|  |  |
| --- | --- |
|  | /storage/emulated/0/.polarisOffice5/polarisTemp/image1.png |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования «Московский технологический университет»  МИРЭА | | |

Институт Информационных Технологий

Кафедра Корпоративных Информационных Систем

**ОТЧЕТ**

по Лабораторной Работе №3

на тему

«Простые типы данных. Ввод и вывод простых типов данных»

по дисциплине

«Процедурное Программирование»

Выполнил студент группы ИСБО-13-13 Акжигитов Р. Р.

Принял Асс. Каф. КИС Алдобаева В. Н.

Выполнено «30» октября 2017 г.

Зачтено «\_\_\_» \_\_\_\_\_\_\_\_\_ 2015 г.

Москва, 2017

**Задание**

Создать приложение - калькулятор, в котором расположены поля ввода 2-х аргументов и набор возможных (предложенных в заданиях) операций над ними. Данные из текстовых полей должны приводятся к нужному типу: числовому или логическому, в зависимости от типа операции. Если в процессе приведения типов возникла ошибка (например, один из аргументов пуст или аргументы различаются по типу). Результат операции вывести в listBox с сохранением истории операций.

**Теоретическая часть**

**Математические функции библиотеки.**

В библиотеке Math существует множество полезных математических функций, таких как e в степени x, степень числа, логарифмы, синусы (др. тригонометрические функции), функция округления Round. Также существуют и константы: число е, число Пи.

**Привидение типов.**

Введенные данные пользователем нужно привести к какому-то типу, в нашем случае либо числовой, либо логический. Пользователь может случайно или нет допустить ошибки при вводе, поэтому перед приведением нужно для начала произвести какие-то подготовительные действия, например, для логического типа – это удаление пробелов по краям, приведение текста к нижнему регистру (для более широкого охвата вариантов введение), также для логического типа мною было предусмотрена возможность введения данных, как на английском, так и на русском языках. Именно поэтому я и не использовал стандартный вариант bool.Parse, ибо он работает только с английским вариантом.

Числовой тип парсится через стандартное решение double.Parse. Поддерживает ввод, как с точкой, так и без. Также для расширения возможностей различных способов ввода: знаки «запятая» и «точка» программа рассматривает одинаково, как отделение дробной от целой части.

**Неточность типа.**

В работе мы использовали «широкодиапазонный» double, который нам позволяет, в отличие от float, хранить больший диапазон (множество) чисел и причем более точных (больше кол-во знаков (чисел) после запятой соответствуют реальности).

**Исключительные ситуации.**

Существует много вариантов, когда пользователь может ввести некорректные данные, причем требуется определить, в каком поле произошла ошибка, также возможно определить какого типа ошибка. Вот пару возможных ситуаций:

1. Пользователь не ввел данные в одно или в оба поля
2. Пользователь ввел некорректные данные, которые не являются ни число, ни логикой, просто строку.
3. Пользователь хотел ввести строку или логику, но ошибся при вводе значения, случайно поставил пробел или лишнюю запятую в числе.
4. Пользователь ввел аргумент поддерживаемый программой (логика или число), однако операция не может его обработать, например, сравнение логики (больше, меньше), сложение, умножение логических данных.
5. Пользователь ввел данные различных типов, причем операция может поддерживать их одновременно (как побитовые и логические операции, «принадлежащие» одним и тем же кнопкам), но программа не может понять, какую конкретно операции хочет в результате получить пользователь, ей нужно, чтобы оба типа были одинаковые. **Решение:** вывести сообщение о разных типах, просьбу уточнить данные.

**Алгоритм решения задачи**

1. Разместить на форме 16 элементов «Кнопка»
2. Задать каждой кнопке в качестве обработчика нажатия вывод случайного числа
3. Нажимать кнопки до тех пор, пока результат не совпадет с ожидаемым 3 раза подряд

1 вариант

1. является ли число степенью 2

2. sin

3. tan

4. e^x

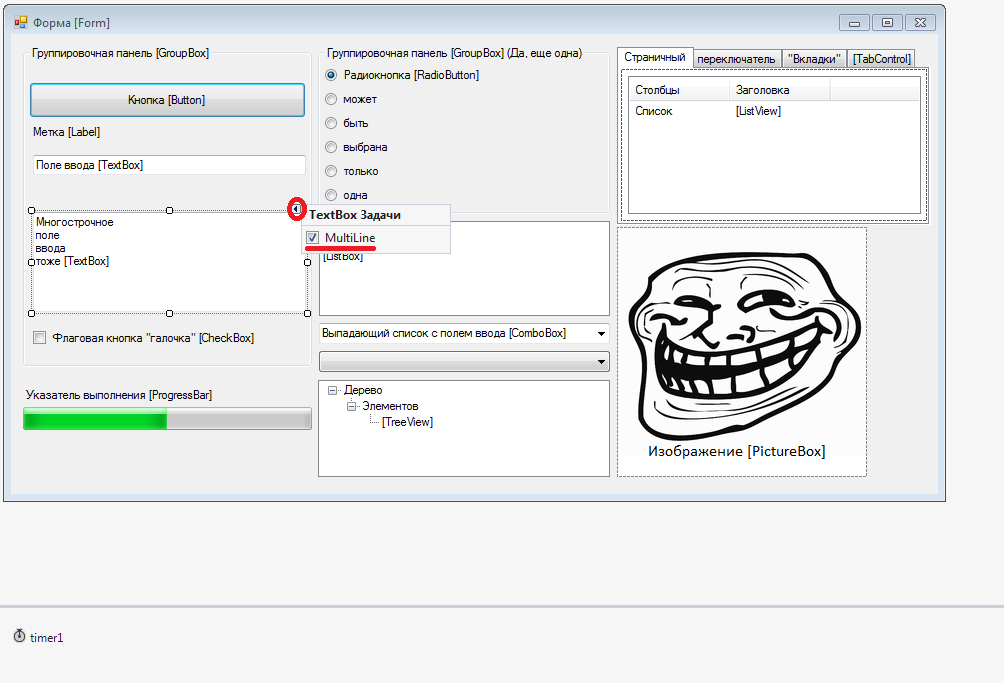
**Тестирование**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Исходные данные | Ожидаемый результат | Фактический результат | Комментарий |
| 1. | 5+3 | 8 | 11 | Не попал |
| 2. | 22/11 | 2 | 3 | Неудача |
| 3. | 7\*11 | 77 | 64 | Повезет в другой раз |
| 4. | 8-3 | 5 | 5 | Бинго! |

**Заключение**

Практическая проверка показала сравнительно малую применимость метода Монте-Карло для решения данных типов задач.

**Исходный код**



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 WindowsFormsApplication13

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

comboBox2.SelectedIndex = 0;

Form2 newForm = new Form2();

newForm.form1 = this;

newForm.Show();

}

}

}