Практическая работа 17
Работа с потоками. Форматированный ввод/вывод

Цель занятия: Получить практические навыки управления потоками вывода-вывода

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio 2010

Краткие теоретические сведения

C# поддерживает параллельное выполнение кода через многопоточность. Поток – это независимый путь исполнения, способный выполняться одновременно с другими потоками.

Программа на C# запускается как единственный поток, автоматически создаваемый CLR и операционной системой (главный поток), и становится многопоточной при помощи создания дополнительных потоков.

Пример и его вывод:

```
using System;
using System.Threading;

class ThreadTest
{
  static void Main()
  {
    Thread t = new Thread(WriteY);
    t.Start();          // Выполнить WriteY в новом потоке
    while (true)
      Console.Write("x"); // Все время печатать 'x'
  }

  static void WriteY()
  {
    while (true)
      Console.Write("y"); // Все время печатать 'y'
  }
}
```

Вывод:xxxxxxxxxxxxxxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyyyyyyyyyyyyxx
xxxxxxxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyxxxxyyyyyyyyyyyyyyyyyyyxxxxxxxx

xxxxyyyyyyyyyyyyyyyyxxxxxxxxxxxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyyyyyx
xxxxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyyxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
yyyyyyyyyyyyyyyyyyyyyyyyxxxxxxxxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyxxxxxxxx
xxxxxxxxxxxxxxxxxxxxyyyyyyyyyyyyyyyyyyyyyyyyyyyxxxxxxxxxxxxxxxxxxxxx
xyyyyyyyyyyyyyyyyxxxxy...

В главном потоке создается новый поток t, исполняющий метод, который непрерывно печатает символ ‘y’. Одновременно главный поток непрерывно печатает символ ‘x’. CLR назначает каждому потоку свой стек, так что локальные переменные хранятся раздельно.

Как работает многопоточность

Управление многопоточностью осуществляет планировщик потоков, эту функцию CLR обычно делегирует операционной системе. Планировщик потоков гарантирует, что активным потокам выделяется соответствующее время на выполнение, а потоки, ожидающие или блокированные, к примеру, на ожидании эксклюзивной блокировки, или пользовательского ввода – не потребляют времени CPU.

На однопроцессорных компьютерах планировщик потоков использует квантование времени – быстрое переключение между выполнением каждого из активных потоков. Это приводит к непредсказуемому поведению, как в самом первом примере, где каждая последовательность символов ‘X’ и ‘Y’ соответствует кванту времени, выделенному потоку. В Windows XP типичное значение кванта времени – десятки миллисекунд – выбрано как намного большее, чем затраты CPU на переключение контекста между потоками (несколько микросекунд).

На многопроцессорных компьютерах многопоточность реализована как смесь квантования времени и подлинного параллелизма, когда разные потоки выполняют код на разных CPU. Необходимость квантования времени все равно остается, так как операционная система должна обслуживать как свои собственные потоки, так и потоки других приложений.

Пример:

```
using System;
using System.Threading; //Именно это пространство имен поддерживает многопоточность
namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         Thread myThread = new Thread(func); //Создаем новый объект потока (Thread)
         myThread.Start(); //запускаем поток
```

```
    for (int i = 0; i < 10; i++ )
    {
        Console.WriteLine("Поток 1 выводит " + i);
        Thread.Sleep(0);
    }
    Console.Read(); //Приостановим основной поток
}
//Функция запускаемая из другого потока
static void func()
{
    for (int i = 0; i < 10; i++)
    {
        Console.WriteLine("Поток 2 выводит " + i.ToString());
        Thread.Sleep(0);
    }
```

Задания

1 Изучить теоретические сведения и задание к работе.

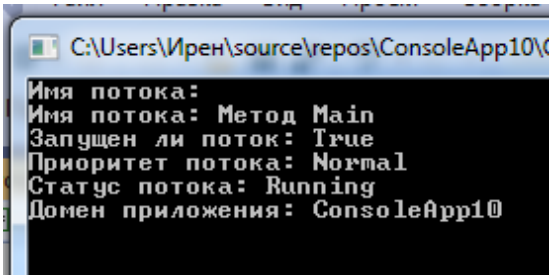
2 Разработать и отладить программный модуль, в котором получить информацию о потоке.

3 Разработать приложение на тему по собственному выбору, демонстрирующее многопоточность.

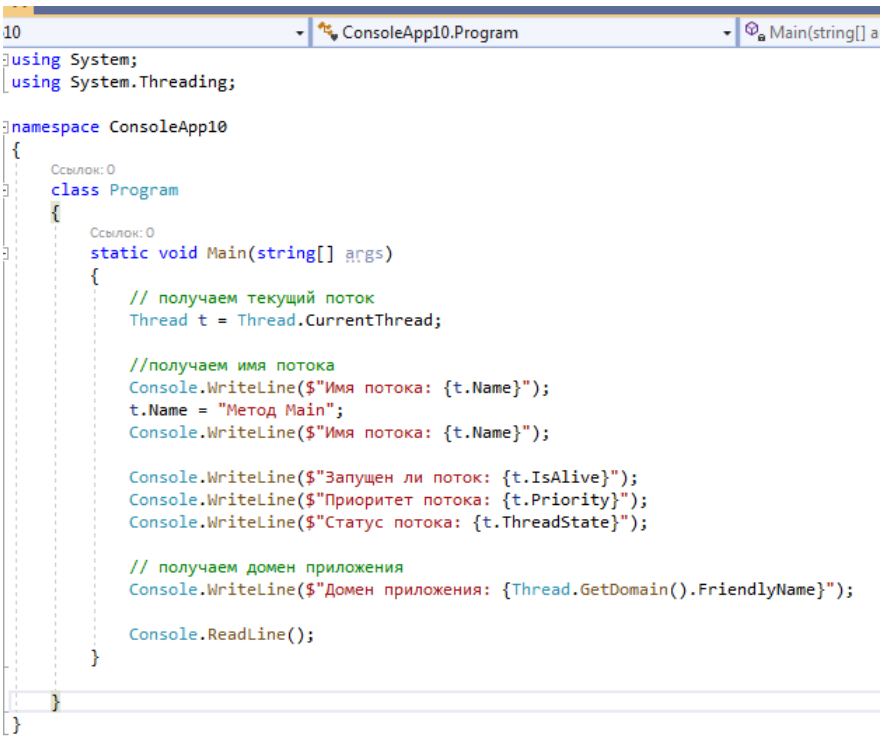
Порядок выполнения работы

Задание 2

Используя свойства и методы для получения информации о потоке, создадим консольное приложение для отображения данных потока, не забудем подключить пространство имен System.Threading. Например, следующую информацию



С помощью кода:



Задание 3 (Примеры выполнения)

Создадим программу, в которой главный и второстепенные потоки параллельно выполняют почти идентичный код они считают до 19 (как повелось, с нуля), перед этим назвав собственный номер.

На что стоит обратить внимание в этой программе. В первую очередь это пространство имен System.Threading. Это пространство имен содержит в себе классы поддерживающие многопоточное программирование. И именно там содержится класс Thread, который мы используем далее в коде. Далее мы создаем объекты потока.

Thread myThread1 = new Thread(func1);

Thread myThread2 = new Thread(func2);

После чего мы запускаем наш поток методом Start() определенном внутри вновь созданного потока.

Метод Thread.Sleep(0) приостанавливает поток, вызвавший его на количество миллисекунд указанных в параметре, если ему передан параметр 0, то поток освобождает оставшуюся часть своего интервала времени для любого потока с таким же приоритетом, готовым к выполнению. Это означает что поток должен приостановиться для того чтобы дать возможность выполнения другому потоку.

```
oleApp8                          ConsoleApp8.Program          func2()
namespace ConsoleApp8
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            Thread myThread1 = new Thread(func1);
            Thread myThread2 = new Thread(func2); //Создаем новый объект потока (Thread)
            myThread1.Start();
            myThread2.Start(); //запускаем поток

            for (int i = 0; i < 20; i++)
            {
                Console.WriteLine("Поток 1 выводит " + i);
                Thread.Sleep(0);
            }
            Console.Read(); //Приостановим основной поток
        }

        //Функции запускаемые из других потоков
        ссылка: 1
        static void func1()
        {
            for (int i = 0; i < 20; i++)
            {
                Console.WriteLine("          Поток 2 выводит " + i.ToString());
                Thread.Sleep(0);
            }
        }
        ссылка: 1
        static void func2()
        {
            for (int i = 0; i < 20; i++)
            {
                Console.WriteLine("                    Поток 3 выводит " + i.ToString());
                Thread.Sleep(0);
            }
Проблемы не найдены.
```

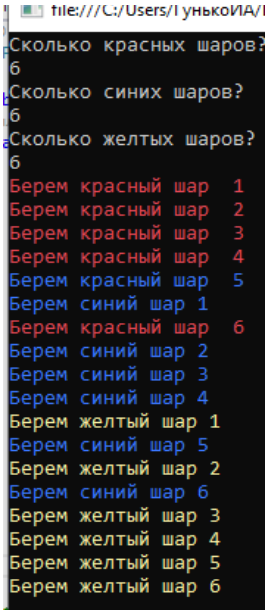
Для ожидания завершения созданного потока в основной программе вызывается метод Thread.Join(). Если его не вызвать программа завершится раньше чем закончится выполнение вторичного потока. По умолчанию в приоритетном потоке запускается функция Main а остальные потоки создаются фоновыми. Именно поэтому мы должны следить за тем, чтобы главный поток не завершился до окончания производных.

В результате ваша программа выведет в консоль примерно следующее

```
C:\Users\Ирен\source\repos\ConsoleApp8\ConsoleApp8\bi...
Поток 1 выводит 0
Поток 1 выводит 1
                    Поток 3 выводит 0
                    Поток 3 выводит 1
                    Поток 3 выводит 2
                    Поток 3 выводит 3
                    Поток 3 выводит 4
                    Поток 3 выводит 5
                    Поток 3 выводит 6
                    Поток 3 выводит 7
                    Поток 3 выводит 8
Поток 1 выводит 2
Поток 1 выводит 3
                    Поток 3 выводит 9
Поток 1 выводит 4
Поток 1 выводит 5
Поток 1 выводит 6
                    Поток 3 выводит 10
Поток 1 выводит 7
Поток 1 выводит 8
                    Поток 3 выводит 11
Поток 1 выводит 9
Поток 1 выводит 10
                    Поток 3 выводит 12
Поток 1 выводит 11
Поток 1 выводит 12
Поток 1 выводит 13
                    Поток 3 выводит 13
                    Поток 3 выводит 14
                    Поток 3 выводит 15
          Поток 2 выводит 0
          Поток 2 выводит 1
                    Поток 3 выводит 16
                    Поток 3 выводит 17
          Поток 2 выводит 2
          Поток 2 выводит 3
          Поток 2 выводит 4
Поток 1 выводит 14
                    Поток 3 выводит 18
                    Поток 3 выводит 19
Поток 1 выводит 15
          Поток 2 выводит 5
          Поток 2 выводит 6
          Поток 2 выводит 7
          Поток 2 выводит 8
          Поток 2 выводит 9
          Поток 2 выводит 10
          Поток 2 выводит 11
Поток 1 выводит 16
Поток 1 выводит 17
Поток 1 выводит 18
Поток 1 выводит 19
          Поток 2 выводит 12
          Поток 2 выводит 13
          Поток 2 выводит 14
          Поток 2 выводит 15
          Поток 2 выводит 16
          Поток 2 выводит 17
          Поток 2 выводит 18
          Поток 2 выводит 19
```

Интересно, что если программу запустить несколько раз, результат вывода каждый раз будет разным.

Или в примере украшения новогодней елки разноцветными шарами может быть получен результат:



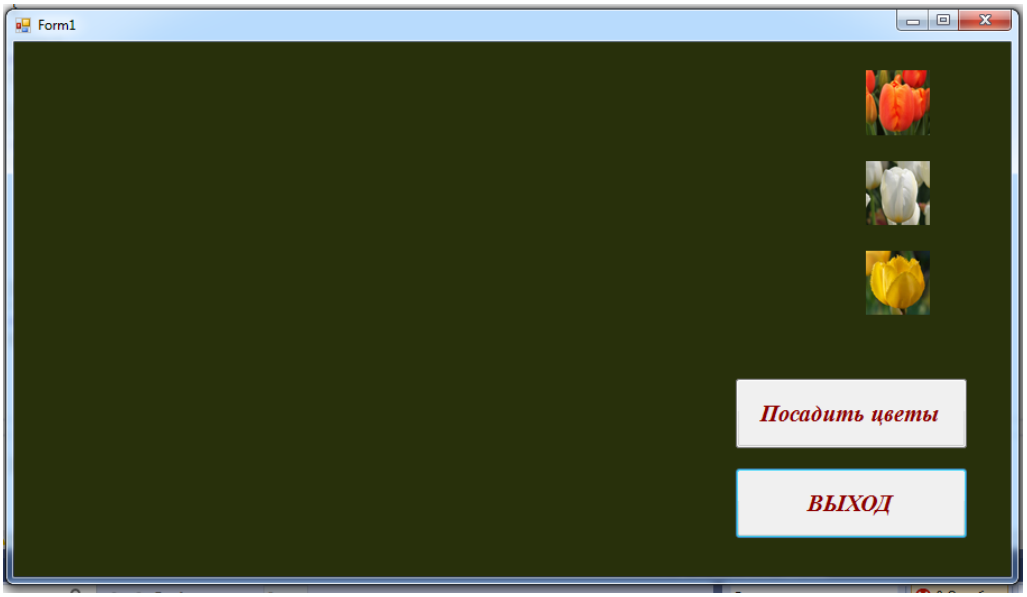
Примеры приложений с использованием Windows Form:

Пример с клумбой

В нем создана форма для размещения PictureBox с тремя видами тюльпанов и двумя кнопками:

«Посадить цветы» для отображения клумбы;

«Выход» для завершения работы приложения.



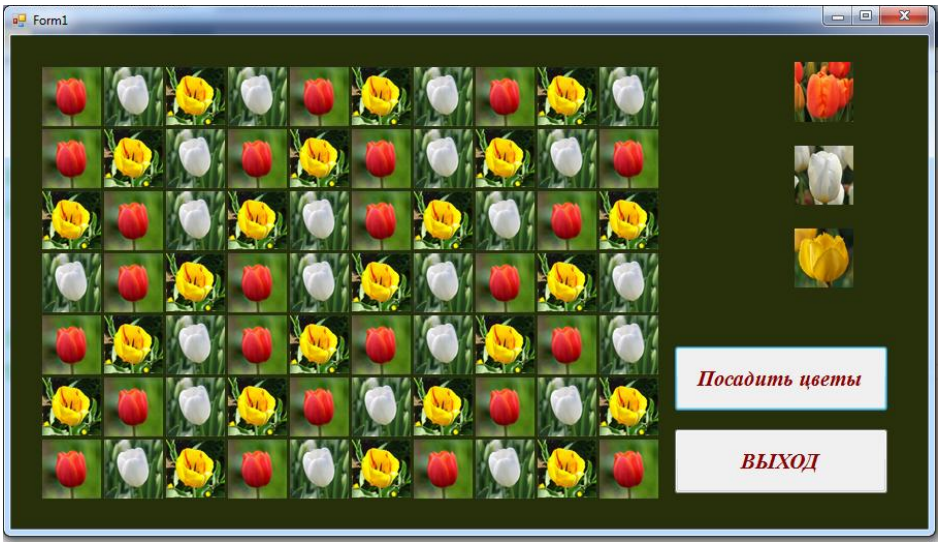
Так как для описания потока используются только статические методы, формирование блоков изображений внутри потока затруднительно, поэтому в потоках создается строковый массив с адресами файлов изображений, а уже потом происходит создание и размещение PictureBox.

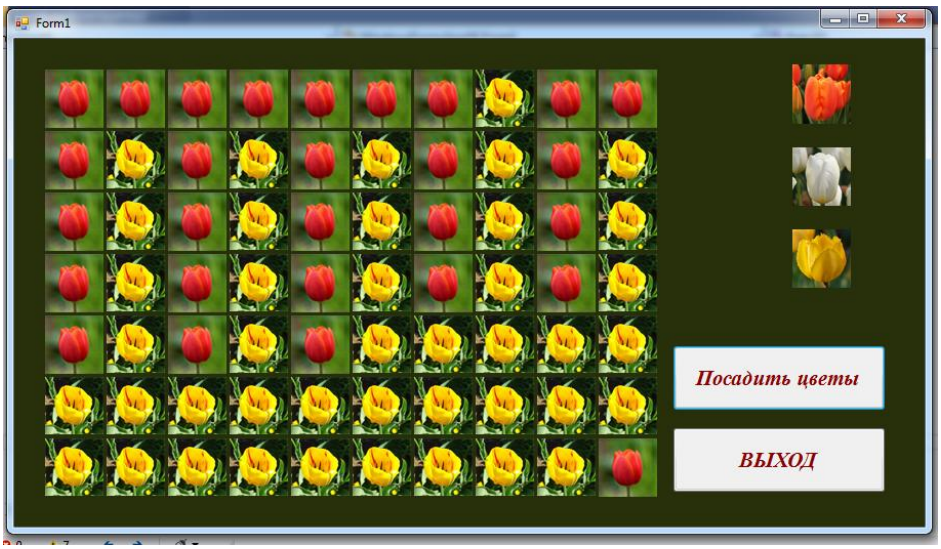
```
Form1.cs [Конструктор]
dowsFormsApp16                WindowsFormsApp16.Form1                func2()

        Thread mThr2 = new Thread(func2);
        mThr1.Start();
        mThr2.Start();
        m = 0;
        while (m < 70)
        {
            ris[m] = @"C:\Users\Ирен\Desktop\Всякое\red.jpg";
            m++;
            Thread.Sleep(0);
        }
        while (i < 7)
        {
            while (j < 10)
            {
                PictureBox a = new PictureBox()
                {
                    Name = "pictureBox" + k.ToString(),
                    Size = new Size(57, 57),
                    Location = new System.Drawing.Point(30 + j * 60, 30 + i * 60),
                };
                a.Image = Image.FromFile(ris[k-1]);
                //MessageBox.Show(k.ToString() + "    " + ris[k-1]);
                a.Size = new System.Drawing.Size(57, 57);
                Controls.Add(a);
                k++;
                j++;
            }
            i++;
            j = 0;
        }
    }

0   7
```

Если несколько раз запустить приложение, на экране будут разные клумбы:

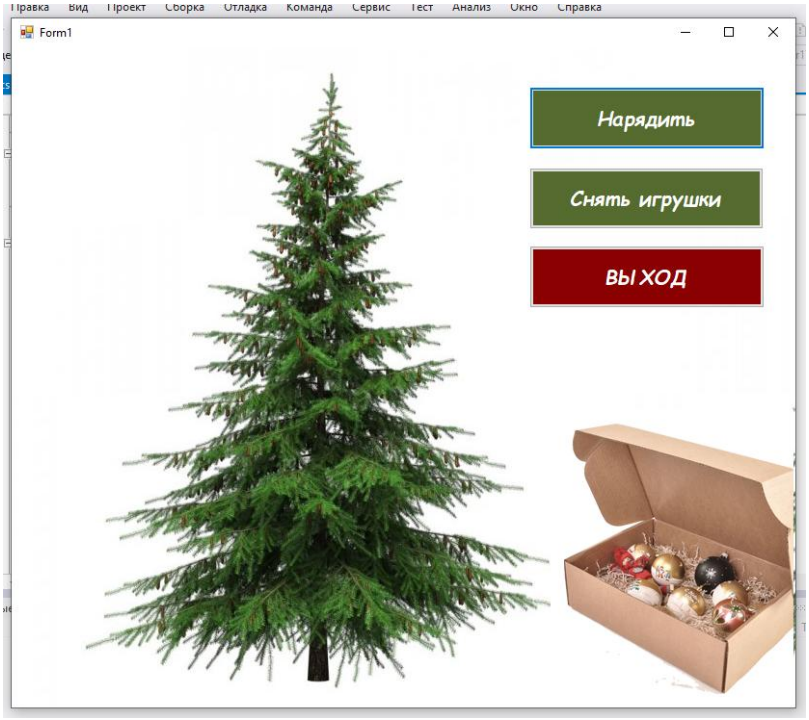




Пример с новогодней елкой

При вызове приложения на экране отображается форма с кнопками:

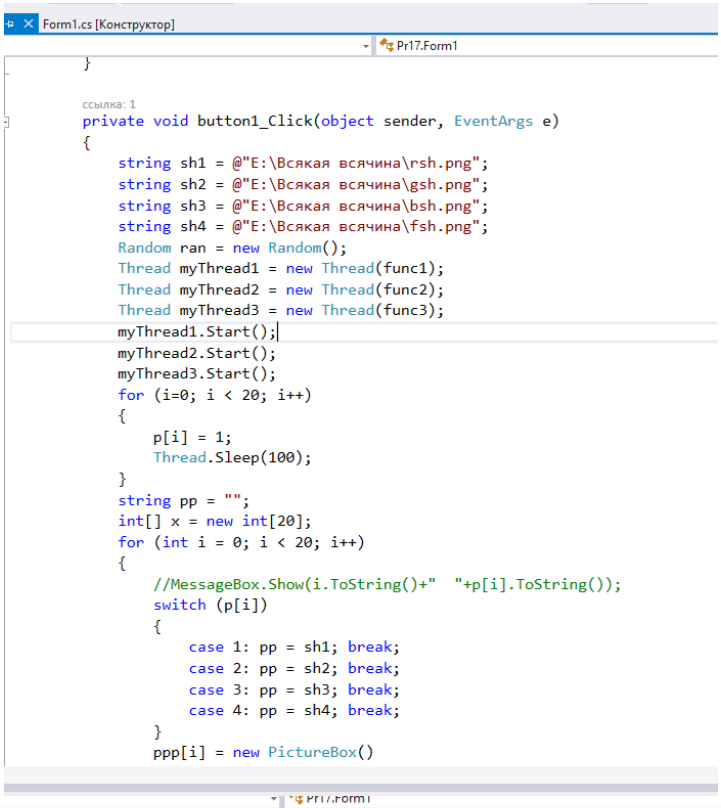
- «Нарядить» - размещает игрушки на форме;
- «Снять игрушки» - очищает форму от изображений игрушек;
- «Выход» - закрывает приложение.



По нажатии кнопки «Нарядить» создаются потоки для выбора одного из 4 видов шариков, а затем динамически создаются PictureBox, которые с помощью генератора случайных чисел размещаются на елке.

```
Form1.cs  Form1.cs [Конструктор]
Pr17                                  Pr17.Form1
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;

namespace Pr17
{
        ссылок: 3
        public partial class Form1 : Form
    {
        static int[] p = new int[20];
        static int i=0;
        public PictureBox[] ppp = new PictureBox[20];
        ссылка: 1
        static void func1()
        {
            for (i = 0; i < 20; i++)
            {
                p[i]=2;
                Thread.Sleep(100);
            }
        }
        ссылка: 1
        static void func2()
        {
            for (i = 0; i < 20; i++)
            {
                p[i] = 3;
                Thread.Sleep(100);
            }
```

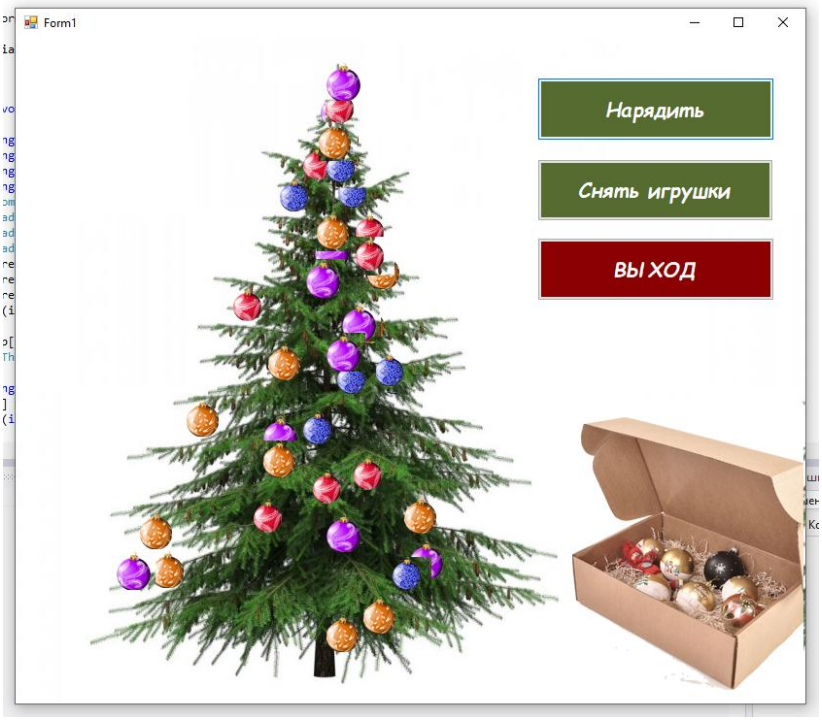



```
                case 4: pp = sh4; break;
            }
            ppp[i] = new PictureBox()
            {
                Name = "PictureBox" + (i + 1).ToString(),
                Size = new Size(40, 40),
                BackColor = Color.Transparent
            };
            x[i] = 320 + i * ran.Next(1, 7) * ran.Next(-2, 2);
            int j = 0;
            while (j < i)
            {
                if (x[j] == x[i]) x[i] = 320 + i * ran.Next(1, 7) * ran.Next(-2, 2);
                j++;
            }
            ppp[i].Location = new System.Drawing.Point(x[i], 25 + i * (30 + ran.Next(-2, 2)));
            ppp[i].Image = Image.FromFile(pp);
            ppp[i].Size = new System.Drawing.Size(40, 40);
            Controls.Add(ppp[i]);
        }
    }

    ссылка: 1
    private void button2_Click(object sender, EventArgs e)
    {
        Close();
    }

    ссылка: 1
    private void button3_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < 20; i++)
        {
           Controls.Remove(ppp[i]);
        }
    }
}
```

Результат:



По нажатии кнопки «Снять игрушки» форма приобретает первоначальный вид.

Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Ответы на контрольные вопросы
7 Вывод

Контрольные вопросы

1 Что такое многопоточность?
2 Что такое поток?
3 Какие существуют типы потоков?
4 Особенности различных типов потоков.
5 В каком пространстве имен сосредоточен функционал для использования потоков?
6 Какие классы определены в этом пространстве имен?
7 Как осуществляется выбор процессором следующего потока для выполнения
8 Как создать поток?

9 Как запускать потоки в C#?
10 Как приостановить потоки в C#?

Используемая литература

1. Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.
2. Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.
3. Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.
4. Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.
5. http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.