

Collections

8.1 Basic Collections

There are three basic collections.

1. The basic collection is often called a *bag*. It stores objects with no ordering of the objects and no restrictions on them.
2. Another unstructured collection is a *set* where repeated objects are not permitted: it holds at most one copy of each item. A set is often from a predefined universe.
3. A collection where there is an ordering is often called a *list*. Specific examples include an *array*, a *vector* and a *sequence*. These have the same idea, but vary as to the methods they provide and the efficiency of those methods.

8.2 The Bag ADT

An ADT or *abstract data type* defines a way of storing data: it specifies only how the ADT can be used and says nothing about the implementation of the structure. (An ADT is more abstract than a Java specification or C++ list of class member function prototypes.)

The Bag ADT might have:

- accessor methods such as `size`, `countOccurrence`, possibly an iterator (which steps through all the elements);
- modifier methods such as `add`, `remove`, and `addAll`; and
- also a `union` method which combines two bags to produce a third.

8.3 The Array Implementation

A common implementation of a collection is a *partially filled array*. This is often expanded every time it needs to be, but rarely shrunk. It has a pointer/counter which keeps track of where the real data ends.

0	1	2	3	4	5	6
Amy	Bo	Carl	Dana	NULL	NULL	NULL

count=4

8.4 Sample Program: StringSet

An array-based implementation of a set of strings.

```
// sample code for a set of strings
// uses partially filled array
#ifndef STRINGSET_H
#define STRINGSET_H
#include <string>
using namespace std;

class StringSet {
public:
    StringSet();
    StringSet(const StringSet &other);
    ~StringSet();
    bool contains(string item) const;
    void add(string item);
    void remove(string item);
    bool operator== ( const StringSet &other ) const;

private:
    static const int MAX_COUNT=100;
    int count;
    string *A;

    friend ostream &operator<< (ostream &, const StringSet &myStringSet);
};

#endif

// sample code for a set of strings
// uses partially filled array
// not order preserving
// not the greatest
// wdg 2009

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

```

StringSet::StringSet() : count(0), A(new string[MAX_COUNT])
{
}

StringSet::StringSet( const StringSet &other)
                    : count( other.count), A(new string[MAX_COUNT])
{
    for(int i=0; i<count; i++)
        A[i] = other.A[i];
}

StringSet::~~StringSet()
{
    delete [] A;
}

bool StringSet::contains(string item) const
{
    for(int i=0; i<count; i++)
        if( item==A[i] )
            return true;
    return false;
}

void StringSet::add(string item)
{
    if( contains(item) )
        return;
    A[count++]=item;
}

void StringSet::remove(string item)
{
    for(int i=0; i<count; i++)
        if( item==A[i] ) {
            A[i]=A[count-1];
            count--;
            return;
        }
}

```

```

        }
    }

bool StringSet::operator==( const StringSet &other ) const
{
    if( count != other.count )
        return false;
    for(int i=0; i<count; i++)
        if( !other.contains(A[i]) )
            return false;
    return true;
}

ostream &operator<< (ostream &out, const StringSet &myStringSet)
{
    for(int i=0; i<myStringSet.count-1; i++)
        out << myStringSet.A[i]+":";
    out << myStringSet.A[myStringSet.count-1];
    return out;
}

```

```

// TestStringSet.cpp - wdg - 2008

```

```

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

```

```

int main( )
{
    StringSet S;
    S.add("A"); S.add("B"); S.add("A"); S.add("D"); S.add("C");
    S.remove("D"); S.remove("A");
    StringSet T(S);
    T.remove("dummy"); T.remove("C"); T.add("C");

    cout << "S is " << S << endl;
    cout << "T is " << T << endl;
    cout << (S==T ? "Equal" : "OOOPPPSSS" ) << endl;
    return 0;
}

```