СибГУТИ

Контрольная работа по дисциплине «Современные технологии программирования 2 »

Новосибирск
15.12.2018

Содержание

Тема

Проектирование и реализация программ в технологии «абстрактных типов данных» и объектно-ориентированного программирования.

Цель

Сформировать практические навыки:

- *проектирования программ в технологии «абстрактных типов данных» и «*объектно-ориентированного программирования*» и построения диаграмм UML;*
- *реализации абстрактных типов данных с помощью классов C#;*
- *использования библиотеки визуальных компонентов VCL для построения интерфейса,*
- *тестирования программ.*

Задание

Спроектировать и реализовать приложение под Windows в соответствии с вариантом, используя классы C# и библиотеку визуальных компонентов для построения интерфейса.

Варианты заданий:

Номер Варианта	Тема	Настройка вида числа	Окно справки
1	Калькулятор простых дробей.	Всегда дробь.	Нет.
2	Калькулятор простых дробей.	Целое или дробь.	Да.
3	Калькулятор простых дробей.	Всегда дробь.	Нет.

4	Калькулятор простых дробей.	Целое или дробь.	Нет.
5	Калькулятор комплексных чисел	Всегда комплексное.	Нет.
6	Калькулятор комплексных чисел	Действительное или комплексное.	Да.
7	Калькулятор комплексных чисел	Всегда комплексное.	Да.
8	Калькулятор комплексных чисел	Действительное или комплексное	Нет
9	Калькулятор чисел в системе счисления с выбранным основанием	Нет.	Нет.
10	Калькулятор чисел в системе счисления с выбранным основанием	Нет.	Да.

Общие требования

Калькулятор обеспечивает вычисление операций: +, -, *. /. Предусмотреть возможность ввода операндов в выражение:

- с помощью командных кнопок интерфейса,
- клавиатуры: цифровой и алфавитно-цифровой.

3. Необходимо реализовать команду (=). которая завершает вычисление выражения. Она выполняет текущую операцию.

4. Интерфейс может иметь следующий вид.

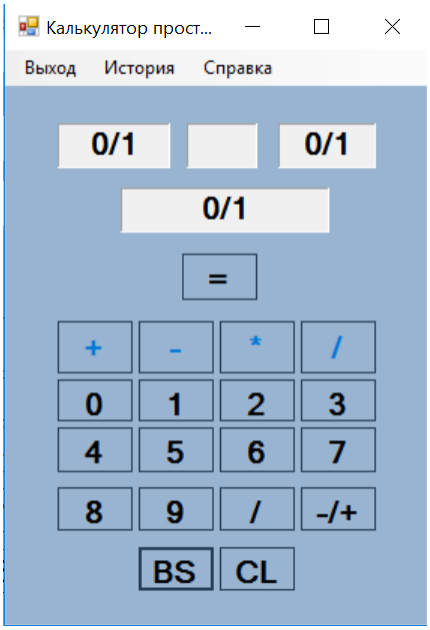


Рис. 1. Интерфейс калькулятора простых дробей.

Приложение должно иметь основное окно для ввода исходных данных, операций и отображения результата и окно для вывода сведений о разработчиках приложения (в зависимости от варианта задания).

6. Вводимые числа выравнивать по правому краю.

7. Для редактирования вводимых значений необходимы команды:

- BackSpase (удалить крайний справа символ отображаемого числа),
- CE (заменить отображаемое число нулевым значением)
- Добавить символ, допустимый в изображении числа (арабские цифры, знак, разделители).

Тема – «Калькулятор чисел в системе счисления с выбранным основанием».

Требования.

1. Калькулятор обеспечивает работу с действительными числами в системах счисления с основанием в диапазоне от 2 до 16.

2. Основание системы счисления – настраиваемый параметр. Настройку можно установить в основном окне или добавить в меню «Настройка».

3. Исходные числа и результат вводятся и выводятся в формате фиксированная точка [-]<p - ичное целое без знака><разделитель>< p - ичная дробь без знака >

Например: 2.34 или -5.0 или 0.456

4. Кнопки для ввода цифровой информации необходимо связать с используемой системой счисления. Для пользователя необходимо сделать доступными кнопки только для ввода цифр используемой системы счисления.

Необходимо предусмотреть следующие варианты (прецеденты) использования калькулятора:

1. Выполнение одиночных операций:
   «операнд1» «операция» «операнд2» «=» «результат»
   Пример. 5.0 + 2.0 = 7.0 (p = 10)

Тема – «Калькулятор простых дробей».

Требования.

1. Калькулятор должен обеспечить ввод и редактирование целых чисел в обычной записи и рациональных дробей в записи:
   [-]<целое без знака>|[-]<числитель><разделитель><знаменатель>.
   <числитель>::= <целое без знака>
   <знаменатель>::= <целое без знака>
   <разделитель>::= ‘/’ | ‘|’
   Например: 1 или ½ или -1|2.

2. Предусмотреть настройку калькулятора на отображение результата в двух форматах: «всегда дробь» или «целое или дробь» (в зависимости от варианта задания). В формате «дробь» результат всегда отображается в виде дроби. В формате «целое или дробь о» результат отображается в виде числа, если дробь может быть сокращена, так что знаменатель равен 1.

Необходимо предусмотреть следующие варианты использования (прецеденты) калькулятора:

1. Выполнение одиночных операций:

   «операнд1» «операция» «операнд2» «=» «результат»

   Пример. 5/1 + 2/1 = 7/1.

Тема – «Калькулятор комплексных чисел».

Требования.

1. Калькулятор обеспечивает ввод комплексных чисел в записи:

[-]<действительная часть><разделитель>[-] <мнимая часть>

<действительная часть>::= <действительное число без знака с целой и\или дробной частями>

<мнимая часть>::= <действительное число без знака с целой и\или дробной частями>

<разделитель>::= ‘i*’

Напимер: -2.35 +i* 3.5 или -2.0 -i* 3.0

2. Предусмотреть настройку калькулятора на отображение результата в двух форматах: «всегда комплексное» или «действительное или комплексное» число (в зависимости от варианта задания). В формате «всегда комплексное» результат всегда отображается в виде комплексного числа. В формате

«действительное или комплексное» результат отображается в виде действительного, если мнимая часть равна 0.

Необходимо предусмотреть следующие варианты использования калькулятора (прецеденты):

1. Выполнение одиночных операций:

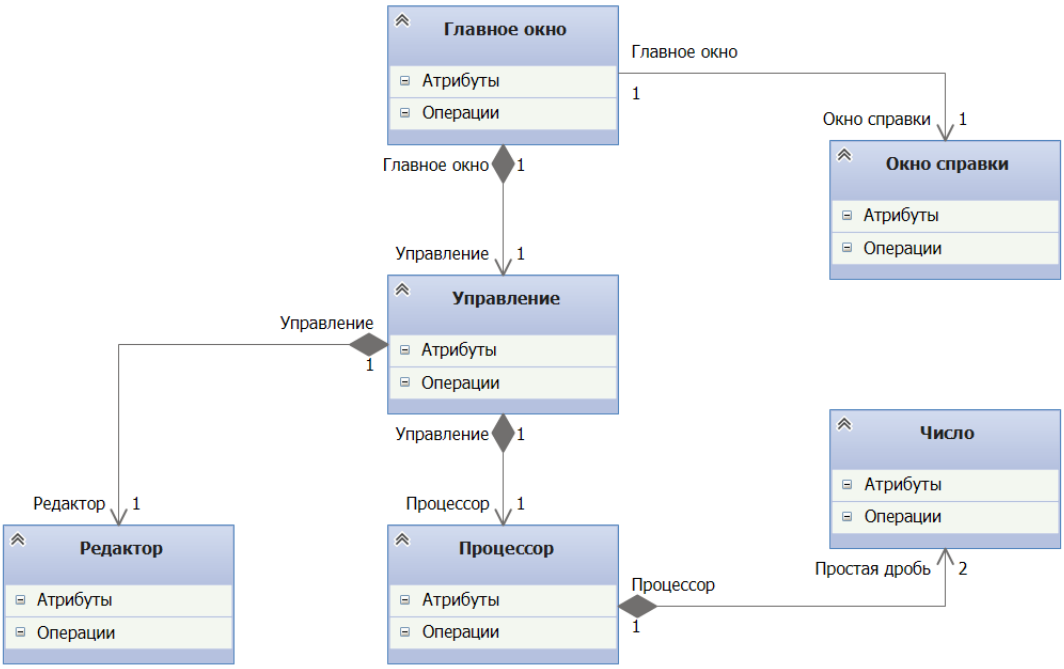
   «операнд1» «операция» «операнд2» «=» «результат»

   Пример. 5.0+i*2.0 + 2.0+i*3.0= 7.0+i*5.0.

Рекомендации к выполнению

1. Для выполнения Контрольной работы, в зависимости от варианта, используйте классы, разработанные вами в практических занятиях:
2. Разработка класса для конвертации чисел из системы счисления с основанием 10 в систему счисления с основанием p (p от 0до 16).
3. Разработка класса редактор чисел в системе счисления с основанием p (p от 0до 16) .
4. Разработка класса управление для конвертора чисел из системы счисления с основанием p (p от 0до 16) в с.сч с основанием 10.
5. Разработка графического интерфейса для конвертера.

6. Диаграмма классов UML для калькулятора представлена на рисунке.



Здесь класс число в зависимости от варианта может быть: р-ичное число, простая дробь, комплексное число.

3. Описание остальных классов, которые войдут в приложение (Управление, Процессор), приведены ниже и описаны как лабораторные работы. Вам их необходимо выполнить и включить в курсовой проект.
4. Последовательность действий пользователя при работе с калькулятором следующая:
   - После запуска приложения калькулятор переходит в состояние «`Состояния.ОперацияВыполнена`». Пользователь вводит левый операнд (для ввода и редактирования обоих операндов и отображения результата используется один и тот же компонент класса Label (метка) поименованный в программе как «`Ввод_и_Результат`»). При вводе первого символа левого операнда калькулятор переходит в состояние «`Состояния.ВводЛевогоОперанда`»;
   - Пользователь выбирает операцию. Введённый операнд отображается в метке по имени «`ЛевыйОперанд`», операция отображается в метке по имени «`Операция`» справа от метки «`ЛевыйОперанд`». Калькулятор переходит в состояние «`Состояния.ВводПравогоОперанда`».
   - Пользователь вводит правый операнд, который отображается в метке по имени «`ПравыйОперанд`» справа от метки по имени «`Операция`»;
   - Пользователь вводит команду «=» (выполнить операцию). Операция выполняется и результат отображается в метке по

имени «Ввод_и_Результат». Калькулятор переходит в состояние «Состояния.ОперацияВыполнена».

Для построения главного окна приложения используйте пример описания класса формы главного окна для Калькулятора простых дробей, приведённый на рисунке ниже:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FracCalculate
{
    public partial class Form1 : Form
    {
        const string zero = "0/1";
        //Управление;
        Control Cntrl = new Control();
        public Form1()
        {
            InitializeComponent();
        }
            //Установка начальных значений на форме:
        public void UpForm()
        {
            //Левый операнд.
            ЛевыйОперанд.Text = zero;
            //Правый операнд.
            ПравыйОперанд.Text = zero;
            //Операция.
            Операция.Text = "";
            //Результат
            Ввод_и_Результат.Text = zero;
        }
        private void Form1_Load(object sender, EventArgs e)
        {
             UpForm();
           //Изменяем состояние калькулятора.
            Cntrl.состояние = Состояния.ОперацияВыполнена;
        }

        //Команда выбрать арифметическую операцию.
        private void operation_Click(object sender, EventArgs e)
        {

            Button b = (Button)sender;
            //Выполняем команду взависимости от состояния калькулятора.
            switch (Cntrl.состояние)
            {
                case Состояния.ВводЛевогоОперанда:
                    {
```

```
                        //Заносим в процессор знак операции и отображаем её
на форме.
                        Cntrl.Operation = b.Text;
                        Операция.Text = Cntrl.Operation;
                        //Заносим в процессор левый операнд и отображаем
его на форме.
                        Cntrl.LeftOperand = Cntrl.Number;
                        ЛевыйОперанд.Text = Cntrl.LeftOperand;
                        //Очищаем редактор и поле для ввода.
                        Ввод_и_Результат.Text = Cntrl.DoEdit(18);
                        //Изменяем состояние калькулятора.
                        Cntrl.состояние = Состояния.ВводПравогоОперанда;
                        break;
                    }
                case Состояния.ВводПравогоОперанда:
                    {
                        //Заносим в процессор знак операции и отображаем её
на форме.
                        Cntrl.Operation = b.Text;
                        Операция.Text = Cntrl.Operation;
                        //Изменяем состояние калькулятора.
                        Cntrl.состояние = Состояния.ВводПравогоОперанда;
                        break;
                    }
            }
        }
        //Команда выполнить арифметическую операцию.
        private void do_operation_Click(object sender, EventArgs e)
        {
            //Выполняем команду взависимости от состояния калькулятора.
            switch (Cntrl.состояние)
            {
                case Состояния.ВводПравогоОперанда:
                    {
                        //Заносим в процессор правый операнд и отображаем
его на форме.
                        Cntrl.RightOperand = Cntrl.Number;
                        ПравыйОперанд.Text = Cntrl.RightOperand;
                        //Выполняем операцию и отображаем её на форме.
                        Ввод_и_Результат.Text = Cntrl.DoOperation();
                        //Устанавливаем состояние - операция выполнена.
                        Cntrl.состояние = Состояния.ОперацияВыполнена;
                        break;
                    }
            }
        }
        //Команды редактирования числа.
        private void common_Click(object sender, EventArgs e)
        {
            //Рассматриваем источник события sender, как командную кнопку.
            Button b = (Button)sender;
            //Определяем номер команды, соответствующий нажатой кнопке.
            int cmd = int.Parse(b.Tag.ToString());
            //Выполняем команду взависимости от состояния калькулятора.
            switch (Cntrl.состояние)
            {
                case Состояния.ОперацияВыполнена:
                    {
                        Cntrl.состояние = Состояния.ВводЛевогоОперанда;
```

```
                        //Очистить поле для ввода числа.
                        UpForm();
                        Ввод_и_Результат.Text = Cntrl.DoEdit(18);
                        break;
                    }
            }
            //Выполняем команду редактирования и отображаем результат.
            Ввод_и_Результат.Text = Cntrl.DoEdit(cmd);
        }

        private void Form1_KeyDown(object sender, KeyEventArgs e)
        {

        }

        private void toolStripMenuItem1_Click(object sender, EventArgs e)
        {
            //Закрываем окно формы и приложение.
            Close();
        }


        private void справкаToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            //Создаём окно справки.
            About w = new About();
            //Показываем окно справки.
            w.Show();

        }
    }
}
```

Содержание отчета

1. Задание.
2. Диаграмма прецедентов UML. Сценарии прецедентов.
3. Диаграмма последовательностей для прецедентов.
4. Диаграмма классов для прецедентов.
5. Спецификации к типам данных (классам приложения).
6. Текст программы.
7. Тестовые наборы данных для тестирования абстрактных типов данных, классов и приложения.
8. Инструкция пользователю.
9. Литература.

Литература

1. Павловская Т.А. C#. Программирование на языке высокого уровня :Учебник для вузов. - СПб. : Питер, 2014. - 432 с. : ил. - (Серия "Учебник для вузов").

2. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.0 на языке C# . 3-е изд.: - СПб.:Питер, 2012. - 928 с. : ил.

Дополнительная литература

В печатном виде

1.Вендров А.М. Проектирование программного обеспечения экономических информационных систем. Учебник. — 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2005. 544 с.: ил.

2.Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем: Учебное пособие для вузов. СПб.: Питер, 2003. – 472с.: ил.

3.Г. Буч. Объектно-ориентированное проектирование с примерами приложений на C++, 2-ое издание. Учебник/: Пер. с англ. - М.: Издательство Бином, СПб.: Невский Диалект, 1999. - 560с.

4.Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер. С англ. - М.: Конкорд, 1992. - 519 с., ил.

5. Фаулер М., Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования: Пер с англ.- М.:Мир,1999.-191с.:ил.

Интернет ресурсы

1. Справочник по C# [Электронный ресурс] URL: https://msdn.microsoft.com/ru-ru/library/618ayhy6.aspx (дата обращения 12.08.16)
2. METANIT.COM. Сайт о программировании [Электронный ресурс] URL: http://metanit.com/sharp/tutorial/ (дата обращения 12.08.16)
3. Моделирование на UML [Электронный ресурс] URL: http://book.uml3.ru/ (дата обращения 17.02.2015)

4. Объектно-ориентированный анализ и проектирование [Электронный ресурс] URL:http://ooad.asf.ru/(дата обращения 25.02.2015)
5. SRC-CODE.NET [Электронный ресурс] URL:http://src-code.net/(дата обращения 09.04.2016)
6. Хабрахабр [Электронный ресурс] URL: https://habrahabr.ru/ (дата обращения 09.04.2016)
7. Informicus [Электронный ресурс] URL: http://www.informicus.ru/Default.aspx?SECTION=6&id=73 (дата обращения 22.04.16)
8. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 285 с.— Режим доступа: http://www.iprbookshop.ru/39552.— ЭБС «IPRbooks», по паролю

Лабораторная работа. Абстрактный тип данных р - ичное число

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C#.

Задание

1. Реализовать абстрактный тип данных «р-ичное число», используя класс C#, в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных, одним из методов тестирования.

Спецификация типа данных «р-ичное число».

ADT TPNumber

Данные

Р-ичное число TPNumber - это действительное число (n) со знаком в системе счисления с основанием (b) (b в диапазоне 2..16), содержащее целую и дробную части. Точность представления числа с (с >= 0). Р-ичные числа изменяемые.

Операции

Операции могут вызываться только объектом р-ичное число (тип TPNumber), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «само число».

Операция	***Описание***
Конструктор	
Начальные значения:	Вещественное число (a) во внутреннем формате, система счисления (b), точность представления числа (c)

Процесс:	Инициализирует поля p-ичного числа: система счисления (b), точность представления (c). В поле (n) созданного числа заносится (a). Например: Конструктор(a,3,3) = число a в системе счисления 3 с тремя разрядами после троичной точки. Конструктор(a,3,2) = число a в системе счисления 3 с двумя разрядами после троичной точки.
Конструктор	
Начальные значения:	Строковое представление p–ичного числа (a), система счисления (b), точность представления числа (c)
Процесс:	Инициализирует поля p-ичного числа: система счисления (b), точность представления (c). В поле (n) созданного числа заносится результат преобразования строки (a) в числовое представление. b-ичное число (a) и основание системы счисления (b) представлены в формате строки. Например: Конструктор(‘20’,’3’,’6’) = 20 в системе счисления 3, точность 6 знаков

	после запятой. Конструктор(‘0’,’3’,’8’) = 0 в системе счисления 3, точность 8 знаков после запятой.
Копировать:	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт копию самого числа (тип TPNumber).
Выход:	p-ичное число.
Постусловия:	Нет.
Сложить	
Вход:	P-ичное число d с основанием и точностью такими же, как у самого числа.
Предусловия:	Нет.
Процесс:	Создаёт и возвращает p-ичное число (тип TPNumber), полученное сложением полей (n) самого числа и числа d.
Выход:	p-ичное число.
Постусловия:	Нет
Умножить	
Вход:	P-ичное число d с основанием и точностью такими же, как у самого числа.
Предусловия:	Нет.

Процесс:	Создаёт и возвращает p-ичное число (тип TPNumber), полученное умножением полей (n) самого числа и числа d.
Выход:	P-ичное число (тип TPNumber).
Постусловия:	Нет.
Вычесть	
Вход:	P-ичное число d с основанием и точностью такими же, как у самого числа.
Предусловия:	Нет.
Процесс:	Создаёт и возвращает p-ичное число (тип TPNumber), полученное вычитанием полей (n) самого числа и числа d.
Выход:	P-ичное число (тип TPNumber).
Постусловия:	Нет.
Делить	
Вход:	P-ичное число d с основанием и точностью такими же, как у самого числа.
Предусловия:	Поле (n) числа (d) не равно 0.
Процесс:	Создаёт и возвращает p-ичное число (тип TPNumber), полученное делением полей (n) самого числа на поле (n) числа d.
Выход:	P-ичное число (тип TPNumber).
Постусловия:	Нет.
Обратить	
Вход:	Нет.

Предусловия:	Поле (n) самого числа не равно 0.
Процесс:	Создаёт p-ичное число, в поле (n) которого заносится значение, полученное как 1/(n) самого числа.
Выход:	P-ичное число (тип TPNumber).
Постусловия:	Нет.
Квадрат	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт p-ичное число, в поле (n) которого заносится значение, полученное как квадрат поля (n) самого числа.
Выход:	P-ичное число (тип TPNumber).
Постусловия:	Нет.
ВзятьРЧисло	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение поля (n) самого числа.
Выход:	Вещественное значение.
Постусловия:	Нет.
ВзятьРСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает p-ичное число (q) в формате строки, изображающей значение поля (n)

	самого числа в системе счисления (b) с точностью (c).
Выход:	Строка.
Постусловия:	Нет.
ВзятьОснованиеЧисло	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение поля (b) самого числа (q).
Выход:	Целочисленное значение
Постусловия:	Нет.
ВзятьОснованиеСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение поля (b) самого числа в формате строки, изображающей (b) в десятичной системе счисления.
Выход:	Строка.
Постусловия:	Нет.
ВзятьТочностьЧисло	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение поля (c) самого числа .
Выход:	Целое значение.
Постусловия:	Нет.

ВзятьТочностьСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение поля (c) самого числа в формате строки, изображающей (c) в десятичной системе счисления.
Выход:	Строка.
Постусловия:	Нет.
УстановитьОснованиеЧисло	
Вход:	Целое число (newb).
Предусловия:	2 <= newb <= 16.
Процесс:	Устанавливает в поле (b) самого числа значение (newb).
Выход:	Нет.
Постусловия:	Нет.
УстановитьОснованиеСтрока	
Вход:	Строка (bs), изображающая основание (b) р-ичного числа в десятичной системе счисления.
Предусловия:	Допустимый диапазон числа, изображаемого строкой (bs) - 2,,16.
Процесс:	Устанавливает значение поля (b) самого числа значением, полученным в результате

	преобразования строки (bs).
Выход:	Строка.
Постусловия:	Нет.
УстановитьТочностьЧисло	
Вход:	Целое число (newc).
Предусловия:	newc >= 0.
Процесс:	Устанавливает в поле (c) самого числа значение (newc).
Выход:	Нет.
Постусловия:	Нет.
УстановитьТочностьСтрока	
Вход:	Строка (newc).
Предусловия:	Строка (newc) изображает десятичное целое >= 0.
Процесс:	Устанавливает в поле (c) самого числа значение, полученное преобразованием строки (newc).
Выход:	Нет.
Постусловия:	Нет.

end TPNumber

Рекомендации к выполнению

1. Тип данных реализовать, используя класс C#.
2. Число храните как поле вещественного типа.

3. Основание системы счисления храните как поле целочисленного типа.
4. Тип данных реализовать в отдельном модуле UPNumber.

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования типа данных.

Контрольные вопросы

1. Что такое инкапсуляция?
2. Как синтаксически представлено поле в описании класса?
3. Как синтаксически представлен метод в описании класса?
4. В чём состоит назначение конструктора?

Лабораторная работа. Абстрактный тип данных простая дробь

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C#.

Задание

1. Реализовать абстрактный тип данных «простая дробь», используя класс C# в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных одним из методов тестирования.

Спецификация типа данных «простые дроби».

ADT TFrac

Данные

Простая дробь (тип TFrac) - это пара целых чисел: числитель и знаменатель (a/b). Простые дроби изменяемые.

Операции

Операции могут вызываться только объектом простая дробь (тип **TFrac**), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «сама дробь».

Конструктор	
Начальные значения:	Пара целых чисел (a) и (b).
Процесс:	Инициализирует поля простой дроби (тип TFrac): числитель значением a, знаменатель - (b). В случае необходимости дробь предварительно сокращается. Например: *Конструктор*(6,3) = (2/1) *Конструктор*(0,3) = (0/3).
Конструктор	
Начальные значения:	Строковое представление простой дроби . Например: ‘7/9’.
Процесс:	Инициализирует поля простой дроби (тип TFrac) строкой f =’a/b’. Числитель значением a, знаменатель - b. В случае необходимости дробь предварительно сокращается. Например:

	Конструктор(‘6/3’) = 2/1 *Конструктор* (‘0/3’) = 0/3
Копировать:	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт копию самой дроби (тип TFrac) с числителем, и знаменателем такими же, как у самой дроби.
Выход:	Простая дробь (тип TFrac). Например: c = 2/1, Копировать(c) = 2/1
Постусловия:	Нет.
Сложить	
Вход:	Простая дробь d (тип TFrac).
Предусловия:	Нет.
Процесс:	Создаёт и возвращает простую дробь (тип TFrac), полученную сложением самой дроби q = a1/b1 с d =a2/b2: ((a1/b1)+(a2/b2)=(a1*b2 + a2*b1)/(b1* b2)). Например: q = 1/2, d = -3/4 q.Сложить(d) = -1/4.
Выход:	Простая дробь (тип TFrac).
Постусловия:	Нет.

Умножить	
Вход:	Простая дробь d (тип TFrac).
Предусловия:	Нет.
Процесс:	Создаёт простую дробь (тип TFrac), полученную умножением самой дроби q = a1/b1 на d = a2/b2 ((a1/b1)*(a2/b2)=(a1* a2)/(b1* b2)).
Выход:	Простая дробь (тип TFrac).
Постусловия:	Нет.
Вычесть	
Вход:	Простая дробь d (тип TFrac).
Предусловия:	Нет.
Процесс:	Создаёт и возвращает простую дробь (тип TFrac), полученную вычитанием d = a2/b2 из самой дроби q = a1/b1: ((a1/b1)-(a2/b2)=(a1* b2-a2*b1)/(b1*b2)). Например: q = (1/2), d = (1/2) q.Вычесть(d) = (0/1).
Выход:	Простая дробь (тип TFrac).
Постусловия:	Нет
Делить	
Вход:	Простая дробь d (тип TFrac).
Предусловия:	Числитель числа d не равно 0.

Процесс:	Создаёт и возвращает простую дробь (тип TFrac), полученное делением самой дроби q = a1/b1 на дробь d = a2/b2: ((a1/b1)/(a2/b2)=(a1* b2)/(a2*b1)).
Выход:	Простая дробь (тип TFrac).
Постусловия:	Нет.
Квадрат	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт и возвращает простую дробь (тип TFrac), полученную умножением самой дроби на себя: ((a/b)*(a/b)=(a* a)/(b* b)).
Выход:	Простая дробь (тип TFrac).
Постусловия:	Нет.
Обратное	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт и возвращает простую дробь (тип TFrac), полученное делением единицы на саму дробь: 1/((a/b) = b/a.
Выход:	Простая дробь (тип TFrac)
Постусловия:	Нет.
Минус	

Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт простую дробь, являющуюся разностью простых дробей z и q, где z - простая дробь (0/1), дробь, вызвавшая метод.
Выход:	Простая дробь (тип TFrac).
Постусловия:	Нет.
Равно	
Вход:	Простая дробь d (тип TFrac).
Предусловия:	Нет
Процесс:	Сравнивает саму простую дробь q и d. Возвращает значение True, если q и d - тождественные простые дроби, и значение False - в противном случае.
Выход:	Булевское значение.
Постусловия:	Нет.
Больше	
Вход:	Простая дробь d (тип TFrac).
Предусловия:	Нет.
Процесс:	Сравнивает саму простую дробь q и d. Возвращает значение True, если q > d, - значение False - в противном случае.
Выход:	Булевское значение.
Постусловия:	Нет.

ВзятьЧислительЧисло	
Вход:	
Предусловия:	Нет.
Процесс:	Возвращает значение числителя дроби в числовом формате.
Выход:	Вещественное значение.
Постусловия:	Нет.
ВзятьЗнаменательЧисло	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение знаменателя дроби в числовом формате.
Выход:	Вещественное значение.
Постусловия:	Нет.
ВзятьЧислительСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение числителя дроби в строковом формате.
Выход:	Строка.
Постусловия:	Нет.
ВзятьЗнаменательСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение знаменателя

	дроби в строковом формате.
Выход:	Строка.
Постусловия:	Нет.
ВзятьДробьСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение простой дроби, в строковом формате.
Выход:	Строка.
Постусловия:	Нет.

end TFracRatio

Рекомендации к выполнению

1. Тип данных реализовать, используя класс C#.
2. Для записи и считывания полей простой дроби использовать свойства (property).
3. Тип данных реализовать в отдельном модуле UFrac.

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования типа данных.

Контрольные вопросы

1. Особенности описания методов класса?
2. Особенности описания и назначение конструктора класса?
3. Видимость идентификаторов в описании класса?
4. Особенности вызова методов применительно к объектам класса?

5. Что такоё абстрактный тип данных?

Лабораторная работа. Абстрактный тип данных комплексное число

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C++.

Задание

1. Реализовать абстрактный тип данных «комплексное число», используя класс C#, в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных одним из методов тестирования.

Спецификация типа данных «комплексное число».

ADT TComplex

Данные Комплексное число TComplex - это изменяемая пара вещественных чисел, представляющие действительную и мнимую части комплексного числа (a + i*b).

Операции

Операции могут вызываться только объектом комплексное число (тип TComplex), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «само число».

Конструктор	
Начальные значения:	Пара вещественных чисел (a) и (b).
Процесс:	Инициализирует поля комплексного

	числа (тип TComplex) значениями: действительную часть - a), мнимую - b. Например: *Конструктор*(6,3)=6 + i*3 *Конструктор*(3,0)=3 + i*0 *Конструктор*(0,0)=0 + i*0
Конструктор	
Начальные значения:	Строка, представляющая комплексное число.
Процесс:	Инициализирует поля комплексного числа (тип TComplex) значениями представленными строкой f = ’a + i*b’: действительную частью значением a, комплексную часть - b. Например: *Конструктор*(‘6+i*3’) = 6+i*3 *Конструктор*(‘0+i*3’) = 0+i*3
Копировать:	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Создаёт и возвращает собственную копию - комплексное число (тип TComplex) с действительной и мнимой частями такими же как у самого числа.
Выход:	Комплексное число (тип TComplex).

	Например: c = 6+i3, Копировать(c) = 6+i3
Постусловия:	Нет.
Сложить	
Вход:	Комплексное число d (тип TComplex).
Предусловия:	Нет.
Процесс	Создаёт и возвращает комплексное число, полученное сложением самого числа q = a1+i*b1 с числом d = a2+i*b2: ((a1+i*b1)+(a2+i*b2)=(a1+a2)+i*(b1+b2)). Например: q = (2 +i*1), d = (2 +i*1), q.Сложить(d) = (4 +i*2).
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Умножить	
Вход:	Комплексное число d (тип TComplex).
Предусловия:	Нет.
Процесс	Создаёт и возвращает комплексное число, полученное умножением самого числа q = a1+i*b1 на число d = a2+i*b2: ((a1+i*b1)*(a2+i*b2)=(a1*a2 - b1*b2)+i*(a1*b2+ a2*b1)).
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.

Квадрат	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Создаёт и возвращает комплексное число (тип TComplex), полученное умножением числа на самого себя: ((a1+i*b1)*(a1+i*b1)=(a1*a1 - b1*b1)+i*(a1*b1+ a1*b1)).
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Обратное	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Создаёт и возвращает комплексное число (тип TComplex), полученное делением единицы на само число 1/((a1+i*b1) = a1/(a1**2 + b1**2) - i* b1/(a1**2 + b1**2)).
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Вычесть	
Вход:	Комплексное число d (тип TComplex)..
Предусловия:	Нет.
Процесс	Создаёт и возвращает комплексное число (тип TComplex), полученное

	вычитанием d = a2 + i b2 из самого себя q = (a1+i*b1): (a1+i*b1)-(a2+i*b2)=(a1-a2)+i*(b1-b2). Например: q = (2 +i*1), d = (2 +i*1)) q.Вычесть(d) = (0 + i0).
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Делить	
Вход:	Комплексное число (d).
Предусловия:	Нет.
Процесс	Создаёт и возвращает комплексное число (тип TComplex), полученное делением самого числа (q) на число (d) ((a1+i*b1)/(a2+i*b2)=(a1*a2 + b1*b2)/(a2**2 + b2**2)+i*(a2*b1 – a1*b2)/(a2**2 + b2**2)).
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Минус	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Создаёт и возвращает комплексное число (тип TComplex), являющееся разностью комплексных чисел z и и самого числа, где z – комплексное

	число (0+i0).
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Модуль	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Вычисляет и возвращает модуль самого комплексного числа (q). Например: q = (2 +i*1), q. Модуль = √(2*2+1*1). q = (i*17), q. Модуль = √(0*0+17*17).
Выход:	Вещественное число.
Постусловия:	Нет.
УголРад	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Возвращает аргумент fi самого комплексного числа q (в радианах). fi = (arcTg(b/a), a>0; pi/2, a = 0, b > 0; arcTg(b/a) + pi, a < 0; -pi/2, a = 0, b <0). Например: q = (1 +i*1), q. УголРад = 0,79.
Выход:	Вещественное число.
Постусловия:	Нет.

УголГрад	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Возвращает аргумент самого комплексного числа q (в градусах). Например: q = (1 +i*1), q. Град = 45.
Выход:	Вещественное число.
Постусловия:	Нет.
Степень	
Вход:	Целое (n).
Предусловия:	Нет.
Процесс	Возвращает целую положительную степень n самого комплексного числа q. $q^n = r^n(\cos(n*fi)+ i*\sin(n*fi))$.
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Корень	
Вход:	Целое (n), целое (i).
Предусловия:	Нет.
Процесс	Возвращает i-ый корень целой положительной степени n самого комплексного числа q. $\sqrt[n]{(q)} = \sqrt[n]{(r)}*(\cos((fi + 2*k*pi)/n)+ i*\sin((fi + 2*k*pi)/n))$. При этом коэффициенту k придается последовательно n значений: k =

	0,1,2…, n - 1 и получают n значений корня, т.е. ровно столько, каков показатель корня.
Выход:	Комплексное число (тип TComplex).
Постусловия:	Нет.
Равно	
Вход:	Комплексное число (d).
Предусловия:	Нет.
Процесс	Сравнивает само комплексное число с числом (d). Возвращает значение True, если они - тождественные комплексные числа, и значение False - в противном случае.
Выход:	Булевское значение.
Постусловия:	Нет.
НеРавно	
Вход:	Комплексное число (d).
Предусловия:	Нет.
Процесс	Сравнивает само комплексное число с числом (d). Возвращает значение True, если само число <> d, - значение False - в противном случае.
Выход:	Булевское значение.
Постусловия:	Нет.
ВзятьReЧисло	

Вход:	Нет
Предусловия:	Нет.
Процесс	Возвращает значение действительной части самого комплексного числа в числовом формате.
Выход:	Вещественное значение.
Постусловия:	Нет.
ВзятьImЧисло	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Возвращает значение мнимой части самого комплексного числа в числовом формате.
Выход:	Вещественное значение.
Постусловия:	Нет.
ВзятьReСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Возвращает значение вещественной части самого комплексного числа в строковом формате.
Выход:	Строка.
Постусловия:	Нет.
ВзятьImСтрока	
Вход:	Нет.

Предусловия:	Нет.
Процесс	Возвращает значение мнимой части самого комплексного числа в строковом формате.
Выход:	Строка.
Постусловия:	Нет.
ВзятьКомплексноеСтрока	
Вход:	Нет.
Предусловия:	Нет.
Процесс	Возвращает значение самого комплексного числа в строковом формате.
Выход:	Строка.
Постусловия:	Нет.

end TComplex

Рекомендации к выполнению

1. Тип данных реализовать, используя класс.
2. Для описания полей комплексного числа использовать свойства (property).
3. Тип данных реализовать в отдельном модуле UComplex.

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования типа данных.

Контрольные вопросы

1. Что такое интерфейс класса?
2. Что такое реализация класса?
3. Особенности описания и назначение деструктора класса?
4. Особенности описания и назначение конструктора копирования?
5. Особенности вызова методов применительно к объектам класса?
6. Что такое сообщение применительно к объектам класса?

Лабораторная работа. Редактор р-ичных чисел

Цель

Сформировать практические навыки реализации классов средствами объектно-ориентированного программирования C++.

Задание

1. Разработать и реализовать класс TEditor «Редактор р-ичных чисел», используя класс C#.

На Унифицированном языке моделирования UML (Unified Modeling Language) наш класс можно описать следующим образом:

РедакторР-ичныхЧисел
строка: String числоЕстьНоль: Boolean добавитьЗнак: String добавитьР-ичную цифру(a: Integer): String добавитьНоль: String забойСимвола: String очистить: String конструктор читатьСтрокаВформатеСтроки: String (метод свойства)

писатьСтрокаВформатеСтроки(a: String) (метод свойства) редактировать(a: Integer): String
Обязанность: ввод, хранение и редактирование строкового представления р-ичных чисел

2. Класс должен отвечать за ввод и редактирование строкового представления р-ичных чисел. Значение р-ичного нуля - ‘0,’. Класс должен обеспечивать:
   - добавление символов, соответствующих р-ичным цифрам (р от 2 до 16);
   - добавление и изменение знака;
   - добавление разделителя целой и дробной частей;
   - забой символа, стоящего справа (BackSpace);
   - установку нулевого значения числа (Clear);
   - чтение строкового представления р-ичного числа;
   - запись строкового представления р-ичного числа;
3. Протестировать каждый метод класса.

Рекомендации к выполнению

1. В классе TEditor опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление редактируемого р- ичного числа, .
2. В классе опишите следующие операции:
   - «число есть ноль», операция возвращает булевское значение True, если «строка» содержит изображение числа равного 0, False – в противном случае;
   - «добавить знак», операция добавляет или удаляет знак «-» из «строка» и возвращает значение «строка»;

- «добавить р-ичную цифру», операция получает целое число (числовое обозначение р-ичной цифры), преобразует его в символ и добавляет к «строка», если это допускает формат, возвращает значение «строка»;
- «добавить ноль», операция добавляет ноль к «строка», если это допускает формат, возвращает значение «строка»;
- «забой символа», операция удаляет крайний правый символ «строка» и возвращает значение «строка»;
- «очистить», операция устанавливает в «строка» строку, изображающую р-ичный 0, возвращает значение «строка»;
- «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
- «конструктор», создаёт объект типа TEditor;
- «читать «строка» в формате строки» - строкового типа (метод свойства), возвращает значение «строка» в заданном пользователем формате;
- «писать «строка» в формате строки», получает значение строкового типа (метод свойства) и заносит его в «строка»;

3. Класс реализуйте в отдельном модуле UEditor. В разделе описания констант опишите следующие константы:
   - «разделитель целой и дробной частей» строкового типа;
   - «строковое представление нуля» строкового типа.

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования класса.

Контрольные вопросы

1. В чём состоит особенность раздела описания класса с уровнем доступа protected?
2. В чём состоит особенность раздела описания класса с уровнем доступа private?
3. В чём состоит особенность раздела описания класса с уровнем доступа public?
4. В чём состоит особенность инициализации полей ссылочного типа и констант в конструкторе?
5. Что такое указатель this?
6. Что такое статические элементы класса?

Лабораторная работа. Редактор простых дробей

Цель

Сформировать практические навыки реализации классов средствами объектно-ориентированного программирования C++.

Задание

1. Разработать и реализовать класс TEditor «Ввод и редактирование простых дробей», используя класс C++.

На Унифицированном языке моделирования UML (Unified Modeling Language) наш класс можно обозначить следующим образом:

РедакторПростыхДробей
строка: String
добавитьРазделитель: String добавитьЗнак: String добавитьЦифру(a: Integer): String добавитьНоль: String

забойСимвола: String очистить: String конструктор читатьСтрокаВформатеСтроки: String (метод свойства) писатьСтрокаВформатеСтроки(a: String) (метод свойства) редактировать(a: Integer): String
Обязанность: ввод, хранение и редактирование строкового представления простых дробей.

2. Класс должен отвечать за посимвольный ввод, хранение и редактирование строкового представления простых дробей. Значение нуля - ‘0|1’. Класс должен обеспечивать:
   - добавление цифры;
   - добавление и изменение знака;
   - добавление разделителя между числителем и знаменателем;
   - забой символа, стоящего справа (BackSpace);
   - установку нулевого значения числа (Clear);
   - чтение строкового представления простой дроби;
   - запись строкового представления простой дроби.
3. Протестировать каждый метод класса.

Рекомендации к выполнению

1. В классе TEditor опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление редактируемой простой дроби.
2. В классе опишите следующие операции:
   - «добавить разделитель», операция возвращает строковое значение, полученное добавлением разделителя между числителем и знаменателем дроби;

- «добавить знак», операция добавляет или удаляет знак «-» из «строка» и возвращает значение «строка»;
- «добавить цифру», операция получает целое число (числовое обозначение арабской цифры), преобразует его в символ и добавляет к «строка», если это допускает формат, возвращает значение «строка»;
- «добавить ноль», операция добавляет ноль к «строка», если это допускает формат, возвращает значение «строка»;
- «забой символа», операция удаляет крайний правый символ «строка» и возвращает значение «строка»;
- «очистить», операция устанавливает в «строка» строку, изображающую дробь 0/1, возвращает значение «строка»;
- «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
- «конструктор», создаёт объект типа TEditor;
- «читать «строка» в формате строки» - строкового типа (метод свойства), возвращает значение «строка» в заданном пользователем формате;
- «писать «строка» в формате строки», получает значение строкового типа (метод свойства) и заносит его в «строка»;

3. Класс реализуйте в отдельном модуле UEditor. В разделе описания констант опишите следующие константы:
   - «разделитель числителя и знаменателя» строкового типа;
   - «строковое представление нуля» строкового типа.

Содержание отчета

1. Задание.
2. Текст программы.

3. Тестовые наборы данных для тестирования класса.

Контрольные вопросы

1. В чём состоят особенности статических полей?
2. В чём состоят особенности статических методов?
3. В чём состоит перегрузка операций для класса?
4. В чём состоит особенность константных методов?

Лабораторная работа. Редактор комплексных чисел

Цель

Сформировать практические навыки реализации классов средствами объектно-ориентированного программирования C++.

Задание

1. Разработать и реализовать класс «Ввод и редактирование комплексных чисел» (TEditor), используя класс C++.

На Унифицированном языке моделирования UML (Unified Modeling Language) наш класс можно обозначить следующим образом:

РедакторКомплексныхЧисел
строка: String добавитьРазделительВещественногоМнимого: String добавитьРазделителmЦелогоДробнлго: String добавитьЗнак: String добавитьЦифру(a: Integer): String добавитьНоль: String забойСимвола: String очистить: String конструктор читатьСтрокаВформатеСтроки: String (метод свойства)

писатьСтрокаВформатеСтроки(a: String) (метод свойства) редактировать(a: Integer): String
Обязанность: ввод, хранение и редактирование строкового представления комплексных чисел

2. Класс должен отвечать за посимвольный ввод, хранение и редактирование строкового представления комплексных чисел. Значение комплексного нуля - ‘0, i* 0,’. Класс должен обеспечивать:
   - добавление цифры;
   - добавление и изменение знака действительной и мнимой частей;
   - добавление разделителя целой и дробной частей действительной и мнимой частей комплексного числа;
   - добавление разделителя мнимой и действительной частей комплексного числа
   - забой символа, стоящего справа (BackSpace);
   - установку нулевого значения комплексного числа (Clear);
   - чтение строкового представления комплексного числа;
   - запись строкового представления комплексного числа.
3. Протестировать каждый метод класса.

Рекомендации к выполнению

1. В классе TEditor опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление редактируемого комплексного числа, .
2. В классе опишите следующие операции:

- «число есть ноль», операция возвращает булевское значение True, если «строка» содержит изображение комплексного числа равного 0, +i 0,, False – в противном случае;
- «добавить знак», операция добавляет или удаляет знак «-» из «строка» и возвращает значение «строка»;
- «добавить цифру», операция получает целое число (числовое обозначение арабской цифры), преобразует его в символ и добавляет к «строка», если это допускает формат, возвращает значение «строка»;
- «добавить ноль», операция добавляет ноль к «строка», если это допускает формат, возвращает значение «строка»;
- «забой символа», операция удаляет крайний правый символ «строка» и возвращает значение «строка»;
- «очистить», операция устанавливает в «строка» строку, изображающую комплексное число 0, +i 0,, возвращает значение «строка»;
- «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
- «конструктор», создаёт объект типа TEditor;
- «читать «строка» в формате строки» - строкового типа (метод свойства), возвращает значение «строка» в заданном пользователем формате;
- «писать «строка» в формате строки», получает значение строкового типа (метод свойства) и заносит его в «строка»;

3. Класс реализуйте в отдельном модуле UEditor. В разделе описания констант опишите следующие константы:

- «разделитель целой и дробной частей действительной и мнимой частей комплексного числа» - строкового типа;
- «разделитель действительной и мнимой частей комплексного числа» - строкового типа;
- «строковое представление нуля» - строкового типа.

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования класса.

Контрольные вопросы

1. Когда в классе необходимо явным образом описать деструктор?
2. Что такое конструктор по умолчанию?
3. Когда в классе необходимо явным образом описать конструктор копирования?

Лабораторная работа. Процессор чисел (в зависимости от варианта)

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C#.

Задание

1. Реализовать класс «Процессор», используя класс C#, в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных, одним из методов тестирования.

Процессор предназначен для хранения двух операндов и знака операции и выполнения операции над операндами.

Спецификация типа «Процессор».

ADT Proc

Данные

Два поля A,B для хранения левого и правого операндов соответственно (тип поля зависит от варианта использования). Значения полей - изменяемые. Поле op для хранения текущей операции. Значение поля – изменяемое.

Операции

Операции могут вызываться только объектом тип Процессор (тип Proc), указатель на который в них передаётся по умолчанию.

Операция	***Описание***
Конструктор без параметров	
Начальные значения:	Нет
Процесс:	Инициализирует поля A,B нулевыми значениями. В поле op заносит пустую строку.
Установить операцию	
Вход:	Строка s, содержащие символ операции.
Предусловия:	Нет.
Процесс:	Заносит в поле op строку s.
Выход:	Нет.
Постусловия:	Нет

Прочитать операцию	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает содержимое поля op.
Выход:	Строка.
Постусловия:	Нет.
ЗаписатьA	
Вход:	Число N.
Предусловия:	Нет.
Процесс:	Заносит в поле A число N.
Выход:	Нет.
Постусловия:	Нет.
ЗаписатьB	
Вход:	Число N.
Предусловия:	Нет.
Процесс:	Заносит в поле B число N.
Выход:	Нет.
Постусловия:	Нет.
ЧитатьA	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Читает число из поля A.
Выход:	Число.
Постусловия:	Нет.

ЧитатьB	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Читает число из поля B.
Выход:	Число.
Постусловия:	Нет.
ВыполнитьОперацию	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Выполняет операцию, хранящуюся в поле op над числами A и B и возвращает результат.
Выход:	Число.
Постусловия:	Нет.

end Proc

Рекомендации к выполнению

1. Тип данных реализовать, используя класс C#.
2. Поля A,B, op реализуйте как публичные свойства.
3. Тип данных реализовать в отдельном модуле Proc.

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования типа данных.

Контрольные вопросы

1. Что такое инкапсуляция?
2. Как синтаксически представлено поле в описании класса?
3. Как синтаксически представлен метод в описании класса?

4. В чём состоит назначение конструктора?

Лабораторная работа. Управление калькулятором (в зависимости от варианта)

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C#.

Задание

1. Реализовать класс «Управление», используя класс C#, в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных, одним из методов тестирования.

Управление включает в себя редактор, процессор и состояние калькулятора. Оно предназначено для выполнения команд процессора и редактора.

Спецификация класса «Управление».

ADT Control

Данные

Поле E (Editor), P (Proc) для хранения редактора и процессора соответственно (редактор и процессор зависят от варианта использования), поле состояние типа Состояние. Значения полей - изменяемые. Тип Состояния – перечисляемый тип с тремя значениями.

```
enum Состояния { ОперацияВыполнена, ВводЛевогоОперанда, ВводПравогоОперанда }
```

Калькулятор может находиться в одном из этих состояний. Допустимые команды, которые может выполнить пользователь, определяются состоянием калькулятора.

Операции

Операции могут вызываться только объектом тип Управление (тип ***Control***), указатель на который в них передаётся по умолчанию.

Операция	***Описание***
Конструктор без параметров	
Начальные значения:	Нет
Процесс:	Инициализирует поля P,E создавая объекты соответствующих типов. В поле «состояние» заносит значение ОперацияВыполнена.
Установить левый операнд. Свойство `LeftOperand`	
Вход:	Строка, содержащая число (зависит от варианта).
Предусловия:	Нет.
Процесс:	Создаёт объект число (зависит от варианта) и заносит в поле A процессора.
Выход:	Нет.
Постусловия:	Нет
Читать левый операнд. Свойство `LeftOperand`	

Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает содержимое поля A процессора P.
Выход:	Строка.
Постусловия:	Нет.
Установить правый операнд. Свойство `RightOperand`	
Вход:	Строка, содержащая изображение числа число (зависит от варианта).
Предусловия:	Нет.
Процесс:	Создаёт объект число (зависит от варианта) и заносит в поле B процессора.
Выход:	Нет.
Постусловия:	Нет
Читать правый операнд. Свойство `RightOperand`	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает содержимое поля B процессора P.
Выход:	Строка.
Постусловия:	Нет.
Читать поле op (операция) процессора P.	

Свойство `Operation`	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает содержимое поля op процессора P.
Выход:	Строка.
Постусловия:	Нет.
Установить op (операция) процессора P. Свойство `Operation`	
Вход:	Строка, содержащая знак арифметической операции.
Предусловия:	Нет.
Процесс:	Заносит строку в поле op процессора P.
Выход:	Нет.
Постусловия:	Нет
Читать поле Number редактора E. Свойство Number	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Читает число из поля ***Number редактора E*** .
Выход:	Строка.
Постусловия:	Нет.
DoEdit (выполнить команду редактора)	

Вход:	Целое cmd– номер команды редактора.
Предусловия:	Нет.
Процесс:	Вызывает операцию редактора E. DoEdit(cmd) .
Выход:	Строка.
Постусловия:	Нет.
ВыполнитьОперацию (`DoOperation()`)	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Вызывает операцию DoOperation() процессора P.
Выход:	Строка.
Постусловия:	Нет.

end Control

Рекомендации к выполнению

1. Тип данных реализовать, используя класс C#.
2. Свойства Состояния, LeftOperand, RightOperand, Operation, Number реализуйте как публичные свойства.
3. Тип данных реализовать в отдельном файле Control.

Описание класса Control может выглядеть следующим образом:

```
    //Набор возможных состояний калькулятора.
    enum Состояния { ОперацияВыполнена, ВводЛевогоОперанда,
ВводПравогоОперанда }
    class Control
    {

        //Процессор.
        Proc P;
```

```
        //Редактор.
        Editor E;
        //Свойство для чтения и записи состояние калькулятора.
        public Состояния состояние { get; set; }
        //Свойство для чтения и записи левого операнда
процессора (поле P.A).
        public string LeftOperand { get { return
P.A.ToString(); } set { P.A = new Frac(value); } }
        //Свойство для чтения и записи правого операнда
процессора (поле P.B).
        public string RightOperand { get { return
P.B.ToString(); } set { P.B = new Frac(value); } }
        //Свойство для чтения и записи операции процессора
(поле P.B).
        public string Operation { get { return P.op; } set {
P.op = value; } }
        //Свойство для чтения и записи редактируемого числа
свойство Number редактора.
        public string Number { get {return E.Number; } }

        public Control()
        {
            P = new Proc();
            E = new Editor();
            состояние = Состояния.ОперацияВыполнена;
        }
        //Выполнить команду редактирования по её номеру c.
        public string DoEdit(int cmd)
        {
            return E.DoEdit(cmd);
        }
        //Выполнить текущую операцию процессора P.
        public string DoOperation()
        {
            return P.DoOprtn().ToString();
        }
    }
```

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования типа данных.

Контрольные вопросы

5. Что такое инкапсуляция?
6. Как синтаксически представлено поле в описании класса?
7. Как синтаксически представлен метод в описании класса?
8. В чём состоит назначение конструктора?