과목: 자료구조

교수: 최재현 교수님

자료구조 <과제 3>

- List 클래스, OrderedList 클래스 작성 -

홍지훈

이름: 홍지훈

학과: 소프트웨어학부

분반: 나

학번 : 20201777

0. 과제

1. 순서가 없는(정렬되지 않은 ) 리스트를 표현하는 List 클래스 작성

2. 순서가 있는(정렬되어 있는) 리스트를 표한하는 OrderedList 클래스 작성

3. 2개 클래스 모두, 생성자 및 소멸자함수를 작성

  -- 멤버 변수는 int \*items, int itemCount, int size    ( 최초 사이즈 5로 시작 )

4. List 클래스의 경우

      - 리스트의 특정 위치에 있는 값을 가져오는 int getItem(int index) 함수 작성  ( index가 범위를 벗어날경우 -99999 반환 )

     -  리스트의 맨끝에 값을 추가하는 void addItem(int itm) 함수 작성

     -  리스트의 특정 위치에 값을 추가하는 void insertItem(int index, int itm) 함수 작성

     -  리스트의 특정 위치에 있는 값을 삭제하는 int removeAt(int index) 함수 작성 ( 성공시 삭제된 값이 반환, 실패시(인덱스오류) -99999 반환)

     -  리스트에 있는 특정 값을 찾아서 삭제하는 int removeItem(int itm) 함수 작성 (성공시 해당값의 위치 반환, 실패시(인덱스오류) -99999 반환)

         ※리스트 맨처음부터 시작하여 최초로 일치하는 곳의 값을 삭제

**-  리스트의 접합 연산을 수행하는 concat(List& list) 함수 구현 ( 2개의 리스트를 연결  - 자기자신과 파라미터로 전달된 리스트 )**

     -  리스트의 모든 값을 출력하는 print() 함수 구현

5. OrderedList 클래스의 경우

      - 리스트의 특정 위치에 있는 값을 가져오는 int getItem(int index) 함수 작성  ( index가 범위를 벗어날경우 -99999 반환 )

     - 리스트에 값을 추가하는 void addItem(int itm) 함수 작성 --> 값추가후 정렬된 상태 유지

     - 리스트의 특정 위치에 있는 값을 삭제하는 int removeAt(int index) 함수 작성 ( 성공시 삭제된 값이 반환, 실패시(인덱스오류) -99999 반환)

     - 리스트에 있는 특정 값을 찾아서 삭제하는 int removeItem(int itm) 함수 작성 ( 성공시 해당값의 위치 반환, 실패시(인덱스오류) -99999 반환)

         ※리스트 맨처음부터 시작하여 최초로 일치하는 곳의 값을 삭제

**-  리스트의 접합 연산을 수행하는 void concat(OrderedList& olist) 함수 구현 ( 2개의 리스트를 연결 - 연결후에도 정렬된 상태 유지 - 자기자신과 파라미터로 전달된 리스트 )**

     -  리스트의 모든 값을 출력하는 print() 함수 구현

6. 위에서 구현된 기능의 동작을 확인할수 있는 main() 함수를 작성

    - 리스트에 최소 20 개 정도의 숫자를 삽입 ( 중복허용 ) 후 각각의 기능 테스트

강의내용 및 교재에 거의 모든 힌트가 존재하지만,

위와 과제와 관련된 문의사항은 질의응답 게시판 및 메일을 이용해 언제든지 질문 가능.

제출방법은 이전과제의 제출방법과 동일

1. 소스코드

1-1. List.h

#pragma once

#include <iostream>

using namespace std;

class List

{

private:

    int\* items;

    int itemCount;

    int size;

public:

    List();

    ~List();

    int getItem(int index);

    void addItem(int itm);

    void insertItem(int index, int itm);

    int removeAt(int index);

    int removeItem(int itm);

    void concat(List& list);

    void print();

};

1-2. OrderedList.h

#pragma once

#include <iostream>

using namespace std;

class OrderedList

{

private:

    int\* items;

    int itemCount;

    int size;

public:

    OrderedList();

    ~OrderedList();

    int getItem(int index);

    void addItem(int itm);

    int removeAt(int index);

    int removeItem(int itm);

    void concat(OrderedList& olist);

    void print();

};

1-3. List.cpp

#include "List.h"

List::List() {

    itemCount = 0;

    size = 5;

    items = new int[size];

}

List::~List() {

    delete[] items;

}

int List::getItem(int index) {

    if (index <= 0 || index >= itemCount)

        return -99999;

    return items[index];

}

void List::addItem(int itm) {

    if (itemCount < size) {

        items[itemCount] = itm;

        itemCount++;

    }

    else {

        int\* newItem = new int[size \* 2];

        for (int i = 0; i < itemCount; i++)

            newItem[i] = items[i];

        newItem[itemCount] = itm;

        itemCount++;

        delete[] items;

        items = newItem;

        size = size \* 2;

    }

}

void List::insertItem(int index, int itm) {

    if (index >= 0 && index < itemCount) {

        itemCount++;

        if (itemCount >= size) {

            int\* newItem = new int[size \* 2];

            for (int i = 0; i < itemCount - 1; i++)

                newItem[i] = items[i];

            delete[] items;

            items = newItem;

            size = size \* 2;

        }

        for (int i = itemCount; i > index; i--)

            items[i] = items[i - 1];

        items[index] = itm;

    }

    else

        cout << "처리할 수 없습니다. index값은 0~" << itemCount << "사이의 값이여야 합니다." << endl;

}

int List::removeAt(int index) {

    if (index <= 0 || index >= itemCount)

        return -99999;

    int temp = items[index];

    for (int i = index; i < itemCount; i++)

        items[i] = items[i + 1];

    itemCount--;

    return temp;

}

int List::removeItem(int itm) {

    for (int i = 0; i < itemCount; i++) {

        if (items[i] == itm) {

            int temp = i;

            for (int j = i; j < itemCount; j++)

                items[j] = items[j + 1];

            itemCount--;

            return temp;

        }

    }

    return -99999;

}

void List::concat(List& list) {

    for (int i = 0; i < list.itemCount; i++) {

        addItem(list.items[i]);

    }

}

void List::print()

{

    for (int i = 0; i < itemCount; i++) {

        cout << items[i] << " ";

        if (i == itemCount - 1)

            continue;

        cout << "-> ";

    }

    cout << endl;

}

1-4. OrderedList.cpp

#include "OrderedList.h"

OrderedList::OrderedList() {

    itemCount = 0;

    size = 5;

    items = new int[size];

}

OrderedList::~OrderedList() {

    delete[] items;

}

int OrderedList::getItem(int index) {

    if (index <= 0 || index >= itemCount)

        return -99999;

    return items[index];

}

void OrderedList::addItem(int itm) {

    if (itemCount < size) {

        items[itemCount] = itm;

        itemCount++;

    }

    else {

        int\* newItem = new int[size \* 2];

        for (int i = 0; i < itemCount; i++)

            newItem[i] = items[i];

        newItem[itemCount] = itm;

        itemCount++;

        delete[] items;

        items = newItem;

        size = size \* 2;

    }

    int temp;

    for (int i = 0; i < itemCount; i++) {

        for (int j = 0; j < itemCount - i - 1; j++) {

            if (items[j] > items[j + 1]) {

                temp = items[j];

                items[j] = items[j + 1];

                items[j + 1] = temp;

            }

        }

    }

}

int OrderedList::removeAt(int index) {

    if (index <= 0 || index >= itemCount)

        return -99999;

    int temp = items[index];

    for (int i = index; i < itemCount; i++)

        items[i] = items[i + 1];

    itemCount--;

    return temp;

}

int OrderedList::removeItem(int itm) {

    for (int i = 0; i < itemCount; i++) {

        if (items[i] == itm) {

            int temp = i;

            for (int j = i; j < itemCount; j++)

                items[j] = items[j + 1];

            itemCount--;

            return temp;

        }

    }

    return -99999;

}

void OrderedList::concat(OrderedList& OrderedList) {

    for (int i = 0; i < OrderedList.itemCount; i++)

        addItem(OrderedList.items[i]);

    int temp;

    for (int i = 0; i < itemCount; i++) {

        for (int j = 0; j < itemCount - i - 1; j++) {

            if (items[j] > items[j + 1]) {

                temp = items[j];

                items[j] = items[j + 1];

                items[j + 1] = temp;

            }

        }

    }

}

void OrderedList::print()

{

    for (int i = 0; i < itemCount; i++) {

        cout << items[i] << " ";

        if (i == itemCount - 1)

            continue;

        cout << "-> ";

    }

    cout << endl;

}

1-5. main.cpp

#include <iostream>

#include "List.h"

#include "OrderedList.h"

using namespace std;

//53 76 12 33 9 0 61 121 200 3 5 77 12 51 98 35 42 39 87 75 46 (21개)

int testnum[25] = { 53, 76, 12, 33, 9, 0, 61, 121, 200, 3, 5, 77, 12, 51, 98, 35, 42, 39, 87, 75, 46 };

//12 23 10 5 31 42 99 13 (8개)

int testnum2[15] = { 12, 23, 10, 5, 31, 42, 99, 13 };

int main(void) {

    cout << "List 테스트" << endl;

    //List 테스트

    List list1;

    List list2;

    //임의로 값을 넣어줌

    for (int i = 0; i < 21; i++)

        list1.addItem(testnum[i]);

    //전체프린트

    list1.print();

    //3번째 인수 불러옴

    cout << "3번째 인수: " << list1.getItem(3) << endl;

    //25번째(존재하지않는) 인수 불러옴

    cout << "25번째 인수: " << list1.getItem(25) << endl;

    //2번째 인수에 300을 삽입

    list1.insertItem(2, 300);

    //전체프린트

    list1.print();

    //26번째(존재하지 않는 인수에 2를 삽입

    list1.insertItem(26, 300);

    //35 추가

    list1.addItem(35);

    //전체프린트

    list1.print();

    //5번째 인수 삭제 (9)

    cout << "5번째 인수 " << list1.removeAt(5) << "삭제" << endl;

    //전체프린트

    list1.print();

    //값이 200인 인수 삭제 (9)

    cout << "값이 200인 인수 " << list1.removeItem(200) << "번째 인수 삭제" << endl;

    //전체프린트

    list1.print();

    //값이 12인 인수 삭제 (3)

    cout << "값이 12인 인수 " << list1.removeItem(12) << "번째 인수 삭제" << endl;

    //전체프린트

    list1.print();

    //임의로 값을 넣어줌

    for (int i = 0; i < 8; i++)

        list2.addItem(testnum2[i]);

    //전체프린트2

    list2.print();

    //list1에 list2 추가

    list1.concat(list2);

    //전체프린트

    list1.print();

    cout << endl << endl << "OrderedList 테스트" << endl;

    //OrderedList 테스트

    OrderedList olist1;

    OrderedList olist2;

    //임의로 값을 넣어줌

    for (int i = 0; i < 21; i++)

        olist1.addItem(testnum[i]);

    //전체프린트

    olist1.print();

    //3번째 인수 불러옴(9)

    cout << "3번째 인수: " << olist1.getItem(3) << endl;

    //30번째(존재하지않는) 인수 불러옴

    cout << "30번째 인수: " << olist1.getItem(30) << endl;

    //35 추가

    olist1.addItem(25);

    //전체프린트

    olist1.print();

    //12번째 인수 삭제 (53)

    cout << "12번째 인수 " << olist1.removeAt(12) << "삭제" << endl;

    //전체프린트

    olist1.print();

    //값이 98인 인수 삭제 (17)

    cout << "값이 98인 인수 " << olist1.removeItem(98) << "번째 인수 삭제" << endl;

    //전체프린트

    olist1.print();

    //값이 12인 인수 삭제 (4)

    cout << "값이 12인 인수 " << olist1.removeItem(12) << "번째 인수 삭제" << endl;

    //전체프린트

    olist1.print();

    //임의로 값을 넣어줌

    for (int i = 0; i < 8; i++)

        olist2.addItem(testnum2[i]);

    //전체프린트2

    olist2.print();

    //list1에 list2 추가

    olist1.concat(olist2);

    //전체프린트

    olist1.print();

    return 0;

}

2. 실행 화면

