Register Now!
Contact Us

Home Project Ideas » Training Programs New » Downloads » Campus Experience » Blog » Contact Us » Search... Go

Application Of Fraction As A Data Type

Code Id	40
Date Updated	11/7/2010
Title	Application of fraction as a data type

Description

```
This Fraction class allows extreme flexibility in working with fractions.  When a
simply the fraction that they are dealing with. Pre/Post conditions have been add
```

Attachments
Source Code fraction2.zip

Codes Snippet

```
//----------------------------------------------------------------
//
// F R A C T I O N  C L A S S
//
// F R A C T I O N . C P P
//
// C L A S S  D E S C R I P T I O N
// ------------------------------
//
// This file, provides a body to the functions declared in
// Fraction.h and must not be seperated from it, lest it stop
// functioning and benefit no one.
//
// This Fraction class allows extreme flexibility in working
// with fractions.  When a fraction is declared and the user
// and or client program implements a fraction with a negative
// denominator this class quickly distributes it to both the
// numinator and denominator to keep with what one would
// normally do with Fractions in a Math class.  Operators are
// defined to work with extreme ease for the user of this class.
// A Reduce fraction is included to allow the user to use to
// simply the fraction that they are dealing with.
//
// Pre/Post conditions have been added to simplify the Class.
//
//----------------------------------------------------------------
Fraction::Fraction()
        :num(0), den(1) // Sets default values for Fraction Class
{
        // Constructor has no need to initialize any other code
        // in it's current state
}
//-------------------------------------------------------
Fraction::~Fraction()
{
        // Deconstructor contains no code at the moment
}
//-------------------------------------------------------
void Fraction::SetNum(BIGGEST_INT newnum)
//pre : Call to set the numerator
//post: new numerator is set
{
        num = newnum;
}

//-------------------------------------------------------
void Fraction::SetDen(BIGGEST_INT newden)
// pre : Call to set the deminator
// post: Validate if negative and set new denominator
{
        den = newden;
        if(den < 0) {
                num *= -1;
                den *= -1;
        }
}
```

Online Enquiry

Course Registration

Recent Posts

- Types of Cloud Computing
- What is Cloud Computing ?
- How to pass a multi-dimensional array to a function?
- Memory Layout of a C Program
- PHP and Its Advantages

```
        if(den < 0) { // Choke out negative denominators as soon as they are inpu
                num *= -1;
                den *= -1;
        }
}
//-------------------------------------------------------
BIGGEST_INT Fraction::Num()
// pre: any call made to use numerator in Fraction class
{
        return(num);
}
//-------------------------------------------------------
BIGGEST_INT Fraction::Den()
// pre: any call made to use denominator in Fraction class
{
        return(den);
}
//-------------------------------------------------------
void Fraction::Reduce()
// pre : Function is called with a Fraction to reduce
// post: Function is reduced as far as possible
{
        BIGGEST_INT rdc = 0; // Reduce value
        if(den > num) // Makes sure the greater number is sent to the Function fi
                rdc = GCD(den, num);
        else if(den < num)
                rdc = GCD(num, den);
        else
                rdc = GCD(num, den);
        num /= rdc; // Simplifies Fraction's Numerator
        den /= rdc; // Simplifies Fraction's Denominator
}
//-------------------------------------------------------
void Fraction::DisplayMixed()
{
        cout << "Mixed value: " << num / den << " " << num % den << "/";
        if(den < 0) /* Don't let the denominator of the mixed number appear negat
                                        if the value is negative it's displayed i
                cout << den * -1;
        else
                cout << den;
        cout << endl;
}
//-------------------------------------------------------
float FracDecVal(Fraction a)
// pre: Function is given one fraction to find the decimal value of
// post: Function returns the decimal value of given fraction
{
        float fracnum = a.num; // Make numerator a floater to prevent loss of data
        float fracden = a.den; // Make denominator a floater to prevent loss of d
        float decval = fracnum / fracden; // Decimal value variable
        return(decval);
}
//-------------------------------------------------------
BIGGEST_INT Fraction::GCD(int num1, int remainder)
// pre : Function is given two values (numerator and denominator)
// post: Function returns the greatest common factor to client
{
        if(remainder == 0)
                return(num1);
        else {
                return(GCD(remainder,num1%remainder));
        }
}
//-------------------------------------------------------
Fraction Fraction::operator +(Fraction d)
// pre : Function is given two valid Fractions to add
// post: Function returns the added value of the Fractions as is
{
        Fraction retfrac;
        retfrac.num = (num * d.den) + (d.num * den);
        retfrac.den = d.den * den;
        return(retfrac);
}
//-------------------------------------------------------
Fraction Fraction::operator -(Fraction d)
// pre : Function is given two valid Fractions to subtract
// post: Function returns the subtracted value of the Fraction as is
{
```

Register Now!
Contact Us

Home Project Ideas » Training Programs New » Downloads » Campus Experience » Blog » Contact Us »

Search... Go

```
        retfrac.num = num * d.num;
        retfrac.den = den * d.den;
        return(retfrac);
}
//-------------------------------------------------------
Fraction Fraction::operator /(Fraction d)
// pre : Function is given two valid Fractions to divide
// post: Function returns the divided value of the Fraction as is
{
        Fraction retfrac;
        retfrac.num = num * d.den;
        retfrac.den = den * d.num;
        if(retfrac.den < 0) { // Just in case the inversion caused the denominato
                retfrac.num *= -1;
                retfrac.den *= -1;
        }
        return(retfrac);
}
//-------------------------------------------------------
Fraction Fraction_Do_Op(int operation, Fraction a, Fraction b)
// pre : Function is given operation and two Fractions
// post: Function completes the operation and returns to client
{
        Fraction rc;
        switch (operation) {
        case FRAC_ADD:
                return (a + b);
        case FRAC_SUB:
                return (a - b);
        case FRAC_MUL:
                return (a * b);
        case FRAC_DIV:
                return (a / b);
        }
        rc.SetNum(1);
        rc.SetDen(1);
        return (rc);
}
//-------------------------------------------------------
```