

Computer Programming

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

Session : Template Class “vector” – Part 2

Quick Recap of Relevant Topics



- Object-oriented programming with structures and classes
- C++ Standard Library
 - The “string” class
 - The “vector” class – some features of it

Overview of This Lecture



- More features of the template class “vector”

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 “vector” class



- For representing and manipulating one dimensional arrays of objects
 - Uses dynamically allocated array to store elements
 - Array can grow or shrink in size
 - Dynamic memory management built in
- “vector” objects are container objects
- Must use **#include <vector>** at start of program
- Large collection of member functions
 - We’ll see only a small subset

Vectors of Complex Data Types

- “vector” is a template class
 - So why restrict to `vector<int>`, `vector<float>`, `vector<char>` ... ?

- Recall

```
class V3 {  
    private: double x, y, z;  
             double length() const { ... }  
    public:  operator+(V3 const &b) const {...}  
             ... Other member functions ...  
};
```

- We can have **`vector<V3> v3Vector(10);`**

Vectors of Complex Data Types

- Vectors of pointers

```
vector<int *> intPtrVec;
```

**Each element of
intPtrVec is an object
of type int ***

```
vector<V3 **> v3PtrPtrVec;
```

**Each element of
v3PtrPtrVec is an
object of type V3****

- Vectors of vectors !

```
vector <vector<int> > x;
```

**Each element of x is
a vector of integers**

Multi-dimensional Vectors

- Vectors of pointers

```
vector<int *> intPtrVec;  
vector<V3 **> v3PtrPtrVec;
```

- Vectors of vectors !

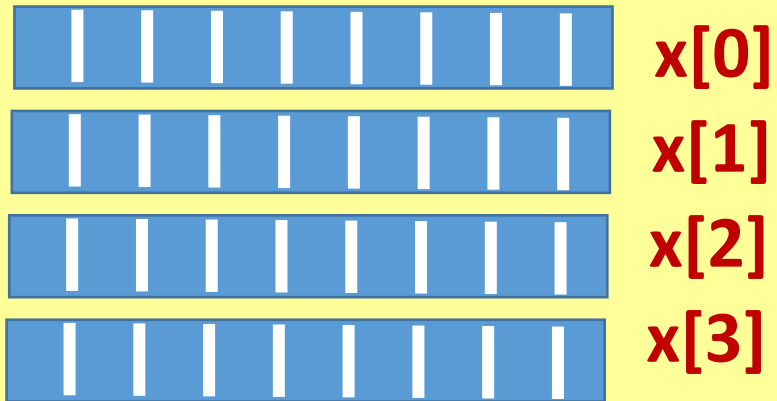
```
vector <vector<int> > x;
```

Note the space

Multi-dimensional Vectors

```
vector<vector<int> > x(4, vector<int>(9));
```

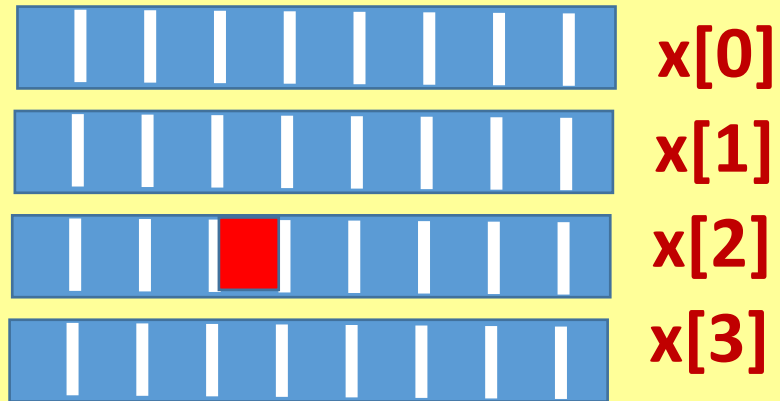
Vector of (vector of integers)



Multi-dimensional Vectors

```
vector<vector<int> > x(4, vector<int>(9));
```

Vector of (vector of integers)

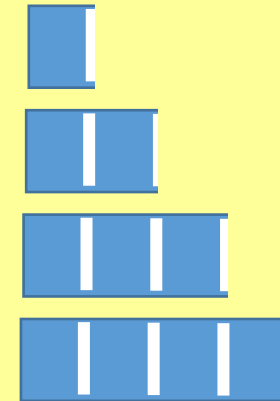


Access:
(x[2])[3] or simply x[2][3]

Multi-dimensional Vectors

```
vector<vector<int> > x(4);  
x[0] = vector<int>(1);  
x[1] = vector<int>(2);  
x[3] = vector<int>(3);  
x[4] = vector<int>(4);
```

Lower triangular matrix



x[0]

x[1]

x[2]

x[3]

A User-defined Matrix Class using Vectors

```
class matrix2D {  
    private: vector<vector<int> > elements;  
    public:  
        matrix2D(size_t m, size_t n) : elements(m, vector<int>(n)) {}  
        int &operator()(size_t i, size_t j) { return elements[i][j]; }  
        size_t n_rows() { return elements.size(); }  
        size_t n_cols() { return elements[0].size(); }  
};
```

alias for unsigned int

**Preferred type for
sizes of objects in
memory**

**Also used for indices
in string, vector, ...**

A User-defined Matrix Class using Vectors

```
class matrix2D {  
    private: vector<vector<int> > elements;  
    public:  
        matrix2D(size_t m, size_t n) : elements(m, vector<int>(n)) { }  
        int &operator() (size_t i, size_t j) { return elements[i][j]; }  
        size_t nrows() { return elements.size(); }  
        size_t ncols() {return elements[0].size(); }  
};
```

**Overloaded
() operator**

A User-defined Matrix Class using Vectors

```
int main() {  
    matrix2D M(5, 5);  
    for (size_t i = 0; i < M.nrows(); i++) {  
        for (size_t j = 0; j < M.ncols(); j++) {  
            M(i, j) = i*i + j*j;  
        }  
    }  
    ... Some other code ...  
}
```

Summary



- Additional features of the “**vector**” class
- Multi-dimensional vectors
- Many more member functions of the “**vector**” class exist