Министерство науки и высшего образования РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа киберфизических систем и управления

Лабораторная работа

по дисциплине «Кроссплатформенное программирование»

«Калькулятор»

Выполнил:

студент гр. 3530902/70201 _____ Матченко Т.И.

подпись, дата

Проверил: _____ Хлопин С.В.

доцент, к.т.н. подпись, дата

Санкт-Петербург

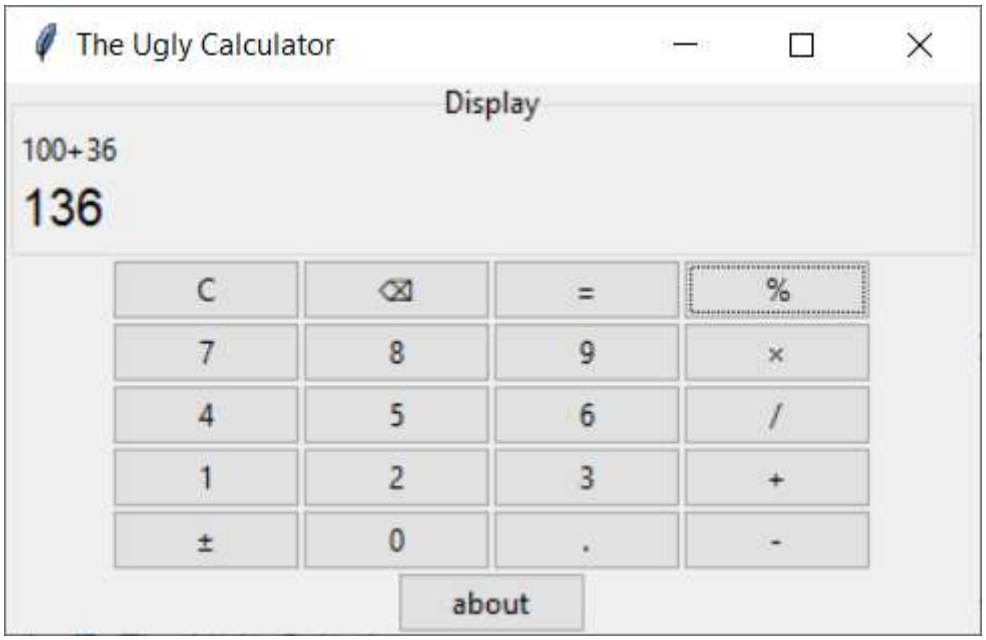
2020

Оглавление

Задание и описание программы

Была написана программа – калькулятор. Окно содержит:

- дисплей – большими жирными символами снизу пишется текущее число или результат, обычными символами сверху пишется история;
- цифровую клавиатуру (кнопки «0» - «9»);
- кнопки арифметических операций (- / + ×);
- кнопка «±» позволяет сменить знак;
- кнопка «.» позволяет вводить дробные числа;
- кнопка «C» стирает и текущее число, и историю, кнопка «⌫» стирает только последний символ в текущем числе;
- кнопка «%» позволяет выполнять операции с процентами;
- кнопка «=» позволяет получить результат;
- кнопка «about» позволяет получить информацию об авторе.



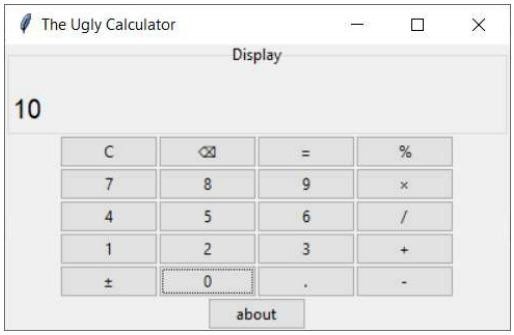
Программа работает так же, как и стандартный калькулятор Windows. С помощью данного калькулятора можно выполнять арифметические операции с вещественными числами. Согласно личному варианту, присутствует и возможность работать с процентами.

Примеры работы программы

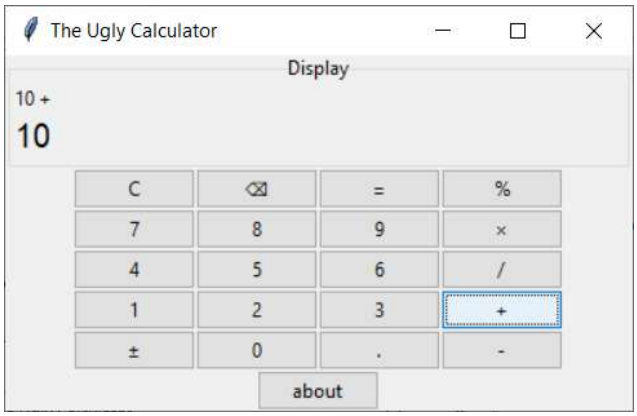
Рассмотрим работу калькулятора на тестовых сценариях.

1. Запись выражения

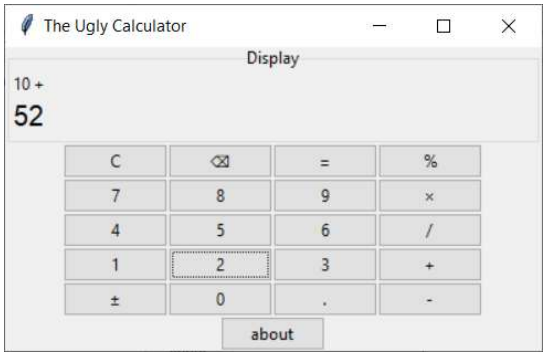
Вводим «10».



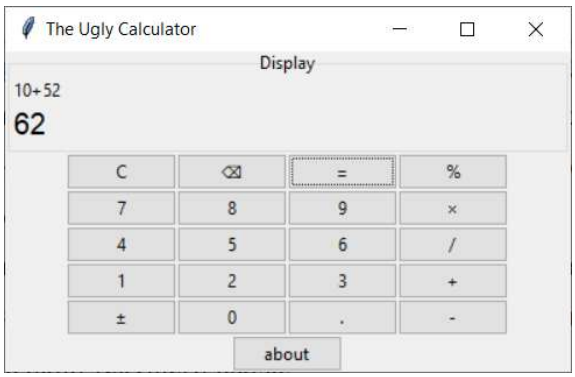
Нажимаем «+».



Вводим «52».

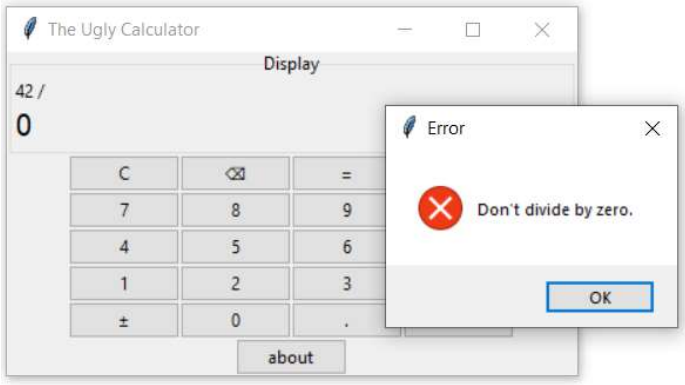


Нажав «=», получаем результат.



2. Деление на 0

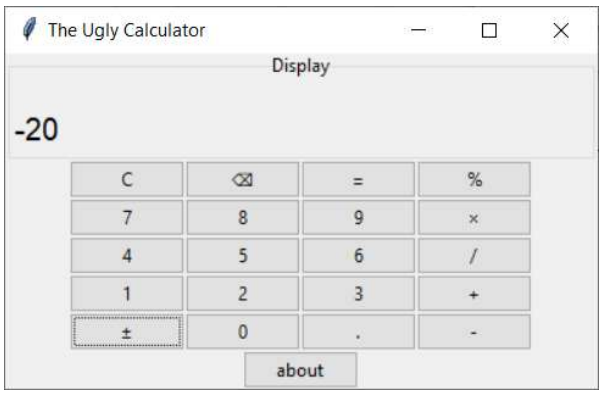
Попытавшись разделить на 0, получаем ошибку.



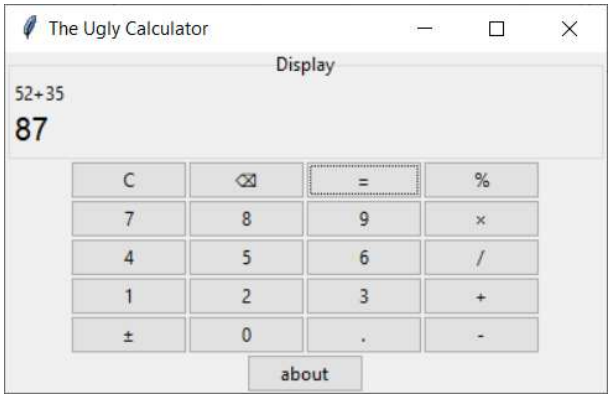
После нажатия «ОК» окно ошибки исчезает, а калькулятор возвращается к изначальному состоянию – на дисплее есть только 0.

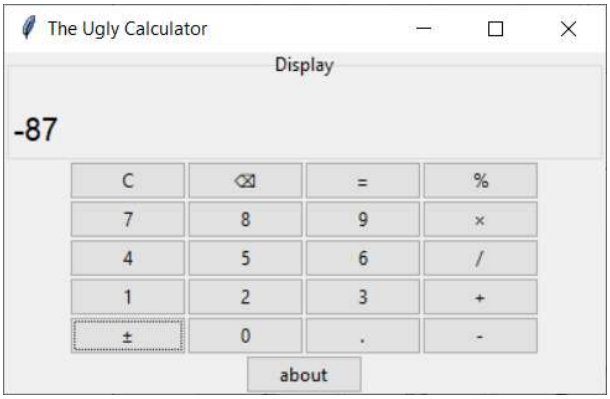
3. Смена знака

Сначала введем число (не 0), потом нажмем ±.

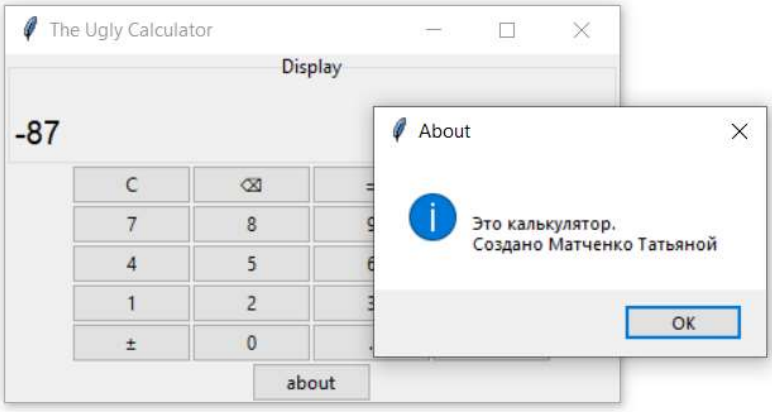


Проверим работу ± на результате выражения:



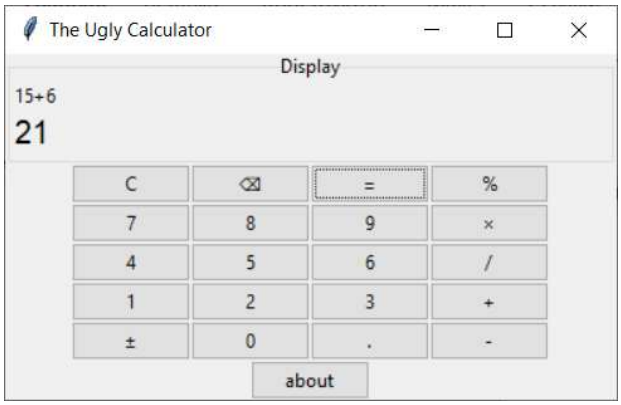


4. About

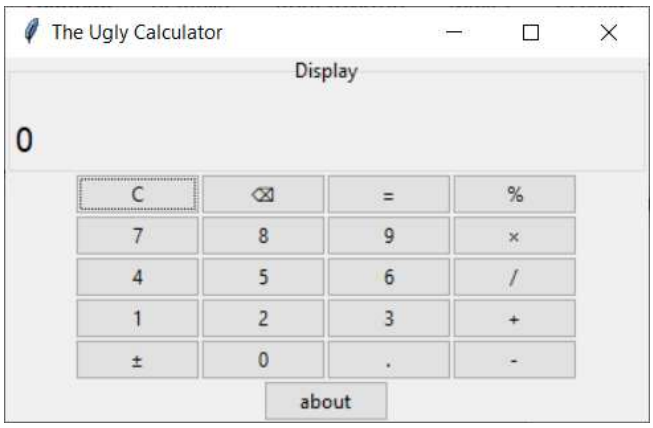


5. Очистка экрана

Введем выражение:

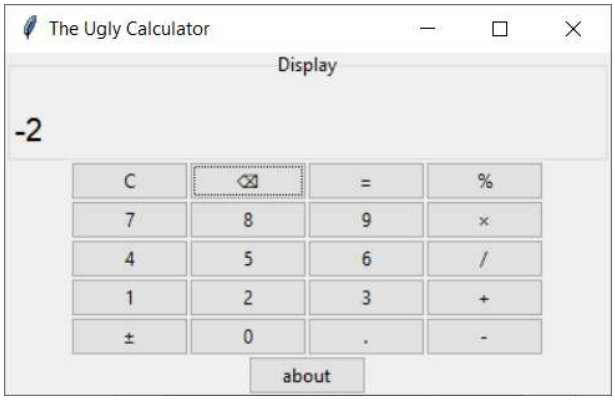


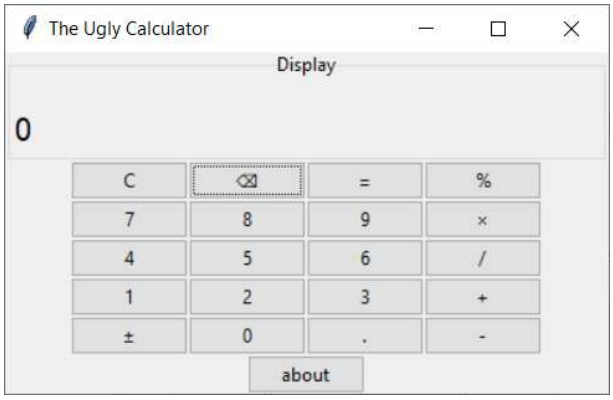
Сотрем:



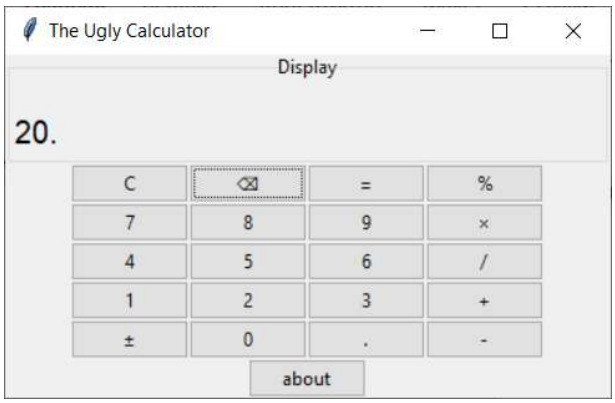
6. Стирание посимвольно

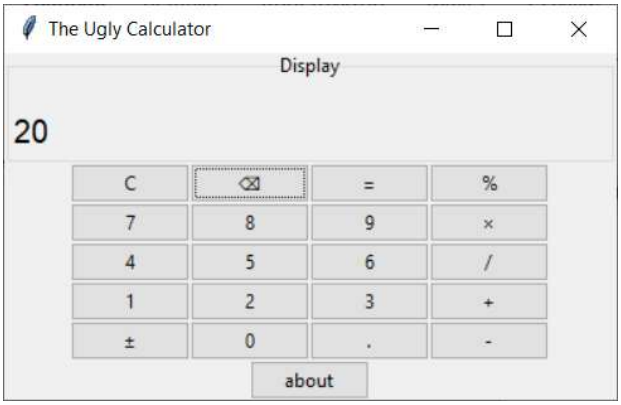
Отрицательные числа:

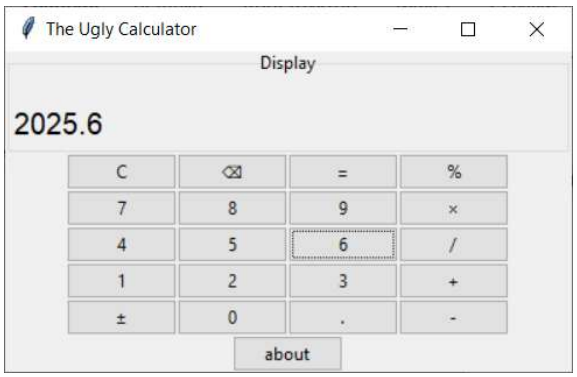




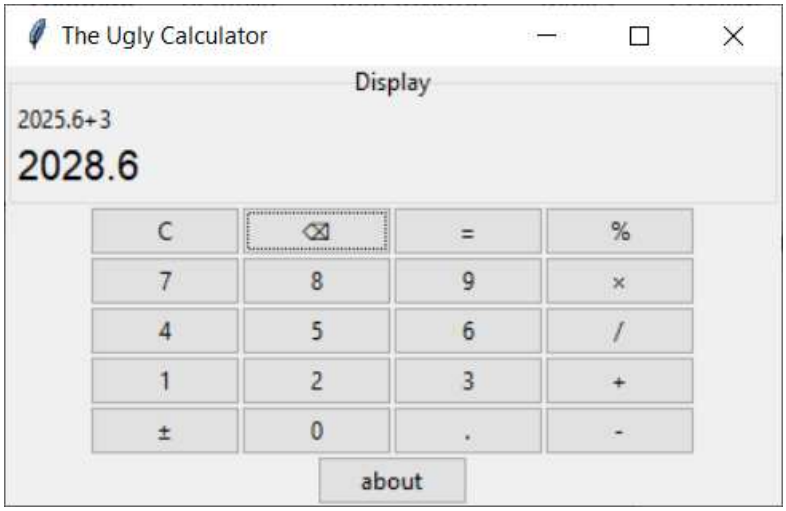
Сотрем точку, чтобы проверить, сможем ли мы поставить ее потом:





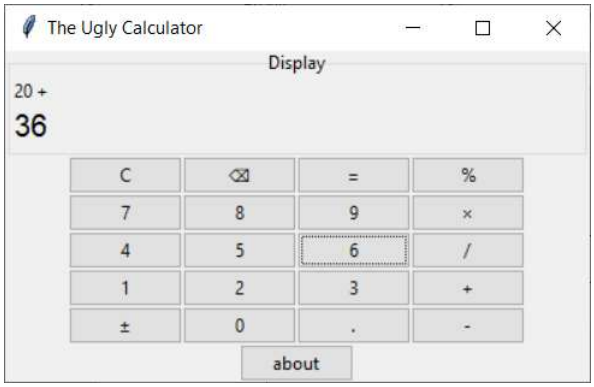


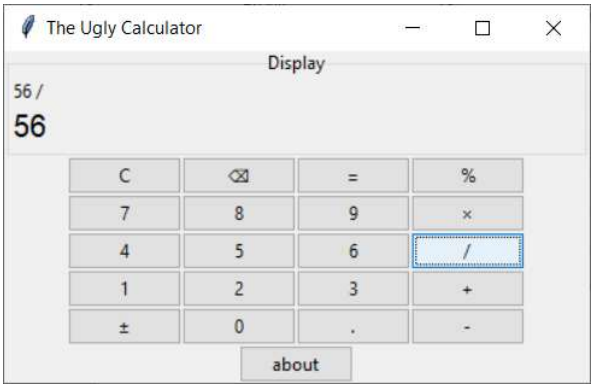
Стирание посимвольно результата выражения не работает и не должно.

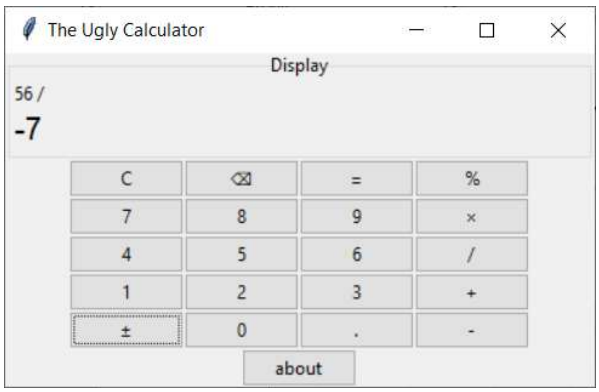


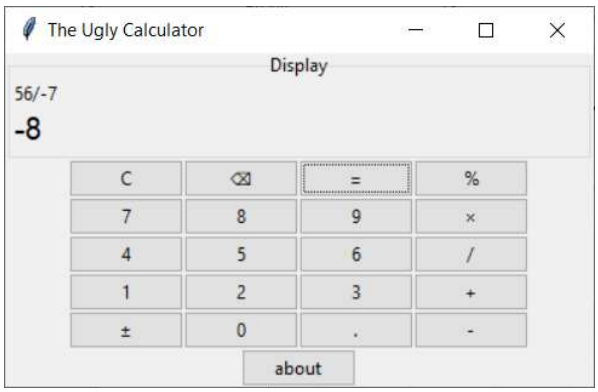
7. Цепные вычисления

Как и в обычном калькуляторе, мы можем получать результат и продолжать работать с ним, не нажимая «=».



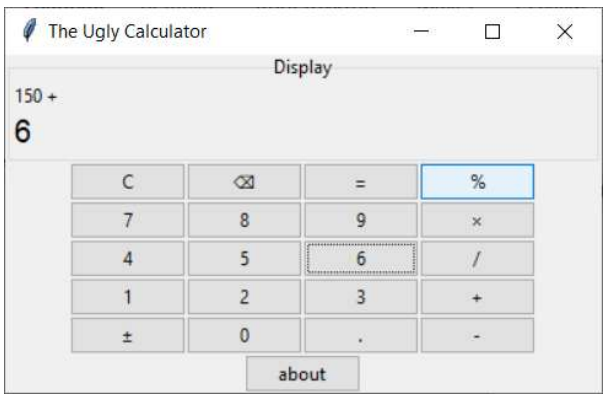


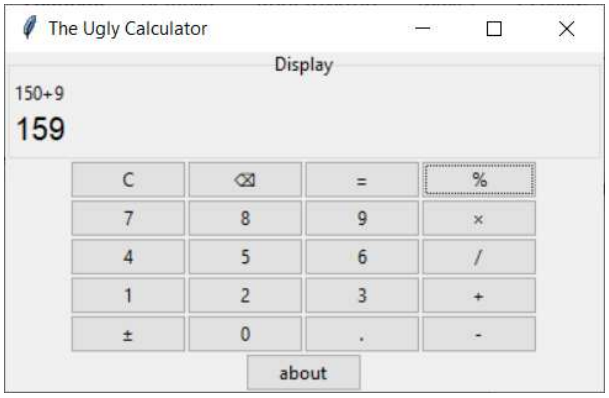




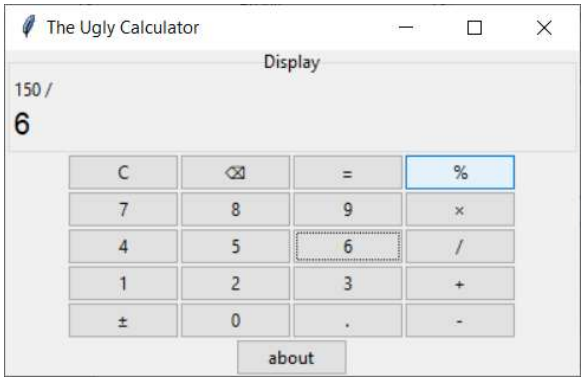
8. Процент

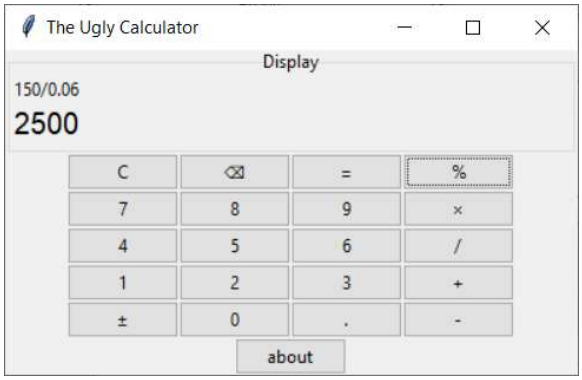
Сложение/вычитание (т.к. работают по одному принципу). 150 + 6%:



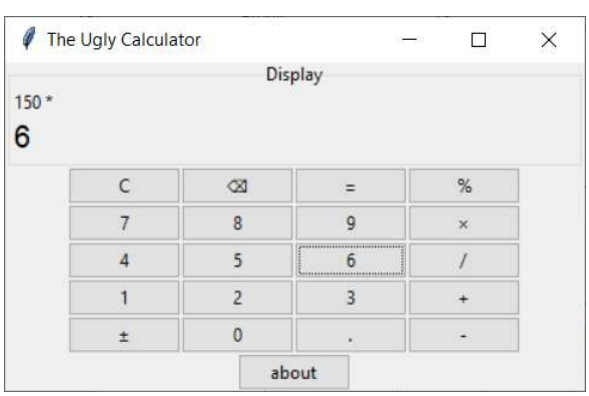


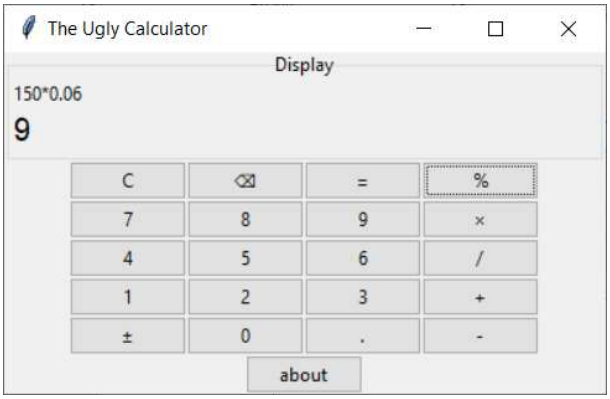
Деление. 150/6% (150 – это 6%. Сложно будет 100% ?)





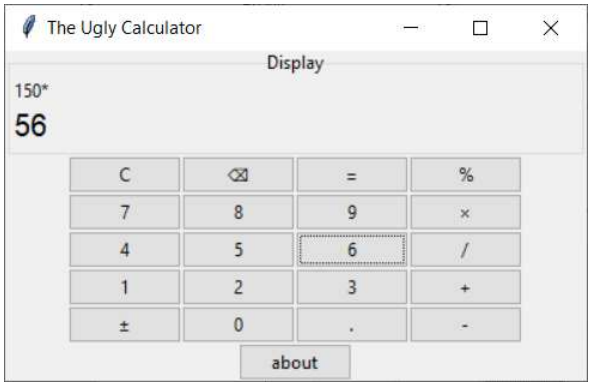
Умножение. 150*6% (6% от 150):

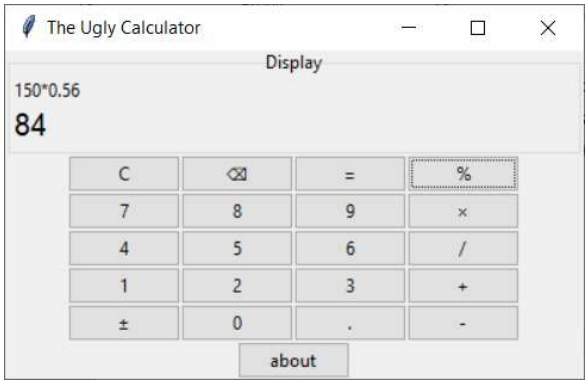




Если в стандартном калькуляторе Windows, которым мы вдохновляемся, после этого начать вводить новое число, первое число (150 в нашем случае) и арифметическая операция (умножение) сохранятся. Вводя новое число после использования процента, мы меняем второй операнд.

56% от 150:





Однако если бы мы не нажали % опять, калькулятор бы просто посчитал 150*56 как обычно, сохранение первого операнда и операции на этом бы закончилось.

Выводы

В ходе лабораторной работы была написана программа – калькулятор с функционалом стандартного приложения «калькулятор» Windows. Программа может выполнять стандартные операции приложения, а также информацию об авторе. Были прописаны основные ограничения – невозможность начать число с нескольких нулей, делить на ноль, ставить в одном числе несколько точек.

Приложение

Листинг:

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from decimal import Decimal

def pretty_decimal(num:Decimal) -> str:
    '''
    Makes a decimal number pretty (rounds to 11 and converts to str).
    '''
    #tmp = num.quantize(Decimal('1.00000000000'), ROUND_HALF_UP, context=Contex
t(prec=40))
    tmp = '{:f}'.format(num)
    if '.' in tmp:
        tmp = (' '+tmp).strip('0')[1:].strip('.')
    return tmp

class CalculatorApp:

    def __init__(self, root):

        self._dot_placed = False
        self._digits_count = 1
        self._choosing_operator = False #возможность смены оператора включена?
        self._needs_erase_number = False # "CE" в калькуляторе
        self._needs_erase_history = False
        self._changing_second_operand = False # появляется после %. num1 и oper
ator сохраняются. меняем только num2
        self._percented = False #костыль, запрещающий дважды нажать %

        def input_number(number):
            self._choosing_operator = False # закончилась возможность смены опе
ратора
            self._percented = False # можно снова использовать процент

            if self._needs_erase_number:
                clean(full=False) # стирается только дисплей

            if self._needs_erase_history:
                    self._history.configure(text='')
                    self._needs_erase_history = False

            # если после операции с процентом нажать на цифру,
            # должны сохраниться num1 и operator
            if self._changing_second_operand:
                self._history.configure(text=pretty_decimal(self._num1)+self._o
perator)
```

```
                self._changing_second_operand = False

            if self._display['text'] == '0':
                # если там стоял просто 0, заменяем его на number
                self._display.configure(text=number)
            else:
                #18 цифр - лимит пользовательского ввода
                if self._digits_count < 18:
                    self._display.configure(text=self._display['text']+number)
                    self._digits_count += 1

        def input_dot():
            if self._needs_erase_number:
                input_number('0')
                input_dot()
                return

            if not self._dot_placed:
                if len(self._display['text']) == 0:
                    self._display.configure(text='0.')
                else:
                    self._display.configure(text=self._display['text']+'.')
                self._dot_placed = True

        def plus_or_minus():
            if len(self._display['text']) > 0 and self._display['text'] != '0':
                if self._display['text'][0] == '-':
                    #если отрицательно, стираем минус в начале
                    self._display.configure(text=self._display['text'][1:])
                else:
                    #если минуса в начале нет, ставим
                    self._display.configure(text='-'+self._display['text'])

                self._choosing_operator = False

                if self._needs_erase_number:
                    self._dot_placed = '.' in self._display['text']
                    self._needs_erase_number = False

                if self._needs_erase_history:
                    self._history.configure(text='')
                    self._needs_erase_history = False

        def backspace():
            if self._needs_erase_number: return
            if len(self._display['text']) > 0:
                #разрешаем снова ставить точку
                if self._display['text'][-1] == '.':
                    self._dot_placed = False
                else:
```

```
                    self._digits_count -= 1

                #стираем символ
                self._display.configure(text=self._display['text'][:-1])

                #если там в итоге остался только знак/пустая строка, ставим нол
ь
                if self._display['text'] == '-' or not self._display['text']:
                    self._display.configure(text="0")
                    self._digits_count = 1

        def clean(full=True):
            self._display.configure(text="0")
            self._dot_placed = False
            self._digits_count = 1
            self._needs_erase_number = False
            if full:
                self._history.configure(text="")
                self._num1, self._num2, self._operator = '', '', ''
                self._changing_second_operand = False

        self._operator = ''
        self._num1 = ''
        self._num2 = ''

        def store_num(oper=''):
            #беру текст с дисплея, привожу в норм вид
            tmp = self._display['text']
            if tmp[-1] == '.': tmp = tmp[:-1]  # обрезаем точку на конце
            if float(tmp) == 0: tmp = '0'  # делаем ноль
            self._display.configure(text=tmp)

            tmp = Decimal(tmp)

            if self._num1 == '':
                # если первый операнд пустой, записываем число в него
                self._num1 = tmp
                self._history.configure(text=pretty_decimal(tmp)+' '+oper)
            else:
                self._num2 = tmp
            self._digits_count = 1
            self._needs_erase_number = True

        def result():
            if self._operator == '+':
                res = pretty_decimal(self._num1 + self._num2)
            elif self._operator == '-':
                res = pretty_decimal(self._num1 - self._num2)
            elif self._operator == '*':
                res = pretty_decimal(self._num1 * self._num2)
```

```
            elif self._operator == '/':
                try:
                    res = pretty_decimal(self._num1 / self._num2)
                except ZeroDivisionError:
                    messagebox.showerror("Error","Don't divide by zero.")
                    clean(True)
                    return

            self._display.configure(text=res)
            self._history.configure(text=pretty_decimal(self._num1) + self._ope
rator + pretty_decimal(self._num2))
            self._digits_count = len([digit for digit in self._display['text']
if digit!='.'])

            #если последним был процент, то сохраняем num1 и operator
            if not self._changing_second_operand:
                self._num1, self._operator = '', ''
            self._num2 = ''



        def operator(oper):
            if self._changing_second_operand:
                self._num1 = ''
                self._changing_second_operand = False
            self._percented = False

            self._needs_erase_history = False

            if not self._choosing_operator:
                store_num(oper)

            if self._num2 != '':
                result()
                self._choosing_operator = False
                operator(oper)

            if not self._choosing_operator:
                self._operator = oper
                self._choosing_operator = True
            else:
                self._history.configure(text=self._history['text'][:-1]+oper)
                self._operator = oper

        def equal():
            if self._needs_erase_number: return

            self._percented = False

            if self._operator != '':
```

```
                self._changing_second_operand = False
                self._choosing_operator = False
                store_num()
                result()
                self._needs_erase_history = True
                self._needs_erase_number = True

        def percent():
            if self._percented: return

            if self._operator != '':
                tmp = self._display['text']
                if tmp[-1] == '.': tmp = tmp[:-1]  # обрезаем точку на конце
                if float(tmp) == 0: tmp = '0'  # делаем ноль
                self._display.configure(text=tmp)

                tmp = Decimal(tmp)

                if self._operator == '+' or self._operator == '-':
                    #превращает num2 в newnum2=num1*0.01*num2
                    newnum2 = self._num1 * Decimal('0.01') * tmp
                else:
                    newnum2 = Decimal('0.01') * tmp

                #поменяем число на дисплее а потом запустим стор нам
                self._display.configure(text=pretty_decimal(newnum2))
                store_num(self._operator)

                #если будет нажата цифра, num1 и oper сохраняется, меняется тол
ько num2.
                # произвести num1 +oper+ newnum2, повторить выбранный оператор
и напечатать цифру
                self._changing_second_operand = True
                self._percented = True
                result()
                self._needs_erase_number = True

        #размещаем интерфейс

        #фреймы
        #фрейм экрана
        self._display_frame = ttk.LabelFrame(
            root, height=70, width=385, text='Display', labelanchor='n')
        self._display_frame.pack()
        self._display_frame.grid_propagate(0)
        #фрейм кнопок
        self._buttons_frame = ttk.Frame(root)
        self._buttons_frame.pack()

        #экран
```

```
        #выражение
        self._history = ttk.Label(self._display_frame, text='')
        self._history.grid(row=0, sticky='w')
        #набор
        self._display = ttk.Label(self._display_frame, text='0')
        self._display.configure(font=40)
        self._display.grid(row=1, sticky='w')

        #кнопочки ряд 1: C ⌫ = %
        ttk.Button(self._buttons_frame, text="C", command=clean).grid(row=2, co
lumn=0)
        ttk.Button(self._buttons_frame, text="⌫
",command=backspace).grid(row=2, column=1)
        ttk.Button(self._buttons_frame, text="=", command=equal).grid(row=2, co
lumn=2)
        ttk.Button(self._buttons_frame, text="%", command=percent).grid(row=2,
column=3)
        #кнопочки ряд 2: 7 8 9 ×
        ttk.Button(self._buttons_frame, text="7", command=lambda: input_number(
"7")).grid(row=3, column=0)
        ttk.Button(self._buttons_frame, text="8", command=lambda: input_number(
"8")).grid(row=3, column=1)
        ttk.Button(self._buttons_frame, text="9", command=lambda: input_number(
"9")).grid(row=3, column=2)
        ttk.Button(self._buttons_frame, text="×", command=lambda: operator('*')
).grid(row=3, column=3)
        #кнопочки ряд 3: 4 5 6 ÷
        ttk.Button(self._buttons_frame, text="4", command=lambda: input_number(
"4")).grid(row=4, column=0)
        ttk.Button(self._buttons_frame, text="5", command=lambda: input_number(
"5")).grid(row=4, column=1)
        ttk.Button(self._buttons_frame, text="6", command=lambda: input_number(
"6")).grid(row=4, column=2)
        ttk.Button(self._buttons_frame, text="/", command=lambda: operator('/')
).grid(row=4, column=3)
        #кнопочки ряд 4: 1 2 3 +
        ttk.Button(self._buttons_frame, text="1", command=lambda: input_number(
"1")).grid(row=5, column=0)
        ttk.Button(self._buttons_frame, text="2", command=lambda: input_number(
"2")).grid(row=5, column=1)
        ttk.Button(self._buttons_frame, text="3", command=lambda: input_number(
"3")).grid(row=5, column=2)
        ttk.Button(self._buttons_frame, text="+", command=lambda: operator('+')
).grid(row=5, column=3)
        #кнопочки ряд 5: ± 0 . -
        ttk.Button(self._buttons_frame, text="±", command=plus_or_minus).grid(r
ow=6, column=0)
        ttk.Button(self._buttons_frame, text="0", command=lambda: input_number(
"0")).grid(row=6, column=1)
```

```
        ttk.Button(self._buttons_frame, text=".", command=input_dot).grid(row=6
, column=2)
        ttk.Button(self._buttons_frame, text="-", command=lambda: operator('-
')).grid(row=6, column=3)
        #кнопка About
        ttk.Button(self._buttons_frame, text="about", command=lambda: messagebo
x.showinfo(
            "About", "\nЭто калькулятор.\nСоздано Матченко Татьяной")).grid(row
=7, column=0, columnspan=4)

        self._buttons_frame.update()
        self._display_frame.update()
        height = self._buttons_frame.winfo_height()+self._display_frame.winfo_h
eight()
        width = self._display_frame.winfo_width()+4
        root.minsize(width, height)


def main():
    root = Tk()
    root.title("The Ugly Calculator")
    app = CalculatorApp(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```