CSCI 2170 **Abstract Data Type, class**

- In many cases, the solution to a problem requires operation on data, for example add, sort, print, and remove operations on data. In these cases, it is better to put the focus on data, and think in terms of what we can do with a collection of data, INDEPENDENTLY of HOW we do it. This is called **data abstraction**. Data abstraction is the fundamental idea of Object Oriented Programming.

- **Abstract Data Type (ADT)** – a collection of data together with a set of operations on that data
  ADT is different from Data Structure – data structure is a construct within a programming language that stores a collection of data. It defines how data is stored in the memory for a program.

  Examples of ADT

  1. favorite movies
     Data
     Operation

  2. Library book catalog
     Data → collection of book records
     Operation → sort the books (sort by author, sort by book due date, sort by related subject, etc.), print out book list, add book to collection, remove out-dated book, display parts of the collection written by the same author …

- When constructing an ADT, there are two steps involved:
  - Specification – indicate precisely what each operation of an ADT does.
  - Implementation –specify how each operation is implemented. This includes selecting a data structure for data involved in ADT using programming language → data structure is part of an ADT's implementation

As a client, using an ADT is like using a vending machine:
we do not grab the snacks directly, we do not care how the snacks are stored..,
we press a button, and expect the corresponding snack to come out.
→
we request for an operation of an ADT, the operation outputs the appropriate result.
We do not need to know how the result is generated, or how data is stored in the ADT.

- **ADT implemented as class:**

class, object
- class is the ADT implemented in C++
- object is an instance of a class

public, private, const
- Data and methods declared in the public section can be accessed by the client program of the class
- Data and methods declared in the private section can not be accessed by the client program of the class
- A const method may not modify the data of a class

constructor (default constructor, other constructors)
- activated when an object of the class is created
- used to initialize data of the object
- a class can have more than one constructor

destructor
- activated when an object of the class exits its scope
- a class can only have one destructor

- destructor is ONLY necessary in a class when data of the class has acquired dynamically allocated memory. In this case, destructor is

- Use **#ifndef / #define / #endif** to prevent multiple inclusion of a class definition

ADT Examples
- ADT Sphere
- ADT Time

- Example AddressBook ADT
  Data : a collection of addresses of friends
  Number of addresses in the address book
  Operation: create the address book → constructor
  add a new address
  find the address of a friend
  print all address of a friend
  Check if two friends live in the same city

Header file:

```
#ifndef ADDRESS_BOOK_H
#define ADDRESS_BOOK_H
struct  AddressStruct;

class AddressBook
{
public:
    AddressBook();
    void AddEntry(AddressStruct newEntry);
    void  DelEntry(AddressStruct newEntry);
    AddressStruct  FindStruct(string name);
    void PrintAll();
    …
private:
     AddressStruct      book[MAX_SIZE];
     int                count;
};
#endif
```

Implementation file:

```
#include "AddressBook.h"
struct AddressStruct
{
     string name;
     string address;
     string  city;
     int      zip;
};

AddressBook::AddressBook()
{
          count =0;
}
```