding if an Element Exists in a Map

IIT Bombay

```
int main() {
  map<string, double> mark
  … Some other code …
  while (true) {
    cout << “Give name of stu
    if (stName == “end”) {cout <<
    else { if (marks.count(stName) > 0) {cout << “Marks: “ << marks[stName] << endl;}
            else { cout << “No student with name: “ << stName << endl;}
    }
  }
  return 0;}
```

Counts number of pairs with key same as stName

Returns 1 if map contains an element with key stName, otherwise returns 0

Finding if an Element Exists in a Map

IIT Bombay

```
int main() {
    map<string, double> marks;
    … Other code …
    map<string, double>::iterator it = marks.find(stName);
    if (it != marks.end()) {cout << "Marks: " << it->second << endl;}
    else { cout << "No student with name: " << stName << endl;}
    return 0;
}
```

Returns an iterator to (key, value) pair with key matching stName

Finding the Count of Elements in a Map

IIT Bombay

```
int main() {
    map<string, double> marks;
    marks["Ajanta"] = 10.0; marks["Bobby"] = 15.0;
    marks["Abdul"] = 25.0;
    cout << "Size : " << marks.size() << endl;
    marks["Alex"] = 14.5;  marks["Ajanta"] = 11.0;
    cout << "Size : " << marks.size() << endl;
    return 0;
}
```

Size : 3

Size : 4

Deleting Elements from a Map

IIT Bombay

```
int main() {
  map<string, double> marks;
  marks["Ajanta"] = 10.0;  marks["Bobby"] = 15.0;
  marks["Abdul"] = 25.0;
  map<string, double>::iterator it = marks.find("Abdul");
  marks.erase(it);  marks.erase("Bobby");
  cout << "Size : " << marks.size() << endl;
  for (it = marks.begin(); it != marks.end(); it++)  {
      cout << it->first << " : " << it->second << endl;
  }
  return 0;
}
```

Size : 1

Ajanta : 10.0

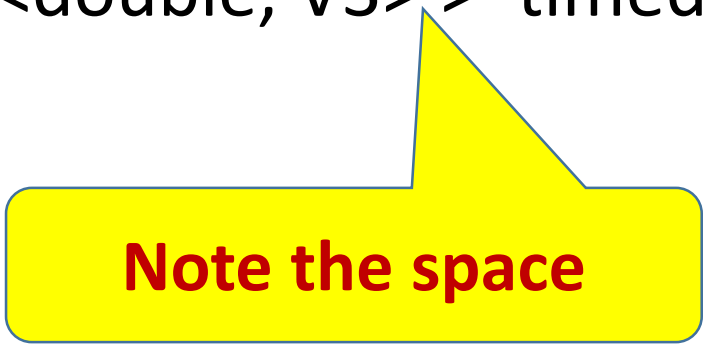
Maps of Complex Data Types

IIT Bombay

- "map" is a template class

  Can be instantiated to maps of complex data types

```
map<string, V3>  flightPosition;
map<string, map<double, V3> >  timedFlightPosition;
```



Summary

IIT Bombay

- "**map**" class and its usage
  - Only some features studied
- Several more features exist …