Somaiya Vidyavihar University
K. J. Somaiya College of Engineering

Batch: A3 **Roll No.: 16010121051**

Experiment No. 1

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Implementation of Abstract Data Type

Objective: Implementation of ADT without using any standard library function

Expected Outcome of Experiment:

CO	Outcome
CO 1	Explain the different data structures used in problem solving.

Books/ Journals/ Websites referred:

https://www.geeksforgeeks.org/abstract-data-types/

Abstract:-

(Define ADT. Why are they important in data structures?)

Abstract Data type (ADT) is a type (or class) for objects whose behaviour is defined by a set of values and a set of operations. The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called "**abstract**" because it gives an implementation-independent view. The process of providing only the essentials and hiding the details is known as **abstraction**.

They are important in data structures because of the following reasons:

● Representation Independence: Most of the program becomes independent of the abstract data type's representation, so that representation can be improved without breaking the entire program.

● Modularity: With representation independence, the different parts of a program become less dependent on other parts and on how those other parts are implemented.

● Interchangeability of Parts: Different implementations of an abstract data type may have different performance characteristics. With abstract data types, it becomes easier for each part of a program to use an implementation of its data types that will be more efficient for that particular part of the program.

Abstract Data Type for Rational Numbers

[for chosen data type write value definition and operator definition)

rationalclass.h

```
#ifndef RATIONALCLASS_H

#define RATIONALCLASS_H


#include<stdbool.h>


class rational {


public:


/*----Constructors*/

rational(void);


rational(int N, int D);


/*----Set the value. Fails if D is not natural*/
```

```
bool Set(int N, int D);


/*----Add*/

rational Add(rational OtherOne);



/*----Subtract*/

rational Subtract(rational OtherOne);


/*----Multiply*/

rational Multiply(rational OtherOne);


/*----Divide*/

rational Divide(rational OtherOne);


/*----Read. Fails if D is not natural*/

bool Read(void);


/*----Write*/

void Write(void);


private:


int Numerator,

   Denominator;
```

```
};


#endif
```

rationalclass.cpp

```
#include <iostream>

#include "rationalclass.h"


using namespace std;

//----Constructors

rational::rational(void) {


Set(0,1);

}

//---------------------------------------------------------------------------

rational::rational(int N, int D) {


Set(N,D);

}

//---------------------------------------------------------------------------

//----Set the value. Fails if D is not natural

bool rational::Set(int N, int D) {


if (D >=1) {

   Numerator = N;
   Denominator = D;

   return(true);

   }

else return(false);

}

//------------------------------------------------------------------------

//----Add

rational rational::Add(rational OtherOne) {


rational Answer;


Answer.Set(Numerator*OtherOne.Denominator + 
Denominator*OtherOne.Numerator,

        Denominator * OtherOne.Denominator);


return(Answer);

}


rational rational::Subtract(rational OtherOne) {


rational Answer;


Answer.Set(Numerator*OtherOne.Denominator - 
Denominator*OtherOne.Numerator,

        Denominator * OtherOne.Denominator);
```

```
return(Answer);

}


rational rational::Multiply(rational OtherOne) {


rational Answer;


Answer.Set(Numerator*OtherOne.Numerator,

        Denominator * OtherOne.Denominator);


return(Answer);

}


rational rational::Divide(rational OtherOne) {


rational Answer;


Answer.Set(Numerator*OtherOne.Denominator,

        Denominator * OtherOne.Numerator);


return(Answer);

}

//------------------------------------------------------------------------

//----Read. Fails if D is not natural

bool rational::Read(void) {
```

```
int NewNumerator,
   NewDenominator;

cin >> NewNumerator >> NewDenominator;

return(Set(NewNumerator,NewDenominator));
}
//--------------------------------------------------------------------------
//----Write
void rational::Write(void) {

cout << Numerator << "/" << Denominator;
}


Userational.cpp
#include <iostream>
#include "rationalclass.h"
#include"rationalclass.cpp"
//--------------------------------------------------------------------------
using namespace std;
int main(void) {

rational R1(2,3),
       R2,
       Sum,
       Subtract,
        Multiply,
        Divide;

cout << "R1 is ";
R1.Write();
cout << " and R2 is ";
R2.Write();
cout << endl;

cout << "Enter N and D for new R2 :";
if (!R2.Read())
    cout << "Doh, must have a positive denominator" << endl;

cout << "R2 is ";
R2.Write();
cout << endl;

Sum = R1.Add(R2);
cout << "Sum is ";
Sum.Write();
cout << endl;

Subtract=R1.Subtract(R2);
cout<< "Subtarction is";
Subtract.Write();
cout<<endl;
```

```
Multiply=R1.Multiply(R2);
cout<<"Multiplication is";
Multiply.Write();
cout<<endl;

Divide=R1.Divide(R2);
cout<<"Division is";
Divide.Write();
cout<<endl;



return(0);
}
```

Abstract Data Type for String and String Functions

astring.h

```
#ifndef RATIONALCLASS_H
#define RATIONALCLASS_H
#include <stdbool.h>
class astring
{
public:
   // Constructors
   astring(void);
   astring(char *s);
```

```
    // String Functions
    int length(void);
    void concat(astring s1, astring s2);
    void copy(astring s);
    bool compare(astring s);

    // display
    void display(void);
    // Read
    void Read(void);
    // Setter
    void Set(char *s, int len);

private:
    char str[1000];
    int len;
};
#endif

asttring.cpp
#include <bits/stdc++.h>
#include "astring.h"
using namespace std;
// Constructors
astring::astring(void){
    char temp[1000];

    Set(temp,0);

}


astring::astring(char* s){

    int c=0;

    while(str[c]!='\0')

        c++;

    Set(s,c);

}


// Setter

void astring::Set(char* s,int l){

    for(int i=0;i<l;i++)

    str[i]=s[i];

    len=l;

}


// length

int astring::length(){

    return len;

}


// Concatenate

void astring::concat(astring s1,astring s2){

    char temp[s1.len+s2.len];
```

```
   for(int i=0;i<s1.len;i++)

      temp[i]=s1.str[i];

   for(int i=s1.len;i<s1.len+s2.len;i++)

      temp[i]=s2.str[i-s1.len];

   Set(temp,s1.len+s2.len);

}


// Copy

void astring::copy(astring s1){

   Set(s1.str,s1.len);

}


bool astring::compare(astring s1){

   if(len==s1.len)

   {

      for(int i=0;i<len;i++)

         if(s1.str[i]!=str[i])

            return false;

      return true;


   }

   return false;

}


// Display

void astring::display(void){
    for(int i=0;i<len;i++)

        cout<<str[i];

}


// Read

void astring::Read(void){

    char temp[1000];

    cin>>temp;

    int c=0;

    while(temp[c]!='\0')

        c++;

    Set(temp,c);

}
```

Useastring.cpp

```
#include <bits/stdc++.h>

#include "astring.h"

using namespace std;

// Constructors

astring::astring(void){

    char temp[1000];

    Set(temp,0);

}
```

```
astring::astring(char* s){
    int c=0;
    while(str[c]!='\0')
        c++;
    Set(s,c);
}

// Setter
void astring::Set(char* s,int l){
    for(int i=0;i<l;i++)
    str[i]=s[i];
    len=l;
}

// length
int astring::length(){
    return len;
}

// Concatenate
void astring::concat(astring s1,astring s2){
    char temp[s1.len+s2.len];
    for(int i=0;i<s1.len;i++)
        temp[i]=s1.str[i];
    for(int i=s1.len;i<s1.len+s2.len;i++)
        temp[i]=s2.str[i-s1.len];
    Set(temp,s1.len+s2.len);
}

// Copy
void astring::copy(astring s1){
    Set(s1.str,s1.len);
}

bool astring::compare(astring s1){
    if(len==s1.len)
    {
        for(int i=0;i<len;i++)
            if(s1.str[i]!=str[i])
                return false;
        return true;

    }
    return false;
}

// Display
void astring::display(void){
    for(int i=0;i<len;i++)
        cout<<str[i];
}
```

```
// Read

void astring::Read(void){

   char temp[1000];

   cin>>temp;

   int c=0;

   while(temp[c]!='\0')

      c++;

   Set(temp,c);

}
```

Abstract Data Type for Complex and Complex Functions

Complexclass.h

```
#define COMPCLASS_H


#include <stdbool.h>


class comp {


public:


//----Constructors

comp(void);


comp(double real, double imaginary);
//----Set the value. Fails if D is not natural

bool Set(double real, double imaginary);


//----Add

comp Add(comp OtherOne);


//----Substract

comp Substract(comp OtherOne);


//----Multiply

comp Multiply(comp OtherOne);


//----Divide

comp Divide(comp OtherOne);


//----Read. Fails if D is not natural

bool Read(void);


//----Write

void Write(void);


private:


double r,i;


};
```

#endif

Complexclass.cpp

```
//-------------------------------------------------------------------------
#include <iostream>
#include<bits/stdc++.h>
#include "complexclass.h"

using namespace std;
//-------------------------------------------------------------------------
//----Constructors
comp::comp(void) {
Set(0,0);
}
//-------------------------------------------------------------------------
comp::comp(double real, double imaginary) {

Set(real,imaginary);
}
//-------------------------------------------------------------------------
//----Set the value. Fails if D is not natural
bool comp::Set(double real, double imaginary) {
   r = real;
   i = imaginary;
   return true;
```

```
}


//------------------------------------------------------------------------

//----Add

comp comp::Add(comp OtherOne) {


comp Answer;


Answer.Set(r+OtherOne.r,i+OtherOne.i);


return(Answer);

}


//------------------------------------------------------------------------

//----Substract

comp comp::Substract(comp OtherOne) {


comp Answer;


Answer.Set(r-OtherOne.r,i-OtherOne.i);


return(Answer);

}

//------------------------------------------------------------------------

//----Multiply

comp comp::Multiply(comp OtherOne) {
```

```
comp Answer;


Answer.Set((r*OtherOne.r) - (i*OtherOne.i), (i+OtherOne.r) + (r+OtherOne.i));


return(Answer);

}


//--------------------------------------------------------------------------

//----Divide

comp comp::Divide(comp OtherOne) {


comp Answer;


Answer.Set((((r*OtherOne.r)+(i*OtherOne.i))/((OtherOne.r)*(OtherOne.r))+((OtherOne.i)*(OtherOne.i)),


((i*OtherOne.r)+(r*OtherOne.i))/((OtherOne.r)*(OtherOne.r))+((OtherOne.i)*(OtherOne.i)));


return(Answer);

}

//--------------------------------------------------------------------------

//----Read. Fails if D is not natural

bool comp::Read(void) {


int NewNumerator,
    NewDenominator;


cin >> NewNumerator >> NewDenominator;


return(Set(NewNumerator,NewDenominator));

}

//--------------------------------------------------------------------------

//----Write

void comp::Write(void) {

cout << r << "+" << i <<"i";

}
```

userComplec.cpp

```
#include <iostream>

#include "complexclass.h"

#include "complexclass.cpp"


using namespace std;


int main(void) {


comp C1,

        C2,

        Sum,

        Diffrence,

        Product,
        Divide;



cout << "Enter real and imaginary for C1 : ";
if (!C1.Read())
        cout << "Doh, must have a positive denominator" << endl;

cout << "Enter real and imaginary for C2 : ";
if (!C2.Read())
        cout << "Doh, must have a positive denominator" << endl;

cout << "C1 is ";
C1.Write();
cout << endl;

cout << "C2 is ";
C2.Write();
cout << endl;

Sum = C1.Add(C2);
Diffrence = C1.Substract(C2);
Product = C1.Multiply(C2);
Divide = C1.Divide(C2);

cout << "Sum is ";
Sum.Write();
```

```
cout << endl;


cout << "Diffrence is ";

Diffrence.Write();

cout << endl;


cout << "Product is ";

Product.Write();

cout << endl;


cout << "Division is ";

Divide.Write();

cout << endl;


return(0);

}
```

Implementation Details:

1. Enlist all the Steps followed and various options explored

Ans: Learnt the use of structure implementation , Operator Overloading, ,Object concepts, etc. Most importantly, leant Abstract Data Types aka ADT with the comparer function the assigned data type.

2. Explain your program logic and methods used.

Ans: The Rational ADT takes in two values and performs basic mathematical operations on it using the following methods:

- **bool Set(int N, int D):** Sets the given values for numerator and denominator
- **rational Add(rational OtherOne) :** Adds two rational numbers
- **rational Subtract(rational OtherOne) :** Subtracts two rational numbers
- **rational Multiply(rational OtherOne) :** Multiplies two rational numbers
- **rational Divide(rational OtherOne)** : Divides two rational numbers
- **rational Simplify() :** Simplifies the given rational number
- **bool Read(void):** Reads the input values from the user
- **void Write(void):** Displays/Prints the value stored in the object

The String(Pstring) ADT takes in one value i.e character array and performs basic string functions on it using the following methods:

- **int length(void):** Returns the length of the string object.
- **void concat(pstring s1, pstring s2):** Concatenates two given strings and stores it inside the object of the third string.
- **void copy(pstring s):** Copies given string and stores it inside the object of the respective string.
- **bool compare(pstring s):** Returns true or false based on if the given string is equal to the string object.
- **void display(void):** Displays the string object
- **void Read(void):** Reads the String from the user
- **void Set(char *s, int len):** Sets the string to given character array and also stores it’s length.

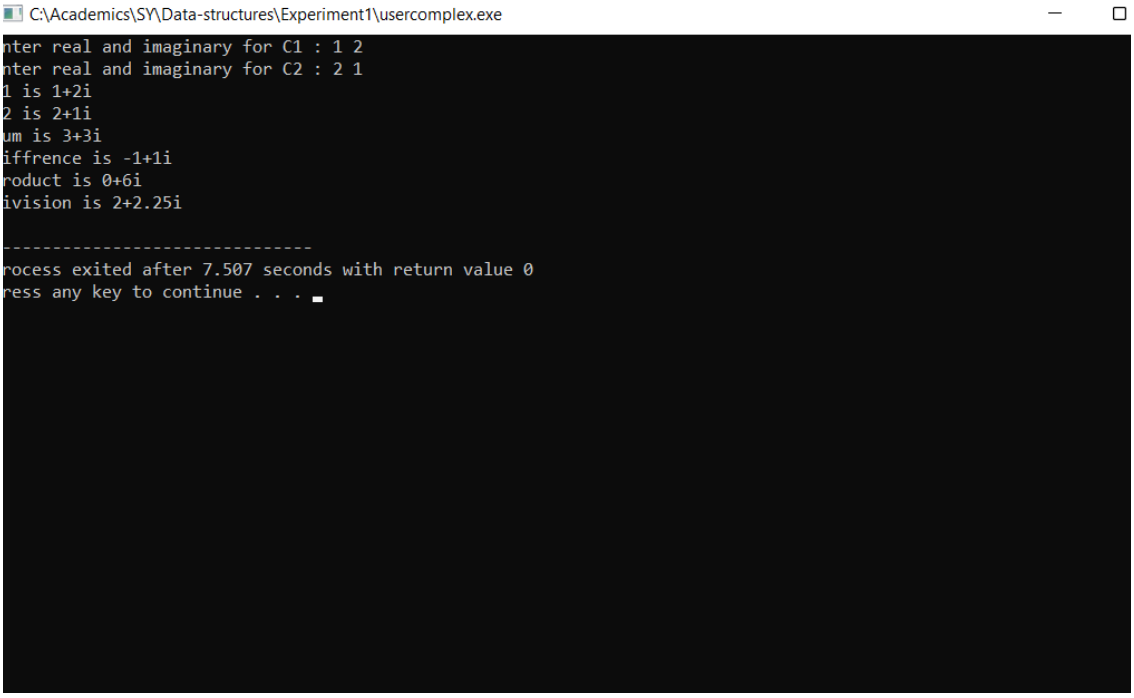
3. Explain the Importance of the approach followed by you.

Ans: The approach of creating a rational number is important because we learnt using ADT to implement different types of data structures by defining a set of functions or rules operating on them. We can use this concept to create more alike data types. The same goes for the Pstring (String) ADT which was created.

Program code and Output screenshots:

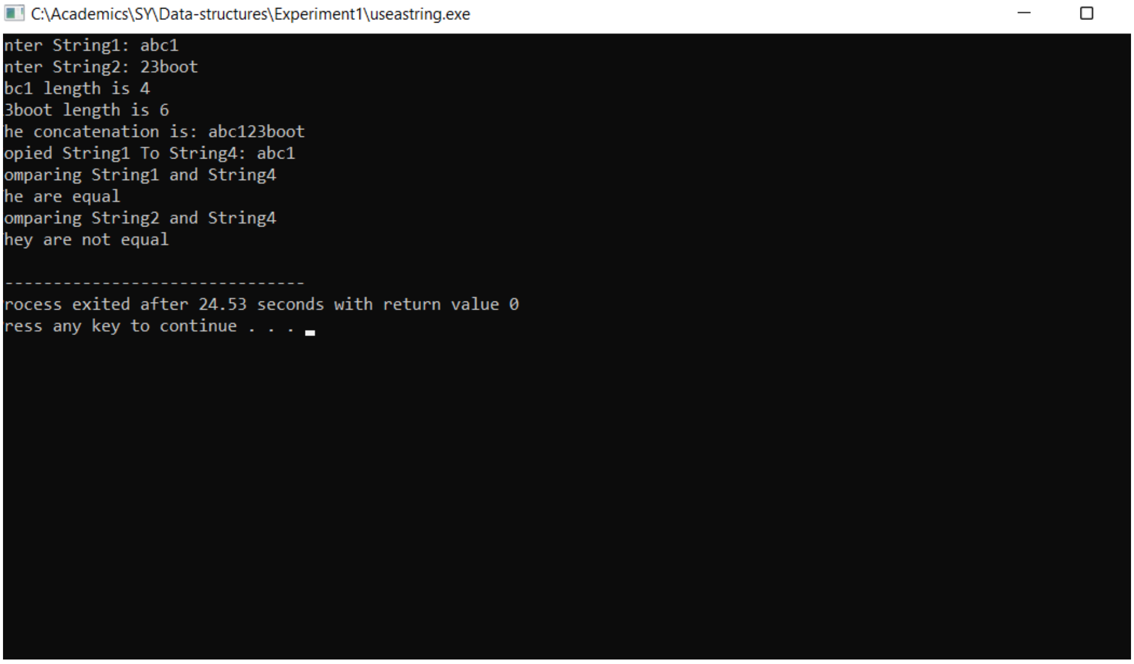
Conclusion:-

Hence Abstract Data Type is implemented and learnt using Rational Numbers and String functions.



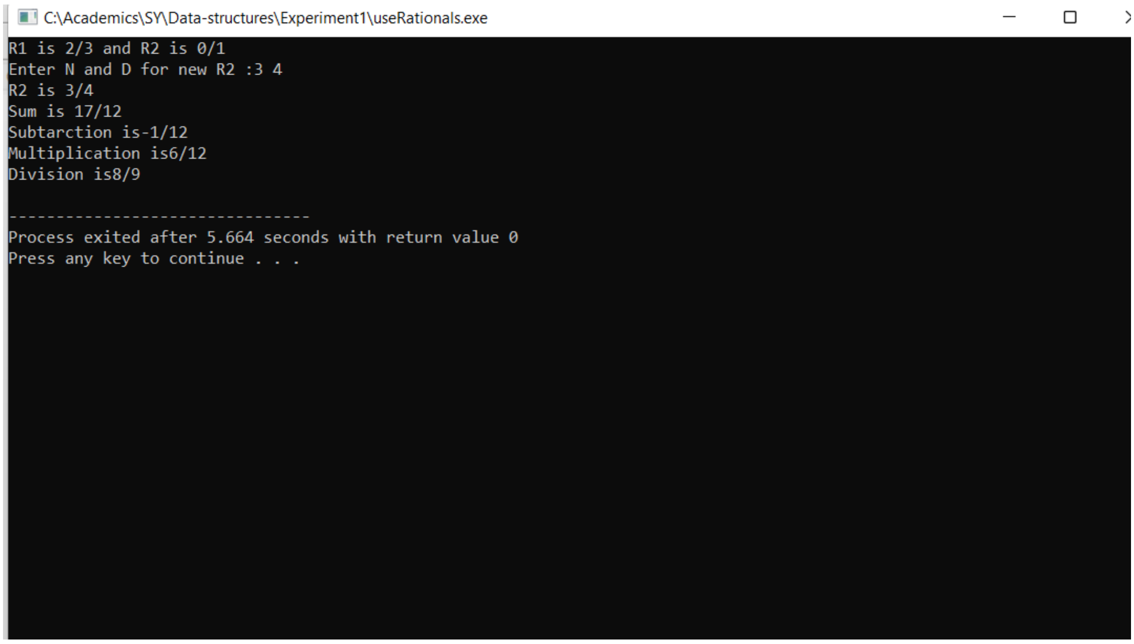
```
nter real and imaginary for C1 : 1 2
nter real and imaginary for C2 : 2 1
1 is 1+2i
2 is 2+1i
um is 3+3i
iffrence is -1+1i
roduct is 0+6i
ivision is 2+2.25i

-------------------------------
rocess exited after 7.507 seconds with return value 0
ress any key to continue . . .
```



```
nter String1: abc1
nter String2: 23boot
bc1 length is 4
3boot length is 6
he concatenation is: abc123boot
opied String1 To String4: abc1
omparing String1 and String4
he are equal
omparing String2 and String4
hey are not equal

-------------------------------
rocess exited after 24.53 seconds with return value 0
ress any key to continue . . .
```


C:\Academics\SY\Data-structures\Experiment1\useRationals.exe

```
R1 is 2/3 and R2 is 0/1
Enter N and D for new R2 :3 4
R2 is 3/4
Sum is 17/12
Subtarction is-1/12
Multiplication is6/12
Division is8/9

--------------------------------
Process exited after 5.664 seconds with return value 0
Press any key to continue . . .
```