CSC 215

Math and Computer Science

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

A Legacy of Excellence

Algorithms - Continued

- Transform – revisited

```
vector<int> v1= {1,2,3,4,5};
vector<int> v2;

transform( v1.begin(), v1.end(), v1.begin(), inc3 );
                                    // v1 = 4,5,6,7,8
v2.resize( v1.size() );
transform( v1.begin(), v1.end(), v2.begin(), inc3 );
     // v1 = 4,5,6,7,8                 v2 = 7,8,9,10,11
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Inc3 Function

- Rule the same data type must be passed in and returned.

  dataType functionName(dataType value);

```
int inc3( int value)
{
    value = value+3;
    return value;
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Reverse

- Reverse function flips the contents between begin and end

```
int arr[10] = {1,2,3,4,5,6,7,8,9,10};
vector<int> v1 = {1,2,3,4,5,6,7,8,9,10};

reverse(  v1.begin(), v1.end() );
                              // v1 = 10,9,8,7,6,5,4,3,2,1
reverse( arr, arr+10);        // arr = 10,9,8,7,6,5,4,3,2,1
reverse( v1.begin() + 2, v1.end() -2 );
                              // v1 = 1,2,8,7,6,5,4,3,9,10
reverse( arr+2, arr+8 );      // arr = 1,2,8,7,6,5,4,3,9,10
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Sort – Increasing Order

```
int arr[10] = {11,21,131,-4,589,695,
               -7,18,89,190};
vector<int> v1 {11,21,131,-4,589,695,
               -7,18,89,190};
sort( v1.begin(), v1.end() );
    // v1 = -7,-4,11,18,21,89,131,190,589,695
sort( arr, arr+10 );
    // arr = -7,-4,11,18,21,89,131,190,589,695
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Partial Sort – Increasing Order

```
int arr[10] = {11,21,131,-4,589,695,-7,18,89,190};
vector<int> v1= {11,21,131,-4,589,695,-7,18,89,190};

sort( v1.begin()+2, v1.end()-2 );
    // v1 = 11,21,-7,-4,18,131,589,695,89,190
sort( arr+2, arr+8 );
    // arr = 11,21,-7,-4,18,131,589,695,89,190
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Sort – Decreasing Order

```
int arr[10] = {11,21,131,-4,589,695,-7,18,89,190};
vector<int> v1= {11,21,131,-4,589,695,-7,18,89,190};

sort( v1.begin(), v1.end(), myOrder );
      // v1 = 695,589,190,131,89,21,18,11,-4,-7
sort( arr, arr+10, myOrder );
     // arr = 695,589,190,131,89,21,18,11,-4,-7
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

myOrder Function

- Boolean function that expects 2 values of the same type
  - bool functionName(dataType left, dataType right);
  - Returns true if left and right are in order, false if out of order
  - Left op right

```
bool myOrder( int left, int right )
{
    return left > right;
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Sort – Decreasing Order - Trickery

- Use the reverse iterators
- Will not work on arrays

```
vector<int> v1= {11,21,131,-4,589,695,-7,18,89,190};

      // reverse iterators
sort( v1.rbegin(), v1.rend() );
      // v1 = 695,589,190,131,89,21,18,11,-4,-7
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Count Function

- The frequency that a particular item is found between two iterators.

```
int times;
int arr[10] = {1,3,2,3,2,1,2,3,2,1};
vector<int> v1 = {1,3,2,3,2,1,2,3,2,1};
times = count( v1.begin(), v1.end(), 2 );
     // times = 4
times = count( arr, arr+10, 2);
     // times = 4
```

Count If Function

- The frequency that a particular item is found between two iterators that matches some criteria.

```
int times;
int arr[10] = {1,3,2,3,2,1,2,3,2,1};
vector<int> v1 = {1,3,2,3,2,1,2,3,2,1};
times = count_if( v1.begin(), v1.end(), isOdd );
      // times = 6
times = count( arr, arr+10, isOdd );
      // times = 6
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

isOdd Function

- Prototype:
  - bool functionName(dataType value)
  - Returns true if the value is to be counted, false if it is not to be counted.

```
bool isOdd( int value )
{
    return(value & 1 ? true : false );
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Max Element

- Returns an iterator to the largest element in the container object

```
int arr[10] = {11,21,131,-4,589,695,-7,18,89,190};
vector<int> v1= {11,21,131,-4,589,695,-7,18,89,190};
int *ptr = nullptr;
vector<int>::iterator it;

it = max_element( v1.begin(), v1.end() );
ptr = max_element( arr, arr+10 );
cout << *it << " " << *ptr << endl; // 695 and 695
```

Min Element

- Returns an iterator to the smallest element in the container object

```
int arr[10] = {11,21,131,-4,589,695,-7,18,89,190};
vector<int> v1 = {11,21,131,-4,589,695,-7,18,89,190};
int *ptr = nullptr;
vector<int>::iterator it;

it = min_element( v1.begin(), v1.end() );
ptr = min_element( arr, arr+10 );
cout << *it << " " << *ptr << endl; // -7 and -7
```

Unique Member Function

- Removes all first element from every group of elements in the container range iterator to iterator

```
vector<int> v1 = {1,1,2,2,4,5,5,5,6,6,6,2,2,3,3,3,5,5,2,2};
vector<int>::iterator it, spot;

spot = unique( v1.begin(), v1.end() );
for( it = v1.begin(); it != spot; it++)
    cout << *it << " ";         // 1 2 4 5 6 2 3 5 2
// v1 gets destroyed, it rearranges the contents
// v1 = 1 2 4 5 6 2 3 5 2 6 6 2 2 3 3 3 5 5 2 2
```

SOUTH DAKOTA
SCHOOL OF MINES & TECHNOLOGY

Unique Member Function

```
vector<int> v1 = {1,1,2,2,4,5,5,5,6,6,6,2,2,3,3,3,5,5,2,2};
vector<int>::iterator it, spot;

sort( v1.begin(), v1.end() );
spot = unique( v1.begin(), v1.end() );
for( it = v1.begin(); it != spot; it++)
    cout << *it << " ";          // 1 2 3 4 5 6
// v1 gets destroyed, it rearranges the contents
// v1 = 1 2 3 4 5 6 2 2 3 3 3 4 5 5 5 5 5 6 6 6
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Merge Function

- Combine two container objects into a third container object
- The container objects must be sorted
- The third container object must contain the space for both containers

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Merge Example

```
int arr[10] = {11,21,131,-4,589,695,-7,18,89,190};
vector<int> v1 = {11,21,131,-4};
vector<int> v2 = {589,695,-7,18,89,190};
vector<int> v3;

sort(v1.begin(), v1.end() );     // -4, 11, 21, 131
sort(v2.begin(), v2.end() );     // -7, 18, 89, 190, 589, 695
v3.resize( v1.size() + v2.size() );
merge(  v1.begin(),v1.end(),   v2.begin(),v2.end(),   v3.begin() );
// -7 -4 11 18 21 89 131 190 589 695
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Find

- Looks for a value between the range first to last iterators
- If the value is not found, an iterator equal to last will be returned.
- There are other find functions available.
    - find_if
    - find_if_not
    - find_end
    - find_first_of
    - adjacent_find

Find example

```
vector<int> v1 = {11,21,131,-4,589,695,-7,18,89,190};
vector<int>::iterator it;

it = find( v1.begin(), v1.end(), 695 );
if( it != v1.end() )
    cout << "Item was found: " << *it << endl;
else
    cout << "Item was not found" << endl;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Find example

```
vector<int> v1 = {11,21,131,-4,589,695,-7,18,89,190};
vector<int>::iterator it;
vector<int>::iterator sit=v1.begin()+1,
                      eit=v1.end()-2;
it = find(sit , eit, 89);
if (it != eit)
    cout << "Item was found: " << *it << endl;
else
    cout << "Item was not found" << endl;
```

SOUTH DAKOTA
SCHOOL OF MINES & TECHNOLOGY

Numeric Library

- #include <numeric>
- Accumulate function
  - Sum the range from first to last

Accumulate Example

```
int arr[10] = {11,21,131,-4,589,695,-7,18,89,190};
vector<int> v1 = {11,21,131,-4,589,695,-7,18,89,190};
int sum1=0, sum2=0;

sum1 = accumulate( v1.begin(), v1.end(), 0 );
sum2 = accumulate( arr, arr+10, 0 );
cout << sum1 << " " << sum2 << endl; // 1733 1733
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Accumulate Example

```
int arr[10] = {11,21,131,-4,589,695,-7,18,89,190};
vector<int> v1 = {11,21,131,-4,589,695,-7,18,89,190};
int sum1=0, sum2=0;

sum1 = accumulate( v1.begin(), v1.end(), 1000 );
sum2 = accumulate( arr, arr+10, 1000 );
cout << sum1 << " " << sum2 << endl; // 2733 2733
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY