Програмування друку на мові C# з використанням Windows Forms

Розглянемо виведення на друк посередині сторінки текст "Hello Printer!". Для цього використовується елемент управління *PrintDocument*, який показано на рис. 1. і має ім'я *printDocument1*. Текст програми наводиться на лістингу 1.

Після натиснення кнопки викликається метод *PrintDocument*1.*Print()*. У результаті виконання методу викликається обробник події *PrintPage* (названий тут *PrintDocument*1*_PrintPage*).

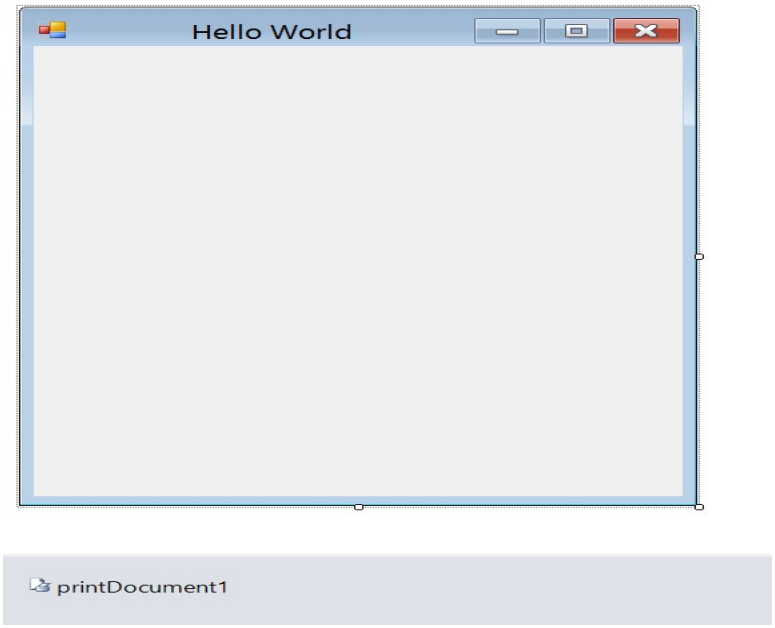


Рис. 1. До організації друку

Значенням параметру *object* обробника *PrintDocument*1*_PrintPage* є раніше створений об'єкт *PrintDocument.* Параметр *PrintPageEventArgs* визначає властивості, що надають відомості про принтер, найважливіша з яких – *Graphics* – аналогічно однойменній властивості параметру *PaintEventArgs* за виключенням того, що *PrintPageEventArgs* надає об'єкт *Graphics* для сторінки принтеру, а не для клієнтської області форми. Об'єкт *Graphics* викликає методи, що виводять графіку на сторінку принтера. Якщо треба надрукувати декілька сторінок, слід встановити властивість *HasMorePages* об'єкту *PrintPageEventArgs* як *true.* Але оскільки друкується лише одна сторінка, залишимо значення властивості за умовчанням – *false.* Метод *DoPage,* викликається як обробником *OnPaint,* так і *PrintDocumentOnPrintPage.*

Як відомо, властивість *ClientSize* форми надає розміри її клієнтської області, виражені у пікселах. Створення графіки в межах клієнтської області вікна відрізняється від виведення графіки на принтер. Сторінка принтера визначається трьома різними областями.

По-перше, повним розміром сторінки. Цю інформацію надає властивість *PageBounds* класу *PrintPageEventArgs.* Його значенням є структура *Rectangle,* її властивості *X* і *Y* дорівнюють 0, а властивість *Width* і *Height* надають розмі-

ри паперу, які за умовчанням, вимірюються в сотих долях дюйма. Наприклад, для листа розміром 8,5 и 11 дюймів значення властивостей *Width* і *Height* об'єкту *PageBounds* дорівнюватиме 850 і 1100. Якщо у принтері за умовчанням задана альбомна, а не книжкова орієнтація сторінки, то вони дорівнюватимуть відповідно 1100 і 850.

По-друге, є *область друку* сторінки. Її розмір зазвичай дуже близький до повного розміру сторінки, окрім полів, недоступних друкуючій головці принтера. Ширина верхнього, нижнього, лівого та правого полів може бути різною. Значенням властивості *VisibleClipBounds* класу *Graphics* є структура *RectangleF,* що надає розмір області друку сторінки. Властивості *X і Y* цієї структури встановлюються в 0, а її властивості *Width* і *Height* дорівнюють горизонтальному та вертикальному розмірам області друку сторінки.

Розмір третьої області розраховується з урахуванням 1-дюймового відступу по периметру сторінки і являє собою границі, в межах яких воліє друкувати користувач. Ця інформація повертається у вигляді структури *Rectangle* властивістю *MarginBounds* об'єкту *PrintPageEventArgs.*

Лістинг 1. (exam1)

```
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;
using System.Drawing.Printing;

namespace exam12{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.ResizeRedraw = true;
            printDocument1.DocumentName = "Doc1";
             Text = "Hello Printer!";
        }

        private void Form1_Click(object sender, EventArgs e)
        {
            printDocument1.Print(); // Печать
        }

        private void printDocument1_PrintPage(object sender,
            PrintPageEventArgs e)
        {
            DoPage(e.Graphics, Color.Black,
            (int)e.Graphics.VisibleClipBounds.Width,
```

```
                    (int)e.Graphics.VisibleClipBounds.Height);
        }
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
          DoPage(e.Graphics, Color.FromArgb(255, 0, 0),
          ClientSize.Width/6, ClientSize.Height/6);
        }
        private void DoPage(Graphics grfx, Color col,int w,int h)
        {
          Pen pen = new Pen(col);
          Rectangle rect = new Rectangle(1,1,w-2, h-2);
          StringFormat ss = new StringFormat();
          ss.Alignment = StringAlignment.Center;
          ss.LineAlignment = StringAlignment.Center;
          grfx.DrawRectangle(pen, rect);
          grfx.DrawString(Text, Font, Brushes.Black, rect, ss);
        }
    }
}
```

Розглянемо виведення на друк даних текстового файлу. Для цього використовується елемент управління *PrintDocument*, який показано на рис. 2 і має ім'я *printDocument*1. Текст програми зчитування даних з текстового файлу і виведення цієї інформації на принтер наводиться у лістингу 2.

Після натиснення кнопки на принтер виводяться дані текстового файлу 1.txt.

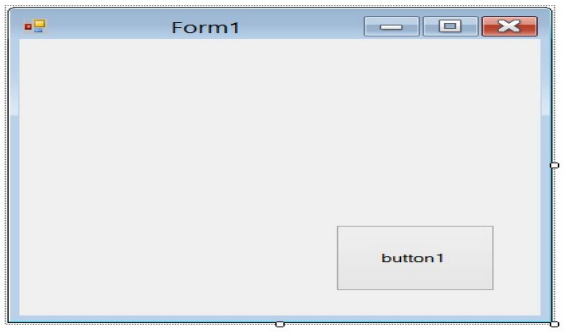


Рис. 2. До виведення на друк текстового файлу

Лістинг 2. (exam2)

```
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;
using System.IO;
using System.Drawing.Printing;

namespace exam13
{
    public partial class Form1 : Form
    {
        Font printFont;
        StreamReader streamToPrint;
        int kod;

        public Form1()
        {
            InitializeComponent();
            printFont = new Font("Arial Cyr", 10);
            kod = 1;
            try
            {
                streamToPrint = new StreamReader("1.txt",
                Encoding.Default);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                kod = 0;
            }
        }
  private void printDocument1_PrintPage(object sender,
     System.Drawing.Printing.PrintPageEventArgs ev)
   {
    float linesPerPage = 0;
    float yPos = 0;
    int count = 0;
    float leftMargin = 100 + ev.Graphics.VisibleClipBounds.Left;
    float topMargin =  ev.Graphics.VisibleClipBounds.Top;
    string line = null;
    linesPerPage = ev.Graphics.VisibleClipBounds.Height /
            printFont.GetHeight(ev.Graphics);
         while (count < linesPerPage &&
               ((line = streamToPrint.ReadLine()) != null))
         {
          yPos = topMargin + (count *
          printFont.GetHeight(ev.Graphics));
          ev.Graphics.DrawString(line, printFont, Brushes.Black,
             leftMargin, yPos);
             count++;
         }
            ev.HasMorePages = line != null;
      }
        void button1_Click(object sender, EventArgs e)
        {
```

```
            if (kod == 0) return;
                  try
                {
                      streamToPrint.BaseStream.Position = 0;
                      printDocument1.Print();
                }
                  catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
        }
        private void Form1_FormClosed(object sender,
                                      FormClosedEventArgs e)
        {
            if(kod==1)streamToPrint.Close();
        }
    }
}
```

Будь-який текстовий редактор повинен мати можливість друку на принтері. Наступна програма має скромні можливості: відкрити у стандартному діалозі текстовий файл, переглянути його у текстовому полі без можливості зміни тексту (*ReadOnly*) і при можливості вивести цей текст на принтер. Використані наступні елементи управління: текстове поле *TextBox*, меню *MenuStrip* з пунктами меню: «Открыть», «Печать» и «Выход», а також елементи управління *OpenFileDialog*, *PrintDialog* (див. рис. 3). Текст програми надано у лістингі 3.

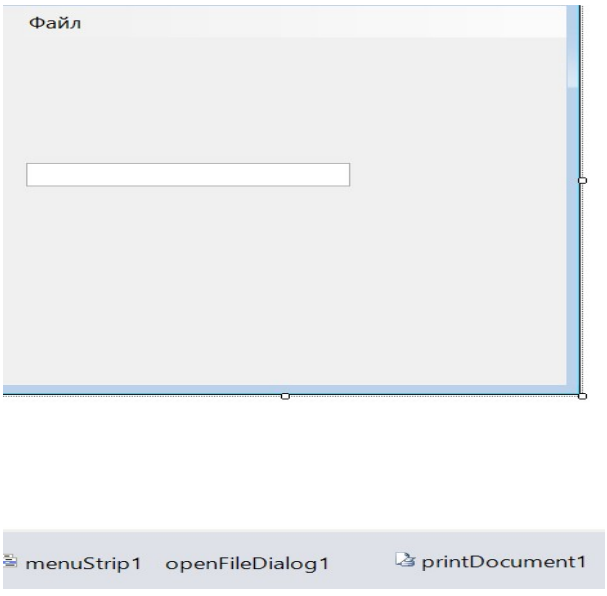


Рис. 3. До друку текстового документа з використанням меню и елементу *openFileDialog*

Лістинг 3. (exam3)

```
using System;
using System.Drawing;
using System.Windows.Forms;
// Інші директиви using видалені, оскільки вони
```

```
// не використовуються у даній програмі

namespace exam14
{
    public partial class Form1 : Form
    {
        System.IO.StreamReader Читатель;
        public Form1()
        {
          InitializeComponent();
          base.Text = "Открытие текстового файла и его печать";
          textBox1.Multiline = true;
          textBox1.Clear();
          textBox1.Size = new System.Drawing.Size(268, 112);
          textBox1.ScrollBars = ScrollBars.Vertical;
          textBox1.ReadOnly = true;
         // До тих пір, поки файл не прочитаний у текстове поле,
         // не повинен бути видно пункт меню "Печать..."
            печатьToolStripMenuItem.Visible = false;
            openFileDialog1.FileName = null;
        }
      private void открытьToolStripMenuItem_Click(object sender,
                                                  EventArgs e)
        { // Клацання на пункті меню "Открыть":
           openFileDialog1.Filter =
            "Текстовые файлы (*.txt)|*.txt|All files (*.*)|*.*";
            openFileDialog1.ShowDialog();
            if (openFileDialog1.FileName == null) return;
          try
           { //Створення потоку StreamReader для читання з файлу
             Читатель = new
                System.IO.StreamReader(openFileDialog1.FileName,
                System.Text.Encoding.GetEncoding(1251));
                // - тут замовлення кодової сторінки Win1251 для
                // кирилиці
                textBox1.Text = Читатель.ReadToEnd();
                Читатель.Close();
                печатьToolStripMenuItem.Visible = true;
            }
           catch (System.IO.FileNotFoundException Exc)
           {
             MessageBox.Show(Exc.Message + "\nНет такого файла",
                             "Ошибка", MessageBoxButtons.OK,
                              MessageBoxIcon.Exclamation);
            }
            catch (Exception Exc)
            {
                // Звіт про інші помилки:
                MessageBox.Show(Exc.Message, "Ошибка",
                    MessageBoxButtons.OK,
                     MessageBoxIcon.Exclamation);
            }
        }
```

```
        private void печатьToolStripMenuItem_Click(object sender,
                                                   EventArgs e)
        { // Пункт меню "Печать"
            try
            {
                Читатель = new
                System.IO.StreamReader(openFileDialog1.FileName,
                System.Text.Encoding.GetEncoding(1251));
                // - тут замовлення кодової сторінки Win1251 для
                // кирилиці
                try { printDocument1.Print(); }
                finally { Читатель.Close(); }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message);
 }
}

        private void printDocument1_PrintPage(object sender,
                System.Drawing, Printing.PrintPageEventArgs e)
        {
            Single linesPerPage = 0;
            Single yPos = 0; int count = 0;
            Single leftMargin = e.MarginBounds.Left;
            Single topMargin = e.MarginBounds.Top;
            topMargin = 0;
            String line = null;
            Font printFont = new Font("Times New Roman", 12.0F);
            // Обчислюємо кількість рядків на одній сторінці
            linesPerPage = e.MarginBounds.Height /
            printFont.GetHeight(e.Graphics);
            // Друкуємо кожний рядок файлу
            while (count < linesPerPage)
            {
                line = Читатель.ReadLine();
                if (line == null) break; // вихід з циклу
                yPos = topMargin + count *
                       printFont.GetHeight(e.Graphics);
                // Друк рядку
                e.Graphics.DrawString(line, printFont,
           Brushes.Black, leftMargin, yPos, new StringFormat());
                count += 1;
            }
            // Друк наступної сторінки, якщо є ще рядки файлу
            if (line != null) e.HasMorePages = true;
            else e.HasMorePages = false;
        }
        private void выходToolStripMenuItem_Click(object sender,
                                                   EventArgs e)
        { // Вихід з програми
            base.Close();
        }
    }
}
```

Тут при обробці події форми *Form1_Load* забороняється користувачеві редагувати текстове поле: *ReadOnly = true*. Також призначаємо властивості друк *ToolStripMenuItem.Visible = false* (пункт меню «Печать»), тобто на початку програми пункт меню «Печать» користувачеві не видно. При натисканні пункту меню «Открыть» викликається стандартний діалог *OpenDialog* і виконується читання текстового файлу з використанням потоку *StreamReader*. Після зчитування файлу стає видимим пункт «Печать». Зверніть увагу на обробку події *PrintDocument*1.*PrintPage*. Тут у змінній *count* відбувається підрахунок рядків на сторінці *linesPerPage*. При перевищенні кількості рядків на сторінці відбувається вихід з циклу, оскільки сторінка роздрукована. Якщо є ще сторінки (l*ine*!= *null*), то змінній *e.HasMorePages* призначаємо *true*, що ініціює знову подію *PrintPage* і підпрограма *PrintDocument*1.*Print* починає свою роботу знову. І так, поки не роздрукуються всі сторінки (*e.HasMorePages = false*). На рис.4. показано результат роботи програми.

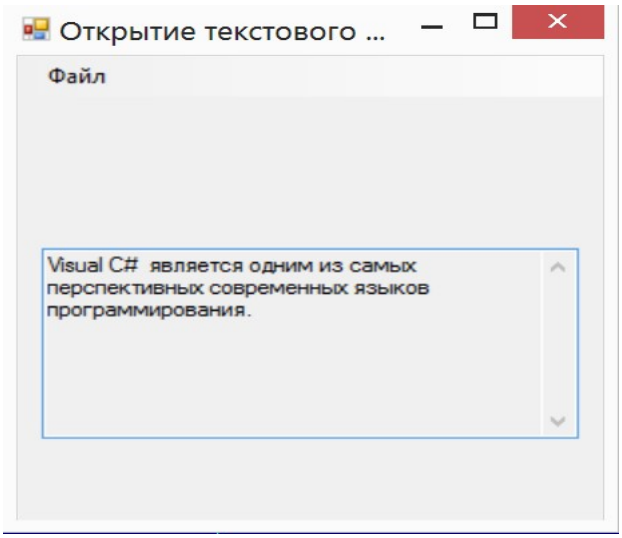


Рис. 4. Результат робото програми друку текстового файлу

У лістингу 4 наводиться програма, яка дозволяє з високою точністю вивести на принтер 10 сантиметрову лінійку. Така ж лінійка виводиться на екран. Однак якщо виміряти на екрані довжину лінійки, вона більш ніж 10 см. Чим менше роздільна здатність екрану, тим довше буде лінійка на екрані. При самій максимальній роздільній здатності екрану розміри лінійки наближаються до 10 см. Було б непогано знати роздільну здатність дисплея, виражене у загальних одиницях виміру, наприклад в точках на дюйм (*dpi*), Але якщо для принтерів цей параметр визначений цілком точно (зазвичай його можна знайти навіть на упаковці принтеру), то для моніторів він залишається досить розпливчастим. Реальна роздільна здатність дисплея в *dpi* визначається двома параметрами: фізичним розміром екрану (зазвичай його вимірюють в дюймах по діагоналі) і роздільною здатністю екрану у пікселях. Оскільки

Windows не знає розміру монітору, вона не може повідомити реальну роздільну здатність екрану. Навіть маючи такі відомості, ОС опиниться у скруті. Тому роздільна здатність екрану оцінюється у кількості пікселів на логічний дюйм. Цими показниками для екрану є властивості класу *Graphics DpiX* і *DpiY* (відповідно по горизонталі і вертикалі).

У лістингу 4 для екрану наводиться конвертування з міліметрів у пікселі:

```
PointF Conv(Graphics grfx, PointF pointf)
{
   pointf.X*= grfx.DpiX  / 25.4f;
   pointf.Y *= grfx.DpiY / 25.4f;
   return pointf;
}
```

Якщо за умовчанням виведення на екран виконується у пікселях, то на принтер у сотих долях реального дюйма. Для принтера конвертування з міліметрів у соті долі дюйма виконується в наступній функції:

```
PointF Conv_pr(Graphics grfx, PointF pointf)
{
   pointf.X *= 100 / 25.4f;
   pointf.Y *= 100 / 25.4f;
   return pointf;
}
```

Результат рішення наводиться на рис. 5.

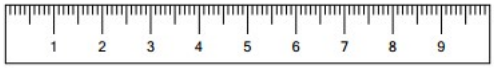
Лістинг 4. (exam4)

```
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 exam15
 {
   public partial class Form1 : Form
   {
       public Form1()
       {
         InitializeComponent();
         Text = "Печать";
       }

      private void Form1_Paint(object sender, PaintEventArgs e)
```

```
        {
            Graphics grfx = e.Graphics;
            Pen pen = new Pen(ForeColor);
            Brush brush = new SolidBrush(ForeColor);
            const int xOff = 10;
            const int yOff = 10;
            grfx.DrawPolygon(pen, new PointF[]
              {
                Conv(grfx, new PointF(xOff, yOff)),
                Conv(grfx, new PointF(xOff + 100, yOff)),
                Conv(grfx, new PointF(xOff + 100, yOff + 12)),
                Conv(grfx, new PointF(xOff, yOff + 12))
              }
        );

StringFormat  fmt = new StringFormat();
fmt.Alignment =  StringAlignment.Center;
//fmt.LineAlignment = StringAlignment.Near;

for (int i=1; i < 100; i++)
{
    if (i % 10 == 0) // Сантиметрові ділення
     {
        grfx.DrawLine(pen,
        Conv(grfx, new PointF(xOff + i, yOff)),
        Conv(grfx, new PointF(xOff + i, yOff + 6)));

        grfx.DrawString((i/10).ToString(), Font, brush,
        Conv(grfx, new PointF(xOff + i, yOff + 7)),fmt);
     }
}
   PointF Conv(Graphics grfx, PointF pointf)
    {
        pointf.X*= grfx.DpiX  / 25.4f;
        pointf.Y *= grfx.DpiY / 25.4f;
        return pointf;
    }
   PointF Conv_pr(Graphics grfx, PointF pointf)
    {
        pointf.X *= 100 / 25.4f;
        pointf.Y *= 100 / 25.4f;
        return pointf;
    }
  private void printDocument1_PrintPage(object sender,
  System.Drawing.Printing.PrintPageEventArgs e)
  {
    Graphics grfx = e.Graphics;
    Pen pen = new Pen(ForeColor);
    Brush brush = new SolidBrush(ForeColor);
    const int xOff = 10;
    const int yOff = 10;
    grfx.DrawPolygon(pen, new PointF[]
    {
```

```
        Conv_pr(grfx, new PointF(xOff, yOff)),
        Conv_pr(grfx, new PointF(xOff + 100, yOff)),
        Conv_pr(grfx, new PointF(xOff + 100, yOff + 12)),
        Conv_pr(grfx, new PointF(xOff, yOff + 12))
      }
      );

      StringFormat fmt = new StringFormat();
      fmt.Alignment = StringAlignment.Center;
      for (int i = 1; i < 100; i++)
      {
          if (i % 10 == 0) // Сантиметрові ділення
          {
              grfx.DrawLine(pen,
              Conv_pr(grfx, new PointF(xOff + i, yOff)),
              Conv_pr(grfx, new PointF(xOff + i, yOff + 6)));
              grfx.DrawString((i / 10).ToString(), Font, brush,
              Conv_pr(grfx, new PointF(xOff + i, yOff + 7)), fmt);
          }
          else if (i % 5 == 0) // Ділення по 5 мм
          {
              grfx.DrawLine(pen,
              Conv_pr(grfx, new PointF(xOff + i, yOff)),
              Conv_pr(grfx, new PointF(xOff + i, yOff + 4)));
          }
          else // Міліметрові ділення
          {
              grfx.DrawLine(pen,
              Conv_pr(grfx, new PointF(xOff + i, yOff)),
              Conv_pr(grfx, new PointF(xOff + i, yOff + 2)));
          }
      }
}
      private void button1_Click(object sender, EventArgs e)
      {
        this.printDocument1.Print();
      }
    }
}
```

а)



б)

Рис. 5. Виведення лінійки на екран (а) і принтер (б)