

PROJECT 3

CSCI 140

C++ Language and Object Development

MAI PHAM

Email – mpham30@student.mtsac.edu

May 09, 2017

Development Environment

MSVS 2012/2015

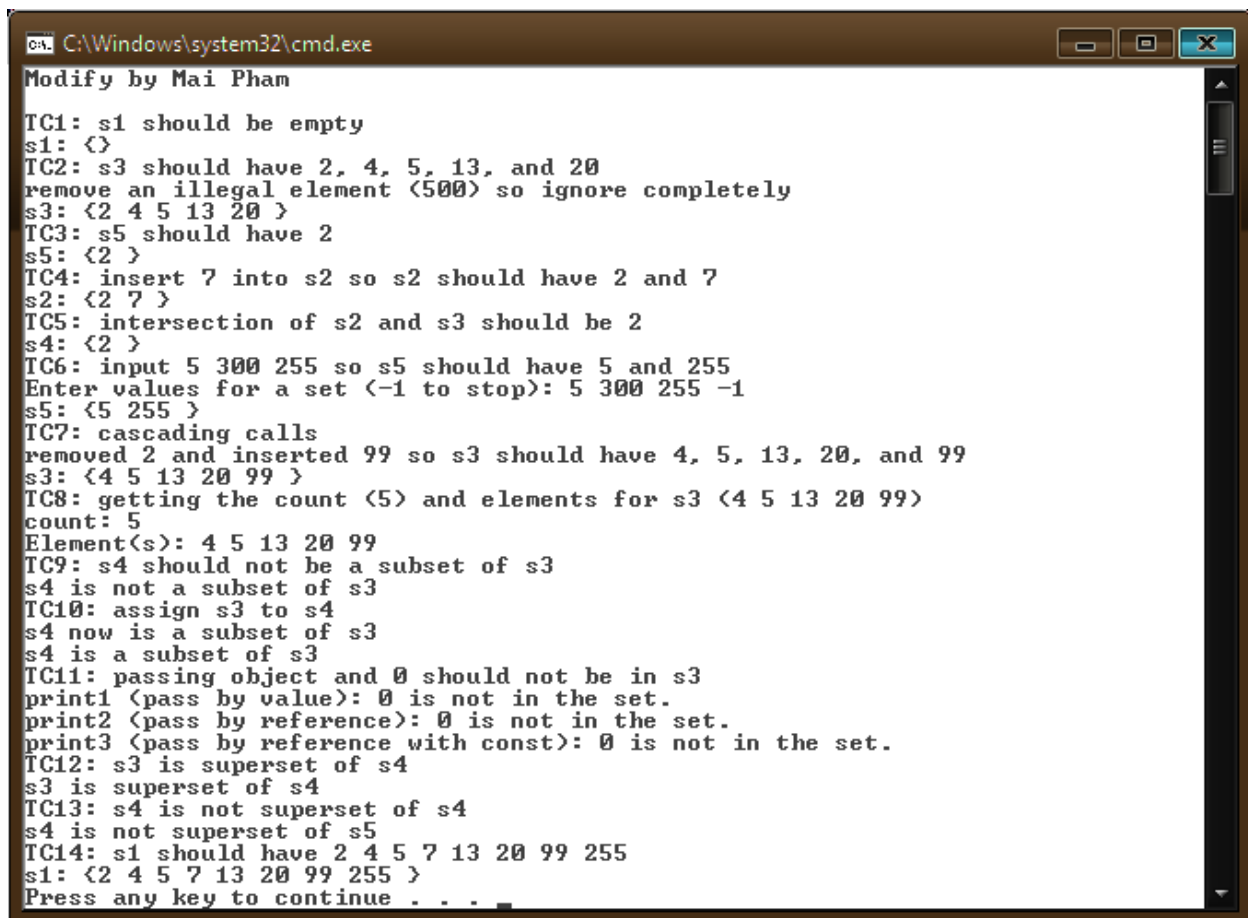
Table of Contents

1. Project Note
2. Input/output
3. Source code (TestIntegerSet.cpp, IntegerSet.cpp, IntegerSet.h)

PROJECT NOTE

Project 3 is to create a class IntegerSet that hold 256 integers, from 0 to 255 in true or false, and some set operations. This class should be able to run the provided test driver by Prof T. Vo. Unlike the previous projects, this project is more simple and straight forward. However, it does bring in new materials which is member/friend functions and operator overloads. Therefore, this project is focusing on learning and working with the syntax of these new materials rather than the logic errors compare to the first two projects. As a conclusion, my project is completed. I did add some additional test cases to the provided test driver which doesn't test the union and the superset operation.

INPUT/OUTPUT



```
C:\Windows\system32\cmd.exe
Modify by Mai Pham
TC1: s1 should be empty
s1: {}
TC2: s3 should have 2, 4, 5, 13, and 20
remove an illegal element (500) so ignore completely
s3: {2 4 5 13 20 }
TC3: s5 should have 2
s5: {2 }
TC4: insert 7 into s2 so s2 should have 2 and 7
s2: {2 7 }
TC5: intersection of s2 and s3 should be 2
s4: {2 }
TC6: input 5 300 255 so s5 should have 5 and 255
Enter values for a set (-1 to stop): 5 300 255 -1
s5: {5 255 }
TC7: cascading calls
removed 2 and inserted 99 so s3 should have 4, 5, 13, 20, and 99
s3: {4 5 13 20 99 }
TC8: getting the count (5) and elements for s3 {4 5 13 20 99}
count: 5
Element(s): 4 5 13 20 99
TC9: s4 should not be a subset of s3
s4 is not a subset of s3
TC10: assign s3 to s4
s4 now is a subset of s3
s4 is a subset of s3
TC11: passing object and 0 should not be in s3
print1 (pass by value): 0 is not in the set.
print2 (pass by reference): 0 is not in the set.
print3 (pass by reference with const): 0 is not in the set.
TC12: s3 is superset of s4
s3 is superset of s4
TC13: s4 is not superset of s4
s4 is not superset of s5
TC14: s1 should have 2 4 5 7 13 20 99 255
s1: {2 4 5 7 13 20 99 255 }
Press any key to continue . . .
```

SOURCE CODE

HEADER FILE

```
/*    Program:          MSVS 2012/2015
    Author:             Mai Pham
    Class:              CSCI 140
    Date:               05/09/2017
    Description:        Project 3 - Integer Set
    I certify that the code below is my own work.
    Exception(s): N/A
*/

//Header File
#ifndef INTEGERSET_H
#define INTEGERSET_H
#include <iostream>
using namespace std;

const int SIZE = 256;

class IntegerSet
{
private:
    bool set[SIZE];
public:
    IntegerSet();
    IntegerSet(int m);
    IntegerSet(const int array[], int n);
    IntegerSet operator+(const IntegerSet &s) const;
    IntegerSet operator*(const IntegerSet &s) const;
    int elements(int arr[]) const;
    bool isElem(int m) const;
    IntegerSet &insert(int m);
    IntegerSet &remove(int a);
    friend istream &operator>>(istream &in, IntegerSet &s);
    friend ostream &operator<<(ostream &out, const IntegerSet &s);
    bool operator<=(const IntegerSet &s) const;
    bool operator>=(const IntegerSet &s) const;
};
#endif
```

IMPLEMENTATION FILE

```
/*    Program:          MSVS 2012/2015
    Author:             Mai Pham
    Class:              CSCI 140
    Date:               05/09/2017
    Description:        Project 3 - Integer Set
    I certify that the code below is my own work.
    Exception(s): N/A
*/

// Implementation File (Member Functions)
#include "IntegerSet.h"
#include <iostream>
```

```
using namespace std;

IntegerSet::IntegerSet()
{
    for (int i = 0; i < SIZE; i++)
        set[i] = false;
}
IntegerSet::IntegerSet(int m)
{
    for (int i = 0; i < SIZE; i++)
        set[i] = false;
    if (m >= 0 && m < SIZE)
        set[m] = true;
}
IntegerSet::IntegerSet(const int arr[], int n)
{
    for (int i = 0; i < SIZE; i++)
        set[i] = false;
    for (int i = 0; i < n; i++)
    {
        if (arr[i] >= 0 && arr[i] < SIZE)
            set[arr[i]] = true;
    }
}
IntegerSet IntegerSet::operator+(const IntegerSet &s) const
{
    IntegerSet a;
    for (int i = 0; i < SIZE; i++)
    {
        if (set[i] || s.set[i])
            a.set[i] = true;
    }
    return a;
}
IntegerSet IntegerSet::operator*(const IntegerSet &s) const
{
    IntegerSet a;
    for (int i = 0; i < SIZE; i++)
    {
        if (set[i] && s.set[i])
            a.set[i] = true;
    }
    return a;
}
int IntegerSet::elements(int arr[]) const
{
    int count = 0;
    for (int i = 0; i < SIZE; i++)
    {
        if (set[i] == true)
        {
            arr[count] = i;
            count++;
        }
    }
    return count;
}
```

```

bool IntegerSet::isElem(int m) const
{
    if (set[m])
        return true;
    else
        return false;
}
IntegerSet &IntegerSet::insert(int a)
{
    if (a >= 0 && a < SIZE)
        set[a] = true;
    return *this;
}
IntegerSet &IntegerSet::remove(int a)
{
    if (a >= 0 && a < SIZE)
        set[a] = false;
    return *this;
}
istream &operator>>(istream &in, IntegerSet &s)
{
    int element;

    for (int i = 0; i < SIZE; i++)
        s.set[i] = false;

    in >> element;
    while (element != -1)
    {
        if (element >= 0 && element < SIZE)
            s.set[element] = true;
        in >> element;
    }
    return in;
}
ostream &operator<<(ostream &out, const IntegerSet &s)
{
    out << "{";
    for (int i = 0; i < SIZE; i++)
    {
        if (s.set[i])
            out << i << " ";
    }
    out << "}";
    return out;
}
bool IntegerSet::operator<=(const IntegerSet &s) const
{
    bool subset = false;
    for (int i = 0; i < SIZE; i++)
    {
        if (set[i])
            if (s.set[i])
                subset = true;
            else
                subset = false;
    }
}

```

```

    }
    return subset;
}
bool IntegerSet::operator>=(const IntegerSet &s) const
{
    bool subset = false;
    for (int i = 0; i < SIZE; i++)
    {
        if (s.set[i])
            if (set[i])
                subset = true;
            else
                subset = false;
    }
    return subset;
}

```

APPLICATION FILE

```

// A partial driver to test IntegerSet class
// Provide by T. Vo for CSCI 140 Project 3 Spring 17
// as a starting point and add more cases as needed.
// Modify by Mai Pham
// Application File

#include <iostream>
#include <string>
#include "IntegerSet.h"
using namespace std;

void print1(IntegerSet s);           // pass by value
void print2(IntegerSet &s);         // pass by reference
void print3(const IntegerSet &s);    // pass by reference with const

int main()
{
    int arr1[6] = {5, 2, 20, 4, 13, 256};
    int arr2[256];                  // can hold up to 256 elements
    int count;
    IntegerSet s1;                  // {}
    IntegerSet s2(2);               // {2}
    IntegerSet s3(arr1, 6);         // {2 4 5 13 20}, 256 was ignored
    IntegerSet s4 = s1;             // {}
    IntegerSet s5 = s2;             // {2}

    cout << "Modify by Mai Pham\n\n";

    cout << "TC1: s1 should be empty\n";
    cout << "s1: " << s1 << endl;    // s1: {}

    cout << "TC2: s3 should have 2, 4, 5, 13, and 20\n";
    s3.remove(500);                 // ignore since it is not a valid element
    cout << "remove an illegal element (500) so ignore completely\n";
    cout << "s3: " << s3 << endl;    // s3: {2 4 5 13 20}

    cout << "TC3: s5 should have 2\n";
}

```

```

cout << "s5: " << s5 << endl;           // s5: {2}

s2.insert(7);                             // s2 is now {2 7}
cout << "TC4: insert 7 into s2 so s2 should have 2 and 7\n";
cout << "s2: " << s2 << endl;           // s2: {2 7}

s4 = s2 * s3;                             // s4 is now {2}, s2 and s3 are still the
same
cout << "TC5: intersection of s2 and s3 should be 2\n";
cout << "s4: " << s4 << endl;           // s4: {2}

cout << "TC6: input 5 300 255 so s5 should have 5 and 255\n";
cout << "Enter values for a set (-1 to stop): ";
cin >> s5;                                // enter: 5 300 255 -1
                                         // s5 is now {5, 255}, 300 was ignore
cout << "s5: " << s5 << endl;           // s5: {5 255}

cout << "TC7: cascading calls\n";
s3.remove(7).remove(2);                   // s3 is now {4 5 13 20}
                                         // 2 was removed and 7 is not in the set
s3.insert(99).insert(5);                  // s3 is now {4 5 13 20 99}
                                         // 99 was inserted and 5 is already in
the set

cout << "removed 2 and inserted 99 so s3 should have 4, 5, 13, 20, and 99\n";
cout << "s3: " << s3 << endl;           // s3: {4 5 13 20 99}

cout << "TC8: getting the count (5) and elements for s3 (4 5 13 20 99)\n";
count = s3.elements(arr2);
cout << "count: " << count << endl;
cout << "Element(s): ";
for (int i = 0; i < count; i++)
    cout << arr2[i] << " ";
cout << endl;

cout << "TC9: s4 should not be a subset of s3\n";
if (s4 <= s3)                             // s4 is not a subset of s3
    cout << "s4 is a subset of s3\n";
else
    cout << "s4 is not a subset of s3\n";

cout << "TC10: assign s3 to s4\n";
s4 = s3;                                 // s4: {4 5 13 20 99}
cout << "s4 now is a subset of s3\n";
if (s4 <= s3)                             // s4 is now a subset of s3
    cout << "s4 is a subset of s3\n";
else
    cout << "s4 is not a subset of s3\n";

cout << "TC11: passing object and 0 should not be in s3\n";
print1(s3);
print2(s3);
print3(s3);

// Create more objects and test cases below
cout << "TC12: s3 is superset of s4\n";

```

```

    if (s4 >= s3)                                // s4 subset s3 = s3 superset s4
        cout << "s3 is superset of s4\n";
    else
        cout << "s3 is not superset of s4\n";

    cout << "TC13: s4 is not superset of s4\n";
    if (s4 >= s5)                                // s4: {4 5 13 20 99} s5: {5 255}
        cout << "s4 is superset of s5\n";
    else
        cout << "s4 is not superset of s5\n";

    cout << "TC14: s1 should have 2 4 5 7 13 20 99 255\n";
    s1 = s2 + s3 + s5;
    cout << "s1: " << s1 << endl;

    return 0;
}

void print1(IntegerSet s)
{
    if (s.isElem(0))
        cout << "print1 (pass by value): 0 is in the set." << endl;
    else
        cout << "print1 (pass by value): 0 is not in the set." << endl;
}

void print2(IntegerSet &s)
{
    if (s.isElem(0))
        cout << "print2 (pass by reference): 0 is in the set." << endl;
    else
        cout << "print2 (pass by reference): 0 is not in the set." << endl;
}

void print3(const IntegerSet &s)
{
    if (s.isElem(0))
        cout << "print3 (pass by reference with const): 0 is in the set." << endl;
    else
        cout << "print3 (pass by reference with const): 0 is not in the set." << endl;
}

```