3. Лабораторная работа №5
Задание №1:

Создать метод, реализующий:
а) Ввод переменной через диалоговое окно InputBox или форму с текстовым полем.
б) Перехват и обработку типовых ошибок ввода/конверсии (Type mismatch, Overflow)
в) Вывод сообщения об ошибках с использованием формы MsgBox/MessageBox и обработать реакцию пользователя на ошибку (прекращение расчётов, возврат к вводу данных и т.п.).
Действия при выборе кнопок пользователем:
Ignore – задать значение по умолчанию (игнорировать ввод).
Cancel/No/Abort – Выход из программы
Retry – Повторить ввод.
Примечание 1. Если исключение не вызывается, а получается значение типа double.NaN или схожие с ним, то сгенерировать исключение вручную.
Примечание 2. В Python использовать, например *tkinter.messagebox*
Примечание 3. Типы данных заданы для C#. Если в выбранном языке их нет заменить на ближайший похожий.
Примечание 4. 1-е задание сдаётся только в течение лабораторной работы

A	B	C	D
Тип переменной	**Тип значка**	**Кнопки**	**Кнопка по умолчанию**
1. Byte 2. Short 3. int 4. uint 5. long 6. ulong 7. float 8. double 9. decimal 10. bool	1. Нет 2. Информационный 3. Вопросительный 4. Восклицательный 5. Ошибка	1. Продолжать вычисления или остановиться (Ok/Cancel) 2. Повторить ввод, игнорировать ввод или остановиться (Abort /Retry/Ignore) 3. Повторить ввод или остановиться (Yes/No) 4. Повторить ввод, остановиться или задать значение по умолчанию (Yes/No/Cancel) 5. Повторить ввод или остановиться (Retry/Cancel)	1. Нет 2. Первая 3. Вторая

Вариант	Задание 1				Задание 2	Вариант	Задание 1				Задание 2
	A	B	C	D			A	B	C	D	
1.	8	1	2	3	8	**41.**	5	1	4	2	15
2.	1	4	3	2	38	**42.**	8	2	2	3	35
3.	6	2	5	1	23	**43.**	4	5	3	1	36
4.	9	5	1	2	7	**44.**	1	3	1	2	6
5.	3	3	4	1	6	**45.**	3	4	5	3	10
6.	4	2	5	3	19	**46.**	6	5	3	1	7
7.	2	4	4	1	34	**47.**	7	1	2	2	11
8.	10	5	2	3	1	**48.**	10	4	1	3	16
9.	7	1	1	2	33	**49.**	9	3	5	1	12
10.	5	3	3	1	4	**50.**	2	2	4	3	25
11.	3	2	5	2	11	**51.**	9	1	3	2	24
12.	7	1	3	3	25	**52.**	3	2	5	2	2
13.	10	5	2	1	31	**53.**	4	5	2	3	29
14.	5	4	1	3	29	**54.**	8	3	1	1	19
15.	4	3	4	2	9	**55.**	10	4	4	1	1
16.	8	5	1	2	12	**56.**	7	5	4	2	34
17.	9	1	5	3	32	**57.**	1	2	2	3	38
18.	2	3	3	1	21	**58.**	6	4	3	1	27
19.	6	2	4	2	35	**59.**	5	3	1	3	20

Вариант	Задание 1				Задание 2	Вариант	Задание 1				Задание 2
	A	B	C	D			A	B	C	D	
20.	1	4	2	3	26	**60.**	2	1	5	2	9
21.	2	3	3	1	13	**61.**	3	4	3	1	22
22.	9	2	4	2	36	**62.**	4	5	5	3	26
23.	4	1	1	3	5	**63.**	7	2	1	2	13
24.	3	5	5	1	3	**64.**	10	3	4	2	10
25.	10	4	2	1	24	**65.**	8	1	2	3	31
26.	5	3	4	2	39	**66.**	6	2	4	1	23
27.	6	2	5	3	27	**67.**	1	1	3	3	32
28.	7	4	1	2	2	**68.**	5	5	5	1	16
29.	1	5	2	3	3	**69.**	2	3	2	2	15
30.	8	1	3	1	22	**70.**	9	4	1	3	14
31.	9	3	5	3	18	**71.**	10	2	2	2	37
32.	7	2	2	1	30	**72.**	1	5	3	1	40
33.	5	5	3	2	37	**73.**	4	1	1	1	8
34.	10	4	4	3	4	**74.**	9	3	4	2	17
35.	4	1	1	1	28	**75.**	8	4	5	3	28
36.	1	2	5	2	20	**76.**					
37.	3	1	4	2	40	**77.**					
38.	8	4	1	1	5	**78.**					
39.	2	3	3	3	14	**79.**					
40.	6	5	2	1	17	**80.**					

Задание №2:

а – 2 балла, б – 2 балла, в – 3 балла

а) Требуется рассчитывать функцию от двух переменных $G(x_i, y_j)$ при определённых пользователем вещественных значениях x_i и y_j. $i=0..m;\ j=0..n$

- Все заданные пользователем значения x_i и y_j будем называть ***наборами исходных данных.***

Создать программу расчёта, где пользователем вводятся:

- Количество наборов исходных данных.
- Вычисление функции $G(x, y)$ оформить в виде отдельной функции (метода) ***с перехватом ошибок***.

Произвести расчёты для *4-х или более* наборов исходных данных. Задание исходных данных для расчётов каждого набора производятся единовременно с листа или формы, а не последовательно. Ввод данных через множество **InputBox**-ов или подобных мини-форм, а также вывод результатов через **MessageBox-ы запрещен!!!**

Примечание: Значения функции не обязательно должны быть корректными в заданных интервалах.

Функции **G(*x*,*y*)**

Вариант	G(*x*,*y*)	Вариант	G(*x*,*y*)	Вариант	G(*x*,*y*)
1.	$lg(cos(x^2))/y$	15.	$y\sqrt{x^5-2}$	29.	$y/arctan(x)$
2.	$tg(x/y)$	16.	$x/(y-2)$	30.	$exp(-y/\lvert x\rvert)$
3.	$y/sin(-x^2)$	17.	$y/\lvert log_2(x)\rvert$	31.	$\sin\left(\sqrt[3]{-2y/x^2}\right)$
4.	$\sqrt{y}/ln(x)$	18.	$\exp\left(-\sqrt{y/(1+x)}\right)$	32.	$\sqrt{\frac{x^3-8}{y}}$
5.	$\sqrt[3]{y}lg(x-5)$	19.	$y\cos\left(2/\sqrt{100-x^2}\right)$	33.	$y\exp\left(\sqrt{x}\right)$
6.	$lg((x^5-7)/2y)$	20.	$y\cos\left(\sqrt[3]{2/x}\right)$	34.	$y\times10^{ln(x)+1}$
7.	$exp(y/sin(x))$	21.	$\sqrt[3]{x^5-2}/y$	35.	$2^{1+\sqrt{xy}}$
8.	$\sqrt{x^3-2y}$	22.	$x/\lvert log_3(y)\rvert$	36.	$y/cos(1/x^2)$
9.	$2^{x/\sqrt{y}}$	23.	$x/(1/y-5)$	37.	$y/arctan(1/x)$
10.	$x/arccos(1/y)$	24.	$x/exp(y)$	38.	$(1/x)*y^2$
11.	$\sqrt{\cos(x+y)}$	25.	$\sqrt{cos(x/y)}$	39.	$ln(y/x)$

Вариант	G(*x,y*)	Вариант	G(*x,y*)	Вариант	G(*x,y*)
12.	$exp(y/ln(x))$	26.	$ln\,(y)\sqrt{x}$	40.	$Log_2(y/x)$
13.	$y/lg(x)$	27.	$ch(x)/exp(y)$	41.	
14.	$\sqrt{x}/ln(y)$	28.	$y\times 2^{ln(x-1)}$	42.	

Наборы исходных данных для вариантов:

X\Y	ФШy	ФТy	ФГy	Пy
ФШx	1.	2.	3.	4.
ФТx	5.	6.	7.	8.
ФГx	9.	10.	11.	12.
Пx	13.	14.	15.	16.
ФШx	17.	18.	19.	20.
ФГx	21.	22.	23.	24.
ФТx	25.	26.	27.	28.
Пx	29.	30.	31.	32.
ФШx	33.	34.	35.	36.
ФГx	37.	38.	39.	40.

- **Фиксированный, с заданным шагом (ФШx, ФШy).**
- **Фиксированный, с заданным количеством точек (ФТx, ФТy)**
- **Фиксированный с заданными границами (ФГx, ФГy)**
- **Произвольный (Пx, Пy)**

Пример ФШx: Задано начальное значение x_0, конечное x_k и шаг h_x.
Пример ФТx: Задано начальное значение x_0, количество точек N_x и шаг h_x.
Пример ФГx: Задано начальное значение x_0, конечное x_k и количество точек N_x.
Пример Пx: Задано количество точек N_x и все значения x_i (i=1.. N_x) в возрастающем порядке.

б) В ходе выполнения программы должен создаваться следующие файлы последовательного доступа:

- Файл ***myProgram.log*** в котором содержатся
  1. Название программы и номер варианта
  2. Дата и время начала выполнения расчёта
  3. Рассчитываемая функция
     (*Примечание*: символы типа корня заменить на соответствующие степени).
  4. Названия файлов, содержащих результаты расчётов по заданным наборам исходных данных (G1.dat, G2.dat и т.д.).
- Файл регистрации ошибок ***myErrors.log*** в котором собрать ошибки в формате:
  1. Имя файла данных (G1.dat, G2.dat и т.д.).
  2. Рассчитываемая функция (*Примечание*: символы типа корня заменить на соответствующие степени).
  3. Аргументы x, y
  4. Тип ошибки (деление на 0, переполнение, и пр.)
- Файлы данных с именами формата G####.dat (#### – номер набора исходных данных), под каждый набор исходных данных. Вывести **в каждый** такой файл следующую информацию:
  1. Рассчитываемая функция (Примечание: символы типа корня заменить на соответствующие степени).
  2. Количество точек для *x* и *y* для аргументов функции G(x, y).
  3. Значения функции G(*x*, *y*) в табличном виде с выравниванием элементов так, чтобы:
     a. **Чётные варианты** – в заголовках строк значения *y*, в заголовках столбцов – *x*
     b. **Нечётные варианты** – в заголовках строк значения *x*, в заголовках столбцов – *y*

Шаблон таблицы для G(x,y) для **нечётных** вариантов:

x\y	**y_0,**	**y_1,**	**y_2…**	**y_n**
x_0,	$G(x_0, y_0)$,	$G(x_0, y_1)$,	$G(x_0, y_2)$,…	$G(x_0, y_k)$
x_1,	$G(x_1, y_0)$,	$G(x_1, y_1)$,	$G(x_1, y_2)$,…	$G(x_1, y_k)$
….				
x_m,	$G(x_n, y_0)$,	$G(x_n, y_1)$,	$G(x_n, y_2)$,…	$G(x_n, y_k)$

x_i и y_j – конкретные числовые значения.

Примечание: В случае, когда для какого-то набора аргументов функция не определена в соответствующей позиции G(x, y) поместить «NaN» или «Null». При выводе сохранять все значащие цифры аргументов и функций!

в) Создать модуль, считывающий из dat-файлов, описанных в myProgram.log, всех данные в соответствующие массивы или табличные элементы управления.

- Предусмотреть возможность ошибок при считывании (достижение конца файла, считывание некорректных данных)
- Считанные данные должны быть доступными к использованию другими программными единицами через заголовок функции в виде соответствующих векторов *x*, *y* и матрицы G.

Лабораторная работа №6

(Варианты берутся по 2-му заданию лабораторной работы 5)

а) Вывести данные по расчётам в файлы с именами формата G####.rez (#### – номер набора исходных данных), под каждый набор исходных данных. При выводе использовать указатели позиционирования.

б) Исходные данные для каждого набора данных вывести:
- для чётных вариантов – в файл **произвольного** доступа G####.rez из предыдущего пункта
- для нечётных вариантов – в отдельный файл **произвольного** доступа Calc.ini

в) Создать модуль, выполняющий считывание из rez-файлов,
- Для номеров вариантов, заканчивающихся на 0…4 – заданные с помощью пар индексов значения.
  Например: Считать из файла 2-го набора (G0002.rez) данные в точках x, y с координатами (0, 5), (1, 7), (3, 1).
- Для номеров вариантов, заканчивающихся на 5…9 – заданные с помощью координат двух углов подматрицы.
  Например: Считать из файла 3-го набора (G0003.rez) подматрицу (0,0)(2,3). Это будет матрица 3×4.

г) Вывести считанные значения или подматрицы на форму.

д) Предусмотреть возможность ошибок при считывании (достижение конца файла, считывание некорректных данных)

е) Считанные данные должны быть доступными к использованию другими программными единицами через заголовок функции. (т.е. не производить прямой вывод на форму)

Примечание 1: Данные о наличии ошибок в ходе вычисления функций можно хранить в отдельных файлах произвольного доступа, либо внутри *.rez файлов с использованием структурирования данных.

Примечание 2. Выполнением данной работы считается реализация доступа к любому элементу файла по положению указателя. Это может быть бинарный или текстовой тип файла. Допустимо использование структурированных файлов любых форматов, допускающих позиционное считывание записей.

Примечание 3. Исходные наборы данных могут задаваться из файла задания, с консоли или с графического пользовательского интерфейса в виде необходимых целых или вещественных чисел в зависимости от варианта.

Примечание 4: В C# можно воспользоваться методом FileStream.Seek() или свойством BinaryReader.BaseStream для позиционирования указателя файла.