# MH1402 Algorithms & Computing II

## Lecture 7 Vectors

Wu Hongjun

# Overview

- **Vectors**
  - **Declaration**
  - **Accessing vector elements**
  - **Member functions of vector**
  - **Vector assignment**
  - **Passing vector to function**
  - **Multidimensional vector**

# Vectors

- **Arrays are widely used in C/C++ (fast)**
- **However, arrays have some drawback**
  - **The size of array must be known at the time of declaration(as specified in the standard), i.e., the array size cannot be modified after declaration**
- **In C++, vector can be used to replace array (no vector in C)**
  - **You can resize a vector**

# Vector

- **The vector in C++ is different from the vectors in math and physics**
- **There is a sequence of elements of the *same type* in a C++ vector**
- **Vector is somehow similar to array, but with significant differences**
  - **Different declaration**
  - **Different when passed to function**
  - **Vector is a type of C++ class (more on class later in this course)**
    - **Vector has member functions**
    - **Those member functions operate on a vector**

# Vector: Declaration

```
#include <vector>
using namespace std;
int main( )
{
   //declare an empty vector foo, elements are int
   vector<int> foo;  //optional to specify the vector size at the time of declaration

   //declare a vector bar with 10 double elements;
   vector<double> bar(10);
   // some codes …
   return 0;
}
```

NOT bar[10]

# Vector: Elements

- Vector elements are indexed starting from 0 (the same as array)
- The values of vector elements are set as 0 if uninitialized (different from array)
- The square brackets can be used to access the elements (the same as array)

Example:

```
vector<int>  foo(10); // ten int elements in vector foo
cout << foo[9];        // prints 0!
foo[2] = 3;
foo[10] = 4;           //out of bound error; program may crash
```

Another Example:

```
vector<int> bar;       //empty vector;
bar[2] = 5;            //likely program crashes
```

# Vector : Member functions

- **There are many vector member functions**

  **http://www.cplusplus.com/reference/vector/vector/**

- **We will learn the commonly used vector member functions**
  - **at, push_back, pop_back,  clear, resize,  size, begin, end, erase, insert**

# Vector : Member function : at

**at(n)** **access the n-th element of a vector,**

**similar to the operator [ ]**

**(difference: at( ) always checks the bounds, and returns error if out of bounds)**

**Example:**

```
vector<int> bar(5);
for (int i = 0; i < 5; i++) bar[i] = 2*i;
cout << bar.at(0) << " "  << bar.at(3) ;   //print:  0 6
cout << bar.at(5) ;                        //program crashes and terminates
```

# Vector : Member function : push_back

**push_back(qux)** **Increase the size of vector by 1,**
**then set the last element as qux**

**Example:**

```
vector<int> bar;        //declare an empty vector bar
bar.push_back(3);    //the size of bar becomes 1; bar[0] = 3
bar.push_back(10);  //the size of bar becomes 2; bar[1] = 10
bar.push_back(13);  //the size of bar becomes 3; bar[2] = 13
```

# Vector : Member function : pop_back

**pop_back()  Decrease the vector size by 1**

**(The last element gets removed from the vector)**

**Example:**

```
vector<int> bar(10);  //declare vector bar with ten elements
                      //all the elements take value 0

bar.pop_back();       //the size of bar becomes 9
bar.pop_back();       //the size of bar becomes 8
```

# Vector : Member function : clear

**clear()** **Reduce the size of a vector to zero**

**(All the elements are removed from the vector)**

**Example:**

```
vector<int> bar(10);   //all the ten elements take value 0
bar[3] = 5;            //bar[3] = 5

bar.push_back(46);   //the size of bar becomes 11;  bar[10] = 46
bar.clear();         //the size of bar becomes 0
```

# Vector : Member function : resize

**resize(n)  Set the size of the vector to n**

**Suppose that before resize(n), there are m elements in a vector:**

1) **if n < m, the last m-n elements are removed from the vector;**
2) **if n = m, no change to the vector;**
3) **if n > m, n-m elements are appended to the vector, and these n-m elements are set to 0;**

# Vector : Member function

**resize(n) example:**

```
vector<int> bar(10); //the size of bar is 10
//some codes ……
bar.resize(100);     //the size of bar becomes 100
```

# Vector : Member function : size

**size()** **Returns the size of the vector**

**Example:**

```
vector<int> bar;                    //declare empty vector bar
cout << bar.size() << endl;    //prints 0: the size is zero
bar.push_back(5);
cout<< bar.size() << endl;     //prints 1: the size is 1
bar.resize(10);
cout<< bar.size() << endl;     //prints 10: the size is 10
bar.clear();
cout<< bar.size() << endl;     //prints 0: the size is 0
```

# Vector : Member function : begin, end

**Iterator is an object that points to some element in a range of elements.**

**begin() Return an iterator referring to the first element of vector**

**end() Return an iterator referring to the pass-the-end element of vector (pass-the-end element: the theoretical element following the last element of a vector)**

# Vector : Member functions : erase

**erase( ) Erase element(s) from a vector**

**The size of the vector gets reduced**

**Erase Example 1:**

```
vector<int> bar(10);
for (int i = 0; i < 10; i++) bar[i] = i;
                    // the elements: 0 1 2 3 4 5 6 7 8 9

bar.erase(bar.begin()+1);  // remove the second element bar[1]
                    // vector size becomes 9
                    // the elements: 0 2 3 4 5 6 7 8 9
```

**Erase Example 2:**

```
vector<int> bar(10);

for (int i = 0; i < 10; i++)  bar[i] = i;
        // the elements are: 0 1 2 3 4 5 6 7 8 9

bar.erase(bar.end()-2);
        // remove the element bar[8], size becomes 9
        // the elements are: 0 1 2 3 4 5 6 7 9
```

**Erase Example 3: (erase more than one elements)**

```
vector<int> bar(10);

for (int i = 0; i < 10; i++)  bar[i] = i;
        //the elements: 0 1 2 3 4 5 6 7 8 9

bar.erase(bar.begin()+2, bar.begin()+4);
        //remove bar[2], bar[3]; but bar[4] not removed
        //vector size becomes 8
        //the elements: 0 1 4 5 6 7 8 9
```

**Erase Example 4:**

```
vector<int> bar(10);

for (int i = 0; i < 10; i++)  bar[i] = i;
        //the elements: 0 1 2 3 4 5 6 7 8 9

bar.erase(bar.begin()+1, bar.end());
        //remove all the elements except the first element
        //the size of bar becomes 1
```

**Erase Example 5:**

```
vector<int> bar(100);

for (int i = 0; i < 100; i++)  bar[i] = i;

int ini = 12, fin = 45;
bar.erase(bar.begin()+ini, bar.begin()+fin+1);
    // remove all the elements from bar[12] to bar[45]
    // (including bar[12] and bar[45])
```

# Vector : Member functions : insert

**insert( )** **Insert element(s) into a vector**

**The size of the vector gets increased**

**Example:**

```
vector<int> bar(10);
for (int i = 0; i < 10; i++)  bar[i] = i;
        //the elements are: 0 1 2 3 4 5 6 7 8 9

 bar.insert(bar.begin()+2, 55);
        //insert 55 as bar[2]
        //size becomes 11
        //the elements are: 0 1 55 2 3 4 5 6 7 8 9
```

# Vector : Member functions

- **Note that erase and insert are expensive for large vector**
  - **Reason: the elements in a vector are stored in contiguous memory space.**
    **A lot of memory accesses (read/write) may be involved for insertion/erase.**
- **For efficient erase and insertion, we may use linked list**
  - **Linked list is another type of data structure (not covered in this course)**
  - **The elements of linked list can be stored at random memory locations**
  - **But it is slow to access (read/write) the elements of linked list**
    **(it is fast to access the elements of vector and array)**

# Vector Assignment

- **Vector assignment**
  - **We can assign one vector to another**
  - **Different from array**
- **Example 1:**

```
vector<int> foo(3);
vector<int> bar(5);
for (int i = 0; i < 5; i++) bar[i] = i;
foo = bar;  //the size of foo is now 5
            //the elements of foo are: 0 1 2 3 4
```

# Vector Assignment

- **Vector Assignment Example 2: Compilation error for type mismatch**

```
vector<double> foo(3);
vector<int> bar(5);

for (int i = 0; i < 5; i++) bar[i] = i;
foo = bar;  //error: no match for assignment operator
```

# Passing Vector to Function

- **Passing a vector to a function is passing by value, not passing by reference**
  - **Different from passing array to function**
- **When passing a vector to a function, the function knows the size of that vector by using the vector member function `size()`**
  - **Different from passing array to function**
- **A function can return a vector (cannot return an array)**

- **Example:**

```
int sum_of_vector(vector<int>);

int main()
{
    int sum;
    vector<int> foo;
    foo.push_back(3);   foo.push_back(4);
    foo.push_back(6);   foo.push_back(8);
    sum = sum_of_vector(foo);
    cout << sum << endl; //prints 21
    return 0;
}

int sum_of_vector(vector<int> bar)
{
    int sum = 0;
    for (int i = 0; i< bar.size(); i++)
        sum += bar[i];
    return sum;
}
```

- **Passing a vector to a function by reference**

```
void increase(vector<int>&);

int main()
{
   int sum;
   vector<int> foo;
   foo.push_back(3); foo.push_back(4);
   foo.push_back(6); foo.push_back(8);
   increase(foo);
   for (int i = 0; i < foo.size(); i++) cout<<foo[i]<<endl;
   return 0;
}
 void increase(vector<int>& bar)
 {
    int sum = 0;
    for (int i = 0; i < bar.size(); i++)
       bar[i] += 3;
    return;
 }
```

```
6
7
9
11
```

- **A function can return a vector (a function cannot return an array)**

```
vector<int> initialize();

int main()
{
   vector<int> foo;
   foo = initialize();
   for (int i = 0;i < foo.size();i++) cout<<foo[i]<<endl;
   return 0;
}

vector<int> initialize()
{
   vector<int> bar;
   bar.push_back(0);  bar.push_back(2);  bar.push_back(4);
   bar.push_back(6);  bar.push_back(8);  bar.push_back(10);
   return bar;
}
```

```
0
2
4
6
8
10
```

# Multidimensional Vector

**Must have space between > >**
**(but not required in code::blocks)**

- **The declaration of multidimensional vector is more complicated than that of multidimensional array**
  - **For example, a two dimensional vector is declared as a vector of one dimensional vectors.**
- **Examples:**

```
vector< vector<int> > foo; //A two-dimensional vector foo
vector< vector<int> > bar(3, vector<int>(4));
        //declare vector bar with 3 rows, 4 columns;
        //declared as a vector of 1D vector<int>(4)
bar[2][3] = 325; //assign 325 to an element of bar
```

# Multidimensional Vector

- **Resizing a multidimensional vector is more complicated than resizing a 1D vector**
- **Example 1: Resizing an empty 2D vector to four rows, five columns (using push_back here)**

```
vector< vector<int> > foo;   //empty vector
for (int i = 0; i < 4; i++)
{
    foo.push_back(vector<int>(5));
}
```

# Multidimensional Vector

- **Example 2: Resizing a 2D vector using resize**

```
vector< vector<int> > foo(3, vector<int>(4));
foo.resize(5); //note that:
        //the first row      foo[0]: 4 elements
        //the second row foo[1]: 4 elements
        //the third row     foo[2]: 4 elements
        //the fourth row  foo[3]: 0 element
        //the fifth row      foo[4]: 0 element
```

# Multidimensional Vector

- **Example 3: Resizing a 2D vector using resize**

```
vector< vector<int> > foo(3, vector<int>(4));
foo[1].resize(7);  //the second row changed to 7 elements
            //note that:
            //the first row  foo[0]: 4 elements
            //the second row foo[1]: 7 elements
            //the third row  foo[2]: 4 elements
```

**(in a 2D vector, the sizes of rows may be different;
in a 2D array, the sizes of rows are always the same)**

# Multidimensional Vector

- **Example 4: Resizing a 2D vector (3 rows, 4 columns) to 5 rows, 7 columns using resize**

```
vector< vector<int> > foo(3, vector<int>(4));
foo.resize(5);          //changed to 5 rows
for (int i = 0; i < foo.size(); i++)
{
    foo[i].resize(7);   //each row is changed to 7 elements
}
```

# Multidimensional Vector

- **Passing a 2D vector to function by reference (example)**

```
void increase(vector < vector<int> >& bar)
{
  for (int i = 0; i < bar.size(); i++)
     for (int j = 0; j < bar[i].size(); j++)
        bar[i][j] += 3;
}
int main()
{
  vector<vector<int> >  foo(3, vector<int>(4));
  increase(foo);
  // ........
}
```