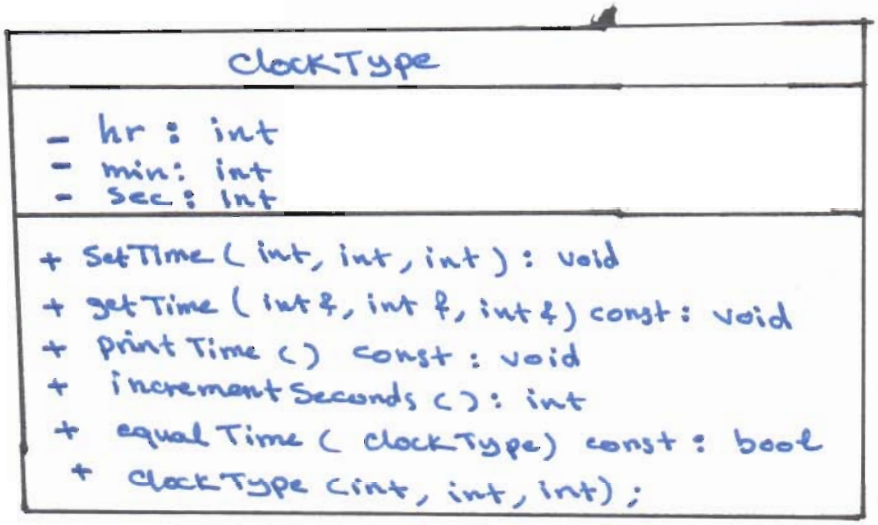
Data structures "Introduction"

objectives:

1. Unified Modeling language Diagrams.
2. Data Abstraction, Classes, and Abstract Data Types (ADT)
3. List as an ADT

① Unified Modeling language Diagrams.

A class and its members can be described graphically using a notation know as Unified Modeling language (UML) notation.



- top box: class Name.
- middle box: data members and their Types
- Last box: member function, parameters list and the return Type.
- + sign: public members.
- - sign: private members
- # sign: protected.

② Data Abstraction, Classes and ADT:

Abstraction is the process of separating the logical properties from the implementation details, Abstraction can be also applied to data.

like any other data type, an ADT has three things associated with it:

- type name
- Domain: the set of values belonging to the ADT
- Set of operations on the data.

ClockType ADT:

dataType name

ClockType

domain

each ClockType value is a time of the day in the form of hours, minutes and seconds.

operations:

Set the Time

Return the Time

print the Time

Increment the time by one second.

Compare the two times to see whether they are equal.

To implement an ADT, we must represent the data and write algorithms to perform the operations.

In C++, classes were specifically designed to handle ADT.

3. List as an ADT

list: set of values of the same type, the most convenient way to represent and process a list is to use an array.

Definition of the list as ADT as the following:

typename

listType

domain

Every element of listType is a set of, say at most 1000 numbers.

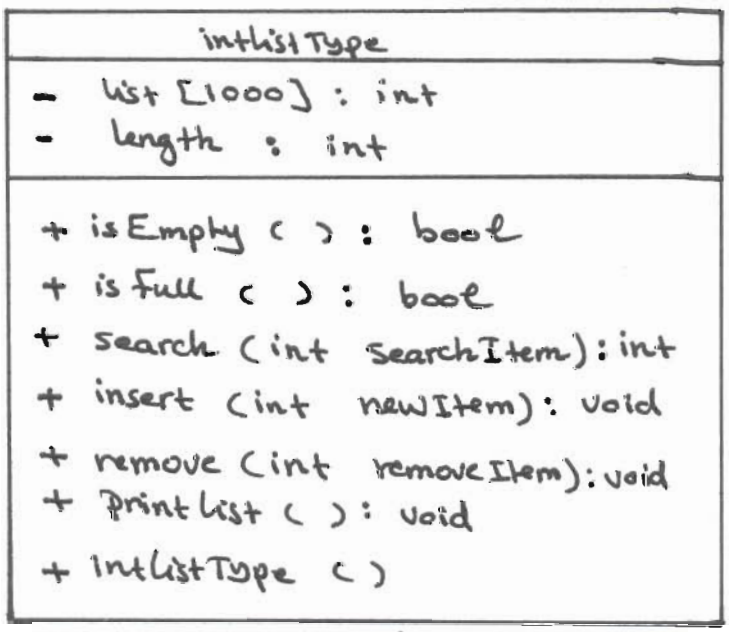
operations

- Check to see whether the list is empty
- Check to see whether the list is full.
- Search the list for a given item.
- Delete an item from the list
- Insert an item in the list
- Sort the list
- Print the list.

ADT list implementation:

```
class intlistType
{
    public :
    bool isEmpty ( );
    // function to determine whether the list is empty.
    // precondition: The list must exist.
    // post condition: Return true if the list is empty,
    //                 false otherwise
```

```
bool     isFull ( );
// Function to determine whether the list is full.
// Precondition : The list must exist.
// Post condition : Return true if the list is full,
// false otherwise
int    search ( int  searchItem);
// Function to determine whether searchItem is in the list.
// post condition : if searchItem is in the list, return
// its index, otherwise it return -1.
void     insert (int  newItem);
// Function to insert newItem in the list.
// Precondition : The list must exist and must not be full.
// Post condition : newItem is inserted in the list and
// length is incremented by one.
void     remove (int  removeitem);
// function to delete removeitem from the list
// Precondition : The list must exist and must not be empty
// post condition : if found, removeitem is deleted from
// the list and the length is decremented by one.
// otherwise an appropriate message is printed.
void   printlist ( );
// function to output the elements of the list
// Precondition : The list must exist.
// Post condition : The elements of the list are printed.
intlist Type ( );
// Default constructor
// post condition : length = 0.
private :
  int   list [1000];
  int   length;
};
```



UML class Diagram of the class intlistType