Objective:

- It will help you understand and thoroughly practice the object passing by value or by references.
- It later part of it will also help you understand the issues related to shallow copy and method to deal with such issues.
- The last task will hit hard on your character arrays skills syntactically but will also help you immensely in sharpen your programming logic teeth ☺.

Task-1: Array ADT

The Array ADT that we discussed today.

Private Members:

- int *data; /* pointer to an array of integers
- int capcity; /* size/capacity of array pointed by data
- int isValidIndex(int index)
  return 1 if index is within bounds otherwise 0.

Public Members:

The class 'Array' should support the following operations

- Array(int cap = 5);
  Sets 'cap' to 'capacity' and initializes rest of the data members accordingly.
  If user sends any invalid value then sets the cap to zero.

- ~Array()
  Free the dynamically allocated memory.

- int & getSet(int index);
  insert value at given index of array.

- int getCapacity()
  returns the size of array.

- void reSize (int newcapacity)
  resize the array to new capacity. Make sure that elements in old array should be preserved in the new array if possible.

You should look at the following function after next lecture.

- Array (Array &)
  makes a deep copy of the received object.

Task-2: Set ADT
The Set ADT, which we also discussed today.
Design an ADT 'Set' whose objects should be able to store an integer set. Size of the set will be given by the user.

Data Members:

- int *data; /* pointer to an array of integers which will be treated as set of integers */
- int noOfElements; // number of elements in the Set
- int capcity; /* maximum possible number of elements that can be stored in the Set */

Supported Operations:

The class 'Set' should support the following operations

1. Set(int cap = 5);
   Sets 'cap' to 'capacity' and initializes rest of the data members accordingly.
   If user sends any invalid value then sets the cap to zero.

2. ~Set()
   Free the dynamically allocated memory.

3. void insert (int element);
   stores the element in the Set.

4. void remove (int element);
   removes the element fromthe Set.

5. void print()
   print the elements of set on console.

6. int getCardinality()
   returns the number of elements in the set.

7. int isMember (int val)
   returns 1 if 'val' is member of the set otherwise return 0.

8. int isSubSet (Set & s2)
   returns 1 if s2 is proper subset of calling object set, return 2 if improper subset otherwise return 0.

9. void reSize (int newcapacity)
   resize the set to new capacity. Make sure that elements in old set should be preserved in the new set if possible.

You should look at the following functions after next lecture.

10. Set(Set & ref)
    The sizzling copy ctor ☺

11. Set calcUnion (Set & s2)
    returns a new Set object which contains the union of 's2' set and calling object set.

12. Set calcIntersection (Set & s2)
    returns a new Set object which contains the intersection of 's2' set and calling object set.

13. Set calcDifference (Set & s2)
    returns a new Set object which contains the difference of 's2' set and calling object set (this object – s2).

14. Set calcSymmetricDifference (Set & s2) (01)
    returns a new Set object which contains the symmetric Difference of 's2' set and calling object set. Where symmetric difference is:
    $A\Delta B = A \bigcup B - A \bigcap B$

Task-3: CString ADT

As discussed/found earlier that you are weak in string manipulation: So through this task, we shall make replica of CString library available in Visual C++, which will surely resolve almost all of your character array issues and will also help in brush up your programming logic. *Do it with full focus because this ADT will be used heavily in upcoming labs and assignments.*

CString ADT

You have to define an Abstract Data Type (ADT) named CString having following data structure:

```
char * data;    // pointer to an array of characters
int size;       //  size of the array (it will always represent amount of memory
                //allocated to data pointer)
```

Methods

1. CString ();
   Default ctor

2. CString (char c);
   Create an object with one character

3. CString(char*);
   Conversion ctor

4. ~CString ();
   Free the resources if any captured by calling object.

5. void input();
   Takes input from console in calling object.

6. char & at(int index);
   index
   Receives the index for string.
   Return Value
   reference of array location represented by index

7. int isEmpty( );
   Tells whether string is empty or not
   Return Value
   return 0 if string empty otherwise nonzero value.

8. int getLenght()
   Return Value
   Return the length of string

9. void display();
   Display on console the string contained in calling object.

10. int find(char* substr, int start=0);
    SubStr
    A substring to search for.
    Start
    The index of the character in the string to begin the search with, or 0 to start from the beginning
    Return Value
    The first character index of the string that matches the sub string, -1 otherwise

11. int find(char ch, int start=0);
    ch
    A character to search for
    Start
    The index of the character in the string to begin the search with, or 0 to start from the beginning
    Return Value
    The character index of the string that matches the c, -1 otherwise

12. int insert(int index, char* substr);
    Index
    The index of the character before which the insertion will take place (start to insert at index and move older elements ahead)
    substr
    A pointer to the sub-string to be inserted
    Return Value
    The length of the changed string

13. int insert(int index, char ch);
    Index
    The index of the character before which the insertion will take place (start to insert at index and move older elements one index ahead)
    ch
    A character to be inserted
    Return Value
    The length of the changed string

14. int remove(int index, int count=1);
    Index
    The zero-based index of the first character in the CString object to delete.
    Count
    The number of characters to be removed
    Return Value
    The length of the changed string

15. int remove (char ch);
    ch
    The character to be removed from a string
    Return Value
    The count of characters removed from the string. Zero if the string is not changed

16. void replace(char new);
    New
    Character replacing all the old characters in old string : string is filled with new char

17. int replace(char old, char new);
    Old
        The character to be replaced by *new*
    New
        Character replacing *Old*
    Return Value
        Return the number of times the old char is replaced by new char

18. int replace(char* old, char* new);
    Old
        A pointer to a string containing the character to be replaced by *new*
    New
        A pointer to a string containing the character replacing *Old*
    Return Value
        Return the number of times the old string is replaced by new string

19. void trim();
    Removes all leading and trailing occurrences of white spaces

20. void trimLeft();
    Removes all white spaces on left side of calling object

21. void trimRight();
    Removes all white spaces on right side of calling object

22. void makeUpper();
    change the string characters to upper case

23. void makeLower();
    change the string characters to lower case

24. void reverse();
    reverse the order of character in the string of calling object.

25. void reSize(int add);
    This function will take an integer variable and increase the string buffer of the object and also preserving the old data at least according to the new size of.

26. void concatEqual(CString& *s2*);
    Concatenate the calling and received object

27. void concatEqual(char * s2);
    Concatenate the calling and received string

28. int isEqual(CString & s2) ;
    compares the calling and received object
    Return Value
        follow the strcmp in the string.h criteria for these comparisons

29. int isEqual(char * s2) ;
    Compares the calling and received string
    Return Value
        follow the strcmp in the string.h criteria for these comparisons

You should look at the following functions after next lecture.

30. CString (CString &);
    Copy ctor

31. CString left(int count) ;

Count
The number of characters to extract from calling object from left side
Return Value
A CString object that contains a copy of the specified range of characters

32. CString right(int count) ;
Count
The number of characters to extract from calling object from right side
Return Value
A CString object that contains a copy of the specified range of characters

33. CString concat(CString& *s2*) ;
Concatenate the calling and received object. No change will come in calling object.
Return Value
Return value concatenated object

34. CString concat(char * *s2*) ;
Concatenate the calling object and received string. No change will come in calling object.
Return Value
Return concatenated object

35. CString tokenzie(char * delim) ;
Returns a Cstring object which contains the substring by extracting it from the calling object CString string depending upon the delimiter characters passed.
See the following Sample Run to further understand the functionality:

```
void main()
{
      CString str(" This, --a sample string. nothing");
      CString token = str.tokeniz (“,.-”);

      // After the execution of above statement:
      // the token contains “ This”
      // the contents of ‘str’ becomes “ --a sample string. nothing”

      // So if we call it again

      token = str.tokenize (“,.-”);

      // then ‘token’ contains “ ”
      // ‘str’ contents becomes “-a sample string. nothing”

      //and calling again

      token = str.tokenize (“,.-”);

      // then token contains “”
      // str contents becomes “a sample string. nothing”

      //and calling again

      token = str.tokenize (“,.-”);

      // then token conatins “a sample string”
      // str contents becomes “ nothing”
}
```

"Twenty years from now you will be more disappointed by the things that you didn't do than by the ones you did do. So throw off the bowlines. Sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream. Discover. "

[- MARK TWAIN -]