

Объектно ориентированное программирование

Домашнее задание №3

Задание:

В этой домашней работы вам необходимо создать приложение с графическим интерфейсом пользователя с использованием WPF. Внешний вид приложения, количество и состав графических компонентов необходимо определить самостоятельно исходя из задания.

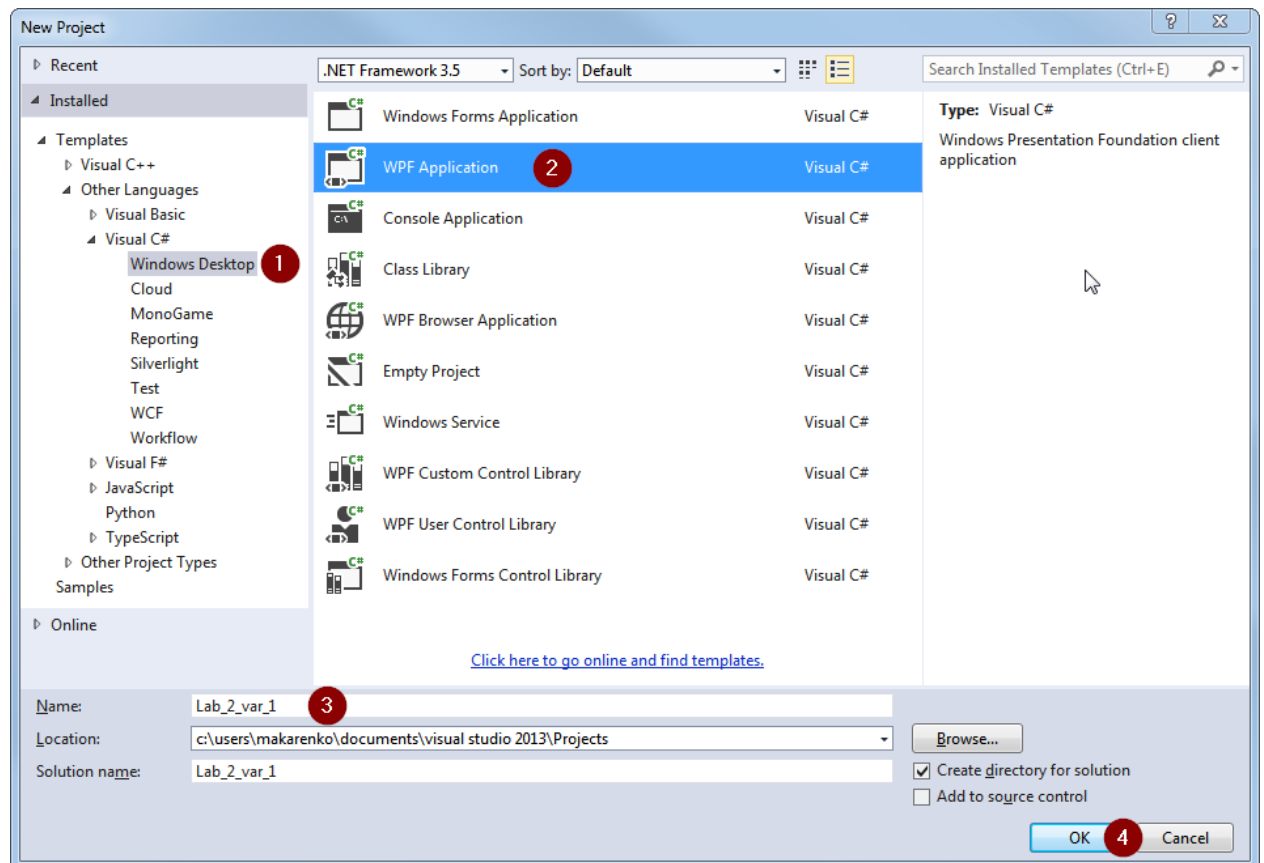
Варианты заданий даны в конце документа.

Пример создания графического приложения

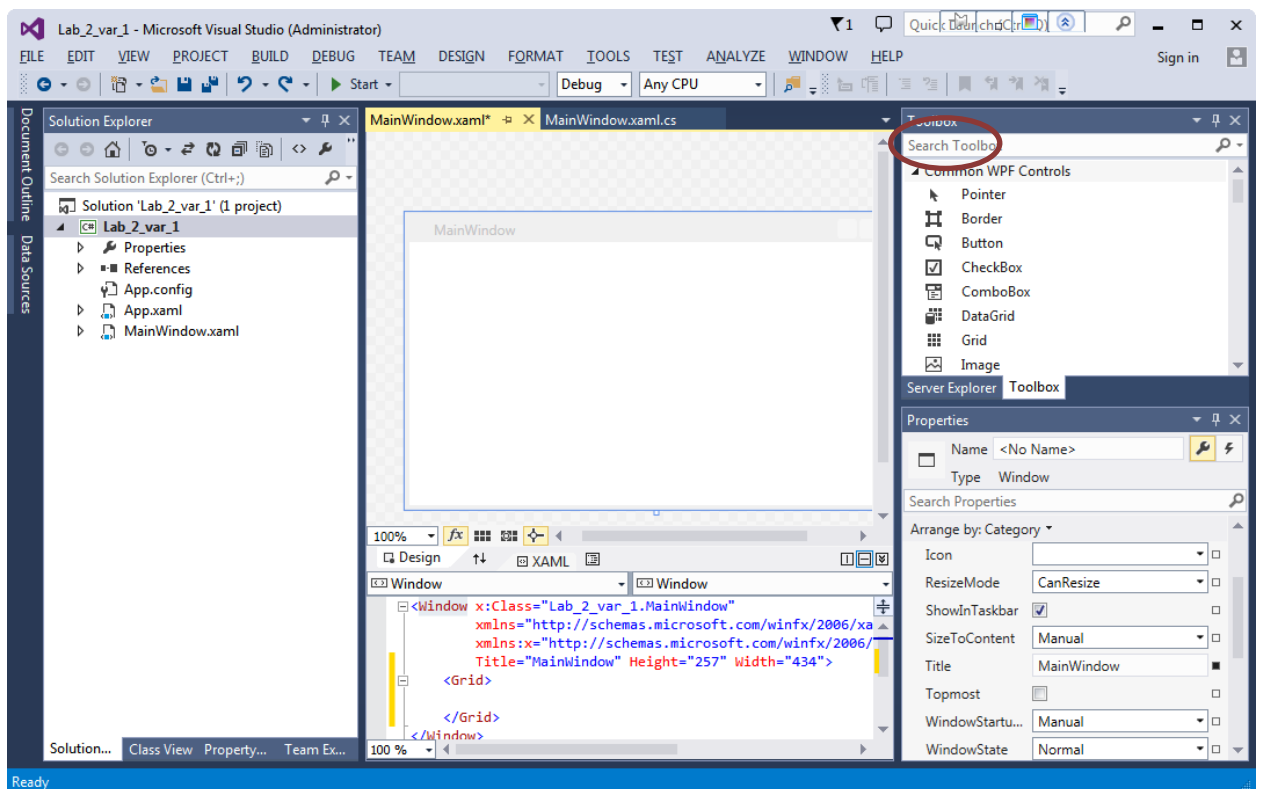
Создадим простую программу, которая будет складывать два введенных числа.

Новый проект

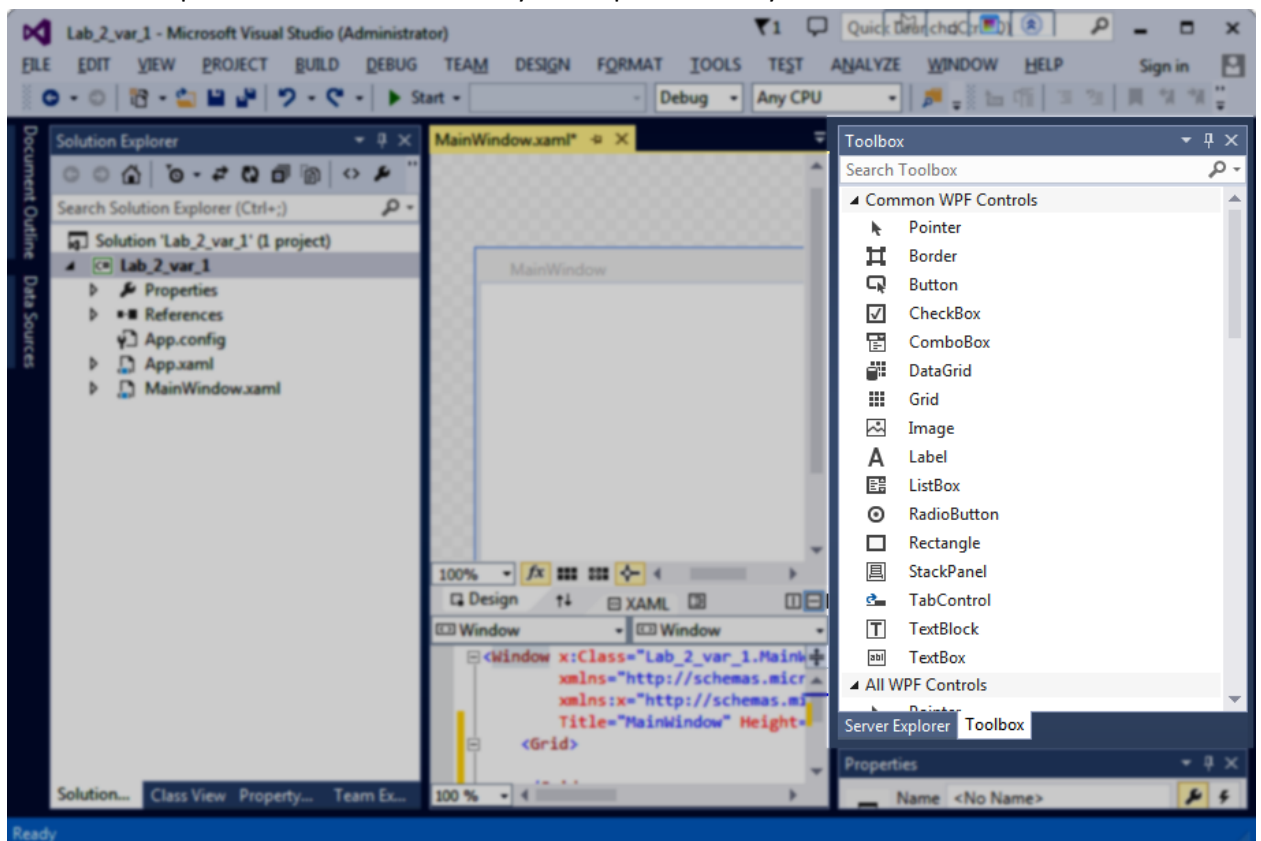
Для создания проекта с использованием WPF, необходимо открыть диалог “New Project” и выбрать тип шаблона – “WPF Application”. Затем задать корректное имя проекта, и нажать «OK».



После этого перед вами откроется окно дизайнера форм, в котором будет открыта ваша первая форма приложения.



По умолчанию форма пустая и не содержит никаких элементов управления, чтобы их добавить следует воспользоваться панелью Toolbox, открыть которую можно нажатием на соответствующую кнопку в правой части окна. Чтобы панель не пропадала после потери фокуса, ее можно закрепить нажатием на иконку с изображением булавки в заголовке.


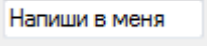
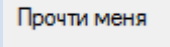
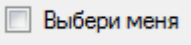
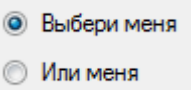
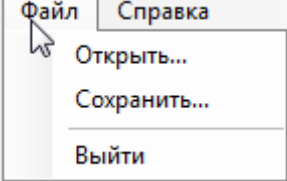
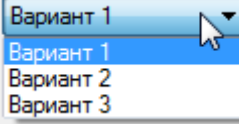


Графические элементы

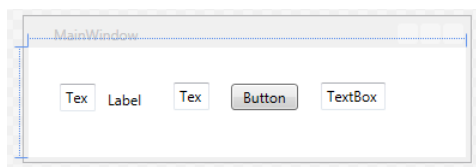
В открывшейся панели Toolbox содержатся стандартные графические элементы: кнопки, поля ввода, тестовые подписи, которые вы можете использовать в своем приложении. Подробное описание доступных элементов можно найти в справочнике разработчика – MSDN, для этого нужно нажать на интересующий элемент и нажать клавишу F1, после чего в браузере откроется страница с описанием элемента.

Чтобы добавить выбранный элемент на форму, нужно либо дважды кликнуть по нему, либо перетащить из Toolbox на форму.

Наиболее часто используемые элементы:

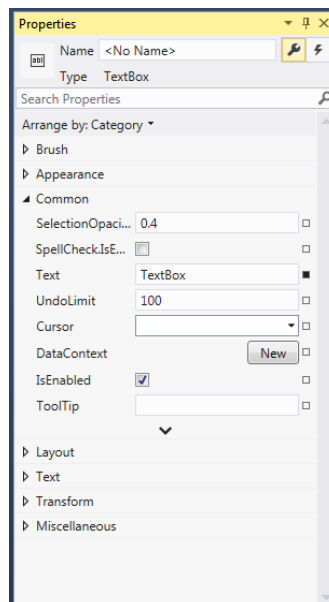
Название	Изображение	Описание
Button		Обычная кнопка
TextBox		Поле ввода, используется для ввода текста с клавиатуры. Может быть однострочным и многострочным
Label		Тестовая метка, используется для подписи полей ввода, списков, а так же для вывода небольших информационных сообщений
CheckBox		Флажок, используется для включения и выключения каких-либо настроек
RadioButton		Переключатель, используется для выбора из нескольких вариантов
Menu		Главное меню программы
ComboBox		Выпадающий список

Добавим на форму несколько элементов: три поля ввода, одну кнопку и одну текстовую метку:



Свойства элементов

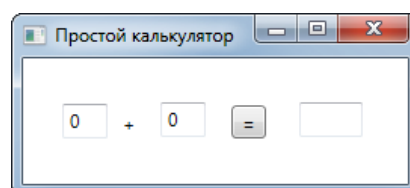
После этого, необходимо настроить свойства каждого элемента. Для этого необходимо открыть и зафиксировать панель “Properties”. В открывшейся панели будут отображаться все свойства активного (выбранного) элемента.



Подробное описание каждого свойства можно узнать в MSDN (выделить свойство и нажать F1).
Здесь приведены только некоторые, наиболее важные свойства.

Название	Описание
Name	Название элемента. Именно по этому имени к элементу можно будет обращаться из кода. Ограничения такие же, как и на именованые переменных в коде. Необходимо всегда задавать корректное и осмысленное значение поля Name!
Content или Text	Текст, отображающийся в элементе.
IsEnabled	«Включенность» элемента. Это свойство включает или выключает возможность взаимодействия с элементом (нажатие на кнопку, ввод текста в поле, переключение флага)
Visibility	Видимость элемента. Позволяет временно скрывать элемент в соответствии с алгоритмом программы
Brush	Цвет текста\фона элемента
AcceptsReturn (только TextBox)	Вывод текста в многострочном режиме
IsReadOnly (только TextBox)	Запрещает редактировать текст в поле ввода
Items (только ComboBox)	Элементы списка

Зададим нужные свойства.



В первую очередь необходимо задать имена элементов, имена нужно подбирать так, чтобы перейдя к написанию кода, можно было легко вспомнить, как каждый из элементов называется. Рекомендуется использовать «венгерскую» нотацию для именования графических элементов, то есть смысловой части имени должно предшествовать двух-трехбуквенное сокращенное название

типа элемента. Например, `TextBox` – `tb`, `Button` – `btn`, `Label` – `lbl` и так далее. Это может пригодиться, в тех случаях, когда вы помните тип элемента, но не помните точно его названия, тогда после ввода первых букв типа, автодополнение редактора подскажет вам все элементы указанного типа. Зададим для добавленных элементов следующие имена `tbA`, `tbB` – для полей ввода первого и второго операндов, `lblPlus` – для символа `+`, `btnCalculate` – для кнопки, по которой будет запускаться вычисление и `tbResult` для поля результата.

Для поля результата нужно выставить свойство `ReadOnly` в `true`, так как пользователь не должен изменять его значение.

Для формы, кнопки и текстовой метки зададим соответствующие значения `Text`.

Обработчики событий

Если запустить программу сейчас, то она запустится, кнопка будет нажиматься, в поля ввода можно будет вводить числа, но никаких вычислений происходить не будет, поскольку мы только создали интерфейс программы, но не написали код, который бы заставил программу делать то, что мы от нее хотим. Теперь нужно организовать взаимодействие с пользователем.

Действия, которые производит пользователь с программой, называются **событиями (events)**.

Событиями могут быть щелчок кнопкой мыши, перемещение мыши, и множество других.

Функции, которые вызываются при наступлении события, называются **обработчиками событий (event handlers)**. При разработке пользовательского интерфейса основная задача, это создание обработчиков событий.

В нашей программе основное событие – это нажатие пользователем на кнопку «=», поэтому необходимо добавить собственный обработчик этого нажатия. Для этого достаточно дважды кликнуть по кнопке на форме. После чего откроется окно с кодом, в котором среда автоматически сгенерирует шаблон обработчика события:

```
private void btnCalculate_Click(object sender, RoutedEventArgs e)
{

}
```

По своей сути, обработчик события – это обычный метод класса формы, который принимает два параметра:

- `object sender` – ссылка на объект который отправил событие (в нашем случае это `btnCalculate`)
- `RoutedEventArgs e` – описание произошедшего события

Так как это обычный метод, то в нем можно писать обычный код:

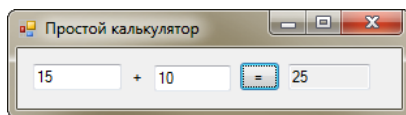
```
private void btnCalculate_Click(object sender, EventArgs e)
{
    int a = int.Parse(tbA.Text);
    int b = int.Parse(tbB.Text);
    int result = a + b;
```

```
tbResult.Text = String.Format("{0}", result);
}
```

- Первая строка: считать значение вписанное в первое текстовое поле, преобразовать его в число и сохранить в переменной **a**.
- Вторая строка: считать значение вписанное во второе текстовое поле, преобразовать его в число и сохранить в переменной **b**.
- Третья строка: сложить два числа и поместить в переменную **result**.
- Четвертая строка: преобразовать результат в строку и вывести в текстовое поле **tbResult**.

Тут видно, что ко всем графическим элементам можно обращаться из кода по тому имени, что задано в поле Name в панели свойств. Так же, из кода, доступны все свойства объекта, к ним можно обращаться через точку после имени объекта.

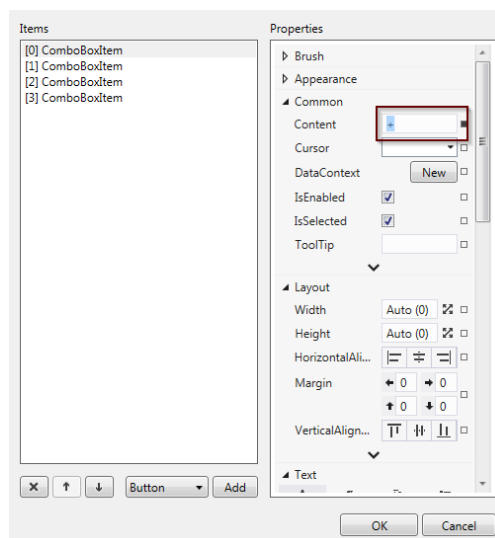
После этого можно запустить программу и проверить ее в действии:



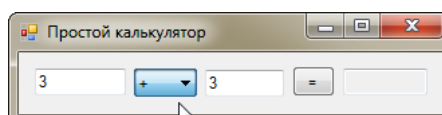
Усложняем задачу

Теперь попробуем добавить возможность не только складывать, но и вычитать, умножать и делить.

Для этого добавим на форму ComboBox, который будет определять операцию которую нужно произвести. В этот ComboBox добавим 4 элемента (свойство Items, тип ComboBoxItem) – «+», «-», «*», «/».



После этого форма должна выглядеть примерно так:



И перепишем обработчик кнопки следующим образом:

```
private void btnCalculate_Click(object sender, RoutedEventArgs e)
{
    int a = int.Parse(tbA.Text);
    int b = int.Parse(tbB.Text);

    int operation = cbOperation.SelectedIndex;
    int result = 0;

    switch (operation)
    {
        case 0:
            result = a + b;
            break;
        case 1:
            result = a - b;
            break;
        case 2:
            result = a * b;
            break;
        case 3:
            result = a / b;
            break;
    }
    tbResult.Text = String.Format("{0}", result);
}
```

Первая часть кода осталась неизменной, разберем вторую половину.

```
int operation = cbOperation.SelectedIndex;
```

Свойство `SelectedIndex` у класса `ComboBox` содержит индекс текущего выбранного элемента. Если вы добавили элементы в том порядке, что указано выше, то индексы будут соответствовать следующим операциям:

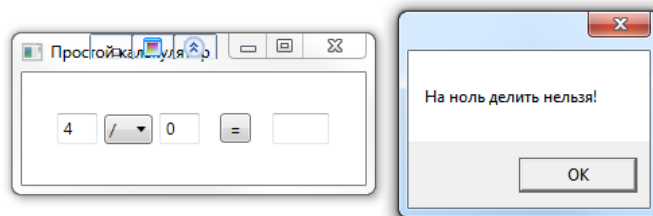
0. Сложение
1. Вычитание
2. Умножение

3. Деление

Таким образом, в переменной operation сохраняется код выбранной операции. Затем, используя этот код, в операторе switch мы выбираем, какое из действий необходимо произвести над операндами. После чего выводим результат в поле результата.

Так как пользователь может ввести второй аргумент равный 0, то может возникнуть ситуация деления на ноль, и ее нужно обработать. Для этого добавим следующие строки в часть отвечающую за деление:

```
case 3:
    if (b == 0)
    {
        MessageBox.Show("На ноль делить нельзя!");
        break;
    }
    result = a / b;
    break;
```



Таким образом, с помощью объекта MessageBox можно показывать различные уведомления для пользователя.

Вообще, для обработки таких ситуаций существует отдельный механизм называющийся Exceptions, но о нем в другой раз.

Варианты заданий:

В этом домашнем задании 10 различных вариантов, свой вариант вы выбираете в соответствии со своим номером в списке по кругу. То есть студент №15 получает 5 вариант, а студент №23 – 3 вариант.

При решении вам могут понадобиться следующие методы:

- Char.IsLetter(char c) – возвращает true, если c – буква и false в ином случае
- Char.IsDigit(char c) – возвращает true, если c – цифра и false в ином случае
- Char.IsPunctuation(char c) – возвращает true, если c – знак препинания и false в ином случае
- Char.IsWhiteSpace(char c) – возвращает true, если c – пробельный знак (пробел, табуляция и тд)
- Char.IsUpper(char c) – возвращает true, если c – прописная буква
- Char.IsLower(char c) – возвращает true, если c – строчная буква

1. Приложение для проверки знаний арифметики для младшеклассников. После запуска приложение отображает:

- a. арифметическое выражение в виде « $x + y = ?$ », где x и y – это числа от 1 до 50, а знак может быть «+» или «-».
- b. 4 варианта ответа – 1 правильный и 3 неправильных (RadioButton)
- c. кнопку «Проверить»
- d. Счетчики правильных и неправильных ответов, изначально равны нулю

После нажатия на кнопку «Проверить» приложение должно сообщать был ли выбран верный вариант и инкрементировать соответствующий счетчик. После этого приложение должно создать новый вариант задания с новыми значениями X и Y и новыми вариантами ответа.

2. Приложение для проверки знаний таблицы умножения для младшеклассников. После запуска приложение отображает:

- a. арифметическое выражение в виде « $x * y = ?$ », где x и y – это числа от 1 до 9.
- b. 4 варианта ответа – 1 правильный и 3 неправильных в виде кнопок при нажатии на которые будет произведена проверка правильности ответа
- c. Счетчики правильных и неправильных ответов, изначально равны нулю

После нажатия на одну из четырех кнопок приложение должно сообщать был ли выбран верный вариант и инкрементировать соответствующий счетчик. После этого приложение должно создать новый вариант задания с новыми значениями X и Y и новыми вариантами ответа.

3. Приложение для шифрования текста при помощи шифра Цезаря

(https://ru.wikipedia.org/wiki/Шифр_Цезаря) с произвольным смещением. Приложение должно позволять вводить произвольный текст кириллицей и задавать произвольное смещение для шифрования. Допускается шифрование только кириллических символов, но шифрование латиницы будет плюсом.

4. Приложение для шифрования текста при помощи шифра Атбаш (<https://ru.wikipedia.org/wiki/Атбаш>). Приложение должно позволять вводить произвольный текст кириллицей, зашифровывать и расшифровывать текст. Допускается шифрование только кириллических символов, но шифрование латиницы будет плюсом.
5. Приложение для шифрования текста при помощи шифра Виженера (https://ru.wikipedia.org/wiki/Шифр_Виженера). Приложение должно позволять вводить произвольный текст кириллицей, вводить кодовое слово, зашифровывать и расшифровывать текст. Допускается шифрование только кириллических символов, но шифрование латиницы будет плюсом.
6. Приложение для перевода текста в код азбуки Морзе (https://ru.wikipedia.org/wiki/Азбука_Морзе). Приложение должно позволять вводить произвольный текст кириллицей и выводить результат переведенный в азбуку Морзе. Допускается шифрование только кириллических символов, но шифрование латиницы будет плюсом.
7. Приложение подсчитывающее количество гласных и согласных букв русского алфавита. Приложение должно позволять вводить текст, выбирать режим подсчета Гласные/Согласные и выводить результат подсчета.
8. Приложение подсчитывающее количество слов во введенном тексте. Приложение должно позволять вводить произвольный текст и выводить количество найденных в нем слов. Словом является последовательность алфавитных символов ограниченная знаками препинания, пробелами, началом или концом строки. При решении задания необходимо учитывать двойные пробелы, пробелы в начале или конце строки, слова разделенные знаком препинания без пробела.
9. Приложение подсчитывающее количество предложений во введенном тексте. Приложение должно позволять вводить произвольный текст и выводить количество найденных в нем предложений. Предложением является последовательность слов, знаков препинания «запятая» «тире» «дефис» разделенных знаками «точка», «восклицательный знак», «вопросительный знак». При решении задания необходимо учитывать многоточия, последовательности ?! и тд.
10. Приложение для записи слов в предложении в обратном порядке. Например «Привет, Мир!» - «тевирП, риМ!». Приложение должно позволять вводить произвольный текст и выводить результат преобразования.