Практическая работа № 5

Создание Windows приложения с использованием BackgroundWorker

На данном практическом занятии мы подробно рассмотрим процесс создания простого многопоточного Windows приложения с использованием элемента управления BackgroundWorker.

1. Создадим новый проект Windows приложения и назовем его, "BackgroundWorkerApplication":

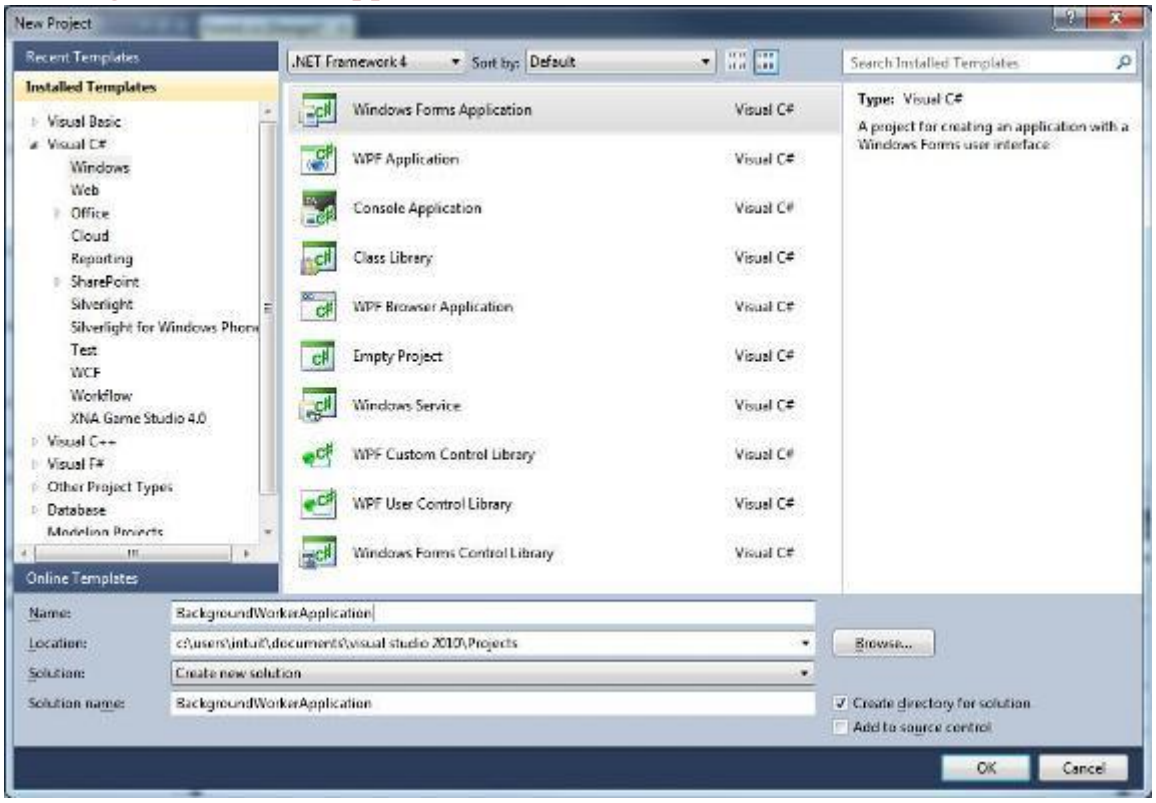

Рис. 1.

2. Разместим три объекта на форме backgroundWorker, statusStrip, timer:

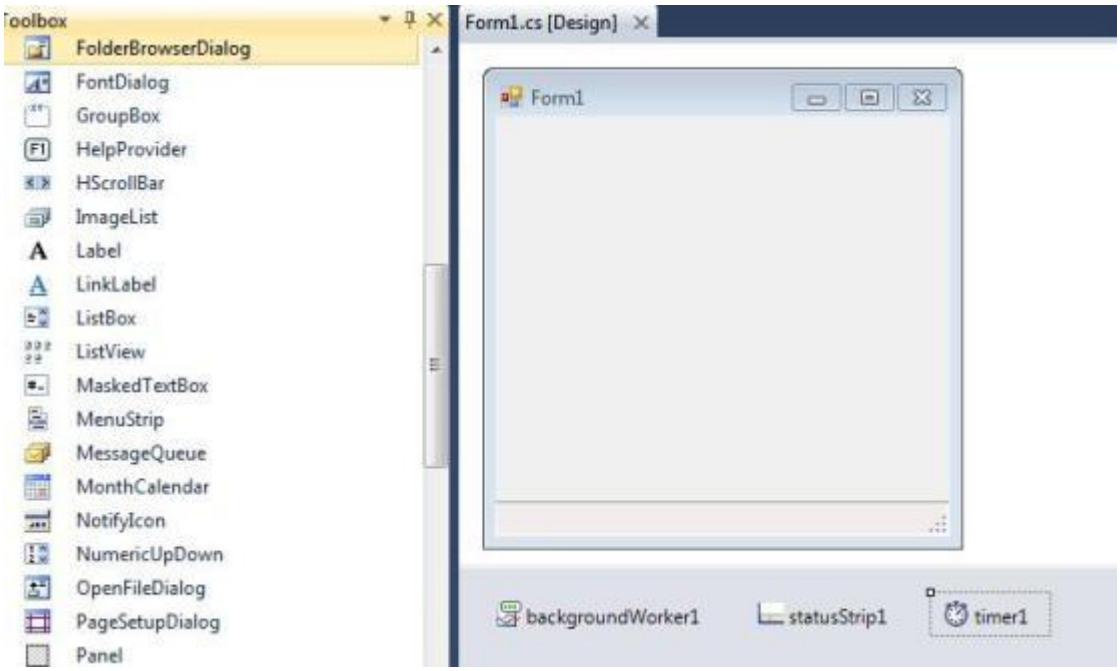

Рис. 2.

- backgroundWorker – объект, в котором будет осуществляться генерация случайных чисел;
- statusStrip - специальный "контейнер" для размещения других элементов;
- timer - объект-таймер, который будет фиксировать время выполнения алгоритма.

3. Теперь, необходимо разместить на форме следующие элементы управления: 2 Button, 2 TextBox, 3 Label, 1 ProgressBar

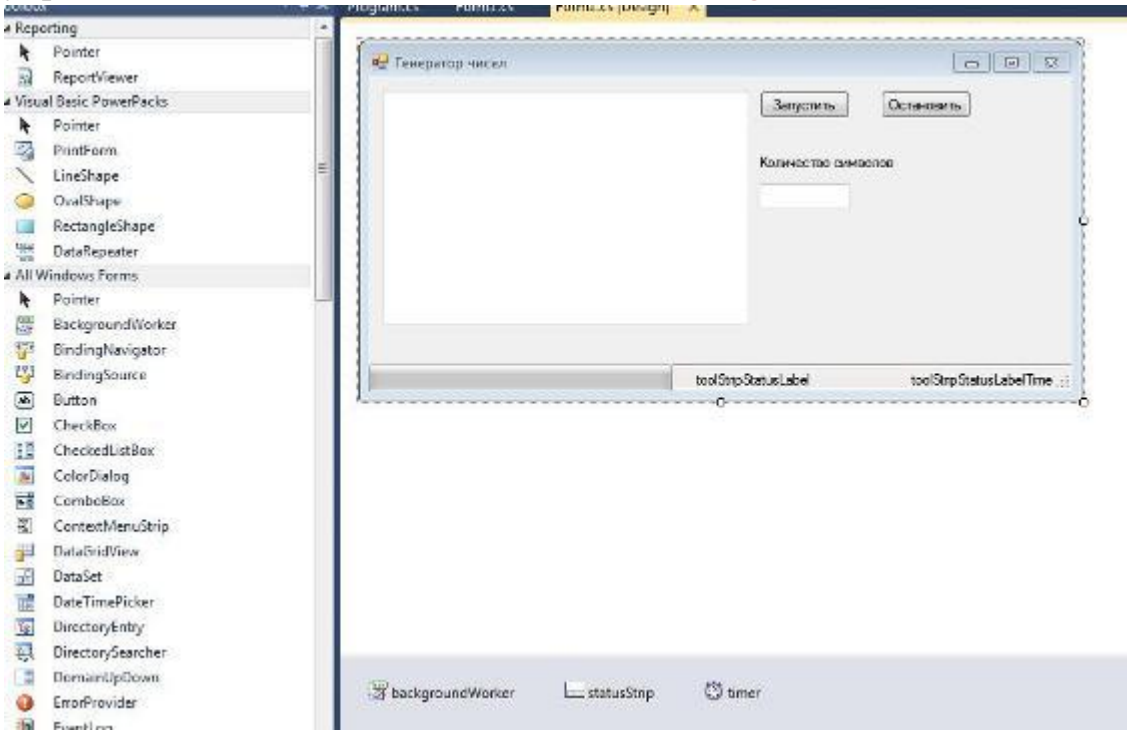

Рис. 3.

- Button - две кнопки, для запуска и остановки процесса генерации случайных чисел;
- TextBox - текстовые поля, для того что бы задать количество генерируемых чисел и для их вывода;
- Label - текстовые метки, для вывода дополнительной информации и надписей;
- ProgressBar - ЭУ визуально отображающий ход выполнения программы.

4. У объекта BackgroundWorker создадим три события:

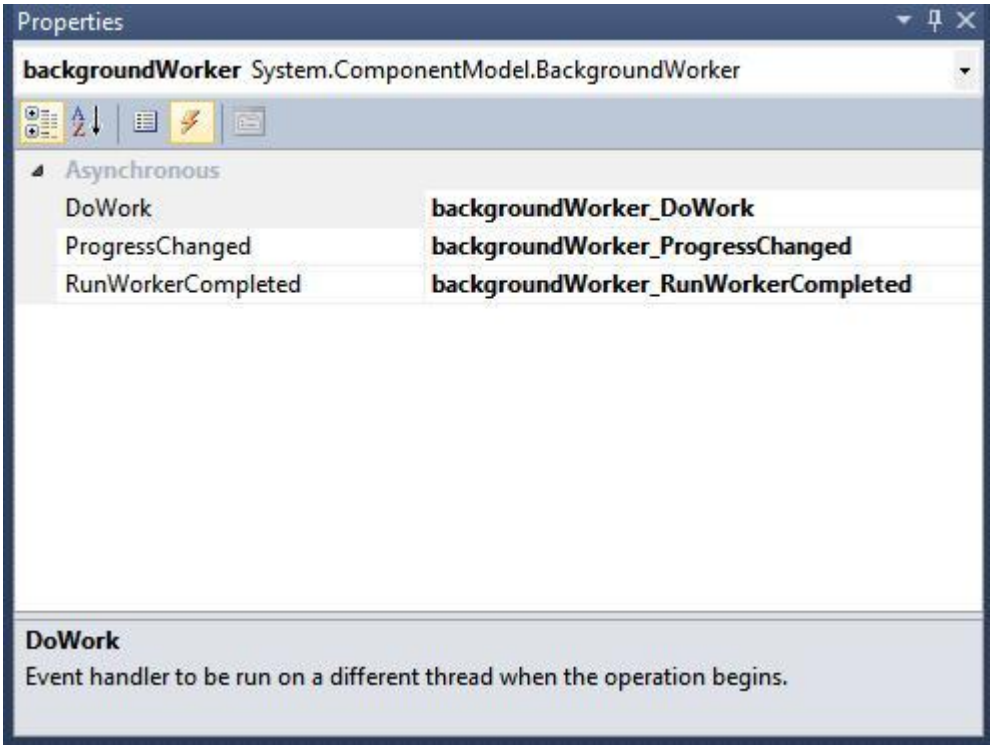

Рис. 4.

- DoWork - событие, которое запускается при вызове метода RunWorkerAsync();
- ProgressChanged - событие, которое запускается при вызове метода ReportProgress();
- RunWorkerCopmpleted - событие, которое возникает, когда выполнение фоновой операции завершено, отменено или вызвало исключение.

5. Также, присвоим свойствам WorkerReportsProgress, WorkerSupportsCancellation, элемента управление BackgroundWorker, значение true:

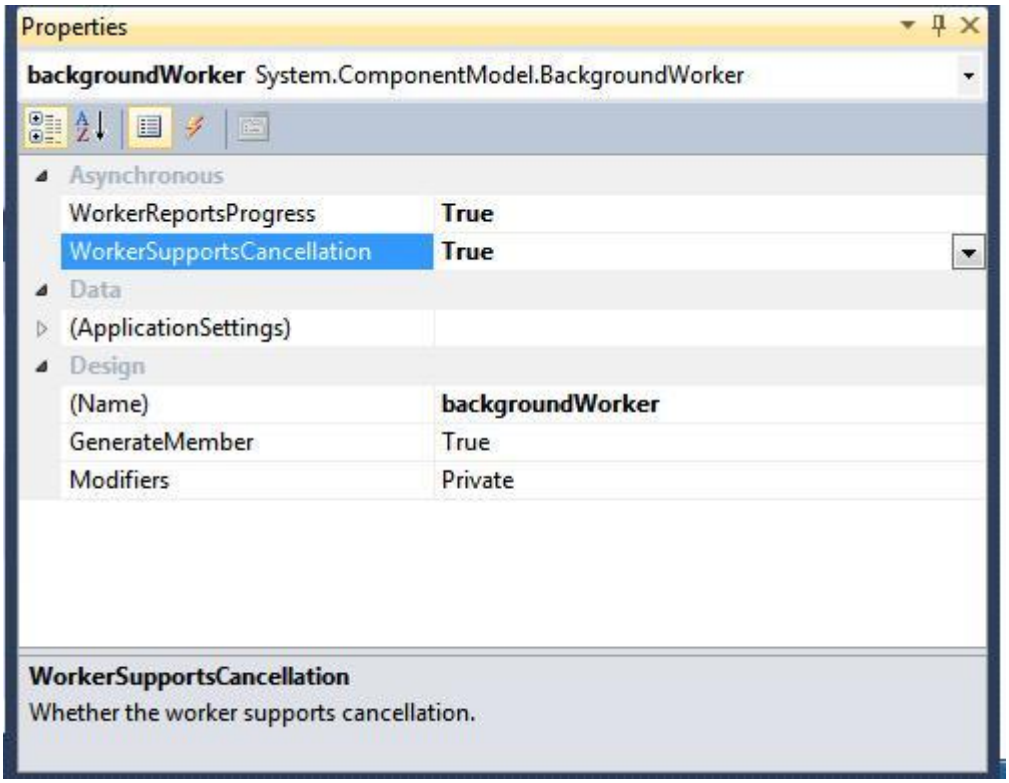

Рис. 5.

- WorkerReportsProgress, - если данное свойств принимает значение true, то пользовательский код может вызвать метод ReportProgress() для инициации события ProgressChanged.
- WorkerSupportsCancellation - если данное свойств принимает значение true, то при использовании объектаBackgroundWorker, можно вызвать метод CancelAsync(), чтобы прервать фоновую операцию.

6. У объекта timer создадим событие Tick, в окне свойств:

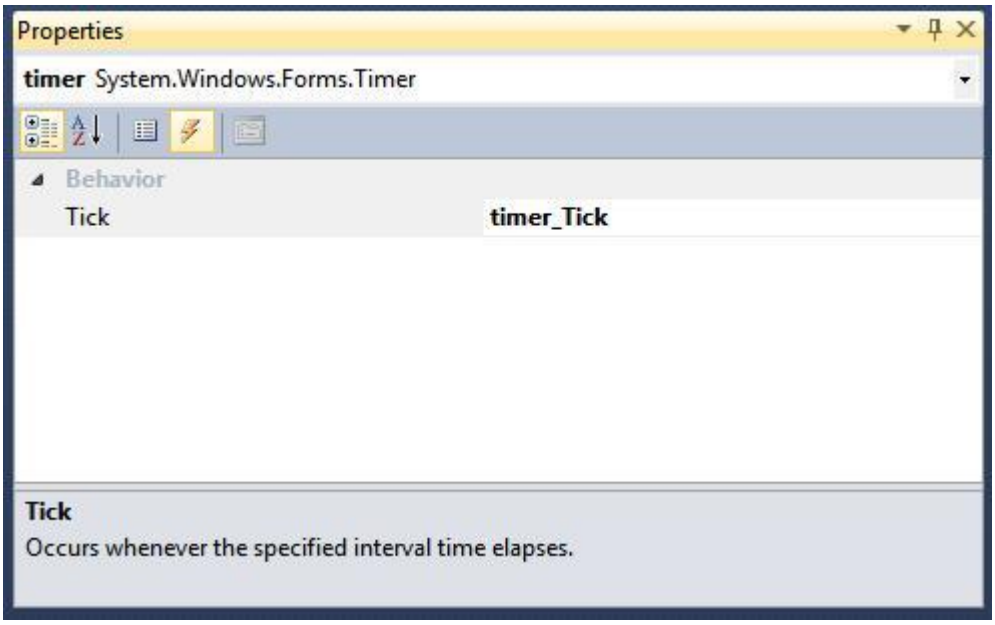

Рис. 6.

7. Теперь, создадим событие Click у кнопки StopButton в окне свойств (Properties):

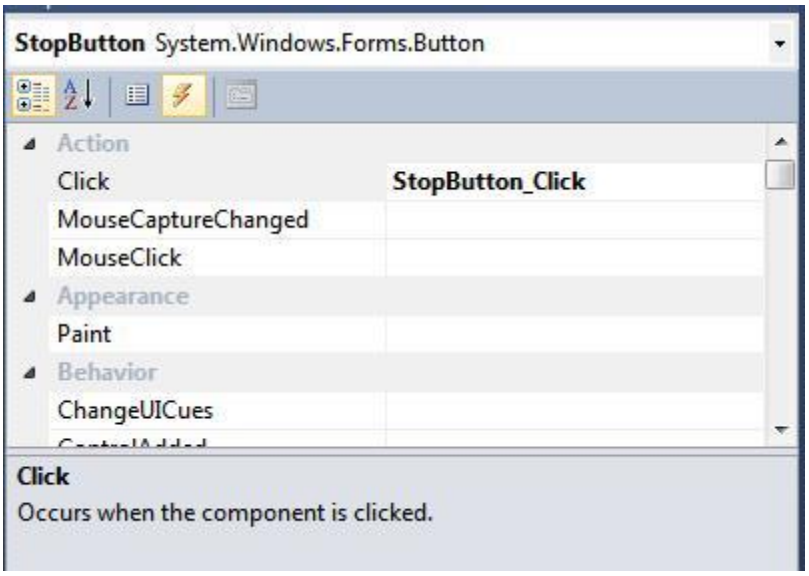

Рис. 7.

8. Создадим событие Click у кнопки StartButton в окне свойств (Properties):

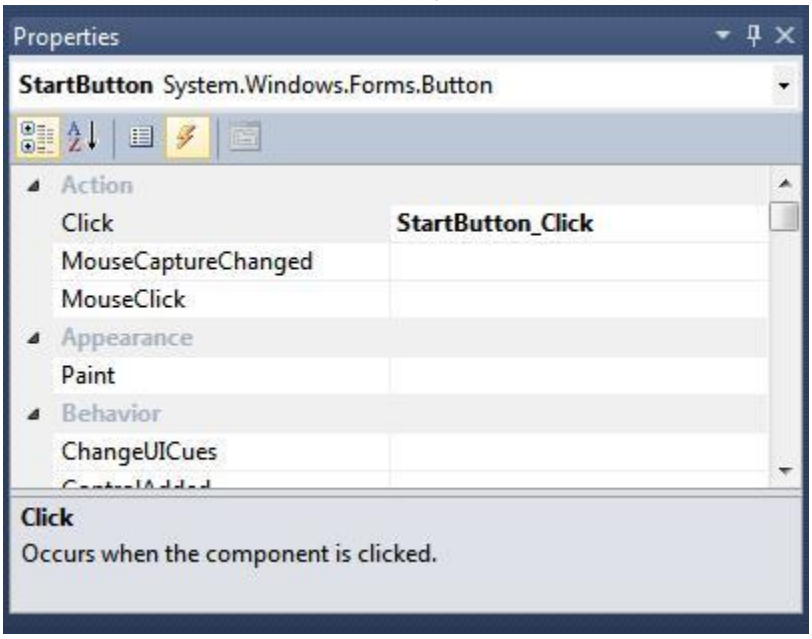

Рис. 8.

9. Создадим метод, который возвращает сгенерированное число в диапазоне от 0 до 100 - randNumber() и присвоим объекту класса DateTime - startDate текущую дату:

```
DateTime startDate = DateTime.Now;
  private  int randNumber()
   {
      Random rnd = new Random();
        return rnd.Next(0,100);
      }
```

10. В событии Tick объекта timer присвоим текстовой метки (toolStripStatusLabelTime) значения интервала времени (TimeSpan ts):

```
private void timer_Tick(object sender, EventArgs e)
      {
```

```
        TimeSpan ts = DateTime.Now.Subtract(startDate);
      string sTime = "  ..." + ts.Minutes.ToString("00") +
        ":" + ts.Seconds.ToString("00") +
        ":" + ts.Milliseconds.ToString("000");
      toolStripStatusLabelTime.Text = sTime;
      }
```

11. В событии StartButton_Click - кнопки StartButton вызовем метод RunWorkerAsync() - объекта BackgroundWorker и запустим таймер с помощью метода Start() объекта timer:

```
private void StartButton_Click(object sender, EventArgs e)
    {
      backgroundWorker.RunWorkerAsync();
        timer.Start();
      }
```

Метод RunWorkerAsync() отправляет запрос для запуска асинхронного выполнения операции (событиеbackgroundWorker_DoWork).

12. В событии StopButton_Click кнопки StopButton, вызовем метод CancelAsync() для остановки выполнения рабочего метода и остановим таймер с помощью метода Stop():

```
private void StopButton_Click(object sender, EventArgs e)
      {
        backgroundWorker.CancelAsync();
         timer.Stop();
      }
```

13. В методе DoWork(), создадим цикл for в котором с помощью метода ReportProgress() будем инициировать событиеbackgroundWorker_ProgressChanged, в случае отмены выполнения фоновой операции присваиваем e.Cancel значениеtrue:

```
private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
      {
        try
        {
          for (int i = 0; i < Convert.ToInt32(CountNumber.Text); i++)
          {
            System.Threading.Thread.Sleep(100);  //замораживаем поток на
100 мс.
            backgroundWorker.ReportProgress(((i * 100) /
(Convert.ToInt32(CountNumber.Text) - 1)));
            if (backgroundWorker.CancellationPending)
            {
              e.Cancel = true;
              return;
```

```
            }    }
        }
         catch (Exception exc) { MessageBox.Show("Ошибка: "+exc); }       }
```

14. В событии backgroundWorker_ProgressChanged, вызываем метод randNumber(), который генерирует и возвращает случайное число. Помещаем сгенерированное число в TextBox с помощью метода AppendText():

```
private void backgroundWorker_ProgressChanged(object sender,
ProgressChangedEventArgs e)
      {
          toolStripProgressBar.Value = e.ProgressPercentage;
           ContainerTextBox.AppendText(randNumber().ToString()+" ");
           toolStripStatusLabel.Text = "Обработка... " +
e.ProgressPercentage.ToString() + "%";
        }
```

Свойство ProgressPercentage - определяет выполненный процент асинхронной задачи, в нашем случае - это значение которое будет рассчитываться в методе ReportProgress() при каждой итерации цикла for (шаг 13).

15. Событие backgroundWorker_RunWorkerCompleted возникает, в случае если фоновая задача завершает выполнение, если фоновая задача отменена, или во время фоновой операции возникла ошибка:

```
private void backgroundWorker_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
      {

         if (e.Cancelled) {
          MessageBox.Show("Задача была отменена");
        }
       else if (e.Error != null)
        {
         MessageBox.Show("В ходе выполнения возникла ошибка: " +
(e.Error as Exception).ToString());
      }
       else {
          timer.Stop();// останавливаем таймер в случае штатного
завершения программы.
           toolStripStatusLabel.Text = "Обработка завершена";
      }
```

16. Запустим программу, вводим количество символов, которые нам надо сгенерировать, и жмем кнопку "Запустить". Запустится процесс генерации случайных чисел:

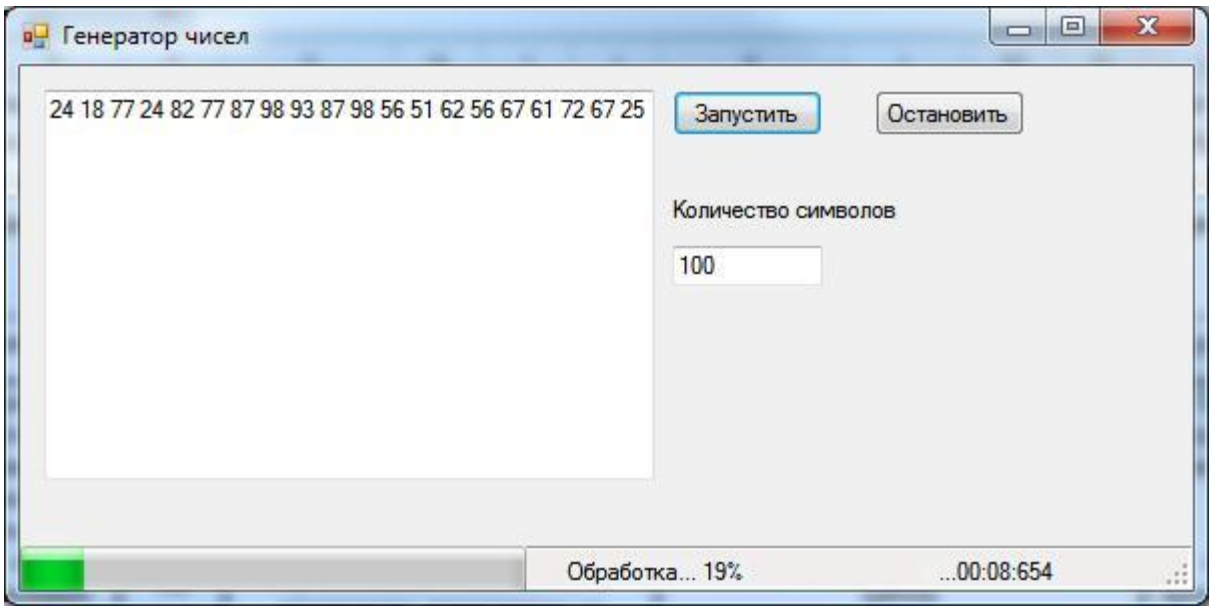

Рис. 9.

17. После завершения процесса генерирования чисел отобразиться следующее:

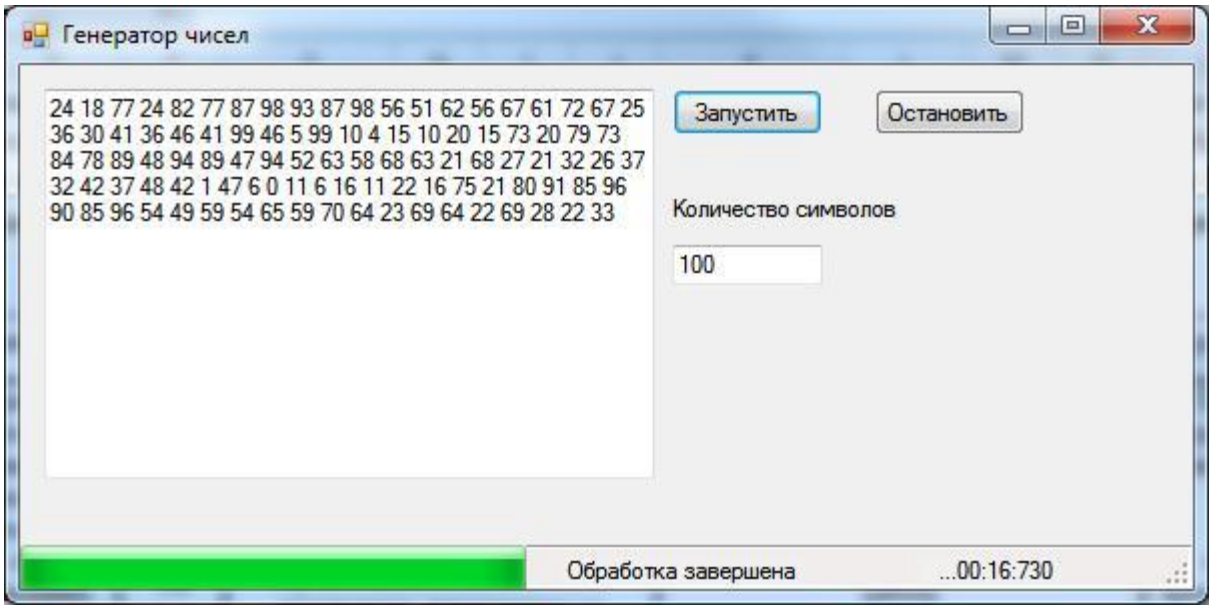

Рис. 10.

Листинг кода программы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace BackgroundWorkerApplication
{
```

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    DateTime startDate = DateTime.Now;
    private  int randNumber()
    {
        Random rnd = new Random();
        return rnd.Next(0,100);
    }
    private void timer_Tick(object sender, EventArgs e)
    {
        TimeSpan ts = DateTime.Now.Subtract(startDate);
        string sTime = "  ..." + ts.Minutes.ToString("00") +
          ":" + ts.Seconds.ToString("00") +
          ":" + ts.Milliseconds.ToString("000");
        toolStripStatusLabelTime.Text = sTime;
    }
    private void backgroundWorker_DoWork(object sender,
DoWorkEventArgs e)
    {
        for (int i = 0; i < Convert.ToInt32(CountNumber.Text); i++)
          {
            System.Threading.Thread.Sleep(100);
            backgroundWorker.ReportProgress(((i * 100) /
(Convert.ToInt32(CountNumber.Text) - 1)));

            if (backgroundWorker.CancellationPending)
            {
                e.Cancel = true;
                return;
            }
          }
    }
    private void backgroundWorker_ProgressChanged(object sender,
ProgressChangedEventArgs e)
    {
      toolStripProgressBar.Value = e.ProgressPercentage;

        ContainerTextBox.AppendText(randNumber().ToString()+" ");
        toolStripStatusLabel.Text = "Обработка... " +
e.ProgressPercentage.ToString() + "%";
    }
```

```
        private void backgroundWorker_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {

            if (e.Cancelled) {
             MessageBox.Show("Задача была отменена");
            }
            else if (e.Error != null)
             {
              MessageBox.Show("В ходе выполнения возникла ошибка: " +
(e.Error as Exception).ToString());
            }
            else {
                timer.Stop();
                toolStripStatusLabel.Text = "Обработка завершена";
             }
        }
        private void StartButton_Click(object sender, EventArgs e)
        {
            backgroundWorker.RunWorkerAsync();
            timer.Start();
        }
        private void StopButton_Click(object sender, EventArgs e)
        {
            backgroundWorker.CancelAsync();
            timer.Stop();
        }
    }
}
```