|  |  |
| --- | --- |
|  | **윈도우프로그래밍**  **LAB 05** |
| **분반** | 2 |
| **학번** | 1815060 |
| **이름** | 문정현 |

**# 실습 문제 1**

#1 Set.cpp 프로그램 소스 및 설명(수정한 결과 소스)

#include <iostream>

#include <algorithm>

using namespace std;

#include "Set.h"

// default constructor creates an empty set

//기본 생성자의 경우 집합의 원소갯수를 0으로 초기화시켜준다

Set::Set(void)

{

size = 0;

}

// array copy function, local to this file

static void array\_copy(int a[], int b[], int n)

{

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

a[i] = b[i];

}

// copy constructor

// 매개변수로 들어온 집합에 저장된 원소들을 읽으면서 새로 생성된 집합 클래스의 원소배 // 열에 모두 저장하고 원소갯수를 s의 원소갯수로 초기화시켜준다.

Set::Set(const Set& s)

{

for (int i = 0; i < s.size; i++) {

elems[i] = s.elems[i];

}

size = s.size;

}

// creates a set from an array

// 배열에 저장된 숫자를 새로 생성된 집합 클래스의 원소배열에 모두 저장하고 원소갯수를 // 배열에 저장된 숫자갯수로 초기화시켜준다.

Set::Set(int a[], int n)

{

// Need to make sure NO overflow occurs on the array elems[]

if (n > 100) { cout << "overflow. can't make set instance" << endl; return; }

else {

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

elems[i] = a[i];

}

size = n;

}

}

// return set size

int Set::getsize()

{

return size;

}

// return the element indexed by i; the lowest index is 1

int Set::get(int i)

{

if (i < 1 || i > size) {

cerr << "error in Set::get(): index out of bounds" << endl;

cerr << "returning 0" << endl;

return 0;

}

else

return elems[i - 1];

}

// set union

Set Set::operator+(Set s)

{

int i, j, n;

int a[2 \* MAXSIZE];

n = size;

array\_copy(a, elems, n);

for (j = 0; j < s.size; j++) {

for (i = 0; i < n; i++)

if (a[i] == s.elems[j])

break;

if (i == n)

a[n++] = s.elems[j];

}

return Set(a,n);

}

// set difference

// 현재 집합에 있는 원소들 중 매개변수로 들어온 집합에도 동일하게 있는 원소는 제외한 // 나머지 원소들을 새로운 집합 클래스 원소로 저장하고 이 집합 객체를 리턴한다.

Set Set::operator-(Set s)

{

int a[MAXSIZE];

int idx=0;

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

if (!(s >> elems[i])) {

a[idx++] = elems[i];

}

}

return Set(a, idx);

}

// set assignment; return the assigned set

Set Set::operator=(Set s)

{

size = s.size;

array\_copy(elems, s.elems, size);

return (\*this);

}

// subset relation test

// 매개변수로 들어온 집합과 equality 이거나 매개변수로 들어온 집합이 subset인 경우 1을 // 리턴하고 아닌 경우 0을 리턴한다.

int Set::operator>=(Set s)

{

return ((\*this) == s) || ((\*this) > s);

}

// set equality test

// 매개변수로 들어온 집합과 구성된 원소가 모두 같고 원소 갯수도 모두 같은 경우 1를 리 // 턴하고 나머지 경우는 0를 리턴한다.

int Set::operator==(Set s)

{

int j;

if (size != s.size) { return false; }

for (j = 0; j < s.size; j++) {

if (!((\*this) >> s.elems[j])) { return false; }

}

return true;

}

// proper subset relation test

// 매개변수로 들어온 집합의 모든 원소들이 멤버함수를 호출한 집합에 저장된 원소에도 존 // 재하고 매개변수로 들어온 집합의 원소의 갯수가 더 작은 경우에만 1를 리턴하고 나머지 // 경우는 0을 리턴한다.

int Set::operator>(Set s)

{

int flag = 1, j;

for (j = 0; j < s.size; j++) {

if (!((\*this) >> s.elems[j])) { flag = 0; break; }

}

return (size > s.size) && flag ;

}

// element membership test; return 1 if (\*this) contains x

// 매개변수로 들어온 숫자가 멤버함수를 호출한 집합의 원소로 존재한 경우에만 1를 리턴하 // 고 나머지 경우는 0을 리턴한다. 집합의 원소가 없는 경우라면 1을 리턴한다

int Set::operator>>(int x)

{

if (size == 0) { return true; }

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

if (elems[i] == x) { return true; }

}

return false;

}

#2 main.cpp 프로그램 소스 및 설명(수정한 결과 소스)

#include <iostream>

using namespace std;

#include "Set.h"

void setprint(Set s)

{

cout << "{";

for (int i = 1; i <= s.getsize(); i++)

cout << s.get(i) << ((i < s.getsize()) ? ", " : "");

cout << "}";

}

int main()

{

int a[] = { 3, 9, 5 };

int b[] = { 2, 4, 5, 8, 10 };

Set s1;

Set s2(a, 3);

Set s3 = s2; // copy constructor is used here

Set s4(b, 5);

cout << "SET UNION TEST" << endl;

s1 = s3 + s4;

cout << "s3 = "; setprint(s3); cout << endl;

cout << "s4 = "; setprint(s4); cout << endl;

cout << "s3 + s4 = "; setprint(s1); cout << endl << endl;

// - 연산을 이용한 s4에서 s4와 s3의 교집합을 제외한 원소들의 집합인 s1을 출력한 // 다.

s1 = s4 - s3;

cout << "SET DIFFERENCE TEST" << endl;

cout << "s4 = "; setprint(s4); cout << endl;

cout << "s3 = "; setprint(s3); cout << endl;

cout << "s4 - s3 = "; setprint(s1); cout << endl << endl;

// >> 연산을 이용해서 리턴값이 1이면 i가 s4의 원소이므로 해당하는 문자열을 출력 // 하고 0을 리턴하는 경우 i가 s4의 원소가 아니라는 문자열을 출력한다.

cout << "SET MEMBERSHIP TEST" << endl;

// test for i = 1, 2, ..., 10 if i is in s4

cout << "s4 = "; setprint(s4); cout << endl;

for (int i = 1; i < 11; i++) {

cout << i;

if (s4 >> i) { cout << " is in s4" << endl; }

else { cout << " is NOT in s4" << endl; }

}

// > 연산을 이용해 리턴값이 1이면 s1이 s4의 proper subset이 맞다는 문자열을 출 // 력하고 0을 리턴할 경우 s1이 s4의 proper subset이 아니라는 문자열을 출력한다.

cout << "\nPROPER SUBSET RELATION TEST" << endl;

cout << "s1 = "; setprint(s1); cout << endl;

cout << "s4 = "; setprint(s4); cout << endl;

// test if s4 > s1

if (s4 > s1) { cout << "s1 is a proper subset of s4" << endl; }

// if not print < s1 is NOT a proper subset of s4 >

else { cout << "s1 is a NOT proper subset of s4" << endl; }

// == 연산을 이용해 1을 리턴하면 문자열로 s2와 s3이 equality라고 출력하고

// 0을 리턴하는 경우에는 equality가 아니라고 출력한다.

cout << "\nSET EQUALITY TEST" << endl;

// test if s2 == s3

cout << "s2 = "; setprint(s2); cout << endl;

cout << "s3 = "; setprint(s3); cout << endl;

if (s2 == s3) { cout << "s2 is equal to s3 "; }

// if not print < s2 is NOT equal to s3 >

else { cout << "s2 is NOT equal to s3"; }

}

#3 실행 화면

