Министерство связи и массовых коммуникаций Российской Федерации (Минкомсвязь РФ)

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(ФГБОУ ВО «СибГУТИ»)

Кафедра ПМ и К

Курсовая работа по дисциплине

«Современные технологии программирования»

«Калькулятор простых дробей»

Вариант №11

Выполнил**:**

студент группы ИП-412

Максимов А. С.

Проверил**:**

старший преподаватель кафедры ПМ и К

Мирошников Д. Ю.

Новосибирск, 2018 г.

Оглавление

[Задание на курсовую работу 3](#_Toc510838155)

[Общие требования 3](#_Toc510838156)

[Варианты использования (прецеденты) 3](#_Toc510838157)

[Прецеденты 4](#_Toc510838158)

[Диаграмма прецедентов и диаграмма классов для прецедентов. 5](#_Toc510838159)

[Сценарии прецедентов. 6](#_Toc510838160)

[Сценарий 1. 6](#_Toc510838161)

[Сценарий 2. 8](#_Toc510838162)

[Сценарий 3. 9](#_Toc510838163)

[Сценарий 4. 10](#_Toc510838164)

[Сценарий 5. 11](#_Toc510838165)

[Сценарий 6. 11](#_Toc510838166)

[Тип данных 15](#_Toc510838167)

[Реализация 22](#_Toc510838168)

[Класс TFrac 22](#_Toc510838169)

[Класс Interface 22](#_Toc510838170)

[Класс TCtrl 25](#_Toc510838171)

[Класс TProc 25](#_Toc510838172)

[Класс TMemory 26](#_Toc510838173)

[Класс TFracEditor 26](#_Toc510838174)

[Инструкция пользователю 28](#_Toc510838175)

[Заключение 30](#_Toc510838176)

# Задание на курсовую работу

## Общие требования

Спроектировать и реализовать калькулятор для выполнения вычислений над простыми дробями, используя классы C#и библиотеку визуальных компонентов для построения интерфейса.

Калькулятор должен обеспечить ввод и редактирование целых чисел в обычной записи и рациональных дробей в записи:

[-]<целое без знака>|[-]<числитель><разделитель><знаменатель>.

<числитель>::= <целое без знака>

<знаменатель>::= <целое без знака>

<разделитель>::= ‘/’ | ‘|’

Предусмотреть настройку калькулятора на отображение результата в двух форматах: «дробь» или «число». В формате «дробь» результат всегда отображается в виде дроби. В формате «число» результат отображается в виде числа, если дробь может быть сокращена, так что знаменатель равен 1.

## Варианты использования (прецеденты)

Необходимо предусмотреть следующие варианты использования калькулятора:

1. Выполнение одиночных операций:   
   «операнд1» «операция» «операнд2» «=» «результат»

Пример. 5/1 + 2/1 = 7/1.

1. Выполнение операций с одним операндом:   
   «операнд» «операция» «=» «результат»

Пример. 5/1 \* = 25/1.

1. Повторное выполнение операции:   
   «=»«результат» «=» «результат»

Пример. 5/1 + 4/1 = 9/1 = 13/1 = 17.

1. Выполнение операции над отображаемым значением в качестве обоих операндов:   
   «результат» «операция» «=» «результат»

Пример. 2/1 + 3/1 = 5/1 = 8/1 + = 16/1.

1. Вычисление функций:   
   «операнд» «Sqr» «результат»

Пример. 5/1 «Sqr» 25/1.

1. Вычисление выражений:   
   «операнд1» «функция1» «операция1» «операнд2» «функция2» «операция2» …«операндN» «операцияN» «=»«результат»
2. Пример.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ввод | 6/1 | Sqr | + | 2/1 | Sqr | / | 10/1 | + | 6/1 | = | |
| Отображаемый результат | 6/1 | 36/1 | 36/1 | 2/1 | 4/1 | 40/1 | 10/1 | 4/1 | 6/1 | | 10/1 |

1. Отображаемое значение может сохраняться в памяти или добавляться к её содержимому.

# Прецеденты

По заданным вариантам использования калькулятора была составлена диаграмма прецедентов (рис. 1), диаграмма классов прецедентов (рис. 2), сценарии и диаграммы последовательностей (рис. 3 - 7) прецедентов.

## Диаграмма прецедентов и диаграмма классов для прецедентов.

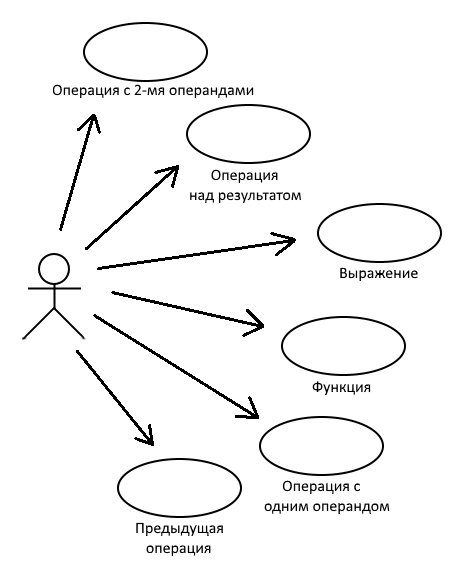


Рисунок 1. Диаграмма прецедентов

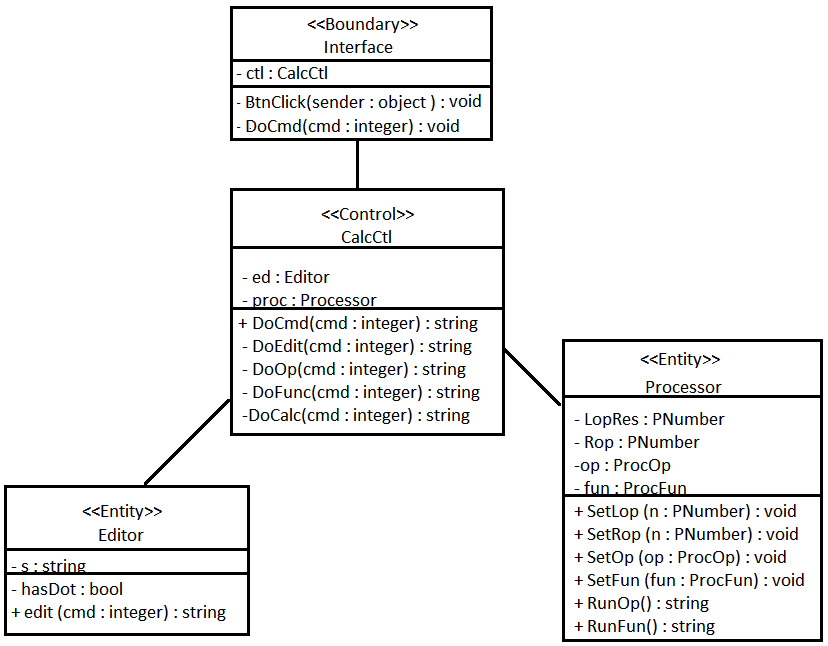


Рисунок 2. Диаграмма классов для прецедентов

## Сценарии прецедентов.

### Сценарий 1.

**Прецедент:** выполнение операции с двумя операндами.

**Основной поток событий:**

1. Пользователь вводит первый операнд.
2. Калькулятор отображает первый операнд.
3. Пользователь вводит операцию.
4. Калькулятор отображает первый операнд.
5. Пользователь вводит второй операнд.
6. Калькулятор отображает второй операнд.
7. Пользователь вводит команду «равно»
8. Калькулятор отображает результат.
9. Вариант использования завершается.

**Альтернативный поток событий 1 (ввод операции или функции без операндов):**

2а1. Калькулятор сбрасывает введенную операцию (функцию) и отображает пустую строку.

2а2. Вариант использования завершается.

**Альтернативный поток событий 2 (ввод функции):**

4б1. Начинается вариант использования «Выполнение функции», куда передаются первый операнд и функция.

4б2. Результат используется в качестве первого операнда.

4б3. Вариант использования продолжается с пункта 2.

**Альтернативный поток событий 3 (повторный ввод операции):**

5а1. Калькулятор меняет операцию на новую.

5а2. Вариант использования продолжается с пункта 4.

**Альтернативный поток событий 4 (ввод «равно» без второго операнда):**

5б1. Вариант использования завершается.

5б2. Начинается вариант использования «Операция с одним операндом».

**Альтернативный поток событий 5 (ввод функции):**

5в1. Калькулятор игнорирует функцию.

5в2. Вариант использования продолжается с пункта 4.

**Альтернативный поток событий 6 (ввод новой операции после второго операнда):**

7а1. Вариант использования завершается.

7а2. Начинается вариант использования «Операция над результатом» с пункта 2, куда передаются результат текущей операции и новая операция.

**Альтернативный поток событий 7 (ввод функции после второго операнда):**

7б1. Начинается вариант использования «Выполнение функции», куда передаются второй операнд и функция.

7б2. Результат используется в качестве второго варианта.

7б3. Вариант использования продолжается с пункта 6.

### Сценарий 2.

**Прецедент:** выполнение операции с одним операндом.

**Основной поток событий:**

1. Пользователь вводит первый операнд.

2. Калькулятор отображает первый операнд.

3. Пользователь вводит операцию.

4. Калькулятор отображает первый операнд.

5. Пользователь вводит команду «равно»

6. Калькулятор отображает результат.

7. Вариант использования завершается.

**Альтернативный поток событий 1 (ввод операции или функции без операндов):**

2а1. Калькулятор сбрасывает введенную операцию (функцию) и отображает пустую строку.

2а2. Вариант использования завершается.

**Альтернативный поток событий 2 (ввод функции):**

4б1. Начинается вариант использования «Выполнение функции», куда передаются первый операнд и функция.

4б2. Результат используется в качестве первого операнда.

4б3. Вариант использования продолжается с пункта 2.

**Альтернативный поток событий 3 (повторный ввод операции):**

5а1. Калькулятор меняет операцию на новую.

5а2. Вариант использования продолжается с пункта 4.

**Альтернативный поток событий 4 (ввод операнда):**

5б1. Вариант использования завершается.

5б2. Начинается вариант использования «Операция с двумя операндами» с пункта 4, куда передаются первый операнд и операция.

**Альтернативный поток событий 5 (ввод функции):**

5в1. Калькулятор игнорирует функцию.

5в2. Вариант использования продолжается с пункта 4.

### Сценарий 3.

**Прецедент:** выполнение предыдущей операции.

**Предусловие:** завершен один из сценариев 1, 2, 3, 4.

**Основной поток событий:**

1. Пользователь вводит команду «равно»

2. Калькулятор отображает результат.

3. Вариант использования завершается.

### Сценарий 4.

**Прецедент:** выполнение операции над результатом (в качестве операндов новой операции выступает результат предыдущей операции).

**Предусловие:** завершен один из сценариев 1, 2, 3, 4.

**Основной поток событий:**

1. Пользователь вводит операцию.

2. Калькулятор отображает предыдущий результат.

3. Пользователь вводит команду «равно»

4. Калькулятор отображает результат.

5. Вариант использования завершается.

**Альтернативный поток событий 1 (ввод команды «равно»):**

2а1. Вариант использования завершается.

2а2. Начинается вариант использования «Выполнение предыдущей операции».

**Альтернативный поток событий 1 (ввод операнда):**

2б1. Вариант использования завершается.

2б2. Начинается вариант использования «Операция с двумя операндами».

**Альтернативный поток событий 2 (повторный ввод операции):**

3а1. Калькулятор меняет операцию на новую.

3а2. Вариант использования продолжается с пункта 2.

**Альтернативный поток событий 3 (ввод операнда):**

3б1. Вариант использования завершается.

3б2. Начинается вариант использования «Операция с двумя операндами» с пункта 4, куда передаются результат и введенная операция.

### Сценарий 5.

**Прецедент:** выполнение функции.

**Основной поток событий:**

1. Пользователь вводит операнд.

2. Калькулятор отображает операнд.

3. Пользователь вводит функцию

4. Калькулятор отображает результат.

5. Вариант использования завершается.

**Альтернативный поток событий 1 (ввод операции):**

4а1. Вариант использования завершается.

4а2. Начинается вариант использования «Выполнение операции с двумя операндами» с пункта 4, куда передаются операнд и операция.

### Сценарий 6.

**Прецедент:** выполнение выражений.

Основной поток событий: состоит из некоторой последовательности предыдущих пяти сценариев, в зависимости от команд пользователя.

1. Диаграммы последовательностей

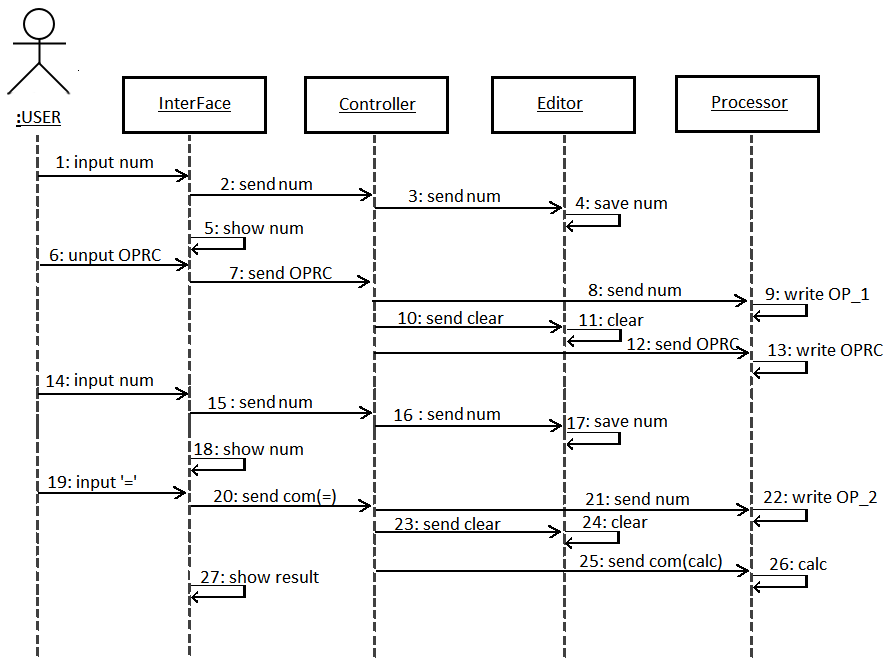


Рисунок 3. Прецедент «Выполнение операции с двумя операндами».

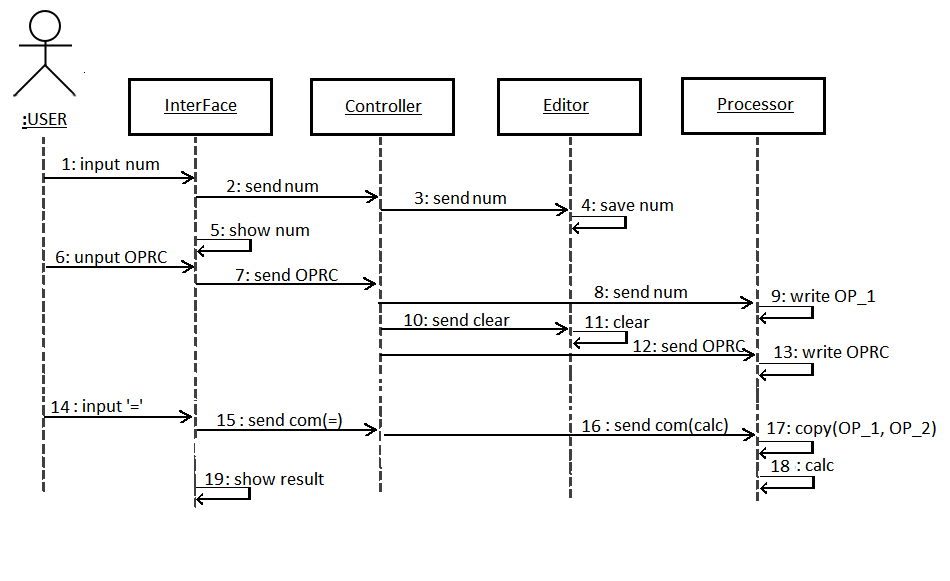


Рисунок 4. Прецедент «Выполнение операции с одним операндом».

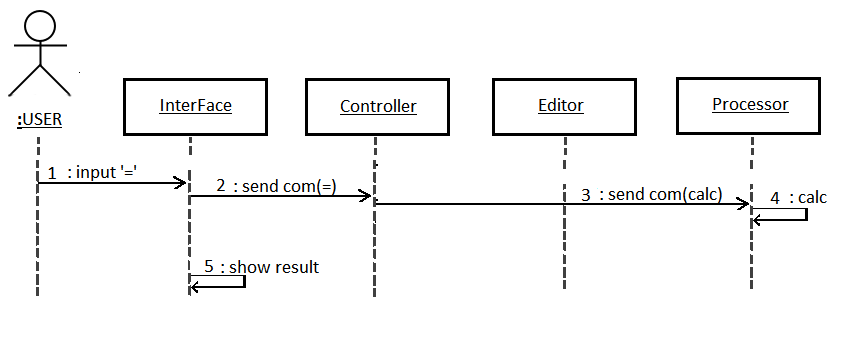


Рисунок 5. Прецедент «Выполнение предыдущей операции».

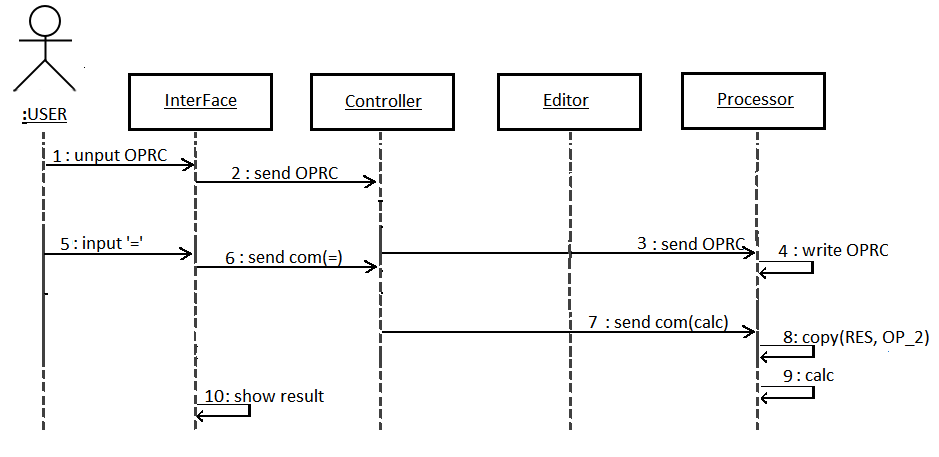


Рисунок 6. Прецедент «Выполнение операции над результатом».

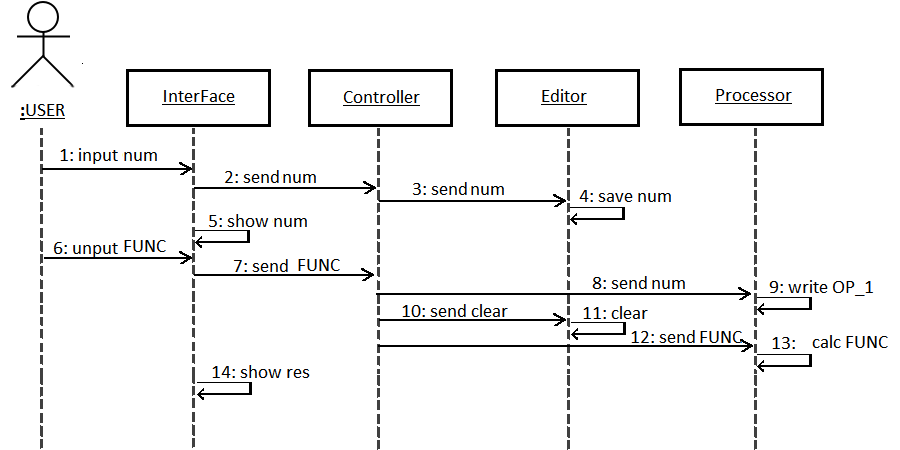


Рисунок 7. Прецедент «Выполнение функции».

# Тип данных

Для возможности выполнения вычислений простых дробей был разработан класс TFrac, предоставляющий соответствующий интерфейс.

**Данные**

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

**Операции**

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

|  |  |
| --- | --- |
| ***Конструктор*** |  |
| Начальные значения: | Пара целых чисел (a) и (b). |
| Процесс: | Инициализирует поля простой дроби (тип TFrac): числитель значением a, знаменатель - (b). В случае необходимости дробь предварительно сокращается.  Например:  *Конструктор*(6,3) = (2/1)  *Конструктор*(0,3) = (0/3). |
|  | |
| ***Конструктор*** |  |
| Начальные значения: | Строковое представление простой дроби . Например: ‘7/9’. |
| Процесс: | Инициализирует поля простой дроби (тип TFrac) строкой f =’a/b’. Числитель значением a, знаменатель - b. В случае необходимости дробь предварительно сокращается.  Например:  *Конструктор*(‘6/3’) = 2/1  *Конструктор* (‘0/3’) = 0/3 |
|  | |
| **Копировать:** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт копию самой дроби (тип TFrac) с числителем, и знаменателем такими же, как у самой дроби. |
| Выход: | Простая дробь (тип TFrac).  Например:  c = 2/1, Копировать(c) = 2/1 |
| Постусловия: | Нет. |
|  | |
| ***Сложить*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную сложением самой дроби q = a1/b1 с d =a2/b2: ((a1/b1)+(a2/b2)=(a1\*b2 + a2\*b1)/( b1\* b2)).  Например:  q = 1/2, d = -3/4  q.Сложить(d) = -1/4. |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Умножить*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт простую дробь (тип TFrac), полученную умножением самой дроби q = a1/b1 на d = a2/b2 ((a1/b1)\*(a2/b2)=(a1\* a2)/( b1\* b2)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Вычесть*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную вычитанием d = a2/b2 из самой дроби q = a1/b1: ((a1/b1)-(a2/b2)=(a1\* b2-a2\*b1)/(b1\*b2)).  Например:  q = (1/2), d = (1/2)  q.Вычесть(d) = (0/1). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет |
|  | |
| ***Делить*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Числитель числа d не равно 0. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученное делением самой дроби q = a1/b1 на дробь d = a2/b2: ((a1/b1)/(a2/b2)=(a1\* b2)/( a2\*b1)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Квадрат*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную умножением самой дроби на себя: ((a/b)\*(a/b)=(a\* a)/( b\* b)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Обратное*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученное делением единицы на саму дробь: 1/((a/b) = b/a. |
| Выход: | Простая дробь (тип TFrac) |
| Постусловия: | Нет. |
|  | |
| ***Минус*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт простую дробь, являющуюся разностью простых дробей z и q, где z - простая дробь (0/1), дробь, вызвавшая метод. |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
|  | |
| ***Равно*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет |
| Процесс: | Сравнивает саму простую дробь q и d. Возвращает значение True, если q и d - тождественные простые дроби, и значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
|  | |
| ***Больше*** |  |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Сравнивает саму простую дробь q и d. Возвращает значение True, если q > d, - значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЧислительЧисло*** |  |
| Вход: |  |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение числителя дроби в числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЗнаменательЧисло*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение знаменателя дроби в числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЧислительСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение числителя дроби в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьЗнаменательСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение знаменателя дроби в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  | |
| ***ВзятьДробьСтрока*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение простой дроби, в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
|  |  |

***end TFrac***

# Реализация

## Класс TFrac

class TFrac

{

private:

int a, b;

void Normalize();

public:

TFrac();

TFrac(int, int);

TFrac(string);

int getA();

int getB();

string AtoString();

string BtoString();

string ToString();

TFrac copy();

TFrac sqr();

TFrac reverse();

~TFrac();

TFrac operator\*(const TFrac &);

TFrac operator+(const TFrac &);

TFrac operator-(const TFrac &);

TFrac operator/(const TFrac &);

TFrac operator-();

bool operator>(const TFrac &) const;

bool operator<(const TFrac &) const;

bool operator>=(const TFrac &) const;

bool operator<=(const TFrac &) const;

bool operator==(const TFrac &) const;

static int TFrac\_comparator(TFrac, TFrac);

void reset()

{

this->a = 0;

this->b = 1;

Normalize();

}

};

## Класс Interface

public ref class MyForm : public System::Windows::Forms::Form

{

MyForm(void)

{

InitializeComponent();

btns[0] = b0;

btns[1] = b1;

btns[2] = b2;

btns[3] = b3;

btns[4] = b4;

btns[5] = b5;

btns[6] = b6;

btns[7] = b7;

btns[8] = b8;

btns[9] = b9;

for(int i = 0 ; i < 10; i++)

btns[i]->Click += gcnew System::EventHandler(this, &MyForm::digit\_click);

button22->Click += gcnew System::EventHandler(this, &MyForm::digit\_click);

cNum1->Text = gcnew String(tctrl.editor.ToString().c\_str());

Memory\_Change(0);

consoleLight = button3->BackColor;

SwitchMode();

SwitchMode();

//

//TODO: добавьте код конструктора

//

}

private: System::Void digit\_click(System::Object^ sender, System::EventArgs^ e) {

unsigned int num = 0;

try

{

num = int::Parse(((System::Windows::Forms::Button^)sender)->Text);

Do\_cmd(num);

}

catch (...)

{

}

}

private: System::Void Memory\_Change(int i)

{

if (i == 1)

{

button25->Enabled = true;

button26->Enabled = true;

button27->Enabled = true;

}

else

{

button25->Enabled = false;

button26->Enabled = false;

button27->Enabled = false;

}

}

private: System::Void Do\_cmd(int i)

{

try

{

string t = tctrl.Do\_Cmd(i);

if (i == (int)Command::C\_MS)

Memory\_Change(1);

if (i == (int)Command::C\_MC)

Memory\_Change(0);

ReloadText();

}

catch (...)

{

tctrl.Reset();

cNum1->Text = "Incorrect input number!";

}

}

private: System::Void ReloadText()

{

string temp = tctrl.editor.ToString();

if (tctrl.editor.IsBEmpty() && !settingsMode)

temp.erase(temp.end() - 2, temp.end());

cNum1->Text = gcnew String(temp.c\_str());

}

private: System::Void MyForm\_KeyDown(System::Object^ sender, System::Windows::Forms::KeyEventArgs^ e) {

int i = -1;

switch (e->KeyCode)

{

case Keys::Back:

i = (int)Command::C\_BS;

break;

case Keys::Delete:

i = (int)Command::C\_CE;

break;

case Keys::Oemplus:

i = (int)Command::C\_EQ;

break;

case Keys::Divide:

i = (int)Command::C\_DIV;

break;

case Keys::Multiply:

i = (int)Command::C\_MUL;

break;

case Keys::Subtract:

i = (int)Command::C\_SUB;

break;

case Keys::Add:

i = (int)Command::C\_ADD;

break;

case Keys::Tab:

SwitchMode();

}

if (i > 0)

Do\_cmd(i);

}

private: System::Void SwitchMode()

{

if (tctrl.editor.getMode() == MODE\_A)

{

tctrl.editor.changeMode(MODE\_B);

button3->BackColor = button2->BackColor;

button6->BackColor = consoleLight;

}

else

{

button6->BackColor = button2->BackColor;

button3->BackColor = consoleLight;

tctrl.editor.changeMode(MODE\_A);

}

}

private: System::Void помощьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show(this, "Калькулятор простых дробей\nВыполнил студент группы ИП-412\nМаксимов А.С.", "Справка");

}

bool settingsMode = true;

private: System::Void комплексноеЧислоToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

комплексноеЧислоToolStripMenuItem->Checked = true;

действительноеЧислоToolStripMenuItem->Checked = false;

settingsMode = true;

ReloadText();

}

private: System::Void действительноеЧислоToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

комплексноеЧислоToolStripMenuItem->Checked = false;

действительноеЧислоToolStripMenuItem->Checked = true;

settingsMode = false;

ReloadText();

}

private: System::Void показатьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show(this, gcnew String(tctrl.history.\_str.c\_str()), "История");

}

private: System::Void очиститьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

tctrl.history.clear();

MessageBox::Show(this,"Очищено", "История");

}

## Класс TCtrl

class TCtrl

{

enum Ctrl\_State { Start, Editing, FunDone, OpChange };

Ctrl\_State state;

TProc <TFrac> processor;

TMemory <TFrac> memory;

TFrac number;

int notation, not\_copy, not\_paste;

public:

History history;

TFracEditor editor;

void Reset();

TCtrl();

~TCtrl();

Ctrl\_State TCtrl::Get\_State();

void Set\_State(Ctrl\_State st);

string TCtrl::Do\_Cmd(int i);

string Do\_Edit(int i);

string Do\_Operation(int i);

string Do\_Func(int i);

string Do\_Calculation(int i);

string Do\_Memory(int i);

string Do\_Clipboard(int i);

};

## Класс TProc

enum TOprtn

{

None,

Add = Command::C\_ADD,

Sub = Command::C\_SUB,

Mul = Command::C\_MUL,

Dvd = Command::C\_DIV

};

enum TFunc

{

Rev = Command::C\_1DIVX,

Sqr = Command::C\_SQR

};

template <class Type>

class TProc

{

Type \*Lop\_Res, \*Rop;

public:

TOprtn Operation;

TFunc Function;

TProc();

~TProc();

void ReSet();

void OprtnClear();

string OprtnRun();

TOprtn OprtnGet();

void OprtnSet(TOprtn);

TFunc FuncGet();

void FuncSet(TFunc);

string FuncRun();

void Lop\_Res\_Set(Type);

void Rop\_Set(Type);

Type Lop\_Res\_Get();

Type Rop\_Get();

};

## Класс TMemory

template <class Type>

class TMemory

{

bool FState;

Type \*FNumber;

public:

enum

{

\_On = true,

\_Off = false

};

void write(Type);

Type get();

void clear();

string getState();

TMemory();

TMemory(string);

TMemory(Type);

~TMemory();

void operator+=(Type);

void operator-=(Type);

};

## Класс TFracEditor

#define MODE\_A true

#define MODE\_B false

enum Command { C\_MC = 16, C\_MR, C\_MS, C\_MPLUS, C\_CE, C\_C, C\_BS, C\_SQR, C\_1DIVX, C\_DIV, C\_MUL, C\_SUB, C\_ADD, C\_EQ, C\_DOT, C\_NEG, C\_CONV, C\_COPY, C\_PAST };

class TFracEditor

{

private:

std::string \_string;

static const std::string \_stringNull;

static const std::string separator;

bool mode;

public:

TFracEditor();

~TFracEditor();

bool IsZero();

std::string ToString();

std::string setSign();

std::string addNum(int);

void changeMode(bool);

std::string erase();

std::string clear();

bool getMode() { return mode; }

string Set\_S(string new\_s)

{

return (\_string = new\_s);

}

string Edit(int i)

{

switch (i)

{

case (int)Command::C\_DOT:

this->addNum(16);

break;

case (int)Command::C\_BS:

this->erase();

break;

case (int)Command::C\_CE:

this->clear();

break;

case (int)Command::C\_NEG:

this->setSign();

break;

default:

this->addNum(i);

break;

}

return \_string;

}

bool IsBEmpty()

{

try

{

int a, b;

sscanf\_s(\_string.c\_str(), "%d/%d", &a, &b);

if (b == 1)

return true;

}

catch (...) {}

return false;

}

};

# Инструкция пользователю

Для ввода доступны выбор числителя или знаменателя.

У всех арифметических операций приоритет одинаковый, а функций приоритет выше, чем у операций. Порядок вычисления слева направо.

Доступна возможность перевода калькулятора в режим действительного числа. Для этого в пункте «Режим» необходимо перевести флаг на пункт «Действительное число».

Ввод чисел можно осуществлять как с помощью интерфейса калькулятора, так и с помощью клавиатуры либо из внутренней памяти калькулятора. Для редактирования ввода есть две команды: «˿» - забивает последний введённый символ и «СЕ» - очищает поле ввода. Для полного сброса состояния калькулятора предусмотрена команда «С».

В калькуляторе предусмотрена память, в которую можно сохранять результаты по команде «MS». По команде «М+» к числу, сохранённому в памяти, прибавляется текущее число. По команде «МС» память очищается. Чтобы считать число из памяти необходимо нажать «MR».

# Заключение

В результате выполнения курсовой работы был разработан и протестирован калькулятор (рис. 8) для простых дробей.

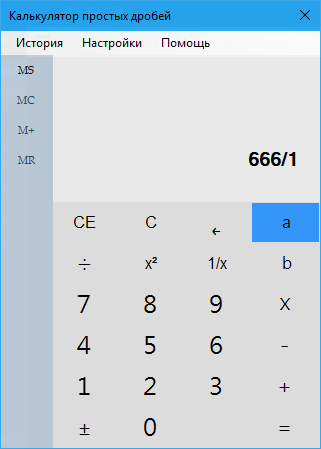


Рисунок 8. Приложение «Калькулятор»

В ходе выполнения были получены следующие практические навыки:

1. проектирования программ в технологии абстрактных типов данных и ООП;
2. построения диаграмм UML;
3. реализации абстрактных типов данных с использованием классов C#;
4. использования библиотеки визуальных компонентов VCL для построения интерфейса;
5. тестирования программ.