

Laboratory Exercise 15 (Ordered Linked Lists)

Topics: Ordered Linked Lists

Goals: Upon successful completion of this lab you should be able to:

1. Insert nodes into the “middle” of a doubly linked list
2. Remove nodes from the “middle” of a doubly linked list
3. Determine if a value is contained in a doubly linked list
4. Use a doubly linked list to store an ordered set
5. Create a new ordered set containing the union of 2 ordered sets
6. Create a new ordered set containing the intersection of 2 ordered sets

Related text sections:

Chapter 17

Laboratory Exercise 16 Instructions

1. Copy your LList2.tmp file from Lab12 to a new Lab 15 directory.
2. Rename your LList2.tmp file OrderedSet.tmp.
3. Rename the class OrderedSet. Change the template type to ST.
4. Move the methods

```
bool InsertFirst (const ST & value);
bool InsertLast (const ST & value);
bool DeleteFirst ();
bool DeleteLast ();
```

from the public section of the class to the private section of the class.
5. Add the following methods to the public section of the OrderedSet class:

```
void Clear ();
bool IsEmpty () const;
bool IsIn (const ST & value) const;
bool Insert (const ST & value);
bool Delete (const ST & value);
OrderedSet operator + (const OrderedSet & other);
OrderedSet operator * (const OrderedSet & other);
```
6. Implement the Clear method to delete all items in the set.
7. Implement the IsEmpty method to return true if there are no nodes in the set and false if there are nodes in the set.
8. Implement the IsIn method to return true if the specified item is in the set and false if it is not in the set.
9. Implement the Insert method to maintain the nodes of the set in ascending order based on the node data. Duplicate data items are not permitted; if the value is already in the list, the method should simply return false. The Insert method should use the InsertFirst and InsertLast methods from the private section of the class when it is appropriate.
10. Implement the Delete method to remove the specified node from the set. If the value is not in the list, the method should simply return false. The Delete method should use the DeleteFirst and DeleteLast methods from the private section of the class when it is appropriate.
11. Implement the output operator (<<) to list the items in the set in set braces ({ and }), separated by commas. For example: { 2, 4, 5, 7, 10 } .
12. Implement the + operator to return an ordered set containing the union of its two ordered set operands. { 1, 3, 5, 7, 9 } + { 1, 2, 3, 5, 7 } = { 1, 2, 3, 5, 7, 9 }
13. Implement the * operator to return an ordered set containing the intersection of its two ordered set operands. { 1, 3, 5, 7, 9 } * { 1, 2, 3, 5, 7 } = { 1, 3, 5, 7 }
14. The full class definition is on the next page:

```

template <class ST>
class OrderedSet
{
private:
    typedef class Node
    {
        public:
            Node ();
            ST data;
            Node * next;
            Node * prev;
    };

public:
    typedef class Iterator
    {
        public:
            Iterator ();
            Iterator (Node * NP);
            const ST operator * () const;
            Iterator operator ++ ();
            Iterator operator ++ (int);
            Iterator operator -- ();
            Iterator operator -- (int);
            bool operator == (const Iterator & other) const;
            bool operator != (const Iterator & other) const;
        private:
            Node * current;
    };

    OrderedSet ();
    OrderedSet (const OrderedSet & other);
    ~OrderedSet ();
    OrderedSet & operator = (const OrderedSet & other);
    bool operator == (const OrderedSet & other);
    int Size () const;
    void Clear ();
    bool IsEmpty () const;
    bool IsIn (const ST & value) const;
    bool Insert (const ST & value);
    bool Delete (const ST & value);
    OrderedSet operator + (const OrderedSet & other);
    OrderedSet operator * (const OrderedSet & other);
    void Forward (void function (const ST & param));
    void Backward (void function (const ST & param));
    Iterator begin () const;
    Iterator rbegin () const;
    Iterator end () const;
    Iterator rend () const;
    ST operator [] (const int & index) const;
    friend ostream & operator << >> (ostream & outs, const OrderedSet<ST> & S);

private:
    Node * first;
    Node * last;
    int size;
    bool InsertFirst (const ST & value);
    bool InsertLast (const ST & value);
    bool DeleteFirst ();
    bool DeleteLast ();
};

```

15. Use the following program to test the Insert method and the output (<<) operator of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 1
#include <iostream>
#include <cstdlib>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<int> S1;
    cout << "Insert values in set:\n";
    S1.Insert (1);
    S1.Insert (5);
    S1.Insert (3);
    S1.Insert (7);
    S1.Insert (-1);
    cout << "Elements in S1: " << S1 << endl;
    return 0;
}
```

16. Use the following program to test the Size and IsEmpty methods of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 2
#include <iostream>
#include <cstdlib>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<char> S1;
    cout << "Test if new set is empty:\n";
    if (S1.IsEmpty ())
        cout << "S1 is empty\n";
    else
        cout << "S1 is NOT empty\n";
    cout << "Insert values in set:\n";
    S1.Insert ('c');
    S1.Insert ('x');
    S1.Insert ('r');
    S1.Insert ('z');
    S1.Insert ('a');
    cout << "Test if modified set is empty:\n";
    if (S1.IsEmpty ())
        cout << "S1 is empty\n";
    else
        cout << "S1 is NOT empty\n";
    cout << "Elements in S1: " << S1 << endl;
    cout << "The size of S1 is: " << S1.Size() << endl;
    return 0;
}
```

17. Use the following program to test the Delete, Clear and IsIn methods of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 3
#include <iostream>
#include <cstdlib>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<int> S1;
    cout << "Insert values 7,12,-3,-4,15,12, and 10 in set:\n";
    S1.Insert (7);
    S1.Insert (12);
    S1.Insert (-3);
    S1.Insert (-4);
    S1.Insert (15);
    S1.Insert (12);
    S1.Insert (10);
    cout << "Elements in S1: " << S1 << endl;
    cout << "The size of S1 is: " << S1.Size() << endl;
    cout << "Test IsIn and delete items found in set:\n";
    if (S1.IsIn (-4))
        cout << "-4 is in S1\n";
    else
        cout << "-4 is not in S1\n";
    S1.Delete(-4);
    if (S1.IsIn (5))
        cout << "5 is in S1\n";
    else
        cout << "5 is not in S1\n";
    S1.Delete(5);
    if (S1.IsIn (7))
        cout << "7 is in S1\n";
    else
        cout << "7 is not in S1\n";
    S1.Delete(7);
    if (S1.IsIn (15))
        cout << "15 is in S1\n";
    else
        cout << "15 is not in S1\n";
    S1.Delete(15);
    cout << "Elements in S1: " << S1 << endl;
    cout << "Test clear:\n";
    S1.Clear();
    cout << "Test attempt to delete item from empty set:\n";
    if (S1.IsIn (7))
        cout << "7 is in S1\n";
    else
        cout << "7 is not in S1\n";
    S1.Delete(7);
    return 0;
}
```

18. Use the following program to test some of the iterator methods of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 4
#include <iostream>
#include <cstdlib>
#include <string>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<string> S1;
    cout << "Insert values in set:\n";
    S1.Insert ("Hello");
    S1.Insert ("CS 215");
    S1.Insert ("students");
    cout << "Elements in S1: " << S1 << endl;
    cout << "Testing begin, rbegin, and *:\n";
    cout << "The contents of begin() is: " << *S1.begin() << endl;
    cout << "The contents of rbegin() is: " << *S1.rbegin() << endl;
    return 0;
}
```

19. Use the following program to test more of the iterator methods and some of the exception handling of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 5
#include <iostream>
#include <cstdlib>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<int> S1;
    S1.Insert (7);
    S1.Insert (12);
    S1.Insert (-3);
    S1.Insert (-4);
    cout << "Elements in S1: " << S1 << endl;
    OrderedSet<int>::Iterator i = S1.begin();
    cout << "Testing iterator = begin(); it contains: " << *i << endl;
    cout << "Testing ++ operators:\n";
    cout << *i++ << ' ';
    cout << *++i << ' ';
    cout << *i << endl;
    for (i = S1.begin(); i != S1.end(); i++)
        cout << "*i = " << *i << ' ';
    cout << endl;
    cout << "Testing ++ exception handling:\n";
    i = S1.end();
    i++;
    return 0;
}
```

20. Use the following program to test more of the iterator methods and some of the exception handling of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 6
#include <iostream>
#include <cstdlib>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<int> S1;
    S1.Insert (7);
    S1.Insert (12);
    S1.Insert (-3);
    S1.Insert (-4);
    cout << "Elements in S1: " << S1 << endl;
    OrderedSet<int>::Iterator i = S1.rbegin();
    cout << "Testing iterator = rbegin(); it contains: " << *i << endl;
    cout << "Testing -- operators:\n";
    cout << *--i << ' ';
    cout << *i-- << ' ';
    cout << *i << endl;
    for (i = S1.rbegin(); i != S1.rend(); i--)
        cout << "*i = " << *i << ' ';
    cout << endl;
    cout << "Testing -- exception handling:\n";
    i = S1.end();
    i--;
    return 0;
}
```

21. Use the following program to test the bracket ([]) operator and some of the exception handling of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 7
#include <iostream>
#include <cstdlib>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<int> S1;
    S1.Insert (6);
    S1.Insert (-5);
    S1.Insert (4);
    cout << "Elements in S1: " << S1 << endl;
    cout << "Testing [] operator:\n";
    int sum = 0;
    for (int i = 0; i < S1.Size(); i++)
        sum = sum + S1[i];
    cout << "The sum of the integers is: " << sum << endl;
    cout << "The contents of S1[-2] is: ";
    cout << S1[-2] << endl;
    return 0;
}
```

22. Use the following program to test the union (+) and intersection (*) operators of the OrderedSet class. Before running the program, predict its output:

```
// OrderedSet test program 8
#include <iostream>
#include <cstdlib>
#include "OrderedSet.tmp"

using namespace std;

int main ()
{
    OrderedSet<int> S1;
    cout << "Insert values 7,12,-3,-4,15,12, and 10 in S1:\n";
    S1.Insert (7);
    S1.Insert (12);
    S1.Insert (-3);
    S1.Insert (-4);
    S1.Insert (15);
    S1.Insert (12);
    S1.Insert (10);
    OrderedSet<int> S2;
    cout << "Insert values -1,2,3,-14,5,12 and 0 in S2:\n";
    S2.Insert (-1);
    S2.Insert (2);
    S2.Insert (3);
    S2.Insert (-14);
    S2.Insert (5);
    S2.Insert (12);
    S2.Insert (0);
    cout << "Elements in S1: " << S1 << endl;
    cout << "Elements in S2: " << S2 << endl;
    OrderedSet<int> S3;
    cout << "Testing S3 = S1 + S2\n";
    S3 = S1 + S2;
    cout << "Elements in S3: " << S3 << endl;
    cout << "Testing S3 = S2 + S1\n";
    S3 = S2 + S1;
    cout << "Elements in S3: " << S3 << endl;
    cout << "Testing S3 = S1 * S2\n";
    S3 = S1 * S2;
    cout << "Elements in S3: " << S3 << endl;
    cout << "Testing S3 = S2 * S1\n";
    S3 = S2 * S1;
    cout << "Elements in S3: " << S3 << endl;
    return 0;
}
```

23. When you are convinced that your OrderedSet template is working correctly, copy your well documented .tmp file to the dropbox as yourlastnameL15.tmp.