Vectors

CSci 588: Data Structures, Algorithms and Software Design

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

Vector

Vectors are part of c++'s standard template library (STL).

This library contains a number of predefined classes and algorithms, with very efficient implementations.

Creating Vectors

Creating Vectors

```
// constructing vectors
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  unsigned int i;

  // constructors used in the same order as described above:
  vector<int> first;                                // empty vector of ints
  vector<int> second (4,100);                       // four ints with value 100
  vector<int> third (second.begin(),second.end());  // iterating through second
  vector<int> fourth (third);                       // a copy of third

  // the iterator constructor can also be used to construct from arrays:
  int myints[] = {16,2,77,29};
  vector<int> fifth (myints, myints + sizeof(myints) / sizeof(int) );

  cout << "The contents of fifth are:";
  for (i=0; i < fifth.size(); i++)
    cout << " " << fifth[i];

  cout << endl;

  return 0;
}
```

Copying Vectors

```cpp
// vector assignment
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> first (3,0);
  vector<int> second (5,0);

  second=first;
  first=vector<int>();

  cout << "Size of first: " << int (first.size()) << endl;
  cout << "Size of second: " << int (second.size()) << endl;
  return 0;
}
```

Size, Capacity, Max Size Reserving Space for and Resizing Vectors

Size of a Vector

```cpp
// vector::size
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myints;
  cout << "0. size: " << (int) myints.size() << endl;

  for (int i=0; i<10; i++) myints.push_back(i);
  cout << "1. size: " << (int) myints.size() << endl;

  myints.insert (myints.begin() + 5,10,100);
  cout << "2. size: " << (int) myints.size() << endl;

  myints.pop_back();
  cout << "3. size: " << (int) myints.size() << endl;

  return 0;
}
```

Capacity and max size of a Vector

```
// comparing size, capacity and max_size
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;

  // set some content in the vector:
  for (int i=0; i<100; i++) myvector.push_back(i);

  cout << "size: " << (int) myvector.size() << "\n";
  cout << "capacity: " << (int) myvector.capacity() << "\n";
  cout << "max_size: " << (int) myvector.max_size() << "\n";
  return 0;
}
```

Checking to see if a Vector is Empty

```
// vector::empty
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;
  int sum (0);

  for (int i=1;i<=10;i++) myvector.push_back(i);

  while (!myvector.empty())
  {
     sum += myvector.back();
     myvector.pop_back();
  }

  cout << "total: " << sum << endl;

  return 0;
}
```

Resizing a Vector

```cpp
// resizing vector
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;

  unsigned int i;

  // set some initial content:
  for (i=1;i<10;i++) myvector.push_back(i);

  myvector.resize(5);
  myvector.resize(8,100);
  myvector.resize(12);

  cout << "myvector contains:";
  for (i=0;i<myvector.size();i++)
    cout << " " << myvector[i];

  cout << endl;

  return 0;
}
```

Resizing a Vector

```
// resizing vector
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;

  unsigned int i;

  // set some initial content:
  for (i=1;i<10;i++) myvector.push_back(i);

  myvector.resize(5);
  myvector.resize(8,100);
  myvector.resize(12);

  cout << "myvector contains:";
  for (i=0;i<myvector.size();i++)
    cout << " " << myvector[i];

  cout << endl;

  return 0;
}
```

resize adds or drops elements from a vector, but will not change it’s capacity.

Reserving more space for a Vector

```
// vector::reserve
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> content;
  size_t filesize;

  ifstream file ("test.bin",ios::in|ios::ate|ios::binary);
  if (file.is_open())
  {
    filesize=file.tellg();

    content.reserve(filesize);

    file.seekg(0);
    while (!file.eof())
    {
      content.push_back( file.get() );
    }
    // print out content:
    vector<int>::iterator it;
    for (it=content.begin() ; it<content.end() ; it++)
      cout << hex << *it;
  }
  return 0;
}
```

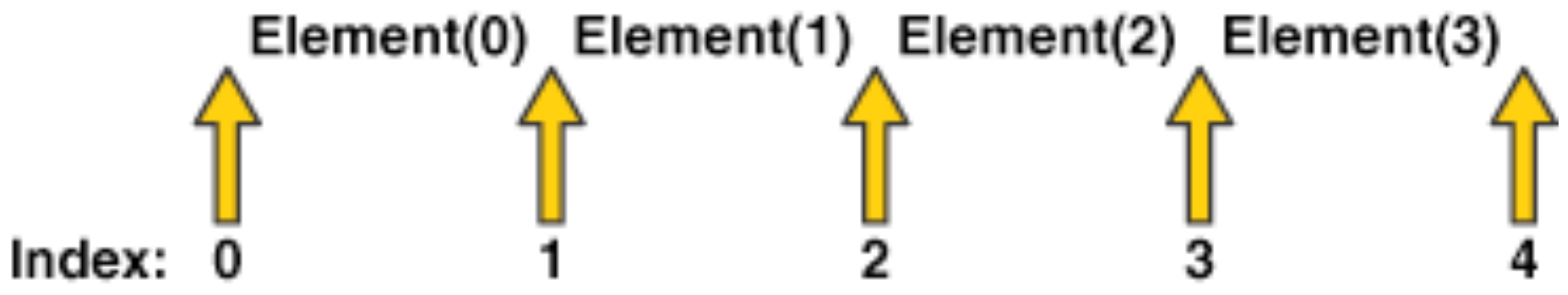
Passing Vectors to Functions

Passing Vectors to Functions

```
#include <iostream>
#include <vector>
using namespace std;
void call_by_value_test(vector<int> v) {
    v[0] = 3;
    v[1] = 6;
    v[2] = 9;
    v[3] = 12;
}
void call_by_reference_test(vector<int> &v) {
    v[0] = 3;
    v[1] = 6;
    v[2] = 9;
    v[3] = 12;
}
int main() {
    vector<int> v1(4,1);

    cout << "initial v1 -- " << v1[0] << " " << v1[1]
         << " " << v1[2] << " " << v1[3] << endl;

    call_by_value_test(v1);
    cout << "after call by value test -- " << v1[0]
         << " " << v1[1] << " " << v1[2] << " " << v1[3] << endl;

    call_by_reference_test(v1);
    cout << "after call by reference test -- " << v1[0]
         << " " << v1[1] << " " << v1[2] << " " << v1[3] << endl;
}
```

Iterating over Vectors

Possible Iterator Positions



iterating over vectors

```
// vector::begin
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;
  for (int i=1; i<=5; i++) myvector.push_back(i);

  vector<int>::iterator it;

  cout << "myvector contains:";
  for ( it=myvector.begin() ; it < myvector.end(); it++ )
    cout << " " << *it;

  cout << endl;

  return 0;
}
```

iterating over vectors 2

```
// vector::end
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;
  for (int i=1; i<=5; i++) myvector.insert(myvector.end(),i);

  cout << "myvector contains:";
  vector<int>::iterator it;
  for ( it=myvector.begin() ; it < myvector.end(); it++ )
    cout << " " << *it;

  cout << endl;

  return 0;
}
```

reverse iteration over vectors

```
// vector::rbegin/rend
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;
  for (int i=1; i<=5; i++) myvector.push_back(i);

  cout << "myvector contains:";
  vector<int>::reverse_iterator rit;
  for ( rit=myvector.rbegin() ; rit < myvector.rend(); ++rit )
    cout << " " << *rit;

  cout << endl;

  return 0;
}
```

inserting at an iterator

```
// inserting into a vector
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector (3,100);
  vector<int>::iterator it;

  it = myvector.begin();
  it = myvector.insert ( it , 200 );

  myvector.insert (it,2,300);

  // "it" no longer valid, get a new one:
  it = myvector.begin();

  vector<int> anothervector (2,400);
  myvector.insert (it+2,anothervector.begin(),anothervector.end());

  int myarray [] = { 501,502,503 };
  myvector.insert (myvector.begin(), myarray, myarray+3);

  cout << "myvector contains:";
  for (it=myvector.begin(); it<myvector.end(); it++)
    cout << " " << *it;
  cout << endl;

  return 0;
}
```

erasing at an iterator

```
// erasing from vector
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  unsigned int i;
  vector<unsigned int> myvector;

  // set some values (from 1 to 10)
  for (i=1; i<=10; i++) myvector.push_back(i);

  // erase the 6th element
  myvector.erase (myvector.begin()+5);

  // erase the first 3 elements:
  myvector.erase (myvector.begin(),myvector.begin()+3);

  cout << "myvector contains:";
  for (i=0; i<myvector.size(); i++)
    cout << " " << myvector[i];
  cout << endl;

  return 0;
}
```

myvector.at(n) vs myvector[n]

at vs []

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<double> myvector(10);

    myvector[11] = 200;    // this may or may not crash; if
                           // it does it will not tell you why
    myvector.at(10) = 200; // this will crash with an error
                           // saying you tried to access something
                           // outside of the bounds of the vector
}
```

at is much safer than using [], but it is a little slower. In general, use at unless you have fully debugged your code and have a need for as much performance as possible.

Vectors as Stacks

pushing to the back of the vector

```
// vector::push_back
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;
  int myint;

  cout << "Please enter some integers (enter 0 to end):\n";

  do {
    cin >> myint;
    myvector.push_back (myint);
  } while (myint);

  cout << "myvector stores " << (int) myvector.size() << " numbers.\n";

  return 0;
}
```

removing the back of the vector

```cpp
// vector::pop_back
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;
  int sum (0);
  myvector.push_back (100);
  myvector.push_back (200);
  myvector.push_back (300);

  while (!myvector.empty())
  {
    sum+=myvector.back();
    myvector.pop_back();
  }

  cout << "The elements of myvector summed " << sum << endl;

  return 0;
}
```

accessing the front of the vector

```
// vector::front
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;

  myvector.push_back(78);
  myvector.push_back(16);

  // now front equals 78, and back 16

  myvector.front() -= myvector.back();

  cout << "myvector.front() is now " << myvector.front() << endl;

  return 0;
}
```

accessing the back of the vector

```
// vector::back
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> myvector;

  myvector.push_back(10);

  while (myvector.back() != 0)
  {
    myvector.push_back ( myvector.back() -1 );
  }

  cout << "myvector contains:";
  for (unsigned i=0; i<myvector.size() ; i++)
    cout << " " << myvector[i];

  cout << endl;

  return 0;
}
```

Clearing and Assigning Vectors

Clearing a Vector

```cpp
// clearing vectors
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  unsigned int i;
  vector<int> myvector;
  myvector.push_back (100);
  myvector.push_back (200);
  myvector.push_back (300);

  cout << "myvector contains:";
  for (i=0; i<myvector.size(); i++) cout << " " << myvector[i];

  myvector.clear();
  cout << "myvector contains:";
  for (i=0; i<myvector.size(); i++) cout << " " << myvector[i];

  myvector.push_back (1101);
  myvector.push_back (2202);

  cout << "\nmyvector contains:";
  for (i=0; i<myvector.size(); i++) cout << " " << myvector[i];

  cout << endl;

  return 0;
}
```

Assigning a Vector

```cpp
// vector assign
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  vector<int> first;
  vector<int> second;
  vector<int> third;

  first.assign (7,100);             // a repetition 7 times of value 100

  vector<int>::iterator it;
  it=first.begin()+1;

  second.assign (it,first.end()-1); // the 5 central values of first

  int myints[] = {1776,7,4};
  third.assign (myints,myints+3);   // assigning from array.

  cout << "Size of first: " << int (first.size()) << endl;
  cout << "Size of second: " << int (second.size()) << endl;
  cout << "Size of third: " << int (third.size()) << endl;
  return 0;
}
```

Operations on Vectors

Sorting a Vector

```
// sort algorithm example
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

bool myfunction (int i,int j) { return (i<j); }

struct myclass {
  bool operator() (int i,int j) { return (i<j);}
} myobject;

int main () {
  int myints[] = {32,71,12,45,26,80,53,33};
  vector<int> myvector (myints, myints+8);               // 32 71 12 45 26 80 53 33
  vector<int>::iterator it;

  // using default comparison (operator <):
  sort (myvector.begin(), myvector.begin()+4);           //(12 32 45 71)26 80 53 33

  // using function as comp
  sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)

  // using object as comp
  sort (myvector.begin(), myvector.end(), myobject);     //(12 26 32 33 45 53 71 80)

  // print out content:
  cout << "myvector contains:";
  for (it=myvector.begin(); it!=myvector.end(); ++it)
    cout << " " << *it;

  cout << endl;

  return 0;
}
```

Binary Search on a Vector

```
// binary_search example
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

bool myfunction (int i,int j) { return (i<j); }

int main () {
  int myints[] = {1,2,3,4,5,4,3,2,1};
  vector<int> v(myints,myints+9);                          // 1 2 3 4 5 4 3 2 1

  // using default comparison:
  sort (v.begin(), v.end());

  cout << "looking for a 3... ";
  if (binary_search (v.begin(), v.end(), 3))
    cout << "found!\n"; else cout << "not found.\n";

  // using myfunction as comp:
  sort (v.begin(), v.end(), myfunction);

  cout << "looking for a 6... ";
  if (binary_search (v.begin(), v.end(), 6, myfunction))
    cout << "found!\n"; else cout << "not found.\n";

  return 0;
}
```

Randomize/Shuffle a Vector

```cpp
// random_shuffle example
#include <iostream>
#include <algorithm>
#include <functional>
#include <vector>
#include <ctime>
#include <cstdlib>
using namespace std;

// random generator function:
ptrdiff_t myrandom (ptrdiff_t i) { return rand()%i;}

// pointer object to it:
ptrdiff_t (*p_myrandom)(ptrdiff_t) = myrandom;

int main () {
  srand ( unsigned ( time (NULL) ) );
  vector<int> myvector;
  vector<int>::iterator it;

  // set some values:
  for (int i=1; i<10; ++i) myvector.push_back(i); // 1 2 3 4 5 6 7 8 9

  // using built-in random generator:
  random_shuffle ( myvector.begin(), myvector.end() );

  // using myrandom:
  random_shuffle ( myvector.begin(), myvector.end(), p_myrandom);

  // print out content:
  cout << "myvector contains:";
  for (it=myvector.begin(); it!=myvector.end(); ++it)
    cout << " " << *it;

  cout << endl;

  return 0;
}
```

Merging Vectors

```
// merge algorithm example
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main () {
  int first[] = {5,10,15,20,25};
  int second[] = {50,40,30,20,10};
  vector<int> v(10);
  vector<int>::iterator it;

  sort (first,first+5);
  sort (second,second+5);
  merge (first,first+5,second,second+5,v.begin());

  cout << "The resulting vector contains:";
  for (it=v.begin(); it!=v.end(); ++it)
    cout << " " << *it;

  cout << endl;

  return 0;
}
```