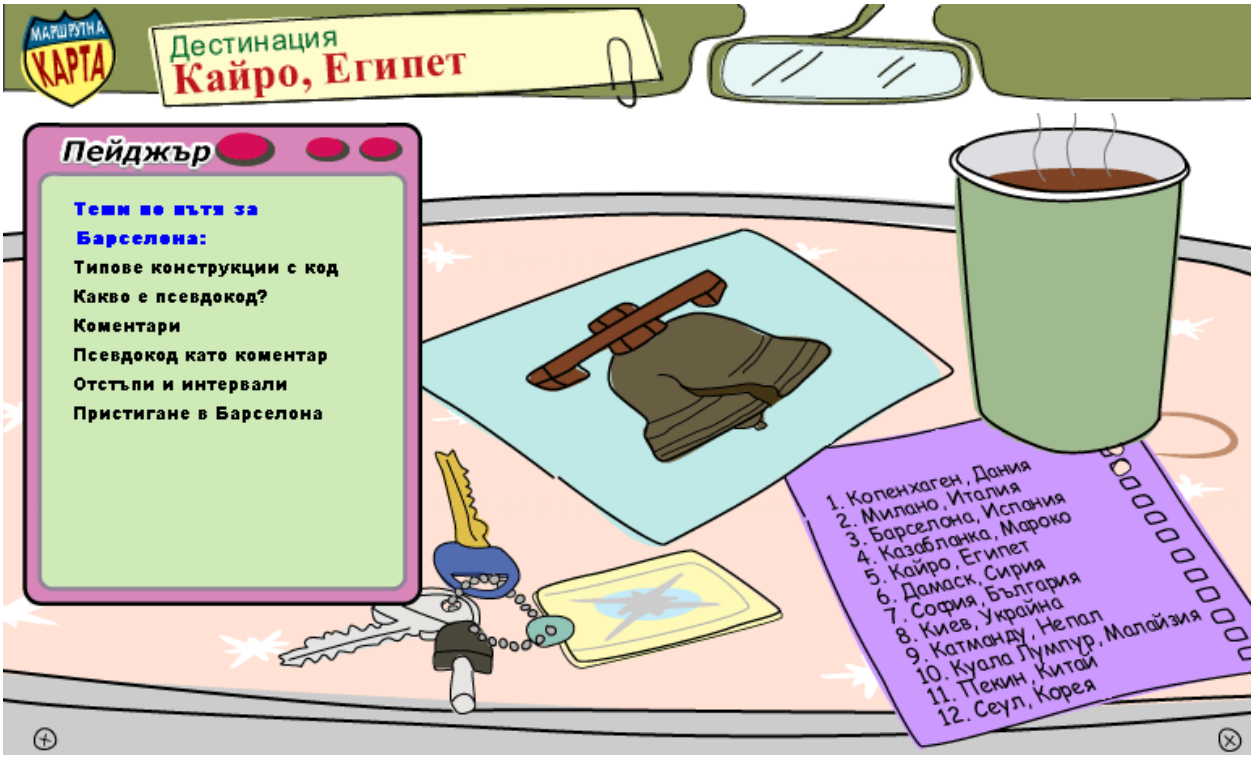
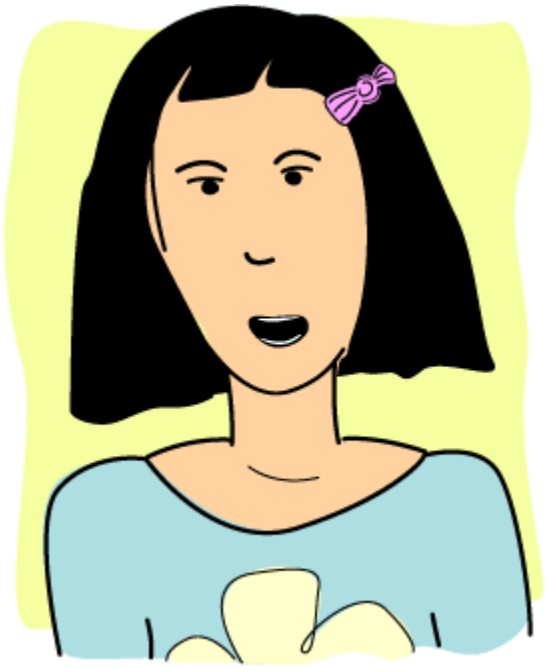

МАРШРУТНА КАРТА
Дестинация
Кайро, Египет
Пейджър
Теми по пътя за
Барселона:
Типове конструкции с код
Какво е псевдокод?
Коментари
Псевдокод като коментар
Отстъпи и интервали
Пристигане в Барселона
1. Копенхаген, Дания
2. Милано, Италия
3. Барселона, Испания
4. Казабланка, Мароко
5. Кайро, Египет
6. Дамаск, Сирия
7. София, България
8. Киев, Украйна
9. Катманду, Непал
10. Куала Лумпур, Малайзия
11. Пекин, Китай
12. Сеул, Корея

Свойства и методи

Свойства



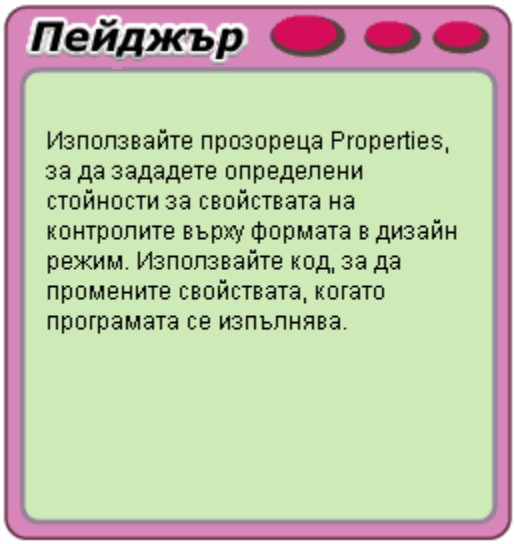
Пътешествието до Казабланка беше забавно! Сигурна съм, че това е едно от любимите места на Клиф. Клиф знае доста за дизайна на формите и контролите. Той ви показа как да създадете форма с контроли и дори ви показа как да промените някои от свойствата на контролите като използвате прозореца Properties.

Аз ще карам до следващата ни дестинация – Кайро, Египет. Били ли сте там? Това е прекрасно място, разположено в делтата на река Нил. Кайро е най-големият град в Африка, който съществува повече от 6,000 години. По пътя ще ви разкажа повече за свойствата. Ще ви покажа как да ги прочитате и записвате като използвате Visual Basic код. След като научите свойствата на контролите, ще ви запозная и с методите на контролите.

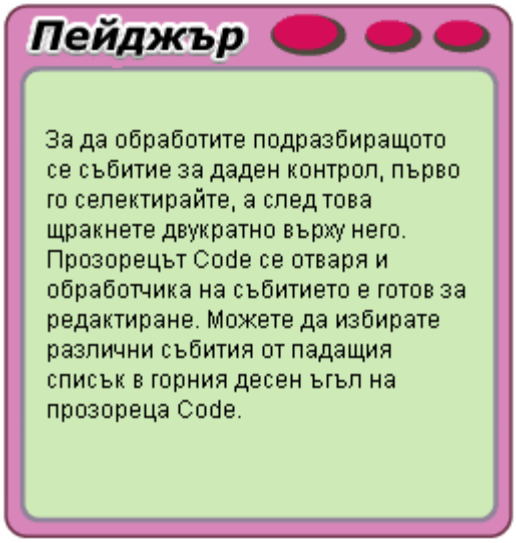
Началото на програмирането

Дизайна на формите и промяната на свойствата на контролите посредством прозореца Properties са забавни, но те са само началото! Прозорецът Properties е лесен за използване, но не е много гъвкав. Какво ще направите, ако потребителят желае да променя цвета на формата, когато работи с програмата ви? Какво ще направите, ако иска да промени размера или стила на шрифта? Как програмата ви ще знае, че потребителят е маркирал поле за отметка или е селектирал радио-бутон? Потребителите нямат достъп до прозореца Properties. Тогава как биха могли да направят нещата по-горе?

Добре, отговорът е: с Visual Basic код! Всичко това, дори повече може да се направи като използвате код. За да прочетете или запишете стойностите на свойствата, трябва да напишете код. Това е много лесно и в същото време могъщо! За мен оттук започва истинското програмиране.



Къде ще напишете кода? Ще използвате страницата с кода на формата, която сте създали. В повечето случаи ще добавяте кода в обработчиците на събитията за контролите. Спомняте ли си асоциираното с бутона събитие? Когато потребителят изпълнява някакво действие, например натискане на бутон, кодът, който обработва събитието се изпълнява, а свойствата на контрола се четат или записват. Добавянето на код за обработка на събитията от контролите е първата стъпка към създаването на истинска работеща програма. Спомняте ли си формата за "спуканата гума"? Тя няма да може да прави каквото и да е преди да добавите код към нея.



Първо ще ви покажа няколко примера за начина, по който можете да четете стойностите на свойствата на контролите като използвате Visual Basic код. След това ще ви покажа как да ги записвате.

Прочитане на свойства посредством код

Нека да разгледаме няколко истински примера за начина, по който могат да се четат свойствата на контролите посредством Visual Basic код. Като начало ще ви покажа как да прочитате някои свойства, които обикновено се задават от прозореца Properties. След това ще ви покажа как да прочитате някои свойства, които потребителят може да променя, ползвайки програмата ви. Да вървим!

Създайте ново Windows приложение с име ReadProperties. Отворете прозореца Toolbox и добавете бутон на формата. Отворете прозореца Properties и задайте стойност "Read Text" за свойството Text на бутона.

Щракнете двукратно върху бутона, за да добавите код за обработка на събитието Button1_Click. Въведете следния код:

```
MessageBox.Show(Button1.Text)
```

Изградете и стартирайте приложението. Натиснете бутона "Read Text". Ще се появи диалогов прозорец със съобщение "Read Text" каквото е заглавието на бутона. Обърнете внимание, че кодът първоначално прочита стойността на свойството Text на бутона след, което я визуализира в диалогов прозорец. Впечатляващо!

Ето друг пример. Добавете друг бутон на формата. Използвайте прозореца Properties Window, за да зададете стойност "Form Size" за свойството Text на бутона. Щракнете двукратно върху бутона, за да добавите код за обработка на събитието Button2_Click. Въведете следния код.

```
MessageBox.Show(Form1.ActiveForm.Height & "," &
Form1.ActiveForm.Width)
```

Изградете и стартирайте проекта. Натиснете бутона "Form Size". Ще се появи диалогов прозорец със съобщение: 300,300 (или нещо близко до това в зависимост от размера на формата). Кодът визуализира стойностите на свойствата Height и Width на формата, разделени с точка! Прекрасно и същевременно просто!

Какви други свойства може да чете Visual Basic кодът? В действителност Visual Basic може да чете всички свойства на формата и нейните контроли, които са показани в прозореца Properties. Вашия код може да

прочита свойства като Height и Width, BackColor, ForeColor, и местоположения, определени от X и Y координати. Някои от свойствата, показани в прозореца Properties, се задават при създаването на формата и не се променят при стартирането на програмата. Ето защо тяхното прочитане не е от особено значение.

Действително полезно е да четете свойствата на контролите, които потребителят може да промени, когато използва програмата ви. Такива са свойството Text на текстовото поле и свойството Checked на полето за отметка. Това е начинът вашата програма да получава информация от потребителя. Какъв регистрационен номер въведе Джин? Коя гума е спукана? Какъв размер гума избра Cliff?

Ето и някои практически примери. Нека да напишем код, който показва дали дадено поле за отметка е маркирано. Отворете прозореца Toolbox и добавете поле за отметка и трети бутон върху формата. Отворете прозореца Properties и задайте стойност "Check Me" за свойството Text на бутона. Щракнете двукратно върху бутона, за да добавите код за обработка на събитието Button3_Click.

Въведете следния код

```
MessageBox.Show(CheckBox1.Checked)
```

Изградете и стартирайте приложението. Натиснете бутона "Check Me". Какво съобщение визуализира диалоговият прозорец? Маркирайте полето за отметка и натиснете бутона "Check Me" отново. Какво съобщение визуализира диалоговият прозорец сега? Всеки път, когато натиснете бутона, диалоговият прозорец ще показва съобщение с текущата стойност на свойството Checked на полето за отметка. Тя може да бъде True или False.

Полезен съвет

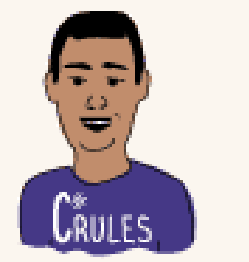
Учениците трябва да знаят как да прочитат и променят свойствата посредством код. Допреди да изучат оператора If, те са ограничени във възможността да направят

А сега един последен пример. Ще ви покажа как да визуализирате текста, който потребителят въвежда в текстово поле.

Добавете текстово поле върху формата. Отворете прозореца Properties и се уверете, че стойността на свойството ReadOnly е False. Задайте стойност "Type Here" на свойството Text. Добавете четвърти бутон и задайте стойност "Read Text" за неговото свойството Text. Щракнете двукратно върху бутона, за да добавите код, който ще обработва събитието Button4_Click. Въведете следния код

```
MessageBox.Show(TextBox1.Text)
```

Изградете и стартирайте приложението. Щракнете двукратно в текстовото поле и натиснете клавиша Backspace, за да изтриете съдържанието му. Въведете някакъв текст. Натиснете бутона "Read Text". Какво съобщение показва диалоговият прозорец? Променете текста в текстовото поле и отново натиснете бутона "Read Text". Какво се случва?



В C# свойствата на контролите се прочитат по аналогичен на Visual Basic начин. Ето пример на C#, който показва стойността на свойството Text на текстово поле:

```
{
MessageBox.Show(textBox1.Text);
}
```

С изключение на фигурните скоби и точката и запетая в края на реда, кодът изглежда както на Visual Basic. Също така в C# обектът textBox1 се пише с малка буква “t”.

Примерите, които ви показах, илюстрират как посредством Visual Basic код можете да четете свойствата на контролите. Това, обаче е само едната страна на нещата, тъй като Visual Basic кодът може да се използва и за записване на нови стойности за свойствата.

Записване на свойства посредством код

Прекрасно е, че можем да прочитаме свойствата на контролите посредством Visual Basic код. Това е начинът, по който могат да се прихванат действията на потребителя при изпълнението на програмата. В зависимост от тези действия програмата може да продължи изпълнението си по различни начини. Как бихте могли да промените свойствата на някои от контролите, когато потребителят стартира програмата? Например, искам винаги, когато потребителят е стартира програмата полето за отметка "Rush Job" да бъде маркирано. Искам винаги когато потребителят натисне бутона "Get Tires" наличните размери на гумите да се визуализират в текстово поле. След като потребителят въведе регистрационния си номер, искам наличните размери на гумите да се попълват в падащия списък.

Разбира се Visual Basic кодът ми позволява да направя всичко това! Задаване на нови стойности за свойствата на контролите е също толкова лесно,както тяхното прочитане. За да направите това, използвайте оператора за присвояване. Спомняте ли си оператора за присвояване, който разгледахме? Този оператор присвоява стойността на една променлива или свойство на друга променлива или свойство. Кодът с оператор за присвояване изглежда като обикновено равенство в математиката.

```
TextBox1.Text = "Code Rules"
```

Първо се изчислява дясната страна на равенството. След това получената стойност се присвоява на лявата страна на равенството.

Полезен съвет

Напомняйте постоянно на учениците, че .Net езиците изпълняват присвояването отдясно наляво.

Важно правило е, че всичко от дясната страна на знака равно не се променя. Стойностите на променливите отдясно се изчисляват, след което се присвояват на променливата отляво. Само променливата

Ето някои примери за това как да задавате нови стойности за свойствата на контролите посредством Visual Basic код. Първият пример задава нова стойност на свойството Text на текстово поле. Това е прекрасен начин да изведете информация за потребителя, вместо да използвате диалогови прозорци. Създайте ново Windows приложение с име Set Properties. Върху формата добавете две текстови полета и един бутон. Използвайте прозореца Properties, за да зададете стойност "Set Text" за свойството Text на бутона. Изчистете текста от двете текстови полета като щракнете двукратно върху свойствата им Text и натиснете клавиша Backspace. Щракнете двукратно върху бутона, з ада добавите код за обработка на събитието Button1_Click. Въведете следния код:

```
TextBox2.Text = TextBox1.Text
```

Изградете и стартирайте приложението. Въведете някакъв текст в първото текстово поле и натиснете бутона "Set Text". Какво се случва? Както виждате текста от първото текстово поле се копира във второто текстово поле. Свойството Text на второто текстово поле приема стойността на свойството Text за първото текстово поле. Visual Basic кодът прочита свойството Text на първото текстово поле и след това го използва, за да зададе стойност за свойството Text на второто текстово поле. Много удобно, нали!

Нека да напишем код, който автоматично маркира едно поле за отметка и автоматично премахва отметката в друго такова, когато програмата се стартира от потребителя. Добавете поле за отметка върху формата и задайте стойност "Rush Job" за свойството му Text като използвате прозореца Property. Добавете второ поле за отметка и задайте стойност "Whenever" за свойството му Text. Щракнете двукратно върху формата (извън двете полета за отметка). Ще се отвори обработчика на събитието Form1_Load. Кодът, който обработва това събитие, се изпълнява когато формата се създава и визуализира на екрана. Въведете следния код:

```
CheckBox1.Checked = True
CheckBox2.Checked = False
```

Изградете и стартирайте приложението. Кое от двете полета за отметка е маркирано? При зареждането на формата свойството Checked на двете полета за отметка се установява автоматично.

Пейджър

Събитието, което се генерира при зареждането на формата, често се използва за инициализация на програмата. При инициализирането се установяват определени стойности преди формата да бъде показана на потребителя. Например, можем да зададем инициализиращи стойности за размера и цвета на формата и да селектираме определени радио-бутони.

Знам, че 14 е най-често използвания размер за гуми. Нека да напишем код, който автоматично да селектира 14 от падащия списък за избор на размер на гума. Първо, добавете падащ списък върху формата. Използвайте прозореца Properties, за да зададете елементите на списъка както следва: 12,13,14,15, и 16. Не забравяйте да използвате клавиша Enter, за да разположите всеки елемент на отделен ред. Щракнете двукратно върху формата (извън падащия списък), за да създадете обработчик на събитието Form1_Load. Въведете следния код:

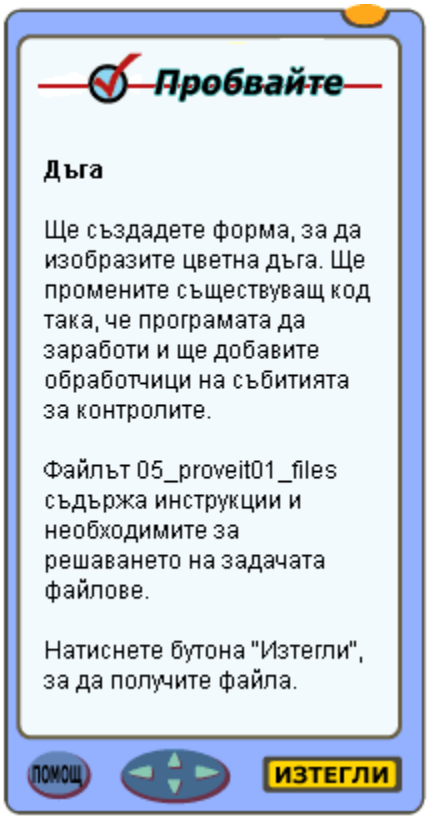
```
ComboBox1.SelectedItem = "14"
```

Изградете и стартирайте приложението. Кой елемент е избран в падащия списък? Ох, сбърках! Най-често използвания размер на гумите е 13. Как ще промените кода, така че в падащия списък да се селектира автоматично 13?

В C# можете да задавате нови стойности на свойствата подобно на Visual Basic. Кодът по-долу показва как можете да промените свойството Checked на поле за отметка посредством код на C#.

```
{
checkBox1.Checked = True;
}
```

05 Пробвайте 01 Дъга

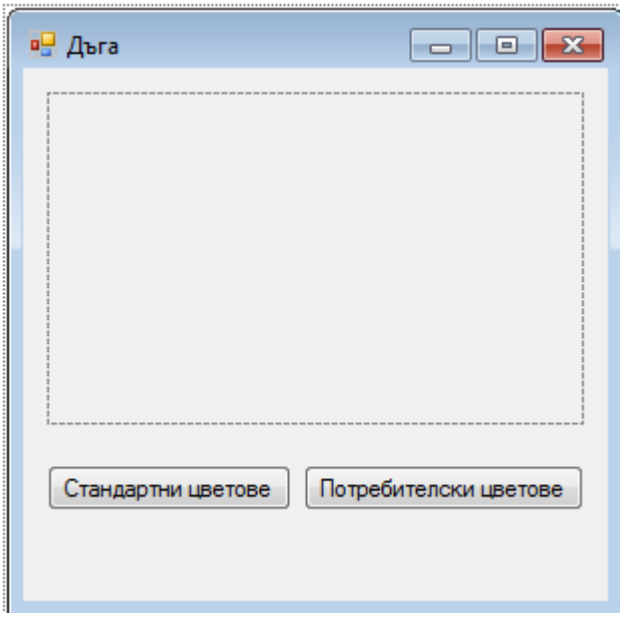


Впечатлени от цветовете на видяна от тях небесна дъга нашите спътници решиха да проверят кой би могъл да я представи с подходяща програма.

Вашата задача е да докажете, че можете да променяте свойствата на обектите посредством истински VB код, а не само като използвате Прозореца за свойства.

Отворете Visual Studio 2010 , изберете командата Open Project от менюто File и намерете файл с име Rainbow.sln. Като алтернативен вариант използвайте My Computer, за да откриете файла Rainbow.sln, след което щракнете двукратно върху него. Средата на Visual Studio ще отвори файла автоматично.

Добавете обекти върху формата като използвате прозореца Toolbox така, че да изглежда както показаната по-долу. Също така променете елементите в прозореца с изходен код както е описано по-долу.



1. Щракнете върху формата, за да установите фокуса върху нея. Променете заглавието й на "Дъга".

2. Добавете контрол за визуализиране на картинка с име PictureBox1.

3. Добавете два бутона с имена Button1 и Button2. Озаглавете първия като "Standard Colors", а втория – като "Custom Colors".

4. Щракнете двукратно върху първия бутон, за да видите съществуващия за него код. Както се вижда за изрисуване на дъгата са предвидени седем цвята. За сега използваният цвят е черен. Ваша задача е да промените кода така, че да използвате стандартните седем цвята: Red, Orange, Yellow, Green, Blue, Indigo, Violet. Като пример заменете кода:

```
MyPen.Color = System.Drawing.Color.Black
```

Със следния код

```
MyPen.Color = System.Drawing.Color.Red
```

Непосредствено след като сте поставили точката след обекта Color ще се появи списък с всички дефинирани в средата на Visual Studio цветове.

5. Щракнете двукратно върху втория бутон, за да видите съществуващият за него код. Вместо черен цвят изберете произволни седем цвята по ваше желание.

Изградете приложението и ако няма грешки изберете опцията Start Debugging от менюто Debug, за да стартирате програмата.

Ако програмата работи правилно, я покажете на учителя си.

Задача с повишена трудност

Ако желаете можете да промените параметрите на функцията DrawArc, за да получите по-интересни резултати.

Полезен съвет

Функцията DrawArc на практика е в елипсовидни граници, определени от правоъгълник. Параметрите определят размера на правоъгълника, където дъгата се вписва. Промяната на тези параметри ще доведе до интересни ефекти.

IntelliSense и точковата нотация

Сега вече знаете как да пишете код на Visual Basic, за да четете и записвате свойствата на формите и контролите върху нея. Мога да се обзаложа какъв ще бъде следващия ви въпрос. Как бихте могли да запомните всички свойства на формите и контролите? Със сигурност не е удобно да превключвате непрекъснато между прозорците, визуализиращи кода и свойствата. Няма ли по-лесен начин?

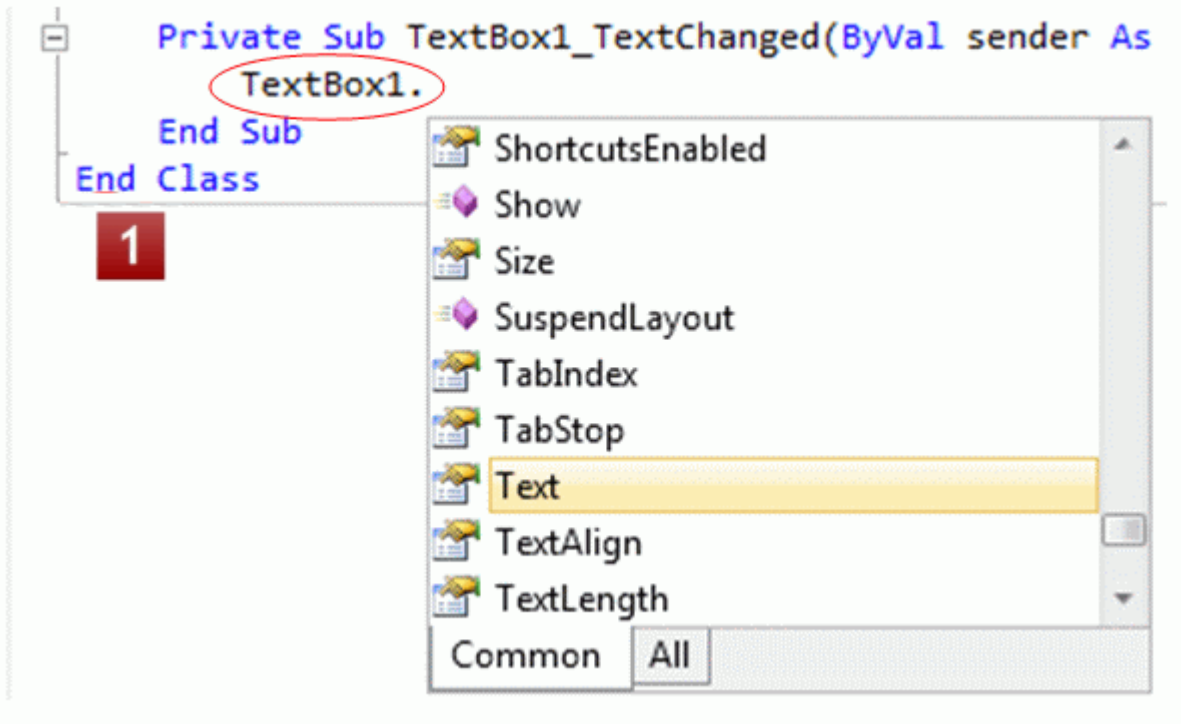
Да, има. Нарича се IntelliSense. Това е помощно средство на Visual Studio. IntelliSense спестява доста време, когато пишете код в прозореца Code, тъй като знае кой контрол използвате и какви свойства има. IntelliSense показва списък със свойства и ви позволява да изберете това, което желаете да използвате. При визуализирането на списъка е маркирано най-често използваното свойство. Такова например е свойството Text на текстовото поле. Използвайте горната и долната стрелка от клавиатурата, за да откриете необходимото ви свойство. Избраното свойство се добавя към кода при натискането на клавиша Tab. Използвайки IntelliSense вие не трябва да помните всички свойства на контролите. Едновременно с това пишете по-малко! Аз използвам това помощно средство през цялото време.

Полезен съвет

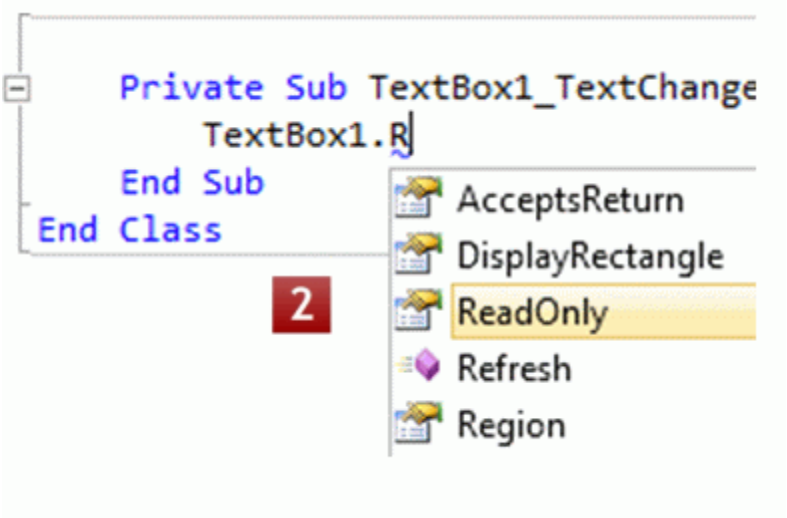
Ако е възможно покажете Intellisense. Поставете точка и изчакайте да се покаже списъка за избор. Ако той не се покаже, то променливата не е декларирана или е написана неправилно.

Пробвайте! Създайте ново Windows приложение с име IntelliSense. Добавете текстово поле и бутон върху формата. Щракнете двукратно върху бутона, за да създадете обработчик на събитието Button1_Click.

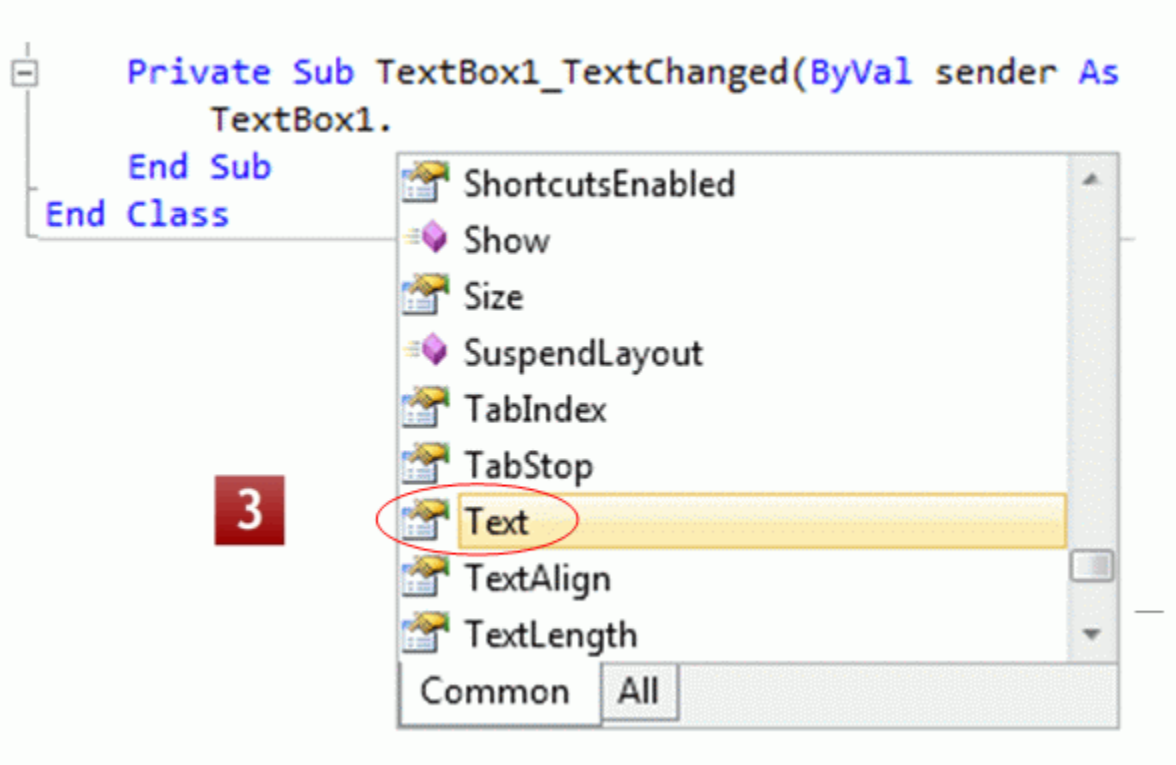
1. Напишете TextBox1 и натиснете клавиша със знака за точка (.). Както виждате IntelliSense показва списък със свойствата на контрола TextBox1. Тъй като контролът TextBox1 е текстово поле, показаният списък е актуален за всички текстови полета, които ще добавите върху формата. Селектирано по подразбиране е най-често използваното свойство за контрола. В нашия случай това е свойството Text.



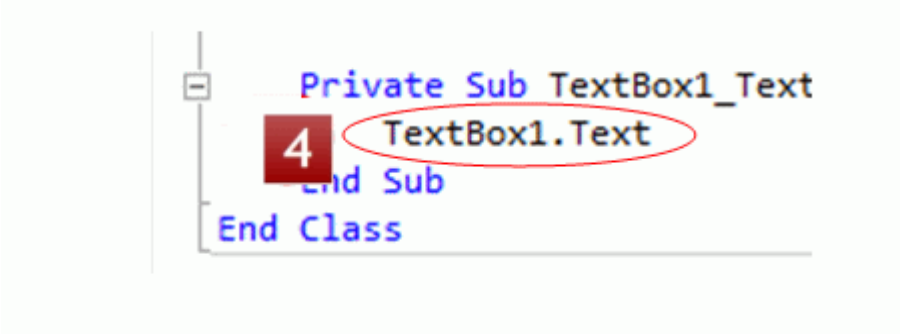
2. Натиснете клавиша с буквата "R". IntelliSense селектира първото свойство в списъка, което започва с буквата "R". Ако продължите да пишете IntelliSense ще открива свойството, което най-много съвпада с въведеното от вас.



3. Използвайте долна стрелка, за да откриете свойството Text. Изберете го.



4. Натиснете клавиша Tab. Вижте какво се случва! Свойството Text се добавя в кода ви след обекта TextBox1. Сега напишете ="Clifford". Кодът трябва да изглежда по следния начин:



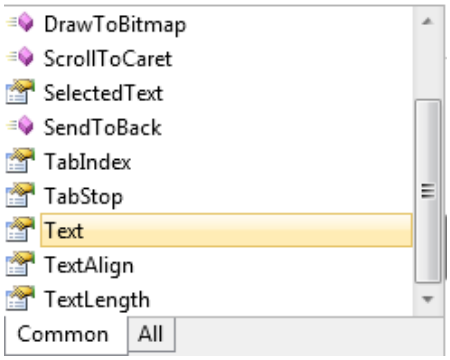
5.

```
TextBox1.Text = "Clifford"
```

Не е ли лесно и бързо? IntelliSense ви помага да ограничите грешките в синтаксиса.

Пейджър

Когато дадено свойство е селектирано в списъка на програмата IntelliSense, се появява прозорец с помощна информация за него. Например този прозорец може да показва дали свойството е достъпно само за четене. Помощната информация показва също и типа стойност, която може да се присвои на свойството, като String (текст) или Integer (цяло число).



Сигурно сте забелязали, че преди всеки елемент от списъка, който IntelliSense визуализира, има икона. Тази икона обозначава свойствата като ръка, показваща списък, а методите като пурпурна кутийка или пурпурен диамант. (Ще ви запозная с методите след малко.) В зависимост от контрола можете да видите и други икони, представящи допълнителни атрибути.

Забелязахте ли, че написахте точка преди IntelliSense списъка да се появи? Това се дължи на факта, че Visual Basic използва тип синтаксис, наречен "Точкова Нотация". След като напишете името на контрола, като TextBox1 например, вие поставяте точка. След това въвеждате и името на свойството. Точката свързва контрола с неговото свойство. Ето базовия синтаксис за референция към свойство на контрол или форма с използване на точкова нотация:

```
FormName.FormProperty()
ControlName.ControlProperty()
```

Вече видяхте много примери, но ето още няколко:

```
TextBox1.Text
CheckBox1.Checked()
Form1.ActiveForm.Height
```

Знаете, че точката свързва контрола с неговото свойство. В последния пример ActiveForm е свойство на формата с име Form1, а Height е свойство на ActiveForm. Поради това са необходими две точки, за да се свържат и двете свойства. IntelliSense знае кога дадено свойство има свое собствено свойство. Ако се съмнявате, поставете друга точка и IntelliSense ще покаже списък с наличните свойства. Ако такъв списък не се покаже, то допълнителни свойства няма и вие трябва да премахнете последната написана точка. Това може да се случи и ако сте сгрешили името на контрола.

IntelliSense е достъпно средство и когато разработвате C# приложения с Visual Studio. То работи по същия начин както при Visual Basic. Само трябва да напишете името на контрола и да поставите точка след него. IntelliSense ще покаже списъка със съществуващите за контрола свойства и методи. Тъй като C# и Visual Basic са базирани на .NET Framework, списъкът със свойствата и методите на всички контроли е еднакъв за всички езици.

Методи

Вероятно сте забелязали, че някои от елементите в списъка, визуализиран IntelliSense, имат икона с пурпурен диамант. Това не са свойства,за методи. Свойствата са характеристики на контролите.Например цвета, броят места и големината на двигателя са характеристики на автомобила. Методите са действия, който контролът може да извършва. Например, действията, които автомобилът може да извършва, са потегляне, завиване на ляво или дясно и спиране. Методите не са достъпни посредством прозореца Properties. Те трябва да се извикват посредством код.

Например, повечето контроли имат методи Hide и Show. Когато извикате метода Hide, контролът не е видим, но все още е на формата. Когато извикате метода Show, контролът става видим. Някои методи ви позволяват да извършвате действия, които обикновено се изпълняват от потребителя. Например, бутонът притежава метод PerformClick, който предизвиква неговото натискане, имитирайки действие от страна на потребителя.

Някои методи изискват допълнителна информация, когато ги извиквате. Тази информация определя начина, по който методът ще се изпълни и резултата, който ще се получи. Тя се задава на метода под формата на т.нар. аргументи. Всеки метод може да изисква един или повече аргумента, някои от които не са задължителни.

Например, методът Show на диалоговия прозорец Messagebox приема като аргумент низ от символи, определящ съобщението, което ще се визуализира. Вие вече използвахте този метод, но вероятно не сте разбрали, че сте извиквали метод с аргумент.

```
MessageBox.Show("Hello World")
```

В кода по-горе текстът "Hello World" е аргумент на метода Show. Аргументът определя какво съобщение ще се визуализира в диалоговия прозорец.

Запомнете, че основната разлика между методите и свойствата е, че методите могат да изпълняват действия. Методите могат да бъдат извиквани посредством код и не са достъпни от прозореца Properties. Някои методи изискват един или повече параметъра (аргументи) или връщат някаква стойност, която можете да използвате в кода си.

Извикване на методи

Синтаксисът за извикване на методи е подобен на този за четене и записване на свойствата. За да свържете контрола с неговия метод, можете да използвате точковата нотация. Някои методи като Button.Hide и Button.Show не приемат аргументи. Други, като MessageBox.Show, трябва да получат аргумент, за да знаят какво да правят. Синтаксисът за извикване на метод е следният:

```
ControlName.MethodName (argument1, argument2, ...)
```

Запомнете, че трябва да подадете аргумент на метода Show, когато използвате диалоговия прозорец MessageBox:

```
MessageBox.Show("Hello World")
```

Както можете да видите, съществена разлика между извикването на метод и прочитането или записването стойност на свойство е, че не се използва знакът за равенство.

Нека да разгледаме някои от любимите ми методи и начина, по който могат да бъдат извикани. Ще ви науча как да скривате и показвате контролите върху формата, да изтривате текста от текстовите полета и да задавате местоположението на курсора. Като начало създайте ново Windows приложение с име Methods. Добавете върху формата два бутона и едно текстово поле. Озаглавете единия бутон като "Show", а другият като "Hide". В обработчика на събитието Button1_Click добавете следния код.

```
TextBox1.Show()
```

В обработчика на събитието Button2_Click добавете следния код.

```
TextBox1.Hide()
```

Изградете и стартирайте приложението. Натиснете бутона "Show" и бутона "Hide". Всеки път, когато натиснете бутон, извиквате метод на контрола Button.

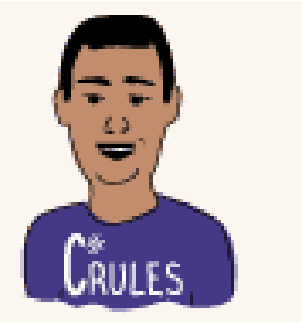
Добавете трети бутон. Озаглавете го като "Reset". В обработчика на събитието Button3_Click добавете следния код.

```
   TextBox1.ResetText()
TextBox1.Focus()
```

Методът ResetText изчиства съдържанието на свойството Text на текстовото поле.
Методът Focus поставя курсора на мишката в текстовото поле.
Изградете и стартирайте приложението. Въведете някакъв текст в текстовото поле и натиснете бутона "Reset". Текстът в текстовото поле се изтрива и курсорът автоматично се позиционира в него, подготвяйки го за ново въвеждане на текст.

Полезен съвет

Алтернативен вариант на `TextBox1.ResetText()` е да присвоите празен низ `TextBox1.Text = ""` Между кавичките няма нищо.



Както можете да предположите, можете да извиквате методи и в C# подобно на Visual Basic. Ето C# код, който изчиства текста от текстово поле и позиционира курсова на мишката в него.

```
{
textBox1.ResetText();
textBox1.Focus();
}
```

Пристигане в Кайро

Ето ни в покрайнините на Кайро, Египет. Научихме доста по пътя. Вече можете да четете и задавате стойности за свойствата на контролите и да извиквате техните методи посредством Visual Basic код. Както видяхте по-голямата част от кода се пише в обработчиците на събитията. За да създадете такъв обработчик, трябва само да щракнете двукратно върху контрола. Прозорецът Code ще се отвори автоматично и обработчика на събитието ще бъде готов за въвеждане на код. За да прочетете или запишете свойство, или извикате метод, трябва да напишете името на контрола, последвано то точка. Помощното средство IntelliSense ще визуализира списък със съществуващите за контрола свойства и методи. Изберете свойството или метода, който ви е необходим, и натиснете клавиша Tab, за да го добавите към кода си.

Имаме достатъчно време, за да изпълним една или две задачи и да направим теста. Готови ли сте?

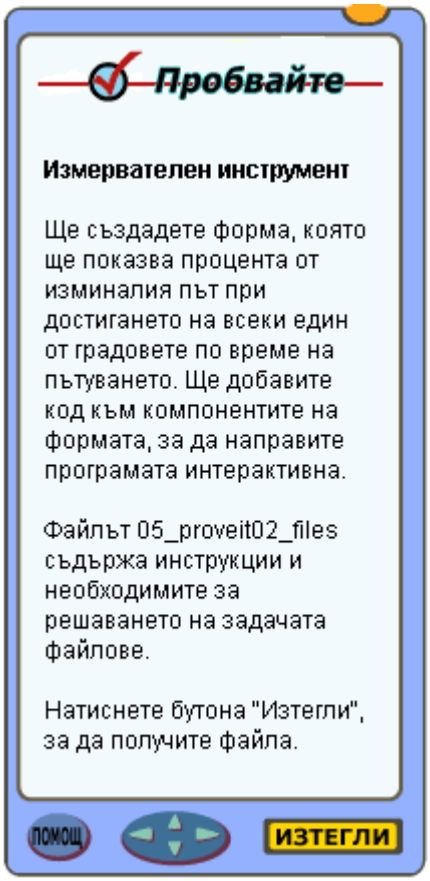
05 Пробвайте 02 Измервателен инструмент

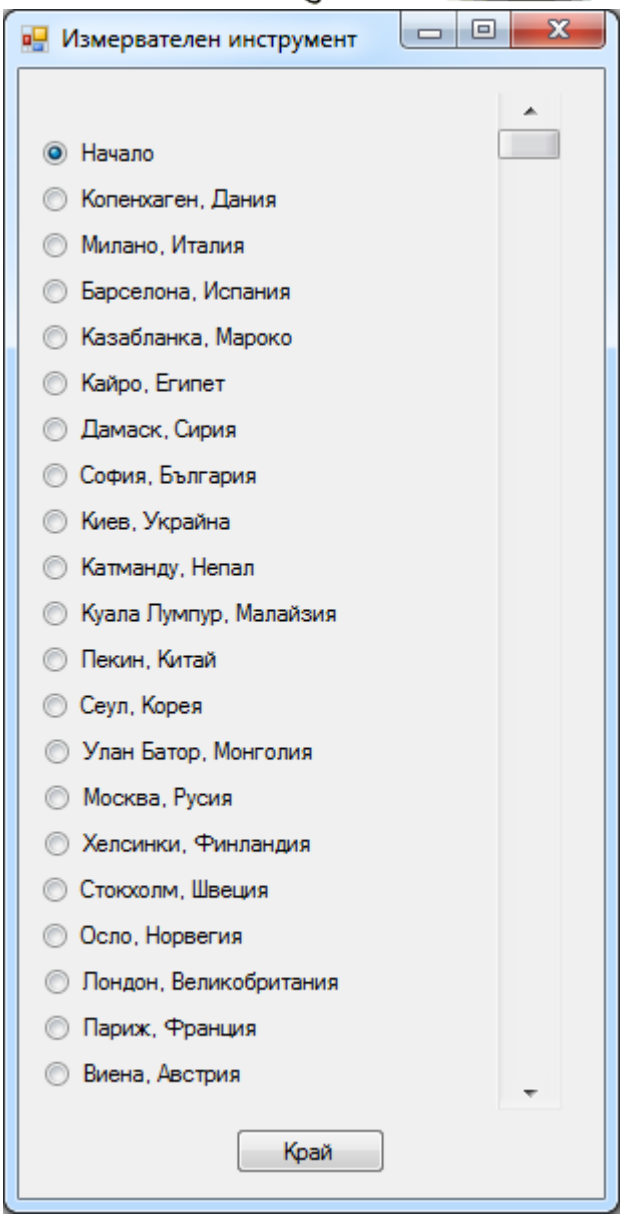
Картата, проследяваща пътуването ни, мотивира Ники да постави като задача създаването на програма, която да показва процента от изминалия път при достигането на всеки един от градовете. Избрахме нашата задача да включва градовете до пристигането в Сеул, Корея.

Вашата задача е да докажете, че можете да променяте свойствата на обектите посредством истински VB код, а не само като използвате прозореца за свойства.

Отворете Visual Studio, изберете командата New Project от менюто File и създайте проект с име Trip Meter.

Добавете обекти върху формата като използвате прозореца Toolbox така, че да изглежда както показаната по-долу. Също така променете елементите в прозореца с изходен код както е описано по-долу.





1. Щракнете върху формата, за да установите фокуса върху нея. Променете заглавието ѝ на "Измервателен инструмент".

2. Добавете 21 радио-бутона. Задайте им ширина от 150 пиксела. Можете да създадете един радио-бутон, след което да го размножите.

3. Подравнете елементите вертикално и ги поставете на равни разстояния един от друг. Знаете ли, че съществуват бутони от лентата с инструменти, които могат да направят това автоматично? Позиционирайте курсора на мишката върху лентата с инструменти, намерете подходящия бутон и го натиснете.

4. Променете текста на радио-бутоните както следва:

Начало:
Копенхаген, Дания
Милано, Италия
Барселона, Испания
Казабланка, Мароко
Кайро, Египет
Дамаск, Сирия
София, България
Киев, Украйна
Катманду, Непал
Куала Лумпур, Малайзия
Пекин, Китай
Сеул, Корея
Улан Батор, Монголия
Москва, Русия,
Хелсинки, Финландия
Стокхолм, Швеция
Осло, Норвегия
Лондон, Беликобритания
Париж, Франция
Виена, Австрия

5. Добавете бутон със заглавие Край и име End. Щракнете двукратно върху него, за да видите асоциирания към него код. Напишете думата End в процедурата, обработваща събитието Button1_Click.

6. Поставете вертикална лента за скролиране върху формата. По подразбиране лентата за скролиране приема стойности от 0 до 100, поради което индикаторът й е голям и се премества малко. За да промените това, задайте стойност 1500 на нейното свойството Maximum.

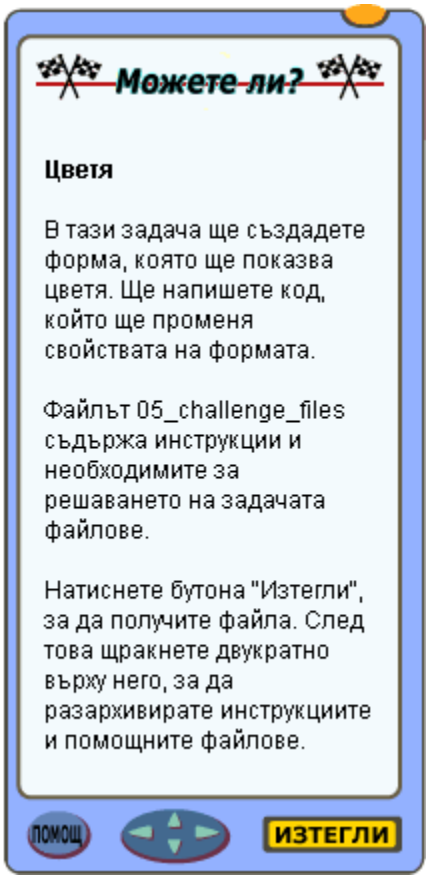
7. Сега вече можете да пишете код. Щракнете двукратно върху първия радио-бутон. В процедурата, обработваща събитието RadioButton1_Click поставете следния ред.

```
VScrollBar1.Value = 0
```

Този код ще премести индикатора на лентата за скролиране на позиция 0 в диапазона 0 – 1500.
За всеки от останалите радио-бутони, използвайте подобен код, като всеки път увеличавате позицията на индикатора с 75. последната стойност трябва да бъде 1500.

Изградете приложението и ако няма грешки изберете опцията Start Debugging от менюто Debug, за да стартирате програмата.
Ако програмата работи правилно, я покажете на учителя си.

05 Можете ли? Цветя



Цветята създават приятно усещане. Ето защо мисля, че трябва да създадем програма за тях.

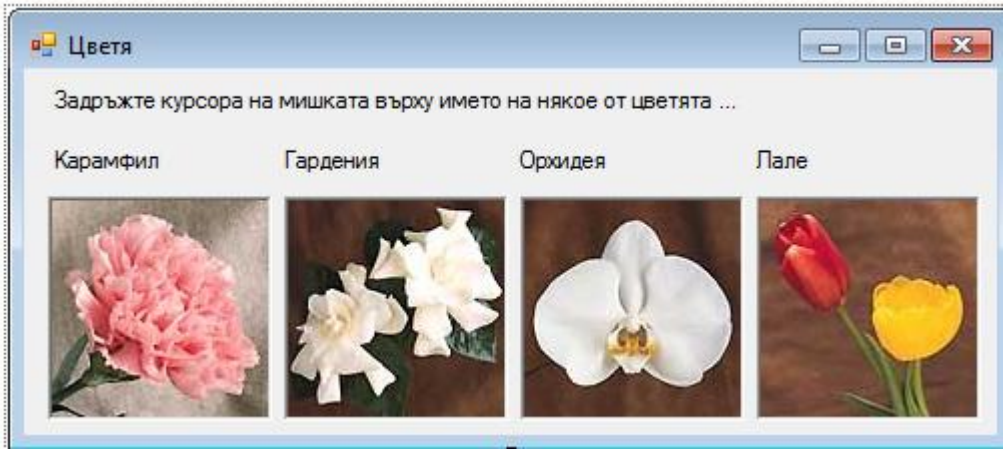
Създайте подобна на показаната по-долу форма:

1. Задайте стойност False на свойството Visible за компонентите, показващи картинки.

2. Задайте стойност Fixed3D на свойството BorderStyle за компонентите, показващи картинки.

3. Задайте стойност AutoSize на свойството SizeMode за компонентите, показващи картинки.

4. Задайте стойност на свойството Image така, че всеки от компонентите да визуализира едно от изображенията, намиращи се в директорията на проекта.



5. Отворете страницата с изходен код. Точно над нея има два падащи списъка. От левия изберете label1, а от десния MouseHover.

6. Използвайте събитието Label1.Mousehover, за да установите стойност True на свойството Visible за компонента PictureBox1. Установете стойност False за същото свойство на останалите три компонента.

7. Направете същите промени за останалите три етикета.

8. В резултат, когато мишката премине през някой от етикетите, програмата трябва да показва само една картинка.

Ако програмата работи правилно, я покажете на вашия учител.

Полезен съвет

Някои студенти ще обработят събитието MouseHover за полето с картинка. Инструкциите изискват да се обработи същото събитие, но за етикета, който показва

Продължение / Обобщение

Осигурете променяна в цвета на формата при промяна на избраната картинка.

Тъй като пристигнахме в Кайро, Египет, трябва да направите друг тест. Късмет!