Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Современные технологии программирования 2

Приложение “Конвертор р1\_р2”

Выполнил: студенты 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Панченко Д.Б.

Скирда А.А

Проверил: ассистент кафедры ПМиК

Агалаков Антон Александрович

Новосибирск, 2021

**Содержание**

1. Задание.
   1. Практическая работа. Конвертор чисел из десятичной системы счисления в систему счисления с заданным основанием.
   2. Практическая работа. Класс Конвертер p\_10 – преобразователь чисел из системы счисления с основанием p в десятичную систему счисления.
   3. Практическая работа. Редактор чисел в системе счисления с основанием p.
   4. Практическая работа. Класс История.
   5. Практическая работа. Класс Управление для конвертора p1\_p2.
   6. Практическая работа. Интерфейс приложения Конвертор p1\_p2.
2. Текст программы.
3. Тестовые наборы данных.

**Задание**

Приложение должно обеспечивать пользователю:

- преобразование действительного числа представленного в системе счисления с основанием p1 в число, представленное в системе счисления с основанием p2; основания систем счисления p1, p2 для исходного числа и результата преобразования выбираются пользователем из диапазона от 2..16;

-возможность ввода и редактирования действительного числа представленного в системе счисления с основанием p2 с помощью командных кнопок и мыши, а также с помощью клавиатуры;

-контекстную помощь по элементам интерфейса и справку о назначении приложения; просмотр истории сеанса (журнала) работы пользователя с приложением – исходные данные, результат преобразования и основания систем счисления, в которых они представлены;

-дополнительные повышенные требования: автоматический расчёт необходимой точности представления результата.

Разработка приложения разбита на практические работы, в каждой из которых реализуется один или несколько логически связанных классов приложения.

**Практическая работа.**

**Конвертор чисел из десятичной системы счисления в систему счисления с заданным основанием**

**Задание**

* + 1. Реализовать преобразователь действительных чисел со знаком из десятичной системы счисления в систему счисления с заданным основанием p, в соответствии с приведенной ниже спецификацией, используя класс. Основание системы счисления p принадлежит диапазону значений от 2 до 16.
    2. Протестировать каждый метод класса.

Спецификация класса «Преобразователь чисел из десятичной системы счисления в систему счисления с заданным основанием p».

# ADT Conver\_10\_p

**Данные**

Преобразователь действительных чисел из десятичной системы счисления в систему счисления с заданным основанием (тип Conver\_10\_p). Основание системы счисления p - это целое число, со значением, принадлежащим диапазону от 2 до 16 и целое число c, определяющее точность представления результата, выраженную в количестве разрядов.

**Операции.** Операции представлены в таблице ниже.

|  |  |
| --- | --- |
| **Do(double n, int p,**  **int c)** | *Выполнить преобразование* |
| Вход: | Десятичное действительное число n. Основание системы счисления p. Точность преобразования дроби, заданная числом разрядов дробной части результата c. Например:  Do(-17.875,16,3) = “-А1.Е” . |
| Процесс: | Выполняет преобразование десятичного действительного числа n, в систему счисления с основанием р и точностью c.  Например: ***Do*** (”-17.875”,16,3) = “-А1.Е”. |
| Выход: | Строка результата.  Например: ***Do***(”-17.875”) = “-А1.Е”. |
| **int\_to\_Char(int d)** | Преобразовать целое значение в цифру системы  счисления с основанием р. |
| Вход: | d – значение типа int – целое, соответствующее цифре  в системе счисления с основанием р. |
| Предусловия: | Нет. |
| Процесс: | Преобразует целое d в соответствующую ему цифру в системе счисления с основанием р, значение типа Char.  Например: int\_to\_Char (14) = “E”. |

|  |  |
| --- | --- |
| Выход: | Значение типа char. |
| Постусловия: | Нет. |
| **int\_to\_P(int n, int p)** | Преобразовать целое в строку. |
| Вход: | n – целое число в системе счисления с основанием 10.  p – основание системы счисления результата. |
| Предусловия: | Нет. |
| Процесс: | Преобразует целое n в строку, содержащую целое число в системе счисления с основанием p. Например:  *int\_to\_P(161, 16) = “A1”* |
| Выход: | Строка. |
| Постусловия: | Нет. |
| **flt\_to\_P(double n, int**  **p, int c)** | Преобразовать дробь в строку. |
| Вход: | n – дробь в системе счисления с основанием 10, p –  основание системы счисления, c – точность представления дроби. |
| Предусловия: | Нет. |
| Процесс: | Преобразует дробь n в строку, содержащую дробь в  системе счисления с основанием p с точностью с. Например: flt\_to\_P(0.9375, 2, 4) «1111» |
| Выход: | Строка. |
| Постусловия: | Нет. |

# end Conver\_10\_p

# Практическая работа.

# Класс Конвертер p\_10 – преобразователь чисел из системы счисления с основанием p в десятичную систему счисления.

Задание

* 1. Реализовать преобразователь действительных (конвертер р\_10) чисел из системы счисления с основанием р в десятичную систему счисления в соответствии с приведенной ниже спецификацией, используя класс. Основание системы счисления р принадлежит диапазону значений от 2 до 16.
  2. Протестировать каждый метод класса.

Спецификация класса «Конвертер р\_10» - преобразователь действительных чисел со знаком из системы счисления с основанием р в десятичную систему счисления.

# ADT Conver\_p\_10 Данные

Преобразователь действительных чисел из заданной системы счисления с основанием p в десятичную систему счисления (тип Conver\_p\_10). Основание системы счисления со значением, принадлежащим диапазону от 2 до 16.

# Операции

Операции приведены в таблице ниже.

|  |  |
| --- | --- |
| **dval(string P\_num, int**  **P)** | ***Выполнить преобразование*** |
| Вход: | P\_num - строковое представление действительного числа в системе счисления с основанием р. Например:  dval(“A5.E”, 16) |
| Процесс: | Выполняет преобразование действительного числа, представленного строкой в числовое представление.  Например:  dval(“A5.E”, 16) = -165.875. |
| Выход: | Вещественное число. |
| Постусловия: | Нет. |

|  |  |
| --- | --- |
| **char\_To\_num(char ch)** | ***Преобразовать символ в целое*** |
| Вход: | ch – значение типа char – символ, изображающий  цифру системы счисления с основанием p. |
| Предусловия: | Нет. |
| Процесс: | Преобразует символ ch в значение целого типа.  Например:  ***PCharToInt(‘A’)*** = 10. |
| Выход: | Вещественное число. |
| Постусловия: | Нет. |
| **convert(string P\_num,**  **int P, double weight)** | ***Преобразовать строку в вещественное число.*** |
| Вход: | P\_num – строка, изображающая цифры целой и дробной частей вещественного числа в системе счисления с основанием p без разделителя. weight –  вес единицы старшего разряда целой части числа. |
| Предусловия: | Нет. |
| Процесс: | Преобразует строку P\_num, содержащую цифры целой и дробной частей вещественного числа в системе счисления с основанием p без разделителя в вещественное число. Например:  convert (“A5E1”, 16, 16) |
| Выход: | Вещественное число. |
| Постусловия: | Нет. |

# end Conver\_p\_10

**Практическая работа.**

**Редактор чисел в системе счисления с основанием p.**

1. Разработать и реализовать класс Editor «Редактор действительных чисел представленных в системе счисления с основанием р», используя класс языка высокого уровня. Основание системы счисления р принимает значение из диапазона 2..16. Все команды редактора удобно пронумеровать, начиная с команды добавить 0 целыми числами от 0. При реализации интерфейса номера команд удобно хранить в свойстве Tag, которое имеется у визуальных компонентов.

Атрибуты и операции класс представлены ниже.

|  |
| --- |
| **Редактор** |
| -число: String; |
| +знак(): string;  +разделитель(): String;  +цифра(n: integer): String  +ноль(): String;  +забой(): String;  +очистить(): String;  +редактировать(n: integer): String;  +читать(): String;  +писать(): String; |

1. Ответственность класса Editor (редактор) – хранение, ввод и редактирование строкового представления числа, представленного в системе счисления с основанием p. Класс должен обеспечивать: добавление символов (AddDigit), соответствующих р-ичным цифрам (р от 2 до 16); добавления нуля (AddZero()); добавление разделителя целой и дробной частей (AddDelim()); забой символа - удаление символа, стоящего справа (BS); очистку - установку нулевого значения числа (Clear); чтение строкового представления р-ичного числа (Number).

Протестировать каждый метод класса

**Практическая работа.**

**Класс История**

1. Разработать и реализовать класс History «История», используя класс языка C#. Класс отвечает за документирование выполнения пользователем переводов чисел. Объекты класса хранят исходные числа, результаты преобразования и основания систем счисления исходного числа и результата.

Атрибуты и операции класс представлены ниже.

|  |
| --- |
| **История** |
| Запись(i: integer): String; |
| ДобавитьЗапись(a: String); |
| Записей(): integer |
| ОчиститьИсторию(); |
| Обязанность: ввод, вывод, хранение данных введённых пользователем и  полученных результатов. |

1. Класс должен отвечать за ввод, вывод, хранение данных введённых пользователем и полученных результатов. Класс должен обеспечивать:
   * добавление записи (ДобавитьЗапись) - строки, содержащей введённое пользователем число, результат его преобразования и основания систем счисления исходной и той, в которую число преобразовано;
   * извлечение записи по её номеру в списке (Запись);
   * очистка списка (ОчиститьИсторию);
   * конструктор (Запись);
   * текущий размер списка в числе записей (Записей);

Протестировать каждый метод класса

**Практическая работа.**

**Класс Управление для конвертора p1\_p2.**

**Задание:**

* 1. Реализовать Управление для «Конвертера p1\_р2».
  2. Протестировать каждый метод класса.

Спецификация класса Управление для «Конвертера p1\_р2».

**ADTControl\_**

**Данные**

Объект класса **Control\_** (Управление) отвечают за координацию действий между классом «Интерфейс» и классами «Редактор», «Конвертер p1\_10», «Конвертер 10\_p2», «История». Объект класса **Control\_** содержат поля: **ed** типа Editor, **his** типа История, и свойства: Pin типа int (основание системы счисления исходного числа), Pout типа int (основание системы счисления результата), **St** типа State (состояние конвертера). Он может находиться в одном из двух состояний: «Редактирование», «Преобразовано». Объекты этого типа изменяемы.

# Операции

Операции представлены в таблице ниже.

|  |  |
| --- | --- |
| **Control\_** | ***Конструктор*** |
| Вход: | Нет. |
| Процесс: | Создаёт объект Управление типа (тип **Control\_**) и  инициирует поля объекта начальными значениями. |
| **DoCommand** | Выполнить команду. |
| Вход: | n - целое значение, номер выполняемой команды. |
| Предусловия: | Нет. |
| Процесс: | В зависимости от значения n и состояния (St) передаёт сообщение объекту Редактор или Преобразователь и изменяет состояние. Возвращает строку результата: либо отредактированное число, либо результат  преобразования. |
| Выход: | Строка. |
| Постусловия: | Нет. |

end Control\_

**Практическая работа.**

**Интерфейс приложения Конвертор p1\_p2.**

**Задание**

1. Реализовать «Интерфейс» приложения «Конвертер р1\_р2», используя библиотечный класс формы и визуальные компоненты.
2. Протестировать методы класса.

Спецификация класса «Интерфейс». Интерфейс приложения представлен на рис. 18.

# ADT TPanel\_p\_p Данные

«Интерфейс» конвертера действительных чисел из системы счисления с основанием p1 в систему счисления с основанием p2 предназначен для: выбора оснований систем счисления p1, p2 из диапазона от 2..16; ввода и редактирования действительного числа со знаком в системе счисления с выбранным основанием p1; отображения результата – представления ввёдённого числа в системе счисления с основанием p2; отображения справки о приложении; отображения истории текущего сеанса работы пользователя с приложением.

«Интерфейс» несёт на себе визуальные компоненты, реализующие выполнения команд преобразователя и объект «Управление» класса Control\_.

Операции. Операции представлены в таблице ниже.

|  |  |
| --- | --- |
| **Наименование** | **Пояснение** |

|  |  |
| --- | --- |
| **trackBar1\_Scroll** | Обработчик события Scroll для компонента  trackBar1. |
| Вход: | object sender, EventArgs e.  sender – указатель на объект, который явился инициатором события Scroll. e - это объект базового класса для классов, содержащих данные о событии. |
| Предусловия: | Пользователь перетаскивает бегунок компонента  trackBar1. |
| Процесс: | Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления р1 исходного числа. Устанавливает новое значение основания системы счисления. Обновляет состояние  командных кнопок. |
| Постусловия: | Обновления выполнены. |
| Выход: | Нет. |
| **numericUpDown1\_ValueChanged** | Обработчик события ValueChanged для  компонента numericUpDown1. |
| Вход: | object sender, EventArgs e. sender – указатель на  объект, который явился инициатором события ValueChanged. |
| Предусловия: | Пользователь изменяет р1 с помощью  компонента numericUpDown1. |
| Процесс: | Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления р1 исходного числа. Устанавливает новое значение основания  системы счисления. Обновляет состояние |

|  |  |
| --- | --- |
|  | командных кнопок. |
| Постусловия: | Обновления выполнены. |
| Выход: | Нет. |
| **trackBar2\_Scroll** | Обработчик события Scroll для компонента  trackBar2. |
| Вход: | object sender, EventArgs e. sender – указатель на объект, который явился инициатором события  Scroll. |
| Предусловия: | Пользователь перетаскивает бегунок компонента  trackBar2. |
| Процесс: | Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления p2 исходного числа. Устанавливает новое значение основания  системы счисления. |
| Постусловия: | Обновления выполнены. |
| Выход: | Нет. |
| **numericUpDown2\_ValueChanged** | Обработчик события ValueChanged для  компонента numericUpDown2. |
| Вход: | object sender, EventArgs e. sender – указатель на объект, который явился инициатором события  ValueChanged. |
| Предусловия: | Пользователь изменяет р2 с помощью  компонента numericUpDown2. |
| Процесс: | Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления р2 результата. Устанавливает новое значение основания системы счисления.  Обновляет состояние командных кнопок. |

|  |  |
| --- | --- |
| Постусловия: | Обновления выполнены. |
| Выход: | Нет. |
| **numericUpDown1\_ValueChanged** | Обработчик события ValueChanged для  компонента numericUpDown1. |
| Вход: | object sender, EventArgs e. sender – указатель на объект, который явился инициатором события  ValueChanged. |
| Предусловия: | Пользователь изменяет р1 с помощью  компонента numericUpDown1. |
| Процесс: | Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления р1 результата. Устанавливает новое значение основания системы счисления.  Обновляет состояние командных кнопок. |
| Постусловия: | Обновления выполнены. |
| Выход: | Нет. |
| **TPanelp\_p\_Load** | Обработчик события Load для компонента  TPanelp\_p. |
| Вход: | (object sender, EventArgs e). sender – указатель на объект, который явился инициатором события  Load. |
| Предусловия: | Форма загружается в память. |
| Процесс: | Устанавливает начальные значения свойств визуальных компонентов формы после загрузки  формы. |
| Выход: | Нет. |
| Постусловия: | Установка свойств выполнена. |
| **DoCmnd(int j)** | Выполнить команду. |
| Вход: | j значение целого типа – номер команды |

|  |  |
| --- | --- |
|  | преобразователя. |
| Предусловия: | Пользователь нажал командную кнопку команды  с номером j. |
| Процесс: | Передаёт сообщение объекту Управление и отображает возвращаемый им результат. Вызывается метод объекта Управление и передаётся номер набранной пользователем  команды Конвертора. |
| Выход: | Нет. |
| Постусловия: | Обновляется состояние Интерфейса. |
| **button\_Click** | Обработчик события Click для командных  кнопок. |
| Вход: | object sender, EventArgs e. sender: object – указатель на объект, который явился  инициатором события Click. |
| Предусловия: | Пользователь нажал командную кнопку. |
| Процесс: | Извлекает из свойства Tag командной кнопки номер соответствующей ей команды. Вызывает метод DoCmnd Интерфейса и передаёт в него  номер команды. |
| Выход: | Нет. |
| Постусловия: | Нет. |
| **TPanelp\_p\_KeyPress** | Обработчик события KeyPress для алфавитно-  цифровых клавиш клавиатуры. |
| Вход: | object sender, KeyPressEventArgs e. |
| Предусловия: | Пользователь нажал алфавитно-цифровую  клавишу клавиатуры. |
| Процесс: | Определяет по нажатой алфавитно-цифровой  клавише номер соответствующей ей команды. |

|  |  |
| --- | --- |
|  | Вызывает метод DoCmnd Интерфейса и передаёт  в него номер команды. |
| Выход: | Нет. |
| Постусловия: | Команда пользователя вызвана. |
| **TPanelp\_p\_KeyDown** | Обработчик события KeyDown для клавиш  управления клавиатуры. |
| Вход: | (object sender, KeyEventArgs e) |
| Предусловия: | Пользователь нажал клавишу управления  клавиатуры. |
| Процесс: | Определяет по нажатой клавише управления номер соответствующей ей команды. Вызывает метод DoCmnd Интерфейса и передаёт в него  номер команды. |
| Выход: | нет. |
| Постусловия: | Команда пользователя вызвана. |
| **UpdateP1** | Выполнить обновления связанные с изменением  р1. |
| Вход: | Нет. |
| Предусловия: | Изменено основание с.сч. р1 исходного числа. |
| Процесс: | Выполняет необходимые обновления при смене  ос. с. сч. р1. |
| Выход: | Нет. |
| Постусловия: | Состояние кнопок обновлено. |
| **UpdateP2** | Выполнить обновления связанные с изменением  р2. |
| Вход: | Нет. |
| Предусловия: | Изменено основание с.сч. р2 результата. |
| Процесс: | Выполняет необходимые обновления при смене  ос. с. сч. р2. |

|  |  |
| --- | --- |
| Выход: | Нет. |
| Постусловия: | Состояние кнопок обновлено. |
| **UpdateButtons** |  |
| Вход: | Нет. |
| Предусловия: | Изменено основание с.сч. р1 исходного числа. |
| Процесс: | Обновляет состояния командных кнопок предназначенных для ввода цифр выбранной  системы счисления p1. |
| Выход: | Нет. |
| Постусловия: | Состояние кнопок обновлено. |
| **выходToolStripMenuItem\_Click** | Команда Выход основного меню класса  TPanelp\_p формы. |
| Вход: | object sender, EventArgs e. |
| Предусловия: | Пользователь кликает мышью на пункте Выход  основного меню формы. |
| Процесс: | Завершает работу приложения. |
| Выход: | Нет. |
| Постусловия: | Приложение завершено. |
| **справкаToolStripMenuItem\_Click** | Команда Справка основного меню класса  TPanelp\_p формы. |
| Вход: | object sender, EventArgs e |
| Предусловия: | Пользователь кликает мышью на пункте Справка  основного меню формы. |
| Процесс: | Показывает окно со справкой по приложению. |
| Выход: | Нет. |
| Постусловия: | Отображено окно справки. |
| **историяToolStripMenuItem\_Click** | Команда История основного меню класса  TPanelp\_p формы. |
| Вход: | object sender, EventArgs e |

|  |  |
| --- | --- |
| Предусловия: | Пользователь кликает мышью на пункте История  основного меню формы. |
| Процесс: | Открывает окно История. |
| Выход: | Нет. |
| Постусловия: | Окно История - открыто. |

end TPanel\_p\_p

**Текст программы**

ADT\_Convert\_10\_p.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//lab1

namespace Converter

{

public class ADT\_Convert\_10\_p

{

public static char Int\_to\_Char(int d)

{

if (d < 0 || d > 15)

{

throw new IndexOutOfRangeException();

}

string allSymbols = "0123456789ABCDEF";

return allSymbols.ElementAt(d);

}

public static string int\_to\_p(long n, long p)

{

if (p < 2 || p > 16)

{

throw new IndexOutOfRangeException();

}

if (n == 0)

{

return "0";

}

if (p == 10)

{

return n.ToString();

}

bool isNegative = false;

if (n < 0)

{

isNegative = true;

n \*= -1;

}

string pNumber = String.Empty;

while(n > 0)

{

pNumber += Int\_to\_Char((int)(n % p));

n /= p;

}

if (isNegative)

{

pNumber += "-";

}

char[] tmp = pNumber.ToCharArray();

Array.Reverse(tmp);

return new string(tmp);

}

public static string flt\_to\_p(double n, int p, int c)

{

if (p < 2 || p > 16)

{

throw new IndexOutOfRangeException();

}

if (c < 0 || c > 100)

{

throw new IndexOutOfRangeException();

}

string pNumber = String.Empty;

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

{

pNumber += Int\_to\_Char((int)(n \* p));

n = n \* p - (int)(n \* p);

}

return pNumber;

}

public static string Do(double n, int p, int c)

{

if (p < 2 || p > 16)

{

throw new IndexOutOfRangeException();

}

if (c < 0 || c > 100)

{

throw new IndexOutOfRangeException();

}

long leftSide = (long)n;

double rightSide = n - leftSide;

if (rightSide < 0)

{

rightSide \*= -1;

}

string leftSideString = int\_to\_p(leftSide, p);

string rightSideString = flt\_to\_p(rightSide, p, c);

return leftSideString + (rightSideString == String.Empty ? "" : ".") + rightSideString;

}

}

}

ADT\_Convert\_p\_10.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace Converter

{

public class ADT\_Convert\_p\_10

{

private static int char\_to\_num(char ch)

{

string allNums = "0123456789ABCDEF";

if (!allNums.Contains(ch))

{

throw new IndexOutOfRangeException();

}

return allNums.IndexOf(ch);

}

private static double convert(string pNumber, int p, double weight)//?

{

if (weight % p != 0)

{

throw new Exception();

}

long degree = (long)Math.Log(weight, p) - 1;

double result = 0.0f;

for(int i = 0; i < pNumber.Length; i++, degree--)

{

result += char\_to\_num(pNumber.ElementAt(i)) \* Math.Pow(p, degree);

}

return result;

}

public static double dval(string pNumber, int p)

{

if (p < 2 || p > 16)

{

throw new IndexOutOfRangeException();

}

foreach (char ch in pNumber)

{

if (ch == '.')

{

continue;

}

if (char\_to\_num(ch) > p)

{

throw new Exception();

}

}

Regex LeftRight = new Regex("^[0-9A-F]+\\.[0-9A-F]+$");

Regex Right = new Regex("^0\\.[0-9A-F]+$");

Regex Left = new Regex("^[0-9A-F]+$");

double Number;

if (LeftRight.IsMatch(pNumber))

{

Number = convert(pNumber.Remove(pNumber.IndexOf('.'), 1), p, Math.Pow(p, pNumber.IndexOf('.')));

}

else if (Left.IsMatch(pNumber))

{

Number = convert(pNumber, p, Math.Pow(p, pNumber.Length));

}

else if (Right.IsMatch(pNumber))

{

Number = convert(pNumber.Remove(pNumber.IndexOf('.'), 1), p, 0);

}

else

{

throw new Exception();

}

return Number;

}

}

}

**ADTEditor.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Converter

{

public class Editor

{

public enum Commands

{

add0, add1, add2, add3,

add4, add5, add6, add7,

add8, add9, addA, addB,

addC, addD, addE, addF,

addDot, BS, CLEAR, exec

};

string number = zero;

const string delim = ".";

const string zero = "0";

public string Number

{

get => number;

set => number = value;

}

public string addDigit(int n)

{

if (n < 0 || n > 16)

{

throw new IndexOutOfRangeException();

}

if (number == zero)

{

number = ADT\_Convert\_10\_p.Int\_to\_Char(n).ToString();

}

else

{

number += ADT\_Convert\_10\_p.Int\_to\_Char(n);

}

return Number;

} public int acc()

{

return number.Contains(delim) ? number.Length - number.IndexOf(delim) - 1 : 0;

} public string addZero()

{

return addDigit(0);

}

public string addDelim()

{

if (number.Length > 0 && !number.Contains(delim))

{

number += delim;

}

return Number;

} public string bs()

{

if (number.Length > 1)

{

number = number.Remove(number.Length - 1);

}

else

{

number = zero;

}

return Number;

}

public string clear()

{

number = zero;

return Number;

}

public string doEdit(Commands j)

{

switch(j)

{

case Commands.add0:

addZero();

break;

case Commands.add1:

addDigit(1);

break;

case Commands.add2:

addDigit(2);

break;

case Commands.add3:

addDigit(3);

break;

case Commands.add4:

addDigit(4);

break;

case Commands.add5:

addDigit(5);

break;

case Commands.add6:

addDigit(6);

break;

case Commands.add7:

addDigit(7);

break;

case Commands.add8:

addDigit(8);

break;

case Commands.add9:

addDigit(9);

break;

case Commands.addA:

addDigit(10);

break;

case Commands.addB:

addDigit(11);

break;

case Commands.addC:

addDigit(12);

break;

case Commands.addD:

addDigit(13);

break;

case Commands.addE:

addDigit(14);

break;

case Commands.addF:

addDigit(15);

break;

case Commands.addDot:

addDelim();

break;

case Commands.BS:

bs();

break;

case Commands.CLEAR:

clear();

break;

default:

return number;

}

return number;

}

}

}

**History.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Converter

{

public class History

{

public struct Record

{

private int p1;

private int p2;

private string number1;

private string number2;

public Record(int p1, int p2, string n1, string n2)

{

this.p1 = p1;

this.p2 = p2;

number1 = n1;

number2 = n2;

}

public List<string> toList()//?

{

return new List<string> { p1.ToString(), number1, p2.ToString(), number2 };

}

}

List<Record> L;

public History()

{

L = new List<Record>();

}

public void addRecord(int p1, int p2, string n1, string n2)

{

Record record = new Record(p1, p2, n1, n2);

L.Add(record);

}

public Record this[int i]

{

get

{

if (i < 0 || i >= L.Count)

{

throw new IndexOutOfRangeException();

}

return L[i];

}

set

{

if (i < 0 || i >= L.Count)

{

throw new IndexOutOfRangeException();

}

L[i] = value;

}

}

public void clear()

{

L.Clear();

}

public int Count()

{

return L.Count();

}

}

}

**ADT\_Control\_.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Converter

{

public class ADT\_Control\_

{

int pin = 10;

int pout = 16;

public History history = new History();

public enum State

{

Edit,

Converted

}

private State state;

internal State St

{

get => state;

set => state = value;

}

public int Pin

{

get => pin;

set => pin = value;

}

public int Pout

{

get => pout;

set => pout = value;

}

public ADT\_Control\_()

{

St = State.Edit;

Pin = pin;

Pout = pout;

}

public Editor editor = new Editor();

public string DoCmnd(Editor.Commands j)

{

if(j == Editor.Commands.exec)

{

double r = ADT\_Convert\_p\_10.dval(editor.Number, Pin);

string res = ADT\_Convert\_10\_p.Do(r, Pout, Acc());

St = State.Converted;

history.addRecord(Pin, Pout, editor.Number, res);

return res;

}

else

{

St = State.Edit;

return editor.doEdit(j);

}

}

private int Acc()

{

return (int)Math.Round(editor.acc() \* Math.Log(Pin) / Math.Log(Pout) + 0.5);

}

}

}

**Form1.cs (интерфейс)**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

//lab6

namespace Converter

{

public partial class Form1 : Form

{

ADT\_Control\_ control\_ = new ADT\_Control\_();

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

label1.Text = control\_.editor.Number;

trackBar1.Value = control\_.Pin;

trackBar2.Value = control\_.Pout;

label2.Text = "0";

UpdateButtons();

}

private void doCmnd(Editor.Commands j)

{

if(j == Editor.Commands.exec)

{

label2.Text = control\_.DoCmnd(j);

}

else

{

if (control\_.St == ADT\_Control\_.State.Converted)

{

label1.Text = control\_.DoCmnd(Editor.Commands.CLEAR);

}

label1.Text = control\_.DoCmnd(j);

label2.Text = "0";

}

}

private void UpdateButtons()

{

foreach(Control i in Controls)

{

if(i is Button)

{

int j = Convert.ToInt16(i.Tag.ToString());

if (j < trackBar1.Value)

{

i.Enabled = true;

}

if ((j >= trackBar1.Value) && (j <= 15))

{

i.Enabled = false;

}

}

}

}

private void trackbar1\_Scroll(object sender, EventArgs e)

{

numericUpDown1.Value = trackBar1.Value;

UpdateP1();

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

trackBar1.Value = Convert.ToByte(numericUpDown1.Value);

UpdateP1();

}

private void UpdateP1()

{

control\_.Pin = trackBar1.Value;

UpdateButtons();

label1.Text = control\_.DoCmnd(Editor.Commands.CLEAR);

label2.Text = "0";

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{

numericUpDown2.ValueChanged -= numericUpDown2\_ValueChanged;

numericUpDown2.Value = trackBar2.Value;

numericUpDown2.ValueChanged += numericUpDown2\_ValueChanged;

UpdateP2();

}

private void numericUpDown2\_ValueChanged(object sender, EventArgs e)

{

trackBar2.Value = Convert.ToByte(numericUpDown2.Value);

UpdateP2();

}

private void UpdateP2()

{

control\_.Pout = trackBar2.Value;

label2.Text = control\_.DoCmnd(Editor.Commands.exec);

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e)

{

Close();

}

private void историяToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form2 history = new Form2();

history.Show();

if (control\_.history.Count() == 0)

{

MessageBox.Show("История пуста", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

for (int i = 0; i < control\_.history.Count(); i++)

{

List<string> currentRecord = control\_.history[i].toList();

history.dataGridView1.Rows.Add(currentRecord[0], currentRecord[1], currentRecord[2], currentRecord[3]);

}

}

private void справкаToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Здесь должна быть полноценная справка по конвертеру.\nНо мы писали диплом :)" , "Справка", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void Form1\_KeyPress(object sender, KeyPressEventArgs e)

{

int i = -1;

if (e.KeyChar >= 'A' && e.KeyChar <= 'F')

i = e.KeyChar - 'A' + 10;

if (e.KeyChar >= 'a' && e.KeyChar <= 'f')

i = e.KeyChar - 'a' + 10;

if (e.KeyChar >= '0' && e.KeyChar <= '9')

i = e.KeyChar - '0';

if (e.KeyChar == '.')

i = 16;

if (e.KeyChar == 8)

i = 17;

if (e.KeyChar == 13)

i = 19;

if (i == -1)

return;

doCmnd((Editor.Commands)Enum.GetValues(typeof(Editor.Commands)).GetValue(i));

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Delete)

{

doCmnd(Editor.Commands.BS);

}

if (e.KeyCode == Keys.Execute)

{

doCmnd(Editor.Commands.exec);

}

if (e.KeyCode == Keys.Decimal)

{

doCmnd(Editor.Commands.addDot);

}

}

private void button\_Click(object sender, EventArgs e)

{

Button but = (Button)sender;

int j = Convert.ToInt16(but.Tag.ToString());

doCmnd((Editor.Commands)Enum.GetValues(typeof(Editor.Commands)).GetValue(j));

}

}

}

**Тестовые наборы данных**

UnitTest.cs

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace ConverterTests

{

[TestClass]

public class Test\_ADT\_Convert\_10\_p

{

[TestMethod]

public void TestDo1()

{

double n = 123.123;

int p = 12;

int c = 3;

string Expect = "A3.158";

string Actual = Converter.ADT\_Convert\_10\_p.Do(n, p, c);

Assert.AreEqual(Expect, Actual);

}

[TestMethod]

public void TestDo2()

{

double n = -144.523;

int p = 3;

int c = 8;

string Expect = "-12100.11201002";

string Actual = Converter.ADT\_Convert\_10\_p.Do(n, p, c);

Assert.AreEqual(Expect, Actual);

}

[TestMethod]

[ExpectedException(typeof(IndexOutOfRangeException))]

public void TestDo3()

{

double n = -12312.1231;

int p = 3;

int c = -8;

Converter.ADT\_Convert\_10\_p.Do(n, p, c);

}

[TestMethod]

[ExpectedException(typeof(IndexOutOfRangeException))]

public void TestDo4()

{

double n = -12312.1231;

int p = -3;

int c = 8;

Converter.ADT\_Convert\_10\_p.Do(n, p, c);

}

[TestMethod]

public void TestIntToChar1()

{

int n = 12;

char ExpectedChar = 'C';

char ActualChar = Converter.ADT\_Convert\_10\_p.Int\_to\_Char(n);

Assert.AreEqual(ExpectedChar, ActualChar);

}

[TestMethod]

public void TestIntToChar2()

{

int n = 3;

char ExpectedChar = '3';

char ActualChar = Converter.ADT\_Convert\_10\_p.Int\_to\_Char(n);

Assert.AreEqual(ExpectedChar, ActualChar);

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestIntToChar3()

{

int n = -12;

Converter.ADT\_Convert\_10\_p.Int\_to\_Char(n);

}

[TestMethod]

public void TestIntToP1()

{

int n = 123;

int p = 12;

string ExpectedString = "A3";

string ActualString = Converter.ADT\_Convert\_10\_p.int\_to\_p(n, p);

Assert.AreEqual(ExpectedString, ActualString);

}

[TestMethod]

public void TestIntToP2()

{

int n = -234567;

int p = 9;

string ExpectedString = "-386680";

string ActualString = Converter.ADT\_Convert\_10\_p.int\_to\_p(n, p);

Assert.AreEqual(ExpectedString, ActualString);

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestIntToP3()

{

int n = 123;

int p = -24;

Converter.ADT\_Convert\_10\_p.int\_to\_p(n, p);

}

[TestMethod]

public void TestFltToP1()

{

double n = 0.123;

int p = 12;

int c = 3;

string ExpectedString = "158";

string ActualString = Converter.ADT\_Convert\_10\_p.flt\_to\_p(n, p, c);

Assert.AreEqual(ExpectedString, ActualString);

}

[TestMethod]

public void TestFltToP2()

{

double n = 0.417;

int p = 9;

int c = 5;

string ExpectedString = "36688";

string ActualString = Converter.ADT\_Convert\_10\_p.flt\_to\_p(n, p, c);

Assert.AreEqual(ExpectedString, ActualString);

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestFltToP3()

{

double n = 1.5;

int p = 12;

int c = 3;

Converter.ADT\_Convert\_10\_p.flt\_to\_p(n, p, c);

}

}

[TestClass]

public class Test\_ADT\_Convert\_p\_10

{

[TestMethod]

public void TestDval1()

{

string Number = "123.321";

int P = 4;

double ExpectedValue = 27.890625;

double ActualValue = Converter.ADT\_Convert\_p\_10.dval(Number, P);

Assert.AreEqual(ExpectedValue, ActualValue, 0.00001);

}

[TestMethod]

public void TestDval2()

{

string Number = "37.53";

int P = 8;

double ExpectedValue = 31.671875;

double ActualValue = Converter.ADT\_Convert\_p\_10.dval(Number, P);

Assert.AreEqual(ExpectedValue, ActualValue, 0.00001);

}

[TestMethod]

public void TestDval3()

{

string Number = "A8F.9C9";

int P = 16;

double ExpectedValue = 2703.611572265625;

double ActualValue = Converter.ADT\_Convert\_p\_10.dval(Number, P);

Assert.AreEqual(ExpectedValue, ActualValue, 0.00001);

}

[TestMethod]

public void TestDval4()

{

string Number = "0.23A5";

int P = 13;

double ExpectedValue = 0.17632435839081264662;

double ActualValue = Converter.ADT\_Convert\_p\_10.dval(Number, P);

Assert.AreEqual(ExpectedValue, ActualValue, 0.00001);

}

[TestMethod]

public void TestDval5()

{

string Number = "9876";

int P = 11;

double ExpectedValue = 13030;

double ActualValue = Converter.ADT\_Convert\_p\_10.dval(Number, P);

Assert.AreEqual(ExpectedValue, ActualValue, 0.00001);

}

[TestMethod]

[ExpectedException(typeof(System.Exception))]

public void TestDval6()

{

string Number = ".A";

int P = 11;

Converter.ADT\_Convert\_p\_10.dval(Number, P);

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestDval7()

{

string Number = "AA";

int P = 77;

Converter.ADT\_Convert\_p\_10.dval(Number, P);

}

[TestMethod]

[ExpectedException(typeof(System.Exception))]

public void TestDval8()

{

string Number = "FFF";

int P = 2;

Converter.ADT\_Convert\_p\_10.dval(Number, P);

}

}

[TestClass]

public class Test\_Editor

{

[TestMethod]

public void TestAddDigit1()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(0);

string ExpectedValue = "0";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAddDigit2()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(0);

editor.addDigit(0);

editor.addDigit(0);

editor.addDigit(0);

editor.addDigit(0);

string ExpectedValue = "0";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAddDigit3()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(0);

editor.addDelim();

editor.addDigit(0);

editor.addDigit(0);

editor.addDigit(0);

editor.addDigit(0);

string ExpectedValue = "0.0000";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAddDigit4()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(15);

editor.addDigit(12);

editor.addDigit(1);

editor.addDelim();

editor.addDigit(1);

editor.addDigit(9);

string ExpectedValue = "FC1.19";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestAddDigit5()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(17);

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestAddDigit6()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(-12);

}

[TestMethod]

public void TestAcc1()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(15);

editor.addDigit(12);

editor.addDigit(1);

editor.addDelim();

editor.addDigit(1);

editor.addDigit(9);

int ExpectedValue = 2;

int ActualValue = editor.acc();

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAcc2()

{

Converter.Editor editor = new Converter.Editor();

int ExpectedValue = 0;

int ActualValue = editor.acc();

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAcc3()

{

Converter.Editor editor = new Converter.Editor();

editor.addDelim();

editor.addDigit(1);

editor.addDigit(9);

editor.addDigit(9);

editor.addDigit(9);

editor.addDigit(9);

int ExpectedValue = 5;

int ActualValue = editor.acc();

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAddDelim1()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(15);

editor.addDigit(15);

editor.addDigit(15);

editor.addDelim();

editor.addDelim();

editor.addDelim();

editor.addDigit(15);

editor.addDigit(15);

editor.addDigit(15);

editor.addDelim();

editor.addDelim();

editor.addDelim();

string ExpectedValue = "FFF.FFF";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAddDelim2()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(0);

editor.addDelim();

editor.addDelim();

editor.addDelim();

editor.addDigit(0);

editor.addDelim();

editor.addDelim();

editor.addDelim();

string ExpectedValue = "0.0";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAddDelim3()

{

Converter.Editor editor = new Converter.Editor();

editor.addDelim();

editor.addDelim();

editor.addDelim();

string ExpectedValue = "0.";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestBs1()

{

Converter.Editor editor = new Converter.Editor();

editor.bs();

editor.bs();

editor.bs();

editor.bs();

editor.bs();

string ExpectedValue = "0";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestBs2()

{

Converter.Editor editor = new Converter.Editor();

editor.bs();

editor.bs();

editor.bs();

editor.bs();

editor.bs();

string ExpectedValue = "0";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestBs3()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(3);

editor.addDigit(3);

editor.addDigit(3);

editor.addDelim();

editor.bs();

string ExpectedValue = "333";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestBs4()

{

Converter.Editor editor = new Converter.Editor();

editor.addDigit(3);

editor.addDigit(3);

editor.addDigit(3);

editor.addDelim();

editor.addDigit(3);

editor.addDigit(3);

editor.addDigit(3);

editor.bs();

editor.bs();

editor.bs();

string ExpectedValue = "333.";

string ActualValue = editor.Number;

Assert.AreEqual(ExpectedValue, ActualValue);

}

}

[TestClass]

public class Test\_History

{

[TestMethod]

public void TestAddRecord1()

{

Converter.History history = new Converter.History();

history.addRecord(12, 4, "23.42", "52.42");

Converter.History.Record ExpectedValue = new Converter.History.Record(12, 4, "23.42", "52.42");

Converter.History.Record ActualValue = history[0];

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestAddRecord2()

{

Converter.History history = new Converter.History();

history.addRecord(3, 7, "11.11", "11.11");

Converter.History.Record ExpectedValue = new Converter.History.Record(3, 7, "11.11", "11.11");

Converter.History.Record ActualValue = history[0];

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestOverride1()

{

Converter.History history = new Converter.History();

history.addRecord(12, 4, "23.42", "52.42");

history.addRecord(12, 4, "23.42", "52.42");

history.addRecord(12, 4, "11", "11");

Converter.History.Record ExpectedValue = new Converter.History.Record(12, 4, "11", "11");

Converter.History.Record ActualValue = history[2];

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

public void TestOverride2()

{

Converter.History history = new Converter.History();

history.addRecord(12, 4, "23.42", "52.42");

history.addRecord(12, 4, "23.42", "52.42");

history.addRecord(12, 4, "11", "11");

Converter.History.Record ToOverride = new Converter.History.Record(1, 1, "1", "1");

history[1] = ToOverride;

Converter.History.Record ExpectedValue = new Converter.History.Record(1, 1, "1", "1");

Converter.History.Record ActualValue = history[1];

Assert.AreEqual(ExpectedValue, ActualValue);

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestOverride3()

{

Converter.History history = new Converter.History();

history.addRecord(3, 7, "11.11", "11.11");

Converter.History.Record Value = history[-1];

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestOverride4()

{

Converter.History history = new Converter.History();

history.addRecord(3, 7, "11.11", "11.11");

Converter.History.Record Value = history[1];

}

[TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]

public void TestOverride5()

{

Converter.History history = new Converter.History();

Converter.History.Record Value = new Converter.History.Record(12, 4, "11", "11");

history[0] = Value;

}

}

}