РЕАЛИЗИРАНЕ НА СТАНДАРТНИ WINDOWS ПРИЛОЖЕНИЯ, СЪДЪРЖАЩИ МЕНЮТА ЧРЕЗ СРЕДАТА VISUAL C# ПРИ ИЗУЧАВАНЕ НА СЪБИТИЙНО ПРОГРАМИРАНЕ В СРЕДНОТО УЧИЛИЩЕ

Стефка Анева

ПУ „П. Хилендарски”, Пловдив, бул. България № 236, stfaneva@uni-plovdiv.bg

Настоящата разработка е посветена на изучаването на събитийно програмиране чрез средата Visual C# в профилираната подготовка по информатика в средното училище. Посочена е примерна класификация на система от задачи за изучаване на учебното съдържание за модула „Събитийно програмиране в среда на графичен потребителски интерфейс“. Дискутирани са средствата и технологиите за реализиране на стандартни Windows приложения, съдържащи менюта в средата Visual C#. Предложена е примерна задача.

Ключови думи: *събитийно програмиране, графичен потребителски интерфейс, елемент, компонент, събитие, събитийна процедура, меню, лента с инструменти, диалози.*

I. Въведение.

Изучаването на събитийно програмиране в профилираната подготовка по информатика в средното училище чрез подходящо подбран набор от учебни задачи с различна степен на сложност дава възможност на учениците да се запознаят с основните принципи и възможности на визуалното програмиране и да усвоят основни технологии и механизми за реализиране на програми, управлявани от събития с достъпен графичен потребителски интерфейс.

II. Основни типове задачи за модула „Събитийно програмиране в среда на графичен потребителски интерфейс”

В курса на обучение за модула „Събитийно програмиране в среда на графичен потребителски интерфейс” [1] трябва да бъдат разгледани следните пет типа задачи:

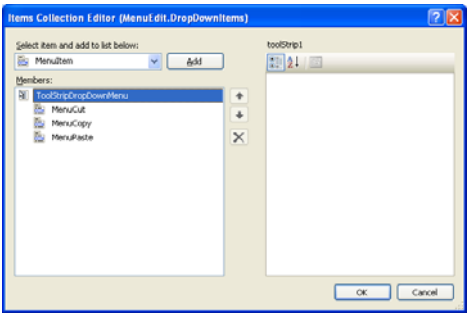
1. Първият тип задачи включва набор от упражнения за усвояване на графичния потребителски интерфейс (ГПИ) и работа с основните елементи от него.
2. Вторият тип задачи е за формиране на знания и умения за реализиране на графика и анимация.
3. Третият тип задачи е посветен на работата с масиви от елементи на ГПИ и използване при необходимост от групиране на специални „елементи-носители” (контейнери) на ГПИ.

4. Четвъртият тип задачи реализира потребителски настроени приложения за връзка с бази от данни.
5. Петият тип задачи включва **създаване на стандартни Windows приложения, съдържащи менюта**.

III. Задачи за формиране на знания и умения за реализиране на Windows приложения, съдържащи менюта в средата Visual C#

✓ *Създаване на менюта във форма на приложение*

Менютата са важен елемент на формите в едно приложение с графичен потребителски интерфейс. Чрез тях потребителят има възможност за бърз достъп при изпълнение на избрана от него операция чрез директен избор на съответната команда от менюто на приложението. Менютата са организирани в йерархична структура. Даден елемент от меню, сам по себе си може да представлява меню, когато се използва за визуализиране на елементи от подменю. Менютата от първо ниво представляват стандартни падащи менюта и образуват лентата с основни менюта за дадено приложение. Те могат да съдържат в себе си списък от **MenuItem** елементи, които представят отделните възможности за избор (команди) от избраното меню.



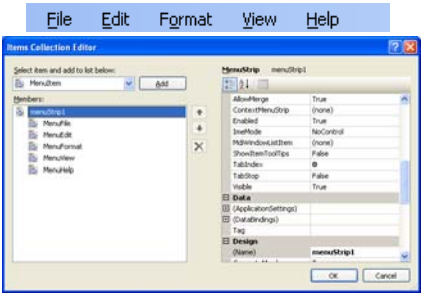
Фиг. 1

Фиг. 2

Този списък може да се визуализира чрез свойството **DropDownItems** на съответното основно меню (напр. Фиг. 1).

Класовете и типовете от пространството **System.Windows.Forms** осигуряват средства за работа с прозорци, диалози, елементи за въвеждане на текст, елементи за избор, менюта, ленти с инструменти и др. [2] За реализиране на менюта в средата Visual C# се използват компонентите **MenuStrip** и **ContextMenuStrip** (Фиг. 2). Тези компоненти нямат графичен образ върху формата на приложението (т.е. добавят се в специално поле под нея) и представляват списък от **MenuItem** елементи за избор. Всеки **MenuItem** елемент вече има реален графичен образ върху формата и може да бъде както команда в приложението, така и родителско меню за други елементи.

При избор на свойството **Items** за компонента **MenuStrip** в прозореца **Properties** може да бъде прегледан по-подробно списъка с основни менюта на дадено приложение. (напр. Фиг. 3)



Фиг. 3

Компонентът **ContextMenuStrip** представлява контекстно меню, което се появява, когато потребителят щракне с десния бутон на мишката върху определен елемент на ГПИ или някъде във формата на приложението. Контекстното меню съдържа списък от **MenuItem** елементи, представляващи отделните команди от менюто.

При необходимост от разделяне на групи от команди със свързана функционалност в дадено падащо меню се използват разделителни линии. Обикновено поставянето на разделителни линии в дадено меню се прави с цел потребителите да могат бързо да се ориентират и да намерят желаните команди за изпълнение.

Клавишите за достъп се използват за обхождане на йерархията от менюта в комбинация с клавиша Alt. Това се постига чрез поставяне на знака амперсанд (&) пред буквата, която ще се използва като клавиш за достъп, в заглавието на менюто или елемента от менюто. Много е важно клавишите за достъп да бъдат уникални на всяко ниво от йерархията на менюта.

Клавишите за бърз достъп се различават от клавишите за достъп, по това, че предизвикват мигновено отваряне на елемент от менюто. За да се достигне до елемент от менюто, използвайки клавиши за достъп трябва да се обходи съответната йерархия на менюта. Клавишите за бърз достъп се свързват към елементи от меню посредством свойство **ShortcutKeys** за конкретното меню.

✓ ***Създаване на лента с инструменти във форма на приложение***

Лентите с инструменти са често използвани при приложенията с графичен потребителски интерфейс и могат да съдържат различни по тип елементи - бутони, комбинирани кутии, текстови полета, етикети и др. За реализиране на ленти с инструменти в средата Visual C# се използва компонента **ToolStrip.**

✓ ***Създаване на лента за състояние във форма на приложение***

Лентите за състояние са още една от типичните компоненти в приложения с ГПИ. Обикновено те се състоят от отделни секции (панели), които могат да съдържат текст или икони. Тези ленти се използват за визуализация на информация, свързана със състоянието на приложението. За реализиране на лента за състоянието в средата Visual C# се използва компонента **StatusStrip**.

✓ ***Диалози за избор на файл***

При създаване на приложения с ГПИ често се налага да се осъществява достъп до локалната файлова система с цел да се реализира възможност за отваряне, модификация и съхраняване на даден тип файлове. За тази цел Windows Forms предоставя стандартни диалози, които осигуряват достъп до стандартно настроените диалогови прозорци, извършващи операции по отваряне и съхранение на файлове (прозорци Open и Save As). Компонентът **OpenFileDialog** представлява диалог за избор на файл при отваряне. Този клас ни позволява да проверим дали файл съществува и да го отворим. Компонентът **SaveFileDialog** представлява диалог за избор на файл при съхраняване. Този клас ни позволява да презапишем съществуващ или да създадем нов файл.

✓ ***Диалози за избор на шрифт и цвят.***

Компонентът **FontDialog** представлява диалог за избор на шрифт чрез визуализиране на стандартния диалогов прозорец **Font** за избор на форматиращи характеристики на шрифт (име на шрифт, размер, стил, цвят, ефекти). Компонентът **ColorDialog** представлява диалог за избор на цвят чрез визуализиране на стандартния диалогов прозорец **Color** за избор на цвят.

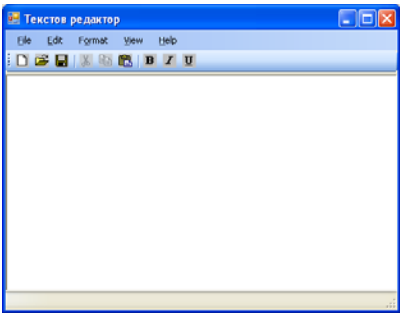
IV. Примерна задача

Създайте приложение, което съдържа следните елементи на ГПИ: **MenuStrip**, **ContextMenuStrip**, **ToolStrip**, **StatusStrip**, **RichTextBox**, **OpenFileDialog**, **SaveFileDialog**, **FontDialog**, **ColorDialog**, както е показано на Фиг. 4. Приложението има за цел да бъдат реализирани действията, свързани със създаване, преглеждане, редактиране, форматиране и съхраняване на текстови файлове. В приложението трябва да бъдат реализирани следните функционални възможности:

1. Да се реализира система от менюта за избор и изпълнение на основните действия, свързани със създаване, преглеждане, редактиране, форматиране и съхраняване на текстови файлове;
2. Да се създаде лента с инструменти за бърз достъп до често използвани действия, свързани със създаване, преглеждане, редактиране, форматиране и съхраняване на текстови файлове. За всеки от бутоните да се зададе съответен подсказващ текст;
3. Да се създаде контекстно меню, включващо командите Cut, Copy и Paste и улесняващо работата по редактиране на текст. Това меню да се визуализира при щракване с десен бутон върху елемента RichTextBox;
4. Да се създаде лента за състояние на приложението, която съдържа етикет, показващ името на текущия файл, с който работи приложението;

5. При изпълнение на действията по копиране, изрязване и поставяне на текст в елемента RichTextBox да се съобрази кога тези команди (бутони) да бъдат активни или не.

Съхранете проекта с име TextFileViewer.



Фиг. 4

Основни цели:

- Учениците да се запознаят с технологията за създаване на менюта в приложение на C#;
- да разберат предназначението и да се запознаят с някои основни свойства на разгледаните в задачата компоненти – **MenuStrip**, **ContextMenuStrip**, **ToolStrip**, **StatusStrip**;
- да се запознаят с предназначението и спецификата на приложение на елемента **RichTextBox**;
- да разберат предназначението и да се запознаят с някои свойства на разгледаните в задачата компоненти за реализиране на диалози за избор на файл, шрифт и цвят – **OpenFileDialog**, **SaveFileDialog**, **FontDialog** и **ColorDialog**;
- да затвърдят своите знания и умения за създаване и извикване на обикновени процедури в C#.

На учениците се предлага следното **решение:**

1) **Първи етап:** Създаване на ГПИ на приложението (фиг. 4), менюто (Таблица № 1) и лентата с инструменти (Фиг. 6) на приложението;

В случая вместо обикновено текстово поле (**TextBox**) ще се използва елемента **RichTextBox**, който дава възможност на потребителя да въвежда и редактира текст, и притежава много по-разширени възможности за форматиране в сравнение с текстовото поле. Методите **LoadFile()** и **SaveFile()** позволяват зареждане и съхраняване на текста в Rich Text Format (RTF) файл или в текстов файл. Свойството **SelectedText** служи за извличане и задаване на областта от текста, която е маркирана. Чрез свойствата **SelectionFont**, **SelectionColor** и **SelectionAlignment** могат да се задават шрифт, цвят и подравняване на текущия маркиран текст.

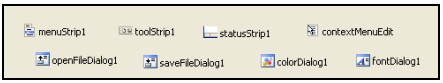
Добавяне на неграфични компоненти към форма на приложение

Неграфичните компоненти нямат графичен потребителски интерфейс. Те не се визуализират върху формата на приложението, а се изобразяват в специална област под нея (Фиг. 5).

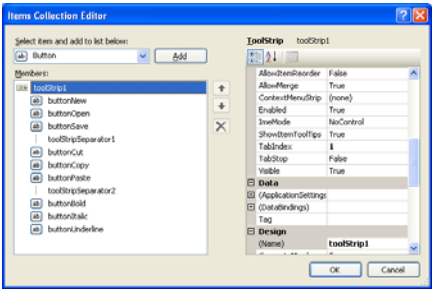
Таблица № 1

Меню на приложението

Тип	Свойство Text	Свойство Name
MenuItem	**&File**	MenuFile
MenuItem	New	MenuNew
MenuItem	Open	MenuOpen
MenuItem	Save	MenuSave
MenuItem	Save As	MenuSaveAs
Separator		
MenuItem	Exit	MenuExit
MenuItem	**&Edit**	MenuEdit
MenuItem	Cut	MenuCut
MenuItem	Copy	MenuCopy
MenuItem	Paste	MenuPaste
MenuItem	**F&ormat**	MenuFormat
MenuItem	Change Font	MenuChangeFont
MenuItem	Back Color	MenuBackColor
MenuItem	**&View**	MenuView
MenuItem	Toolbar	MenuViewToolBar
MenuItem	StatusBar	MenuViewStatusBar
MenuItem	**&Help**	MenuHelp
MenuItem	About	MenuAbout



Фиг. 5



Фиг. 6

2) **Втори етап:** Настройка на някои свойства на елементите на ГПИ в режим на проектиране (таблица № 2).

Таблица № 2

Елементи на ГПИ	Име на елемент	Свойства
OpenFileDialog	openFileDialog1	DefaultExt=*.rtf; Filter= RTF Files (*.rtf)|*.rtf|TXT Files (*.txt)|*.txt
SaveFileDialog	saveFileDialog1	DefaultExt=*.rtf; Filter= RTF Files (*.rtf)|*.rtf|TXT Files (*.txt)|*.txt

3) Трети етап: Добавяне на програмен код към елементите.

✓ Реализиране на събитийна процедура **MenuNew_Click**.

```
private void MenuNew_Click(object sender, EventArgs e)
  { richTextBox1.Text = ""; toolStripStatusLabel1.Text = "";
    openFileDialog1.FileName = ""; saveFileDialog1.FileName = "";}
```

✓ Реализиране на събитийна процедура **MenuOpen_Click**.

```
private void MenuOpen_Click(object sender, EventArgs e)
  { openFileDialog1.FileName = ""; openFileDialog1.DefaultExt = "*.rtf";
    openFileDialog1.Filter = "RTF Files (*.rtf)|*.rtf|TXT Files (*.txt)|*.txt";
  if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK &&
      openFileDialog1.FileName.Length > 0)
      { richTextBox1.LoadFile(openFileDialog1.FileName);
        toolStripStatusLabel1.Text = openFileDialog1.FileName;
        saveFileDialog1.FileName = openFileDialog1.FileName;} }
```

✓ Реализиране на събитийна процедура **MenuSave_Click**.

```
private void MenuSave_Click(object sender, EventArgs e)
  { if (saveFileDialog1.FileName == "")
      { MenuSaveAs_Click(sender, e); }
    else  { if (saveFileDialog1.FilterIndex == 1)   // в случай, че е избран тип на файл - RTF
              richTextBox1.SaveFile(saveFileDialog1.FileName,
                                    RichTextBoxStreamType.RichText); // съхр. в RTF формат
            else if (saveFileDialog1.FilterIndex == 2)    // при избран тип на файл - TXT
              richTextBox1.SaveFile(saveFileDialog1.FileName,
                                    RichTextBoxStreamType.PlainText);}}// съхр. в TXT формат
```

✓ Реализиране на събитийна процедура **MenuSaveAs_Click**.

```
private void MenuSaveAs_Click(object sender, EventArgs e)
  { saveFileDialog1.DefaultExt = "*.rtf";
    saveFileDialog1.Filter = "RTF Files (*.rtf)|*.rtf|TXT Files (*.txt)|*.txt";
    if (saveFileDialog1.FilterIndex == 1)
     { saveFileDialog1.DefaultExt = "*.rtf";   // разширение на файл по подразбиране  - *.rtf
      if (saveFileDialog1.ShowDialog() ==  System.Windows.Forms.DialogResult.OK &&
         saveFileDialog1.FileName.Length > 0)
       richTextBox1.SaveFile(saveFileDialog1.FileName,
                             RichTextBoxStreamType.RichText);}
    else if (saveFileDialog1.FilterIndex == 2)
     { saveFileDialog1.DefaultExt = "*.txt";  // разширение на файл по подразбиране  - *.txt
      if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK &&
         saveFileDialog1.FileName.Length > 0)
       richTextBox1.SaveFile(saveFileDialog1.FileName,
                             RichTextBoxStreamType.PlainText);}
    toolStripStatusLabel1.Text = saveFileDialog1.FileName;}
```

✓ Реализиране на събитийна процедура **MenuExit_Click**.

```
private void MenuExit_Click(object sender, EventArgs e)   { Close();}
```

✓ Реализиране на събитийна процедура **MenuCut_Click**.

```
private void MenuCut_Click(object sender, EventArgs e)
   { Clipboard.Clear(); Clipboard.SetText(richTextBox1.SelectedText);
     richTextBox1.SelectedText = "";}
```

✓ Реализиране на събитийна процедура **MenuCopy_Click**.

```
private void MenuCopy_Click(object sender, EventArgs e)
   { Clipboard.Clear(); Clipboard.SetText(richTextBox1.SelectedText);}
```

✓ Реализиране на събитийна процедура **MenuPaste_Click**.

```
private void MenuPaste_Click(object sender, EventArgs e)
   { richTextBox1.SelectedText = Clipboard.GetText();}
```

✓ Реализиране на събитийна процедура **MenuChangeFont_Click**.

```
private void MenuChangeFont_Click(object sender, EventArgs e)
   { fontDialog1.ShowColor = true; fontDialog1.ShowDialog(); // визуализация на диалога Font
     richTextBox1.SelectionFont = fontDialog1.Font;
     richTextBox1.SelectionColor = fontDialog1.Color;}
```

✓ Реализиране на събитийна процедура **MenuBackColor_Click**.

```
private void MenuBackColor_Click(object sender, EventArgs e)
   { colorDialog1.ShowDialog(); richTextBox1.BackColor = colorDialog1.Color;}
```

✓ Реализиране на събитийна процедура **MenuViewToolBar_Click** за избор дали да се визуализира лентата с инструменти във формата на приложението.

```
private void MenuViewToolBar_Click(object sender, EventArgs e)
   { toolStrip1.Visible = !toolStrip1.Visible;
     MenuViewToolBar.Checked = !MenuViewToolBar.Checked;}
```

✓ Реализиране на събитийна процедура **MenuViewStatusBar_Click** за избор дали да се визуализира лентата за състояние във формата на приложението.

```
private void MenuViewStatusBar_Click(object sender, EventArgs e)
```

```
  { statusStrip1.Visible = !statusStrip1.Visible;
    MenuViewStatusBar.Checked = !MenuViewStatusBar.Checked;}
```

✓ Реализиране на събитийна процедура **MenuAbout_Click**.

```
private void MenuAbout_Click(object sender, EventArgs e)
  { MessageBox.Show("Кратък текстов редактор", "Информация за програмата",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);}
```

✓ Реализиране на съответните събитийни процедури на елементите от контекстното меню за редактиране, включващо командите Cut, Copy и Paste.

```
private void ContextMenuCut_Click(object sender, EventArgs e) { MenuCut_Click(sender, e);}
private void ContextMenuCopy_Click(object sender, EventArgs e)
  { MenuCopy_Click(sender, e);}
private void ContextMenuPaste_Click(object sender, EventArgs e)
  { MenuPaste_Click(sender, e);}
```

✓ Реализиране на обикновена процедура **clipboard_enabled**, която определя дали съответните команди от менютата и бутони за редактиране да са активни или не.

```
private void clipboard_enabled()
  { if (richTextBox1.SelectedText != "")
    { ContextMenuCut.Enabled = true; MenuCut.Enabled = true;
      ContextMenuCopy.Enabled = true; MenuCopy.Enabled = true;
      buttonCut.Enabled = true; buttonCopy.Enabled = true;}
    else
    { ContextMenuCut.Enabled = false; MenuCut.Enabled = false;
      ContextMenuCopy.Enabled = false; MenuCopy.Enabled = false;
      buttonCut.Enabled = false; buttonCopy.Enabled = false;} }
```

✓ Реализиране на събитийна процедура **richTextBox1_MouseDown**.

```
private void richTextBox1_MouseDown(object sender, MouseEventArgs e)
    { clipboard_enabled();
      if (e.Button == MouseButtons.Right)  { contextMenuEdit.Show(Form1.MousePosition);}}
```

✓ Реализиране на събитийна процедура **richTextBox1_KeyDown**.

```
private void richTextBox1_KeyDown(object sender, KeyEventArgs e)  { clipboard_enabled(); }
```

✓ Реализиране на събитийна процедура **richTextBox1_MouseMove**.

```
private void richTextBox1_MouseMove(object sender, MouseEventArgs e)
  {  clipboard_enabled();}
```

✓ Реализиране на събитийна процедура **Form1_Load** за начална инициализация.

```
private void Form1_Load(object sender, EventArgs e)
  { clipboard_enabled();
    MenuViewToolBar.Checked = true; MenuViewStatusBar.Checked = true; }
```

✓ Реализиране на действието на бутоните от лентата с инструменти.

```
private void buttonNew_Click(object sender, EventArgs e)   { MenuNew_Click(sender, e);}
private void buttonOpen_Click(object sender, EventArgs e) { MenuOpen_Click(sender, e);}
private void buttonSave_Click(object sender, EventArgs e)  { MenuSave_Click(sender, e);}
private void buttonCut_Click(object sender, EventArgs e)    { MenuCut_Click(sender, e);}
private void buttonCopy_Click(object sender, EventArgs e)  { MenuCopy_Click(sender, e);}
private void buttonPaste_Click(object sender, EventArgs e) { MenuPaste_Click(sender, e);}
private void buttonBold_Click(object sender, EventArgs e)
```

```
{ bool i,b,u; i = false; b = false; u = false;
  if (richTextBox1.SelectionFont != null)
    { System.Drawing.Font currentFont = richTextBox1.SelectionFont;
      System.Drawing.FontStyle newFontStyle;  if (currentFont.Italic == true) i = true;
      if (currentFont.Bold == true) b = true;  if (currentFont.Underline == true) u = true;
      if (b)   { if ((u) && (i)) newFontStyle = FontStyle.Italic | FontStyle.Underline;
                 else if (u) newFontStyle = FontStyle.Underline;
                 else if (i) newFontStyle = FontStyle.Italic;
                 else newFontStyle = FontStyle.Regular; }
      else  { if ((u) && (i)) newFontStyle = FontStyle.Bold | FontStyle.Italic |
                                             FontStyle.Underline;
              else if (u) newFontStyle = FontStyle.Bold | FontStyle.Underline;
              else if (i) newFontStyle = FontStyle.Bold | FontStyle.Italic;
              else newFontStyle = FontStyle.Bold; }
      richTextBox1.SelectionFont = new Font(
            currentFont.FontFamily,currentFont.Size,newFontStyle);} }
```

Аналогично се постъпва и при реализиране на събитийните процедури **buttonItalic_Click** и **buttonUnderline_Click**.

V. Заключение.

В настоящия материал е представен проект за организация на учебния процес и методика за провеждане на профилирано обучение по информатика за темата „*Събитийно програмиране в среда на графичен потребителски интерфейс”,* като за целта се използва средата на Visual C#. Предложена е примерна задача за запознаване с основните възможности, които предоставя Visual C# за създаване на Windows приложения, съдържащи менюта. Важен момент при реализиране на такъв тип задачи е разглеждане и прилагане на технологията за създаване на приложения с многодокументен интерфейс.

<u>В резултат от обучението на този етап ученикът трябва да:</u>

- знае предназначението на компонентите за реализиране на система от менюта и умее да ги използва при създаване на приложения с подходящ дизайн и оформление на ГПИ;
- усъвършенства и развие своите умения за правилен подбор на подходящи елементи на ГПИ в съответствие с необходимата функционалност на приложението;
- познава предназначението на компонентите за реализиране на стандартни диалози при избор на файл, шрифт и цвят;
- се запознае с технологията за създаване на приложения с многодокументен интерфейс и да разбере и осмисли принципа на работа на формите в MDI режим.

През последните няколко години за преподаване на модула „Събитийно програмиране в среда на графичен потребителски интерфейс” в средното училище се използваше предимно средата на Visual Basic 6.0. В [3] е

дискутирана технологията за създаване на стандартни Windows приложения, съдържащи менюта чрез използване на средата Visual Basic 6.0 и са представени подобни задачи. В тази среда при проектиране на система от менюта се използваше специален редактор и отделните менюта можеха да разпознават само събитие Click.

На този етап от обучението по събитийно програмиране могат да бъдат разгледани и набор от задачи, даващи възможност за творческо прилагане на усвоените знания и умения за ефективно решаване на реални практически проблеми и задачи [4]. Например, от монография [5] могат да бъдат разгледани числени алгоритми, които да бъдат реализирани със средствата на събитийното програмиране като приложения с ГПИ и да се навигират със система от менюта. Системите от менюта намират приложения при изграждането на информационни системи [6], [7] и специализиран софтуер [8].

Литература

[1] МОН, Дирекция „Политика в общото образование”, Учебни програми III част за ЗП и ЗПП – IX, X, XI и XII клас, София, 2003.

[2] Наков, С. и колектив, Програмиране за .Net Framework, том 2, Барс, 2007.

[3] Гъров К., Ст. Анева, За задачите в модула “Събитийно програмиране с Visual Basic 6.0”. Задачи, реализиращи приложения за връзки с бази от данни. Задачи за приложения – тип меню, сп. “Математика и информатика”, приложение на кн. 3, 2004 г., стр. 1-20 (от приложението)

[4] Гъров, К., Задачите в обучението по информатика и информационни технологии, Сборник доклади на Национална конференция „Образованието в информационното общество”, Пловдив, 27-28.05.2010, 95-101.

[5] Iliev, A., N. Kyurkchiev, Nontrivial Methods in Numerical Analysis: Selected Topics in Numerical Analysis, LAP LAMBERT Academic Publishing GmbH & Co. KG, Saarbrucken (2010).

[6] Valchanov, N., T. Terzieva, V. Shkurtov, A. Iliev. Approaches in Building and Supporting Business Information Systems, Сборник доклади от Международна научна конференция „Информационни технологии в управлението на бизнеса”, Варна, 16-17.10.2009, 100-105.

[7] Valchanov, N., T. Terzieva, V. Shkurtov, A. Iliev, Architecture of extensible computations driven systems, Сборник доклади на Тридесет и деветата пролетна конференция на Съюза на математиците в България, Албена, 6-10.04.2010, 207-211.

[8] Илиев, А., Г. Христозов. Някои практически приложения на софтуерна система за представяне на динамични модели, Сборник доклади от Юбилейна конференция „Науката, образованието и времето като грижа”, Смолян, 30.11-01.12.2007, 64-68.