Лабораторная работа №5:
Арифметические операции с целыми числами

Цель:

Целью данной работы является получение навыков реализации арифметических операций с двоичным представлением целых чисел

Задание:

Разработать набор WPF программ на языке C#, реализующих следующие функции:

1. Сложение целых чисел, представленных в двоичной форме. Сигнатура функции может выглядеть следующим образом:
   ```
   static int[] binaryAdd(int n1, int n2)
   ```
   где параметры — это массивы, содержащие двоичное представление числа, а возвращаемое значение – массив, содержащий результат сложения в двоичном виде.

   Примечание: необходимо реализовать алгоритм побитового сложения чисел.

2. Вычитание целых чисел, представленных в двоичной форме. Сигнатура функции может выглядеть следующим образом:
   ```
   static int[] binarySub(int n1, int n2)
   ```
   где параметры — это массивы, содержащие двоичное представление числа, а возвращаемое значение – массив, содержащий результат вычитания в двоичном виде.

   Примечание: для реализации алгоритма побитового вычитания, рекомендуется использовать алгоритмы получения отрицательных чисел и побитового сложения.

3. Умножение целых чисел, представленных в двоичной форме. Сигнатура функции может выглядеть следующим образом:
   ```
   static int[] binaryMul(int n1, int n2)
   ```
   где параметры — это массивы, содержащие двоичное представление числа, а возвращаемое значение – массив, содержащий результат умножения в двоичном виде.

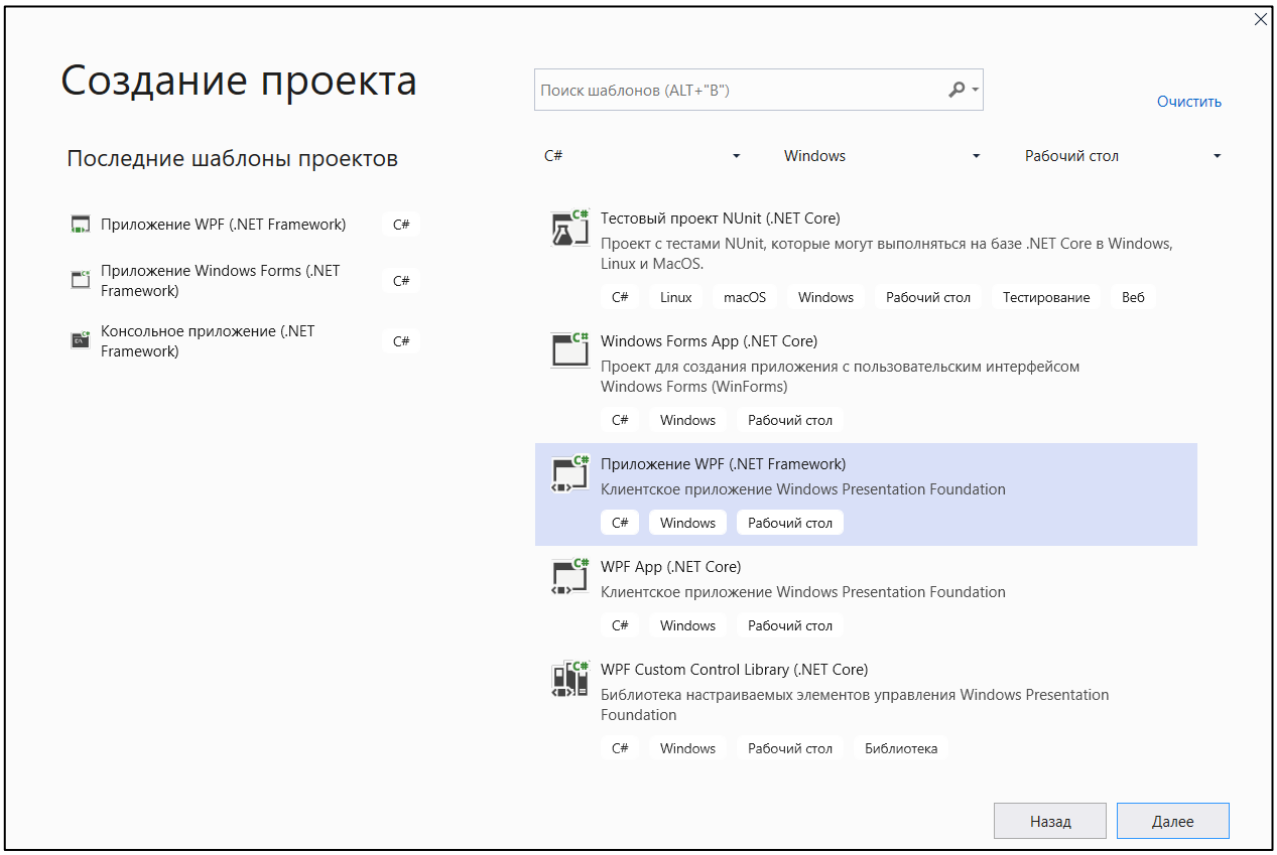
   Примечание: для реализации алгоритма побитового умножения, рекомендуется использовать алгоритмы побитового сдвига влево и побитового сложения.

4. Деление целых чисел, представленных в двоичной форме. Сигнатура функции может выглядеть следующим образом:
   ```
   static int[] binaryDiv(int n1, int n2)
   ```
   где параметры — это массивы, содержащие двоичное представление числа, а возвращаемое значение – массив, содержащий результат деления в двоичном виде.

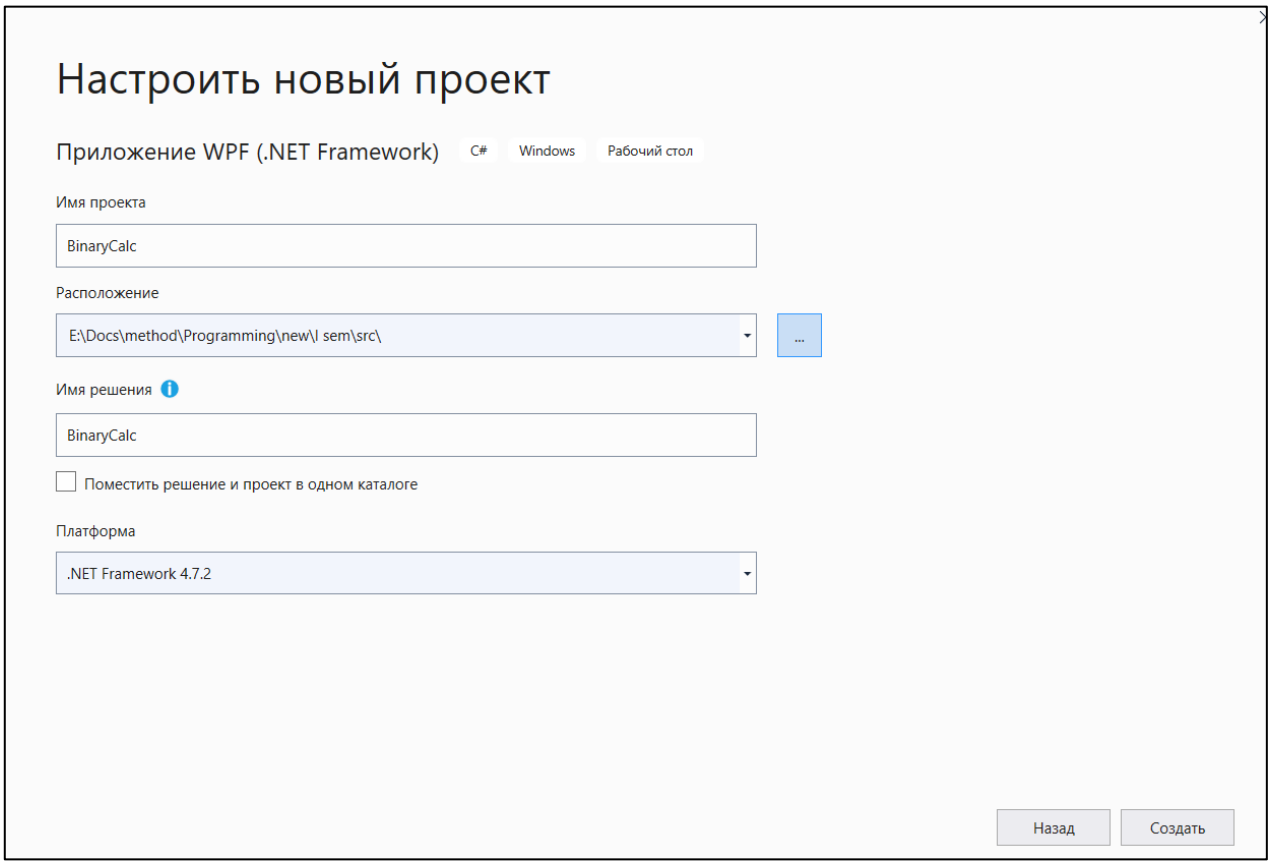
   Примечание: для реализации алгоритма побитового деления, рекомендуется использовать алгоритмы побитового сдвига вправо и побитового вычитания.

Краткая справка:

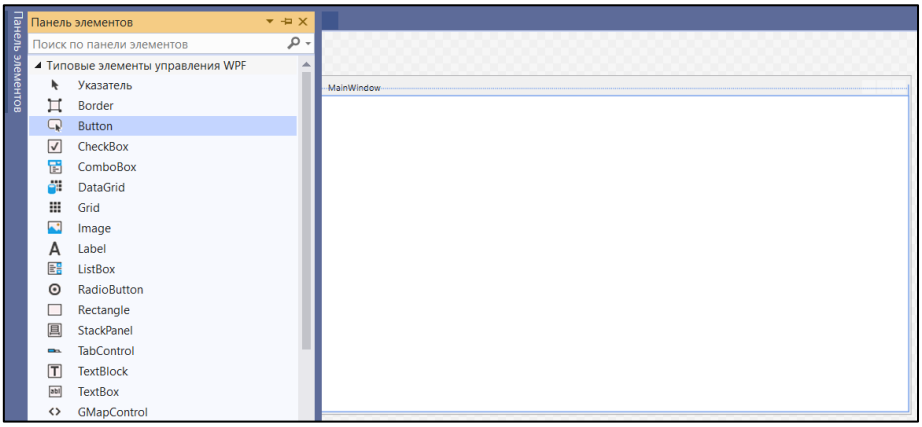
Для выполнения данной лабораторной работы, вам понадобится создать **проект WPF**. В Visual Studio 2019 это можно сделать, перейдя на страницу “Создание проекта” и выбрав “Приложение WPF”:



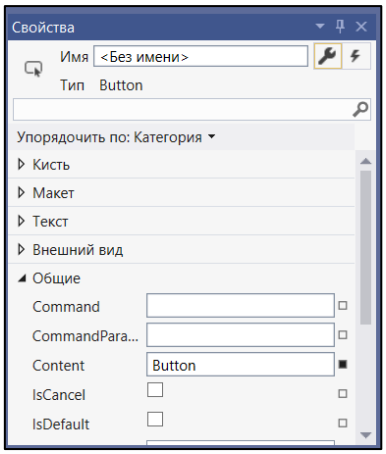
При создании проекта желательно указать осмысленные имя и расположение:

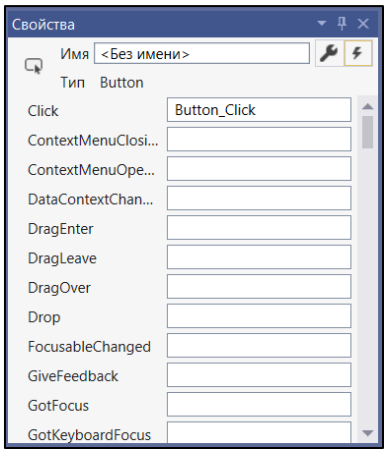


После завершения создания проекта, вы попадёте на экран формы приложения. На форме приложения можно размещать различные элементы интерфейса пользователя. Для организации работы программы, вам понадобится компонент типа “Кнопка”. Добавить его можно выбрав в меню “Панель элементов” и переместив на форму вашего приложения:

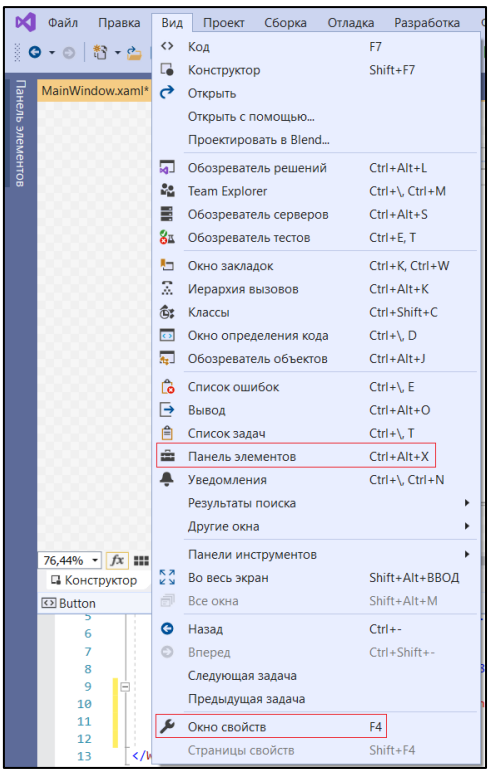


Каждый компонент имеет список свойств, где вы можете настроить отображение этого компонента, и список событий, где вы можете указать функции, вызываемые при определённых событиях:

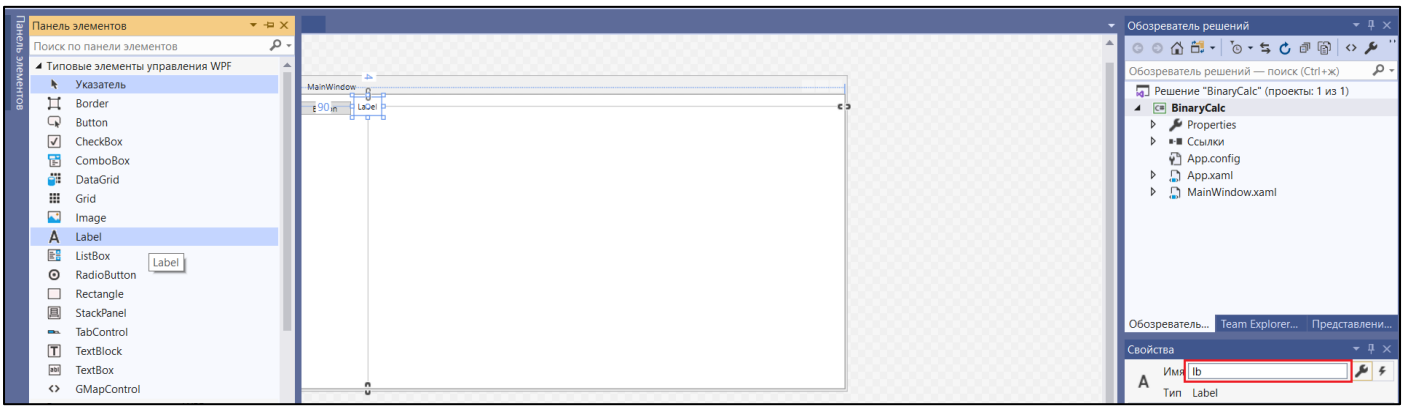




В случае, если панели элементов и окна свойств у вас нет, добавить их можно используя меню “Вид”:



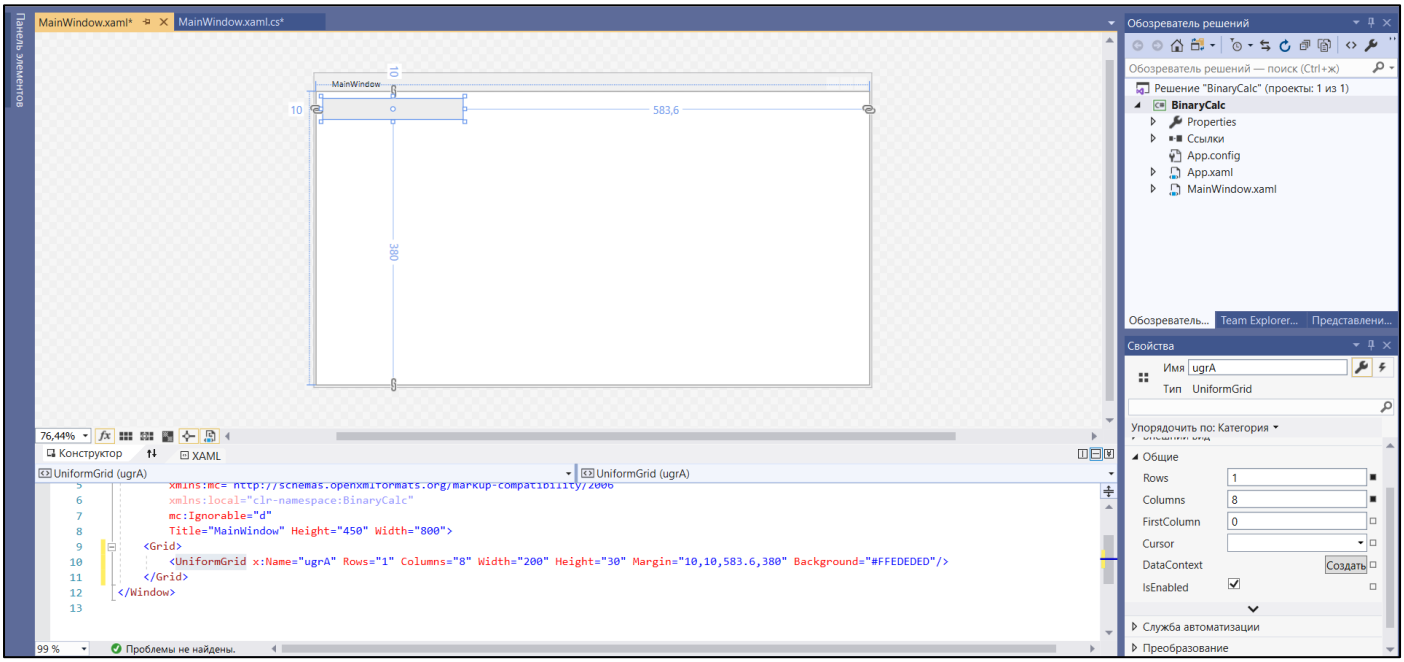
Для **вывода** текстовой информации, можно использовать компонент типа “Этикетка”:



Ниже приведён пример программы, которая выводит в компонент Label с именем lb содержимое переменной A, по нажатию кнопки:

```
16  namespace BinaryCalc
17  {
18      /// <summary>
19      /// Логика взаимодействия для MainWindow.xaml
20      /// </summary>
        Ссылок: 2
21      public partial class MainWindow : Window
22      {
23          int A = 0;
24
            Ссылок: 0
25          public MainWindow()
26          {
27              InitializeComponent();
28          }
29
            Ссылок: 0
30          private void Button_Click(object sender, RoutedEventArgs e)
31          {
32              if (A == 0) A = 1;
33              else A = 0;
34
35              lb.Content = A;
36          }
37      }
38  }
```

Помимо визуальных компонентов, доступных на панели элементов, существуют не визуальные компоненты, которые можно описать самостоятельно в редакторе XAML. Одним из них является **UniformGrid**:

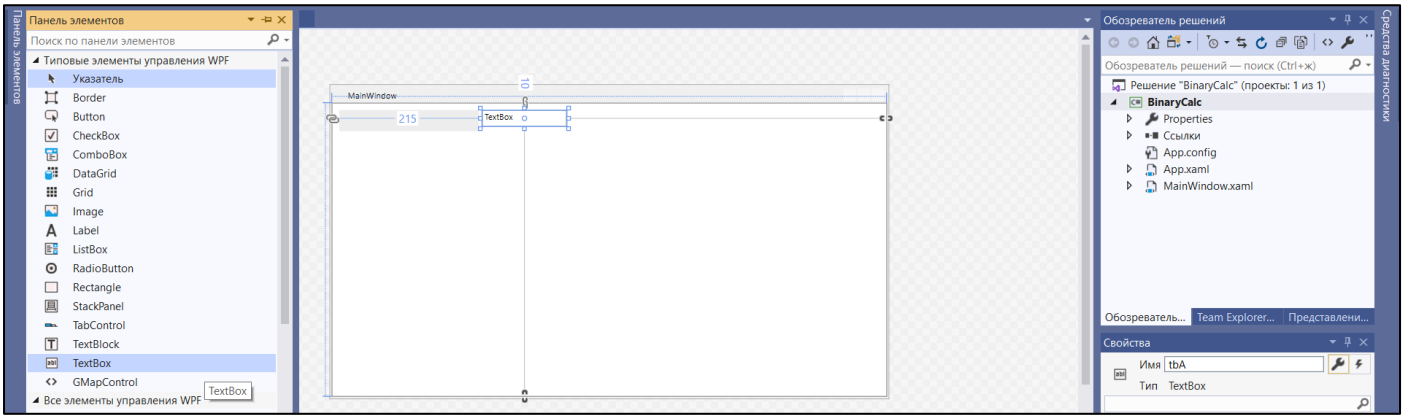


UniformGrid может быть использован для автоматического позиционирования элементов интерфейса в определённой области. Ниже приведён исходный код, который добавляет в компонент типа UniformGrid с именем ugrA, 8 кнопок и ассоциирует их с массивом из 8ми элементов:

```
21      public partial class MainWindow : Window
22      {
23          int[] A = new int[8]; //массив, для хранения числа в двоичном виде
24
            Ссылок: 0
25          public MainWindow()
26          {
27              InitializeComponent();
28
29              for (int i = 0; i < 8; i++)     //добавление 8ми кнопок для переключения бит числа в двоичном виде
30              {
31                  Button btn = new Button();  //создание новой кнопки
32
33                  btn.Tag = i;                //присвоение кнопки номера
34
35                  btn.Width = 20;             //установка размеров кнопки и отображаемого на ней текста
36                  btn.Height = 20;
37                  btn.Content = "0";
38
39                  btn.Click += Button_Click;  //назначение функции, которая будет вызываться при нажатии на кнопку
40
41                  ugrA.Children.Add(btn);     //добавление кнопки
42              }
43          }
44
            ссылка: 1
45          private void Button_Click(object sender, RoutedEventArgs e)
46          {
47              int ind = (int)(((Button)sender).Tag);  //получение номера нажатой кнопки
48
49              if (A[ind] == 0) A[ind] = 1;            //изменение бита, соответствующего этой кнопке в массиве
50              else A[ind] = 0;
51
52              ((Button)sender).Content = A[ind];      //обновление текста, отображаемого на кнопке
53          }
54      }
55  }
```

Описанную выше программу, можно использовать для реализации ввода двоичных чисел размером в один байт.

Для ввода текстовых либо числовых значений может быть использован компонент “Текстовое поле”:



Ниже приведён обработчик нажатия на кнопку, который обращается к текстовому полю с именем tbA и пытается сконвертировать его содержимое в целое число:

```
55          private void Button_Click_1(object sender, RoutedEventArgs e)
56          {
57              int a = 0;
58
59              if (int.TryParse(tbA.Text, out a) == false) //попытка получить из текстового поля числовое значение, и записать его в переменную a
60                  MessageBox.Show("Введено не корректное значение.");
61          }
```

Метод TryParse возвращает значение типа bool: true, если конвертирование прошло успешно и false, если нет. Первым параметром является строка, которую необходимо сконвертировать, вторым параметром является переменная, в которую будет записан результат, если конвертация прошла успешно.

Таблица истинности для **сложения двоичных чисел** выглядит как:

A	B	+
0	0	0
0	1	1
1	0	1
1	1	10

Соответственно, сложение пары целых чисел в двоичном представлении будет выглядеть как:

$$0111 + 0101 = 1100$$

По сути, процесс сложения может быть представлен как последовательное прибавление единиц к тем битам первого числа, где они есть во втором числе:

Шаг	A	B	Операция (A=A+B)	результат
1	0111	0001	A = 0111 + 0001	1000
2	1000	0000	A = 1000 + 0000	1000
3	1000	0100	A = 0100 + 0100	1100
4	1100	0000	A = 0000 + 0000	1100

Операция вычитания, может быть реализована через уже имеющиеся операции сложения и представления отрицательного числа. Например, операция 7-5 в двоичном виде можно записать как:

$$0000\ 0111 - 0000\ 0101 \rightarrow 0000\ 0111 + \ 1111\ 1011 = 0000\ 0010$$

Примечание: если при сложении, возникает ситуация, когда к старшему разряду числа, равному единице, прибавляется единица, то старший разряд числа становится равен нолю, а оставшаяся единица теряется. Пример:

$$1111\ 1111 + 0000\ 0001 = 0000\ 0000$$

Операция умножения, может быть реализована через операции сложения и побитовый сдвиг влево. Пример:

$$0000\ 0111 * 0000\ 0101 = 0010\ 0011$$

Пошагово, процесс может быть записан как:

Шаг	A	B	Сдвиг A	Операция (A = A+((A<<Сдвиг A))*B)	Результат
1	0000 0000	1	0	A = 0000 0000 + (0000 0111 * 1)	0000 0111
2	0000 0111	0	1	A = 0000 0111 + (0000 1110 * 0)	0000 0111
3	0000 0111	1	2	A = 0000 0111 + (0001 1100 * 1)	0010 0011
4	0010 0011	0	3	A = 0010 0011 + (0011 1000 * 0)	0010 0011

Операция деления, может быть реализована через операции вычитания и побитовый сдвиг вправо.

Список литературы:

Учебник. Создание первого приложения WPF в Visual Studio 2019: https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/getting-started/walkthrough-my-first-wpf-desktop-application

Более подробную информацию о компонентах можно получить в следующих источниках:

Элементы управления Windows Forms и эквивалентные элементы управления WPF https://msdn.microsoft.com/ru-ru/library/ms750559.aspx

Класс UniformGrid: https://msdn.microsoft.com/ru-ru/library/system.windows.controls.primitives.uniformgrid(v=vs.110).aspx

Класс Button: https://msdn.microsoft.com/ru-ru/library/system.windows.controls.button(v=vs.110).aspx

Двоичные числа и двоичная арифметика: https://www.intuit.ru/studies/curriculums/16009/courses/541/lecture/12186