Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра ПМиК

Лабораторная работа № 1
по дисциплине «Современные технологии программирования2»
«Приложение “Конвертер”»

Выполнил: студент 4 курса
гр. ИП-013
Копытина Татьяна Алексеевна
Проверил: ассистент
к. ПМиК Агалаков А.А.

Новосибирск, 2024

Оглавление

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 несколько практических работ, 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

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 по одному из структурных критериев (C0,C1,C2). Критерий задаёт преподаватель.

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) = “-A1.E” .
Процесс:	Выполняет преобразование десятичного действительного числа n, в систему счисления с основанием p и точностью c. Например: ***Do*** (”-17.875”,16,3) = “-A1.E”.
Выход:	Строка результата.

	Например: ***Do***(”-17.875”) = “-A1.E”.
int_to_Char(int d)	Преобразовать целое значение в цифру системы счисления с основанием p.
Вход:	d – значение типа int – целое, соответствующее цифре в системе счисления с основанием p.
Предусловия:	Нет.
Процесс:	Преобразует целое d в соответствующую ему цифру в системе счисления с основанием p, 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 c. 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

1. Реализовать преобразователь действительных (конвертер p_10) чисел из системы счисления с основанием p в десятичную систему счисления в соответствии с приведенной ниже спецификацией, используя класс. Основание системы счисления p принадлежит диапазону значений от 2 до 16.
2. Протестировать каждый метод класса по одному из структурных критериев (C0,C1,C2). Критерий задаёт преподаватель.

Спецификация класса «Конвертер p_10» - преобразователь действительных чисел со знаком из системы счисления с основанием p 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 p. 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

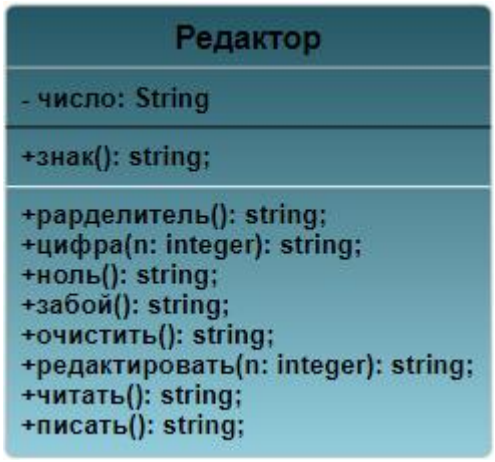
Практическая работа.

Редактор чисел в системе счисления с основанием p

1. Разработать и реализовать класс Editor «Редактор действительных чисел, представленных в системе счисления с основанием p», используя класс

языка высокого уровня. Основание системы счисления p 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 класса представлены ниже.



2. 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 p-ичным цифрам (p 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 p-ичного числа (Number).
3. Протестировать каждый метод класса по одному из структурных критериев (C0,C1,C2). Критерий задаёт преподаватель.

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.

2. 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);

3. Протестировать каждый метод класса по одному из структурных критериев (C0,C1,C2). Критерий задаёт преподаватель.

Практическая работа.

Класс Управление для «Конвертора p1_p2»

1. Реализовать Управление для «Конвертера p1_p2».
2. Протестировать каждый метод класса по одному из структурных критериев (C0,C1,C2). Критерий задаёт преподаватель.

Спецификация класса Управление для «Конвертера p1_p2».

ADT Control_

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»

1. Реализовать «Интерфейс» приложения «Конвертер p1_p2», используя библиотечный класс формы и визуальные компоненты.
2. Протестировать методы класса по одному из структурных критериев (C0,C1,C2). Критерий задаёт преподаватель.

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:

1. выбора оснований систем счисления p1, p2 из диапазона от 2..16;
2. ввода и редактирования действительного числа со знаком в системе счисления с выбранным основанием p1;
3. отображения результата – представления введённого числа в системе счисления с основанием p2;
4. 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 p1 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 p1 с помощью компонента numericUpDown1.
Процесс:	Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления p1 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 p2 с помощью компонента numericUpDown2.
Процесс:	Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления p2 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 p1 с помощью компонента numericUpDown1.
Процесс:	Обновляет свойства визуальных компонентов формы, связанных с изменением основания системы счисления p1 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)

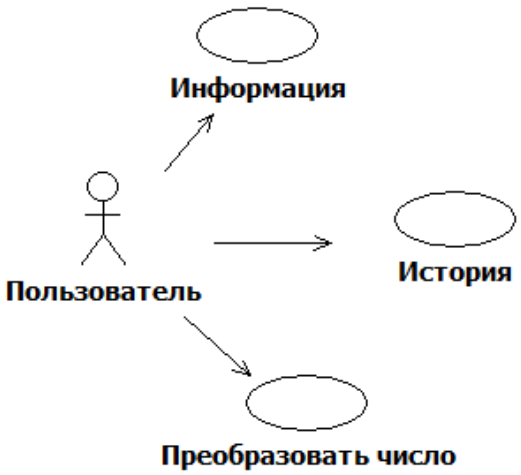
Предусловия:	Пользователь нажал клавишу управления клавиатуры.
Процесс:	Определяет по нажатой клавише управления номер соответствующей ей команды. Вызывает метод DoCmnd Интерфейса и передаёт в него номер команды.
Выход:	нет.
Постусловия:	Команда пользователя вызвана.
UpdateP1	Выполнить обновления связанные с изменением p1.
Вход:	Нет.
Предусловия:	Изменено основание системы счисления p1 исходного числа.
Процесс:	Выполняет необходимые обновления при смене ос. система счисления p1.
Выход:	Нет.
Постусловия:	Состояние кнопок обновлено.
UpdateP2	Выполнить обновления, связанные с изменением p2.
Вход:	Нет.
Предусловия:	Изменено основание системы счисления p2 результата.
Процесс:	Выполняет необходимые обновления при смене ос. система счисления p2.
Выход:	Нет.
Постусловия:	Состояние кнопок обновлено.
UpdateButtons	
Вход:	Нет.
Предусловия:	Изменено основание системы счисления p1 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

Реализация

Функциональные требования представлены диаграммой прецедентов (use-case диаграммой) расположенной ниже.



Сценарий для прецедента «Преобразовать число»

Основной поток событий

1) Пользователь выбирает основание системы счисления p1 исходного числа.
2) Пользователь выбирает основание системы счисления p2 результата.
3) Пользователь вводит действительное число, представленное в системе счисления с основанием p1.
4) Пользователь вводит команду «Преобразовать».
5) Система выводит введённое пользователем число, представленное в системе счисления с основанием p2.
6) Система сохраняет исходные данные и результат преобразования в Историю.

Альтернативный поток событий 1. Введённое пользователем число выходит за границы допустимого диапазона.

3.1. Пользователь получает окно с сообщением.

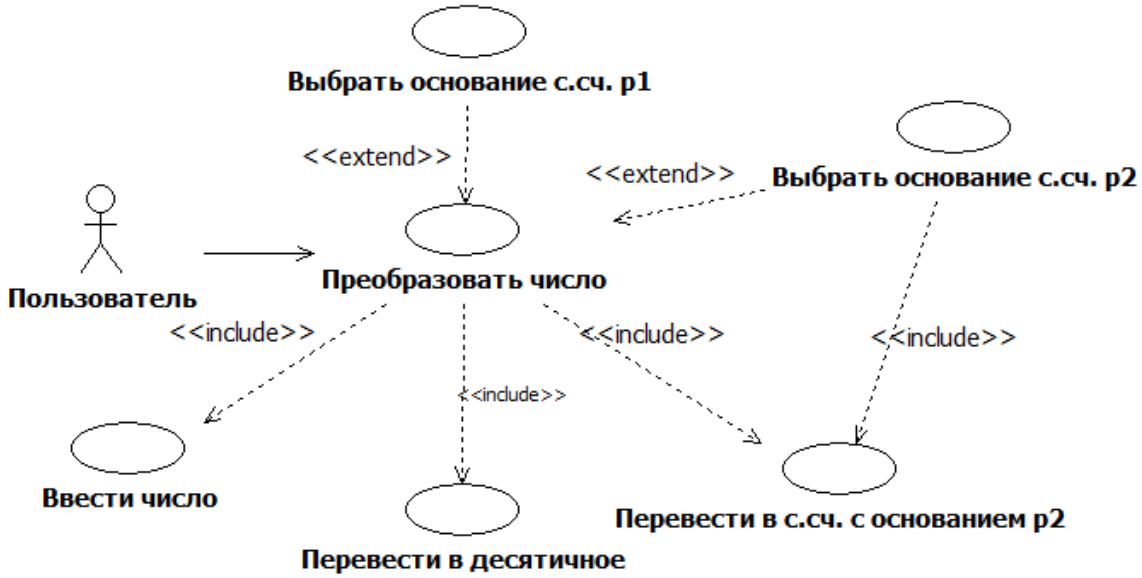
3.2. Приложение переходит в режим Ввод и редактирование.

Альтернативный поток событий 2. Количество разрядов в результате превышает размер поля вывода визуального компонента.

4.1. Пользователь получает окно с сообщением.

4.2. Приложение переходит в режим Ввод и редактирование.

Можно осуществить декомпозицию прецедента. Преобразовать, в результате мы получим для него следующую диаграмму:



Сценарий для прецедента «Преобразовать»

Предусловие

Завершён ввод и редактирования исходного числа.

Основной поток событий

1) Пользователь вводит команду «Преобразовать».
2) Система выводит введённое пользователем число, представленное в системе счисления с основанием p2.
3) Система сохраняет исходные данные и результат преобразования в Историю.

Альтернативный поток событий 1. Введённое пользователем число выходит за границы допустимого диапазона.

1.1. Пользователь получает окно с сообщением.

1.2. Приложение переходит в режим Ввод и редактирование.

Альтернативный поток событий 2. Количество разрядов в результате превышает размер поля вывода визуального компонента.

1.1. Пользователь получает окно с сообщением.

1.2. Приложение переходит в режим Ввод и редактирование.

Сценарий для прецедента «Выбрать основание системы счисления p2»

Предусловие

Прецедент «Преобразовать» завершён.

Основной поток событий

1) Пользователь изменяет основания систем счисления p2.
2) Введённое пользователем число отображается в системе счисления с выбранным основанием.

Альтернативный поток событий 1. Количество разрядов в результате превышает размер поля вывода визуального компонента.

3.1. Пользователь получает окно с сообщением.

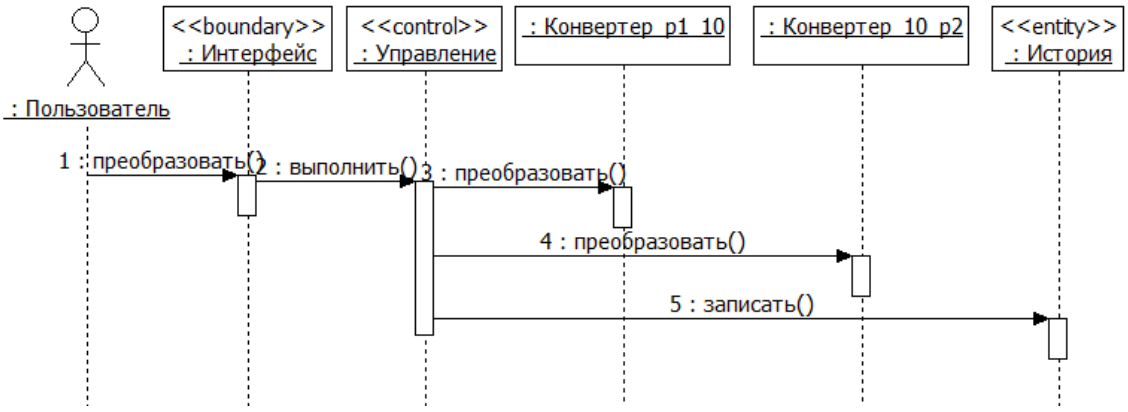
3.2. Приложение переходит в режим Ввод и редактирование.

Диаграмма классов модели объектно-ориентированного анализа.

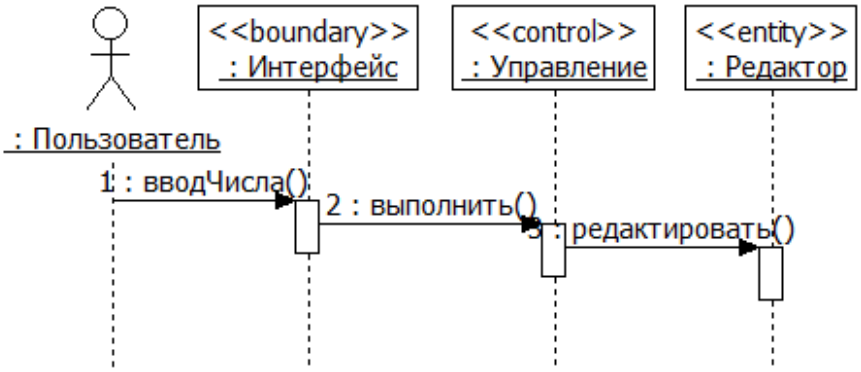
Проанализировав прецеденты, можно выделить следующие классы анализа приложения. Они представлены на диаграмме классов анализа ниже.

Обмен сообщениями между объектами. Диаграмма последовательностей.

Давайте спроектируем обмен сообщениями между объектами в процессе выполнения прецедента «*Преобразовать*». Добавим объект класса Управление для организации обмена сообщениями между объектами в ходе выполнения прецедента. На диаграмме последовательностей приведённой ниже приведёна последовательность сообщений между объектами в основном потоке событий прецедента «*Преобразовать*».



На диаграмме последовательностей приведённой ниже приведёна последовательность сообщений между объектами в процессе реализации прецедента «*Ввести число*».



На диаграмме последовательностей приведённой ниже приведёна последовательность сообщений между объектами в процессе реализации прецедента «*История*».

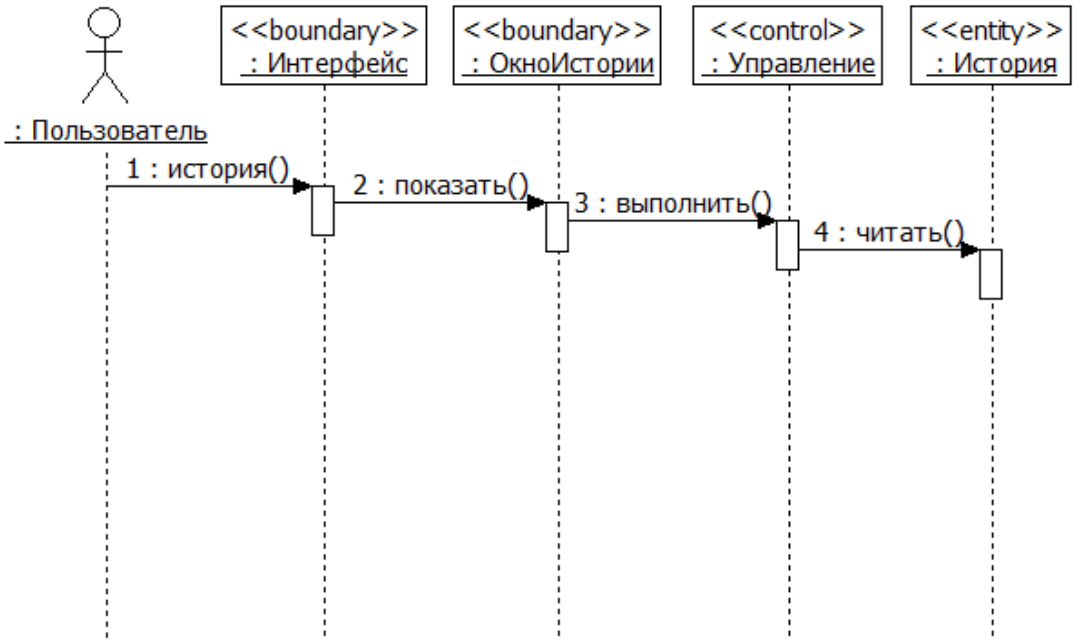
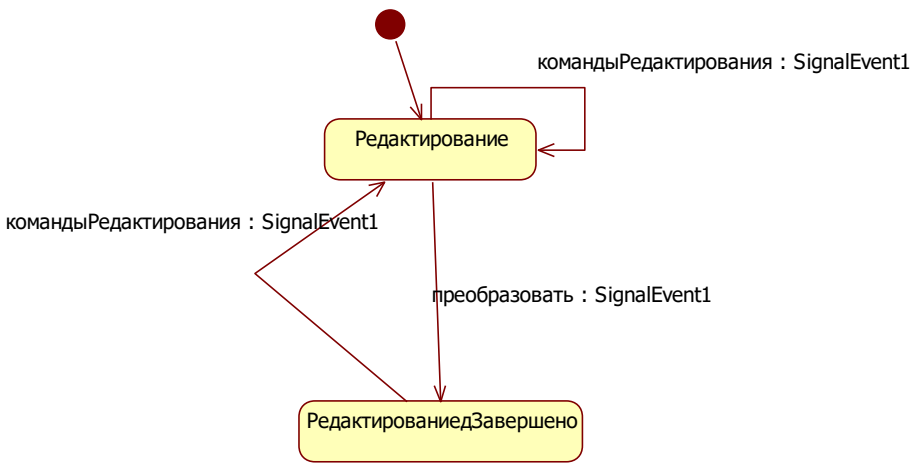


Диаграмма классов проекта

Проанализировав сообщения, которыми обмениваются классы в процессе выполнения прецедентов можно построить следующую диаграмму классов проекта. Для упрощения взаимодействия между классами в процессе работы приложения добавим класс «*Управление*». Тогда наша диаграмма примет следующий вид:

Из диаграммы классов видно, что объект класса «*Интерфейс*» вызывает методы класса «*Управление*» и «*Справка*». Объект же класса «Управление» в свою очередь вызывает методы объектов классов «*Редактор*», «История» и «*Конвертор_p1_10*», «*Конвертор_10_p2*». Диаграмма состояний для объекта класса «Управление» представлена ниже:



Объект класса «*Управление*» может находиться в двух состояниях: «*Редактирование*» и «*Редактирование завершено*».

Демонстрация работы

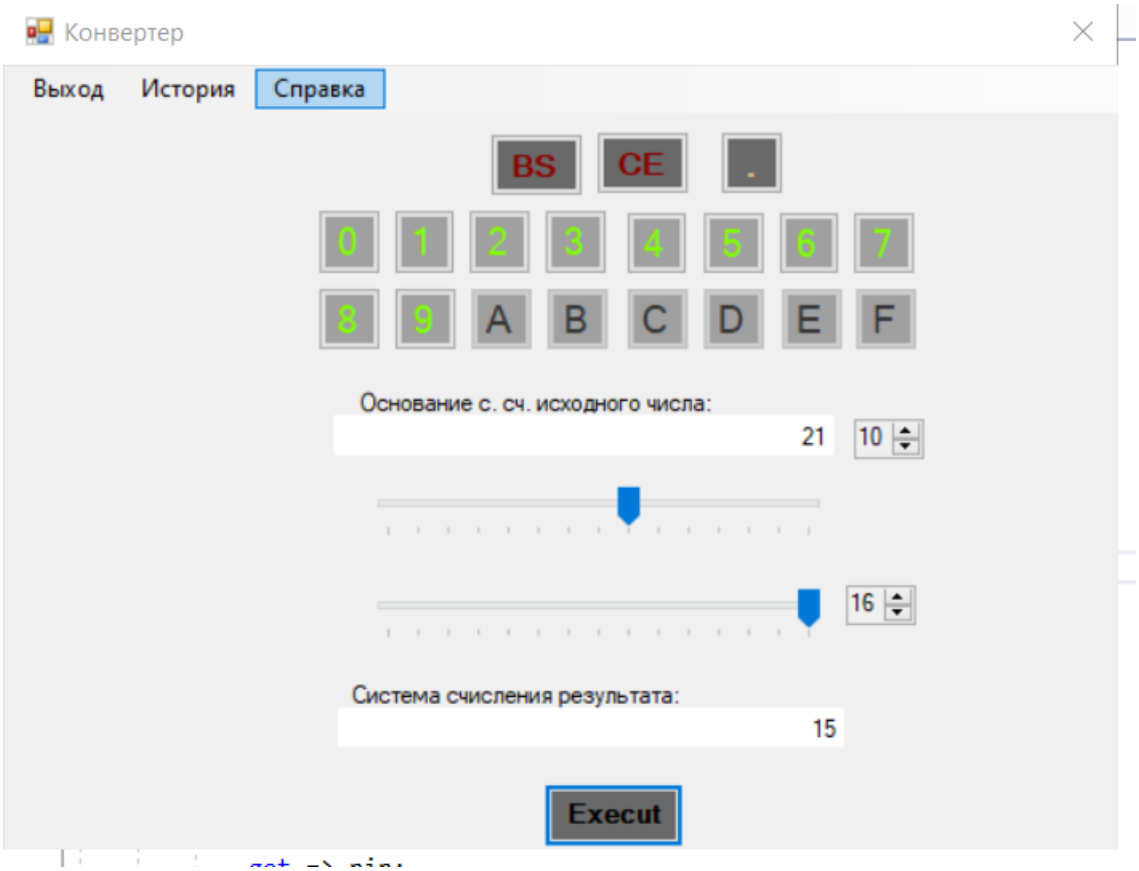


Рисунок 1. Перевод из 10-тичной системы счисления в 16-ричную.

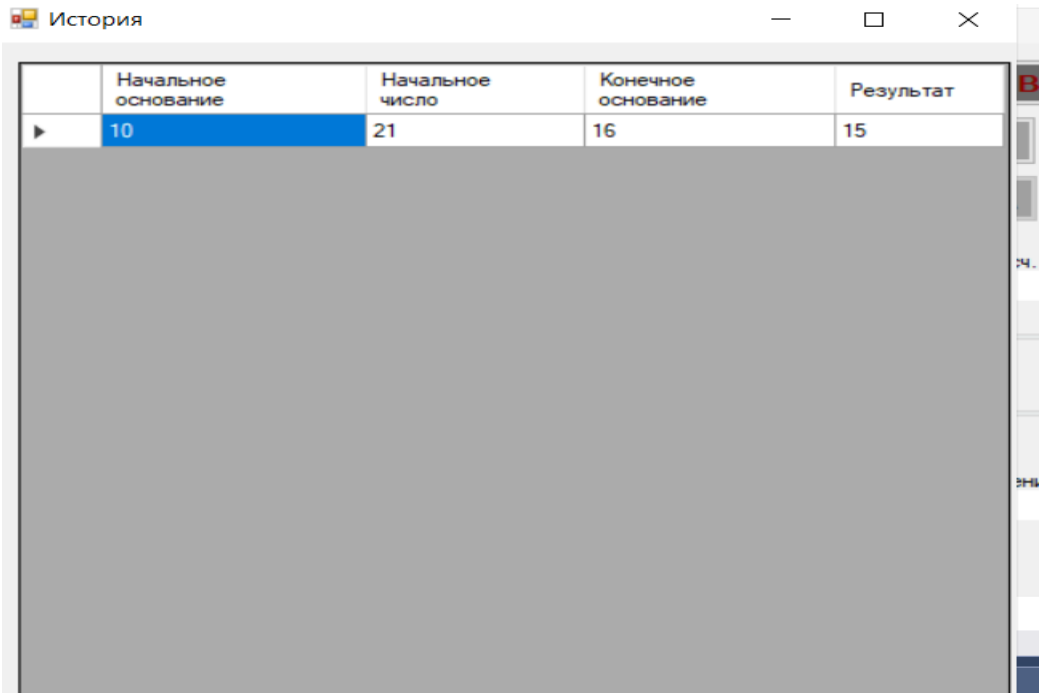


Рисунок 2. История

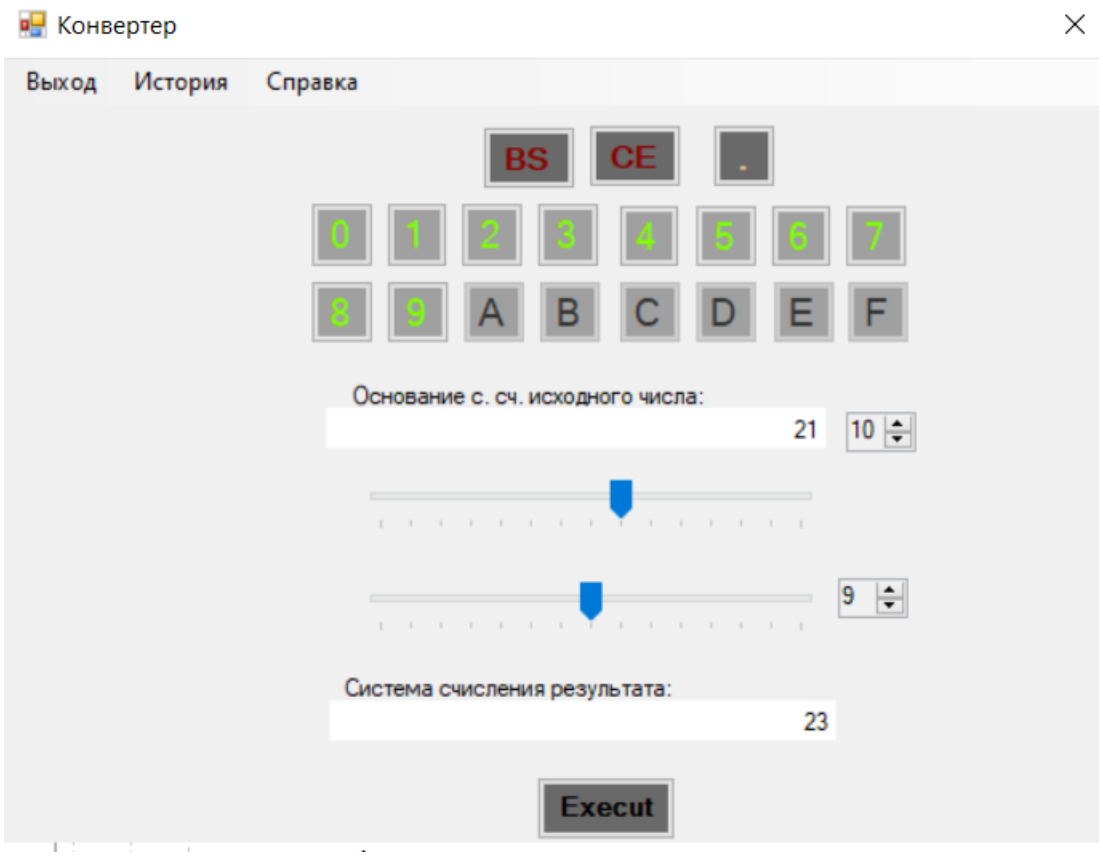


Рисунок 3. Перевод из 10-тичной системы счисления в 9-тиричную.

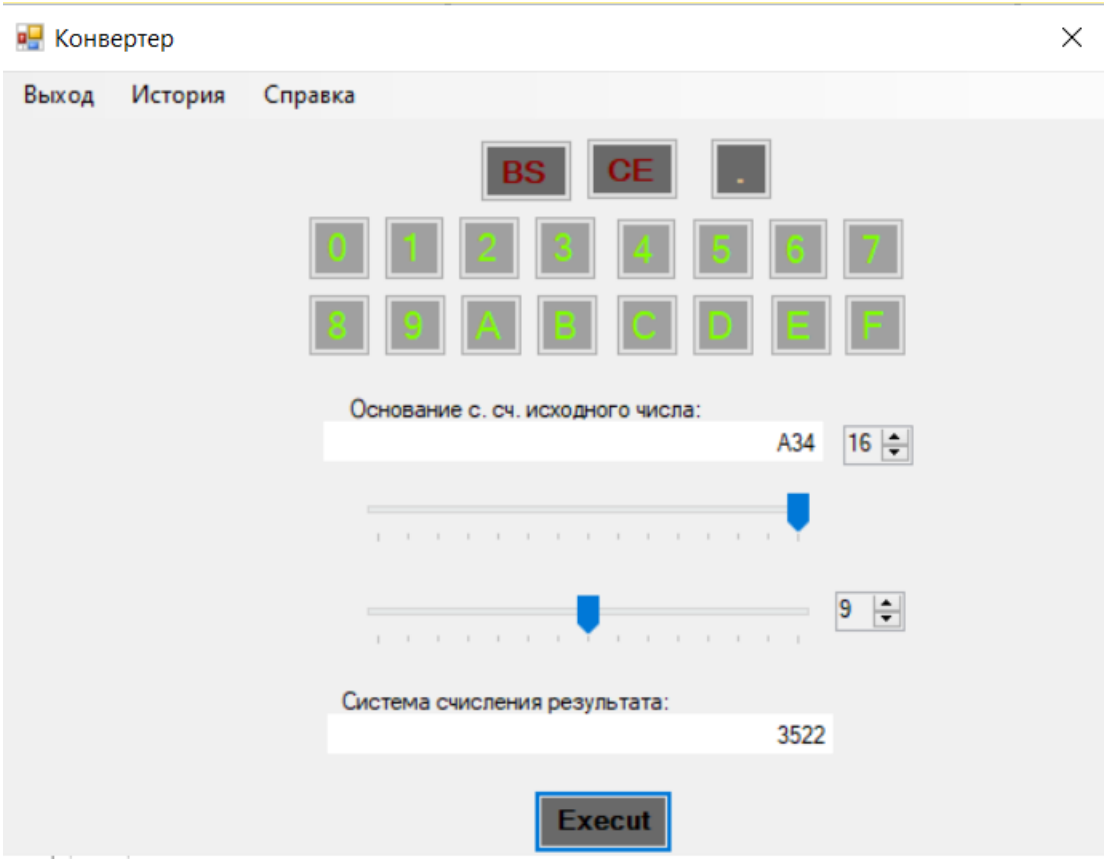


Рисунок 4. Перевод из 16-ричной в 9-ричную.

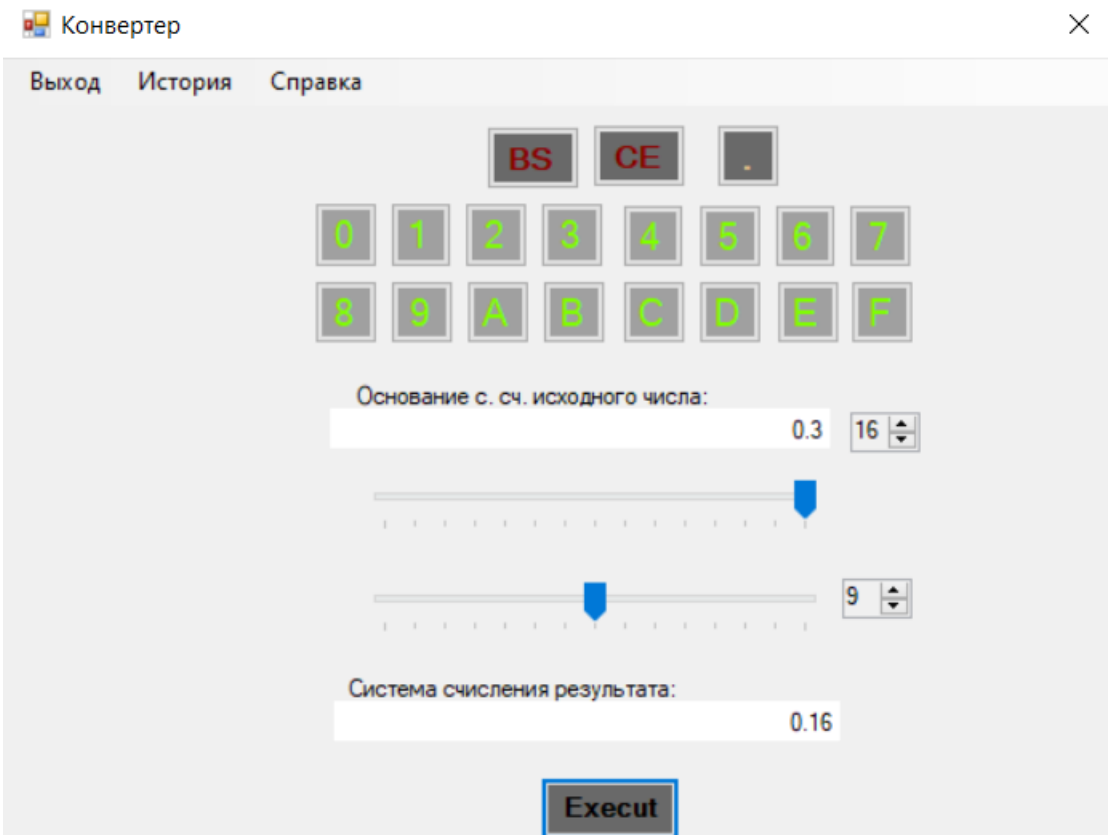


Рисунок 5. Демонстрация работы “точки”

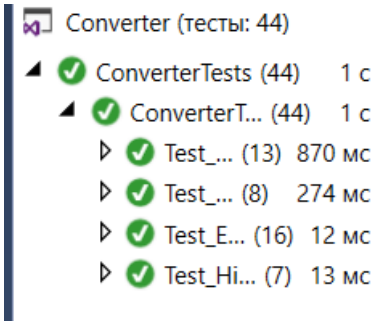


Рисунок 6. Результаты тестов

Вывод

Мы научились работать в среде Visual Studio, а именно разрабатывать в ней модульные тесты для тестирования наших функции и классов на языке C#. А также реализовали приложение конвертер. В процессе выполнения работы мы изучили: отношения между классами: ассоциация, агрегация, зависимость, их реализацию средствами языка программирования высокого уровня; этапы разработки приложений в технологии ООП; элементы технологии визуального программирования; диаграммы языка UML для документирования разработки.

Список литературы

1. Подбельский В.В., Фомин С.С.инт Курс программирования на языке Си: учебник. – М.:ДМК Пресс, 2012 – 384 с.
2. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов. - СПб. : Питер, 2014 - 432 с. : ил. - (Серия "Учебник для вузов").
3. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4 на языке C# . 3-е изд.: - СПб.:Питер, 2012 - 928 с. : ил.
4. Котляров, В. П. Основы тестирования программного обеспечения : учебное пособие для СПО / В. П. Котляров. — Саратов : Профобразование, 2019 — 335 с. — ISBN 978-5-4488-0364-2. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/86202.html (дата обращения: 21.08.2020). — Режим доступа: для авторизир. пользователей

Приложение

Листинг 1. **UnitTest.cs**

```
using
Microsoft.VisualStudio.TestTools.Uni
tTesting;

using Converter;


namespace ConverterTestProj

{

    [TestClass]

    public class
Test_ADT_Convert_10_p

    {

        [TestMethod]

        public void
TestConvertControllerDecimalP()

        {

            double n = 123.123;

            int p = 12;

            int c = 3;

            string Expect =
"A3.158";

            string Actual =
Converter.ConvertControllerDecimalP.
Do(n, p, c);

            Assert.AreEqual(Expect,
Actual);

        }



        [TestMethod]

        public void
TestConvertControllerDecimalP1()

        {

            double n = -144.523;

            int p = 3;

            int c = 8;

            string Expect = "-
12100.11201002";

            string Actual =
Converter.ConvertControllerDecimalP.
Do(n, p, c);

            Assert.AreEqual(Expect,
Actual);

        }



        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]

        public void
TestConvertControllerDecimalP2()

        {

            double n = -12312.1231;

            int p = -3;

            int c = 8;

            string Actual =
Converter.ConvertControllerDecimalP.
Do(n, p, c);

        }



        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]

        public void
TestConvertControllerDecimalP3()

        {

            double n = -12312.1231;

            int p = -3;

            int c = 8;

            string Actual =
Converter.ConvertControllerDecimalP.
Do(n, p, c);

        }
```

```
        [TestMethod]
        public void
TestConvertControllerDecimalP4()
        {
            int n = 12;
            char ExpectedChar = 'C';
            char ActualChar =
Converter.ConvertControllerDecimalP.
IntChar(n);

Assert.AreEqual(ExpectedChar,
ActualChar);
        }

        [TestMethod]
        public void
TestConvertControllerDecimalP5()
        {
            int n = 3;
            char ExpectedChar = '3';
            char ActualChar =
Converter.ConvertControllerDecimalP.
IntChar(n);

Assert.AreEqual(ExpectedChar,
ActualChar);
        }

        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]
        public void
TestConvertControllerDecimalP6()
        {
            int n = -12;

Converter.ConvertControllerDecimalP.
IntChar(n);
        }

        [TestMethod]
        public void
TestConvertControllerDecimalP7()
        {
            int n = 123;
            int p = 12;
            string ExpectedString =
"A3";
            string ActualString =
Converter.ConvertControllerDecimalP.
IntP(n, p);

Assert.AreEqual(ExpectedString,
ActualString);
        }

        [TestMethod]
        public void
TestConvertControllerDecimalP8()
        {
            int n = -234567;
            int p = 9;
            string ExpectedString =
"-386680";
            string ActualString =
Converter.ConvertControllerDecimalP.
IntP(n, p);

Assert.AreEqual(ExpectedString,
ActualString);
        }

        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]
        public void
TestConvertControllerDecimalP9()
        {
            int n = 123;
            int p = -24;
            string Actual =
Converter.ConvertControllerDecimalP.
IntP(n, p);
```

```
        }


        [TestMethod]
        public void
TestConvertControllerDecimalP10()
        {
            double n = 0.123;
            int p = 12;
            int c = 3;
            string ExpectedString =
"158";
            string ActualString =
Converter.ConvertControllerDecimalP.
DblP(n, p, c);

Assert.AreEqual(ExpectedString,
ActualString);
        }


        [TestMethod]
        public void
TestConvertControllerDecimalP11()
        {
            double n = 0.417;
            int p = 9;
            int c = 5;
            string ExpectedString =
"36688";
            string ActualString =
Converter.ConvertControllerDecimalP.
DblP(n, p, c);

Assert.AreEqual(ExpectedString,
ActualString);
        }


        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]
        public void
TestConvertControllerDecimalP12()
        {
            double n = 1.5;
            int p = 12;
            int c = 3;
            string Actual =
Converter.ConvertControllerDecimalP.
DblP(n, p, c);
        }
    }


    [TestClass]
    public class
Test_ADT_Convert_p_10
    {
        [TestMethod]
        public void
TestConvertControllerPDecimal()
        {
            string Number =
"123.321";
            int P = 4;
            double ExpectedValue =
27.890625;
            double ActualValue =
Converter.ConvertControllerPDecimal.
Dval(Number, P);

Assert.AreEqual(ExpectedValue,
ActualValue, 0.00001);
        }


        [TestMethod]
        public void
TestConvertControllerPDecimal1()
        {
            string Number = "37.53";
            int P = 8;
```

```
            double ExpectedValue =
31.671875;
            double ActualValue =
Converter.ConvertControllerPDecimal.
Dval(Number, P);

Assert.AreEqual(ExpectedValue,
ActualValue, 0.00001);
        }

        [TestMethod]
        public void
TestConvertControllerPDecimal2()
        {
            string Number =
"A8F.9C9";
            int P = 16;
            double ExpectedValue =
2703.611572265625;
            double ActualValue =
Converter.ConvertControllerPDecimal.
Dval(Number, P);

Assert.AreEqual(ExpectedValue,
ActualValue, 0.00001);
        }

        [TestMethod]
        public void
TestConvertControllerPDecimal3()
        {
            string Number =
"0.23A5";
            int P = 13;
            double ExpectedValue =
0.17632435839081264662;
            double ActualValue =
Converter.ConvertControllerPDecimal.
Dval(Number, P);

Assert.AreEqual(ExpectedValue,
ActualValue, 0.00001);
        }

        [TestMethod]
        public void
TestConvertControllerPDecimal4()
        {
            string Number = "9876";
            int P = 11;
            double ExpectedValue =
13030;
            double ActualValue =
Converter.ConvertControllerPDecimal.
Dval(Number, P);

Assert.AreEqual(ExpectedValue,
ActualValue, 0.00001);
        }

        [TestMethod]

[ExpectedException(typeof(System.Exc
eption))]
        public void
TestConvertControllerPDecimal5()
        {
            string Number = ".A";
            int P = 11;

Converter.ConvertControllerPDecimal.
Dval(Number, P);
        }

        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]
        public void
TestConvertControllerPDecimal6()
        {
            string Number = "AA";
            int P = 77;
```

```
Converter.ConvertControllerPDecimal.
Dval(Number, P);
        }

        [TestMethod]

[ExpectedException(typeof(System.Exc
eption))]
        public void
TestConvertControllerPDecimal7()
        {
            string Number = "FFF";
            int P = 2;

Converter.ConvertControllerPDecimal.
Dval(Number, P);
        }
    }

    [TestClass]
    public class Test_Editor
    {
        [TestMethod]
        public void
TestEditorController()
        {

Converter.EditorController editor =
new Converter.EditorController();
            editor.addDigit(0);
            string ExpectedValue =
"0";
            string ActualValue =
editor.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestEditorController1()
        {

Converter.EditorController editor =
new Converter.EditorController();
            editor.addDigit(0);
            editor.addDigit(0);
            editor.addDigit(0);
            editor.addDigit(0);
            editor.addDigit(0);
            string ExpectedValue =
"0";
            string ActualValue =
editor.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestEditorController2()
        {

Converter.EditorController editor =
new Converter.EditorController();
            editor.addDigit(0);
            editor.addDelim();
            editor.addDigit(0);
            editor.addDigit(0);
            editor.addDigit(0);
            editor.addDigit(0);
            string ExpectedValue =
"0.0000";
            string ActualValue =
editor.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }
```

```
        [TestMethod]
        public void TestEditorController3()
        {

Converter.EditorController editor = new Converter.EditorController();
            editor.addDigit(15);
            editor.addDigit(12);
            editor.addDigit(1);
            editor.addDelim();
            editor.addDigit(1);
            editor.addDigit(9);
            string ExpectedValue = "FC1.19";
            string ActualValue = editor.getNumber();

Assert.AreEqual(ExpectedValue, ActualValue);
        }


        [TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]
        public void TestEditorController4()
        {

Converter.EditorController editor = new Converter.EditorController();
            editor.addDigit(17);
        }


        [TestMethod]

[ExpectedException(typeof(System.IndexOutOfRangeException))]
        public void TestEditorController5()
        {

Converter.EditorController editor = new Converter.EditorController();
            editor.addDigit(-12);
        }


        [TestMethod]
        public void TestEditorController6()
        {

Converter.EditorController editor = new Converter.EditorController();
            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 TestEditorController7()
        {

Converter.EditorController editor = new Converter.EditorController();
            int ExpectedValue = 0;
            int ActualValue = editor.acc();

Assert.AreEqual(ExpectedValue, ActualValue);
```

```
        }

        [TestMethod]
        public void TestEditorController8()
        {

Converter.EditorController editor = new Converter.EditorController();
            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 TestEditorController9()
        {

Converter.EditorController editor = new Converter.EditorController();
            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.getNumber();

Assert.AreEqual(ExpectedValue, ActualValue);
        }

        [TestMethod]
        public void TestEditorController10()
        {

Converter.EditorController editor = new Converter.EditorController();
            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.getNumber();

Assert.AreEqual(ExpectedValue, ActualValue);
        }

        [TestMethod]
        public void TestEditorController11()
        {
```

```
Converter.EditorController editor =
new Converter.EditorController();
            editor.addDelim();
            editor.addDelim();
            editor.addDelim();
            string ExpectedValue =
"0.";
            string ActualValue =
editor.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestEditorController12()
        {

Converter.EditorController editor =
new Converter.EditorController();
            editor.bs();
            editor.bs();
            string ExpectedValue =
"0";
            string ActualValue =
editor.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestEditorController13()
        {

Converter.EditorController editor =
new Converter.EditorController();
            editor.bs();
            editor.addDigit(1);
            editor.addDigit(2);
            editor.bs();
            string ExpectedValue =
"1";
            string ActualValue =
editor.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestEditorController14()
        {

Converter.EditorController editor =
new Converter.EditorController();
            editor.addDigit(3);
            editor.addDigit(3);
            editor.addDigit(3);
            editor.addDelim();
            editor.bs();
            string ExpectedValue =
"333";
            string ActualValue =
editor.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestEditorController15()
        {

Converter.EditorController editor =
new Converter.EditorController();
            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.getNumber();

Assert.AreEqual(ExpectedValue,
ActualValue);
        }
    }

    [TestClass]
    public class Test_History
    {
        [TestMethod]
        public void
TestHistoryController()
        {

Converter.HistoryController history
= new Converter.HistoryController();
            history.addRecord(12, 4,
"23.42", "52.42");

Converter.HistoryController.Record
ExpectedValue = new
Converter.HistoryController.Record(1
2, 4, "23.42", "52.42");

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

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestHistoryController1()
        {

Converter.HistoryController history
= new Converter.HistoryController();
            history.addRecord(3, 7,
"11.11", "11.11");

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

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

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestHistoryController2()
        {

Converter.HistoryController history
= new Converter.HistoryController();
            history.addRecord(12, 4,
"23.42", "52.42");
            history.addRecord(12, 4,
"23.42", "52.42");
            history.addRecord(12, 4,
"11", "11");

Converter.HistoryController.Record
ExpectedValue = new
Converter.HistoryController.Record(1
2, 4, "11", "11");

Converter.HistoryController.Record
ActualValue = history[2];
```

```
Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]
        public void
TestHistoryController3()
        {

Converter.HistoryController history
= new Converter.HistoryController();
            history.addRecord(12, 4,
"23.42", "52.42");
            history.addRecord(12, 4,
"23.42", "52.42");
            history.addRecord(12, 4,
"11", "11");

Converter.HistoryController.Record
ToOverride = new
Converter.HistoryController.Record(1
, 1, "1", "1");
            history[1] = ToOverride;

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

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

Assert.AreEqual(ExpectedValue,
ActualValue);
        }

        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]
        public void
TestHistoryController4()
        {

Converter.HistoryController history
= new Converter.HistoryController();
            history.addRecord(3, 7,
"11.11", "11.11");

Converter.HistoryController.Record
Value = history[-1];
        }

        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]
        public void
TestHistoryController5()
        {

Converter.HistoryController history
= new Converter.HistoryController();
            history.addRecord(3, 7,
"11.11", "11.11");

Converter.HistoryController.Record
Value = history[1];
        }

        [TestMethod]

[ExpectedException(typeof(System.Ind
exOutOfRangeException))]
        public void
TestHistoryController6()
        {

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

Converter.HistoryController.Record
Value = new
Converter.HistoryController.Record(1
2, 4, "11", "11");
            history[0] = Value;
        }
    }
```

```
}
```

Листинг 2. **Controller.cs**

```
public class ConvertControllerDecimalP
{

    public static string Do(double n, int p, int c)
    {
        if (p < 2 || p > 16)
            throw new IndexOutOfRangeException();
        if (c < 0 || c > 10)
            throw new IndexOutOfRangeException();

        long leftSide = (long)n;

        double rightSide = n - leftSide;
        if (rightSide < 0)
            rightSide *= -1;

        string leftSideString = IntP(leftSide, p);
        string rightSideString = DblP(rightSide, p, c);

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

    public static char IntChar(int d)
    {
        if (d > 15 || d < 0)
        {
            throw new IndexOutOfRangeException();
        }
        string allSymbols = "0123456789ABCDEF";
        return allSymbols.ElementAt(d);
    }

    public static string IntP(long n, int 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 buf = "";
        while (n > 0)
        {
            buf += IntChar((int)n % p);
            n /= p;
        }
```

```
        if (isNegative)
            buf += "-";

        char[] chs = buf.ToCharArray();
        Array.Reverse(chs);
        return new string(chs);
    }

    public static string DblP(double n, int p, int c)
    {
        if (p < 2 || p > 16)
            throw new IndexOutOfRangeException();
        if (c < 0 || c > 10)
            throw new IndexOutOfRangeException();

        string pNumber = String.Empty;
        for (int i = 0; i < c; i++)
        {
            pNumber += IntChar((int)(n * p));
            n = n * p - (int)(n * p);
        }
        return pNumber;
    }
}
```

Листинг 3. **Controller.cs**

```
        public static double Dval(string p_num, int p)
        {
            if (p < 2 || p > 16)
                throw new IndexOutOfRangeException();

            double buf = 0d;
            if (p_num.Contains("."))
            {
                string[] lr = p_num.Split('.');
                if (lr[0].Length == 0)
                    throw new Exception();
                char[] chs = lr[0].ToCharArray();

                Array.Reverse(chs);
                for (int i = 0; i < chs.Length; i++)
                {
                    if (Char_to_num(chs[i]) > p)
                        throw new Exception();
                    buf += Char_to_num(chs[i]) * Math.Pow(p, i);
                }
                char[] chsr = lr[1].ToCharArray();
                for (int i = 0; i < chsr.Length; i++)
                {
                    if (Char_to_num(chsr[i]) > p)
                        throw new Exception();
                    buf += Char_to_num(chsr[i]) * Math.Pow(p, -(i + 1));
                }
            }
            else
            {
                char[] chs = p_num.ToCharArray();

                Array.Reverse(chs);
                for (int i = 0; i < chs.Length; i++)
                {
                    if (Char_to_num(chs[i]) > p)
                        throw new Exception();
                    buf += Char_to_num(chs[i]) * Math.Pow(p, i);
                }
            }
            return buf;
        }

        public static double Char_to_num(char ch)
        {
            string allNums = "0123456789ABCDEF";
            if (!allNums.Contains(ch))
```

```
                throw new IndexOutOfRangeException();
            return allNums.IndexOf(ch);

        }
```

```
        public static double Convert(string p_num, int p, double weight)
        {
            return 0d;
        }
    }
}
```

Листинг 4. **Editor.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Converter
{
    public class EditorController
    {
        string number = "";
        const string zero = "0";
        const string delim = ".";


        public string getNumber() { return number; }


        public string addDigit(int n)
        {
            if (n < 0 || n > 16)
                throw new IndexOutOfRangeException();
            if (number == zero)
                number = ConvertControllerDecimalP.IntChar(n).ToString();
            else
                number += ConvertControllerDecimalP.IntChar(n);
            return number;
        }
        public int acc()
        {
            if (number.Contains(delim))
            {
                string[] chs = number.Split('.');
                return chs[1].Length;
            }
            return 0;
        }
        public string addZero()
        {
            number += zero;
            return number;
        }
        public string addDelim()
        {
            if (number.Length == 0)
            {
                addZero();
            }
            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 = "";
        return number;
    }
    public string doEdit(int
j)
    {
        if (j < 16)
        {
            addDigit(j);
        }
        switch (j)
        {
            case 16:
                addDelim();
                break;
            case 17:
                bs();
                break;
            case 18:
                clear();
                break;
            case 19:
                break;
        }
        return number;
    }
}
```

Листинг 5. **History.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Converter
{

    public class HistoryController
    {
        public struct Record
        {
            int p1, p2;
            string number1, number2;
            public Record(int p1, int p2, string number1, string number2)
            {
                this.p1 = p1;
                this.p2 = p2;
                this.number1 = number1;
                this.number2 = number2;
            }
            public List<string> toList()
            {
                return new List<string> { p1.ToString(), number1, p2.ToString(), number2 };
            }
        }

        List<Record> L;
        public HistoryController()
        {
            L = new List<Record>();
        }

        public void addRecord(int p1, int p2, string number1, string number2)
        {
            L.Add(new Record(p1, p2, number1, number2));
        }

        public void clear()
        {
            L.Clear();
        }

        public int count()
        {
            return L.Count;
        }

        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;
            }
        }
    }
```

Листинг 6. **Controller.cs**

```
class Controller
{
    int _pIn = 10;
    int _pOut = 16;
    const int _accuracy = 10;
    private State _state;

    public HistoryController history = new HistoryController();
    public enum State { Edit, Converted }
    internal State _State { get => _state; set => _state = value; }
    public int Pin { get => _pIn; set => _pIn = value; }
    public int Pout { get => _pOut; set => _pOut = value; }

    public Controller()
    {
        _State = State.Edit;
        Pin = _pIn;
        Pout = _pOut;
    }

    public EditorController editor = new EditorController();

    public string doCmnd(int j)
    {
        if (j == 19)
        {
            double var = ConvertControllerPDecimal.Dval(editor.getNumber(), (Int16)Pin);
            string result = ConvertControllerDecimalP.Do(var, (Int32)Pout, Accuracy());
            _State = State.Converted;

            history.addRecord(Pin, Pout, editor.getNumber(), result);
            return result;
        }
        else
        {
            _State = State.Edit;
            return editor.doEdit(j);
        }
    }

    private int Accuracy()
    {
        return (int)Math.Round(editor.acc() * Math.Log(Pin) / Math.Log(Pout) + 0.5);
    }
}
```

Листинг 7. **Form1.cs**

```
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 Converter
{
    public partial class Form1 : Form
    {
        Controller control_ = new Controller();
        public Form1()
        {

InitializeComponent();
        }


        private void Form1_Load(object sender, EventArgs e)
        {
            label1.Text = control_.editor.getNumber();
            trackBar1.Value = control_.Pin;
            trackBar2.Value = control_.Pout;
            label2.Text = "0";
            UpdateButtons();
        }


        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(obje
ct sender, EventArgs e)
        {
            trackBar1.Value     =
Convert.ToByte(numericUpDown1.Va
lue);
            UpdateP1();
        }

        private void UpdateP1()
        {
            control_.Pin        =
trackBar1.Value;
            UpdateButtons();
            label1.Text         =
control_.doCmnd(18);
            label2.Text = "0";
        }

        private              void
trackBar2_Scroll(object   sender,
EventArgs e)
        {
            numericUpDown2.Value
= trackBar2.Value;
            this.updateP2();
        }
        private              void
numericUpDown2_ValueChanged(obje
ct sender, EventArgs e)
        {
            trackBar2.Value     =
Convert.ToByte(numericUpDown2.Va
lue);
            this.updateP2();
        }
        private void updateP2()
        {
            control_.Pout       =
trackBar2.Value;
            label2.Text         =
control_.doCmnd(19);
        }

        private              void
exitToolStripMenuItem_Click(obje
ct sender, EventArgs e)
        {
            Close();
        }

        private              void
historyToolStripMenuItem_Click(o
bject 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(c
urrentRecord[0],
currentRecord[1],
```

```
currentRecord[2],
currentRecord[3]);
            }
        }

        private              void
infoToolStripMenuItem_Click(obje
ct sender, EventArgs e)
        {
            MessageBox.Show("\n"
+
                "ИП-013
Копытина, Семилетко.", "Справка",
MessageBoxButtons.OK,
MessageBoxIcon.Information);

        }

        private  void  doCmnd(int
j)
        {
            if (j == 19)
                label2.Text     =
control_.doCmnd(j);
            else
            {
                if
(control_._State               ==
Controller.State.Converted)
                    label1.Text
= control_.doCmnd(18);
                label1.Text     =
control_.doCmnd(j);
                label2.Text     =
"0";
            }
        }

        private              void
Form1_KeyPress(object     sender,
KeyPressEventArgs e)
        {
            int i = -1;
            if (e.KeyChar >= 'A'
&& e.KeyChar <= 'F')
            {
                i               =
(int)e.KeyChar - 'A' + 10;
            }
            if (e.KeyChar >= '0'
&& e.KeyChar <= '9')
            {
                i               =
(int)e.KeyChar - '0';
            }
            if (e.KeyChar == '.')
            {
                i = 16;
            }
            if ((int)e.KeyChar ==
8)
            {
                i = 17;
            }
            if ((int)e.KeyChar ==
13)
            {
                i = 19;
            }
            if        ((i        <
control_.Pin) || (i >= 16))
            {
                doCmnd(i);
            }
        }
```

```
        private              void
Form1_KeyDown(object      sender,
KeyEventArgs e)
        {
            if    (e.KeyCode    ==
Keys.Delete) {
                doCmnd(18);
            }
            if    (e.KeyCode    ==
Keys.Execute) {
                doCmnd(19);
            }
            if    (e.KeyCode    ==
Keys.Decimal) {
                doCmnd(16);
            }
        }


        private              void
button_Click(object       sender,
EventArgs e)
        {
            Button     but     =
(Button)sender;
            int        j       =
Convert.ToInt16(but.Tag.ToString
());
                doCmnd(j);
            }

        private              void
label1_Click(object       sender,
EventArgs e)
        {

        }

        private              void
label3_Click(object       sender,
EventArgs e)
        {

        }

        private              void
label2_Click(object       sender,
EventArgs e)
        {

        }
    }
}
```

Листинг 8. **Form2.cs**

```
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 Converter
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }
    }
}
```