
МАРШРУТНА КАРТА
Дестинация
София, България
Пейджър
Теми по пътя за София:
София, България
Операции
Аритметични оператори
Оператори за работа с низ
Операторът Not
Използване на оператори в изрази
Операторите в действие
Дебъгване на код
Пристигане в София
1. Копенхаген, Д
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

Оператори

София, България

Готови сме да продължим. Следващата ни дестинация е София, България! Клиф вече ви показа как да използвате променливите и оператора за присвояване, така че сега ще разгледаме някои основни програмни средства. По пътя за София ще ви покажа как да използвате операторите. Веднъж след като се научите да използвате операторите във Visual Basic, вие ще можете да ги използвате и във всеки друг програмен език. Работата с операторите е лесна! Когато наближим София, ще ви покажа някои средства за дебъгване, които са част от Visual Studio. Те ще ви помогнат да избегнете грешките в кода си, така че той ще се компилира всеки път!

Оператори

И така, след като операторите са толкова лесни, защо до сега не сте чували за тях? Е, шансовете са на ваша страна! Примери за оператори са събирането, изваждането, умножението и делението. Тях сте използвали още в началното си училище, но не сте ги наричали така. В програмирането има и други оператори, които извършват математически изчисления и различни операции над низови стойности.
Обикновено операторите работят с две стойности, въпреки че някои от тях могат да използват и една стойност. Стойностите се наричат операнди. Ето базовия синтаксис за използване на операторите:
Операнд1 Оператор Операнд2
Например

3 + 4

В израза 3 + 4 има два операнда (3 и 4), свързани посредством един оператор (+). Операторите извършват операция (в случая събиране) над два операнда (3 и 4).
Използването на операторите е лесно и забавно! С тях можете да създавате сложни формули. Нека да започнем с аритметичните оператори – събиране, изваждане, умножение и деление. След това ще ви покажа как да обединявате низове и как да използвате оператора NOT.

Аритметични оператори

Математически правила! Обичам математиката! Харесва ми да решавам математически проблеми по цял ден, но понякога се уморявам и започвам да правя грешки. Компютърните програми се справят чудесно с математическите проблеми. Те не правят допълнителни грешки и никога не се изморяват, извършвайки една и съща операция многократно. Не звучи ли това прекалено хубаво, за да е истина? Е, има малък проблем. Въпреки, че могат да извършват неограничен брой изчисления, програмите не могат да структурират решението. Това е задача на програмиста.

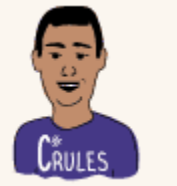
Ето защо е толкова важно да разберете как да използвате аритметичните оператори. Вие трябва да кажете на програмата какъв математически проблем трябва да реши. Тъй като компютрите са много подходящи за извършване на математически изчисления, всички езици за програмиране поддържат стандартно множество от аритметични оператори и използват почти еднакви символи за тяхното представяне в кода. Не е ли много лесно?

Ето списък с най-често срещаните аритметични оператори и символите, с които те се представят във всекидневието и кода. Забележете, че само един от символите е различен. В кода оператора за умножение се представя със символа *, а във всекидневието – със символа x.

Оператор	Символ във всекидневието	Символ в кода
Събиране	+	+
Изваждане	-	-
Умножение	x	*
Деление	/	/

Да разгледаме няколко примера с аритметични оператори.

```
3 + 4
18 - 2
33 * 3
66 / 6
```



C# предоставя същите аритметични оператори както Visual Basic. Използват се +, -, * и / за събиране, изваждане, умножение и деление.

Оператори за работа с низ

Не всички програми, които пишете извършват изчисления. Понякога се налага програмата да работи с низове от символи. Знаете ли, че във Visual Basic и другите езици за програмиране можете да събирате низове? В действителност това не е обичайното събиране на числа. Събирането на низове се нарича конкатенация и реализира обединяване на два низа в общ низ. Във Visual Basic оператора за конкатенация е амперсанд (&). Можете да го използвате както оператора за събиране, но с низови вместо с числови стойности.
Ето и няколко примера:

"Bill" & "Mike
"1" & "2"

Пейджър

Когато сливате два символни низа с оператора за конкатенация, Visual Basic. NET не добавя автоматично празни разстояния. Тяхното добавяне трябва да се укаже в кода. Можете да използвате един от следните два методи:

```
"Bill" & " " & "Mike"
```

или

```
"Bill " & "Mike"
```

В C# за конкатенация на низове не се използва символът амперсанд (&). Вместо него се използва знакът плюс (+). Разгледайте следния код:

```
fullName = "Bill " + "Bob";
```

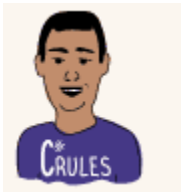
Операторът Not

Операторът Not е специален оператор, който се изпълнява над един операнд. Операндът се интерпретира като булева стойност (True или False). Операторът Not променя стойност False на True и стойност True на False. Разгледайте примерите.

```
Not (True)
Not (Not (True))
```

Не съм много сигурен защо бихте използвали последния пример, но както и да е!

Хей, в C# вместо Not се използва удивителен знак (!). Прегледайте следния C# код.




```
textBox3.Visible = !(textBox1.Visible);
```

Използване на оператори в изрази

И така как се използват операторите в кода? В повечето случаи ще ги използвате в изразите с оператора за присвояване. Припомнете си, че тези изрази изглеждат като равенства в математиката. Операторите и операндите се поставят вдясно от знака равно. Дясната страна на равенството се изчислява, след което получената стойност се присвоява на лявата страна на равенството. Резултатът от изчислението и променливата вляво от знака равно трябва да са от един и същ тип. В противен случай кодът няма да се компилира. Ето няколко примера, използващи аритметични оператори и оператор за конкатенация в изрази.

```
Dim FormWidth as Integer
FormWidth = 200 + 300

Dim FormHeight as Integer
FormHeight = 1000 / 2

Dim WinnebagoName as String
WinnebagoName = "Code" & "Bus"
```

Операторите в действие

Примерите, които ви дадох са елементарни, защото във всеки израз имаше по един оператор и два операнда. В действителност ще ви се наложи да дефинирате формули и да извършвате сложни изчисления, което ще изисква множество оператори и операнди. За щастие, в повечето езици за програмиране можете да обединявате в един израз толкова оператори и операнди, колкото с ви необходими. Заедно с това можете да използвате и променливи като операнди. Точно така! Операторите също така работят и с променливи стига те да са от подходящ тип. Нещо повече можете да присвоите стойност на променлива, която участва в самото изчисление. Нека да разгледаме няколко по-сложни примера. Първият пример изчислява обиколката на автомобилна гума.

```
Dim TireCircum As Single
Dim TireDiam As Integer
Dim PiValue As Single
PiValue = 3.14159
TireDiam = 18
TireCircum = TireDiam * PiValue
```

Следващият пример изчислява средната стойност на бензина за Май и Април. Забележете, че използвам скоби, за да съм сигурен, че операцията събиране ще се извърши преди операцията деление.

```
Dim MarchCost As Single = 123.66
Dim AprilCost As Single = 231.45
Dim MarchGallons As Single = 87.4
Dim AprilGallons As Single = 157.2
Dim CostPerGallon As Single
CostPerGallon = (MarchCost + AprilCost) /
(MarchGallons+ AprilGallons)
```

Пейджър

Можете да използвате скоби в кода си, както когато решавате математически задачи. Когато поставите нещо в скоби, то се изпълнява първо. Например,

(5+7)/(1+5)=2

Ето, (5+7) се изпълнява първо, (1+5) се изпълнява второ, а след това се изпълнява операцията деление. Отговорът е 2.

Без скоби ще получим различен резултат:

5+7/1+5 =17

Използвайте скоби в кода, за да укажете на програмата реда, в който трябва да се изпълняват пресмятанията.

Оператори за сравнение

Спомняте ли си, че можете да присвоите на променлива резултата от изчисление, в което тя също участва.? Ще попитате как става това? Разгледайте следния пример и се опитайте да предположите какво ще се покаже в диалоговия прозорец.

```
Dim MileCounter as Integer
MileCounter = 100
MileCounter = MileCounter + 200
MileCounter = MileCounter + 400
MsgBox (MileCounter)
```

Диалоговият прозорец ще визуализира числото 700, което е стойност на променливата MileCounter. Да проследим как става това. Първият ред от кода декларира променливата MileCounter, а вторият ред я инициализира със стойност 100. Третият ред взима текущата стойност на променливата и добавя към нея 200, така че новата й стойност става 300. Запомнете, че дясната страна на равенството се изчислява винаги първа. След това лявата страна (MileCounter) приема изчислената стойност (300). Четвъртият ред взима текущата стойност на променливата MileCounter и добавя към нея 400, така че новата й стойност става 700. Накрая стойността на променливата се визуализира в диалоговия прозорец MessageBox. Лявата страна на равенството не се интересува от това как се получава резултата от изчислението в дясната страна. Достатъчно е да се използват съвместими типове.

Задаването на стойност на променлива от израз, в който участва самата променлива, е често използвана практика в програмирането с езиците от Visual Studio. Така се спестяват действията, свързани с декларирането на друга променлива за временно съхраняване на резултата от изчислението. Както ще видите от примерите, тази техника е подходяща за извършване на серийни изчисления. Знаете ли, че най-често използвания ред от код вероятно е променлива = променлива + 1? Той се използва за преброяване на итерациите при изпълнението на цикли. По нататък в нашето пътешествие ще ви разкажа и за циклите.

Операторът Not и операторът за конкатенация в действие

Сега ще ви покажа един пример с оператора за конкатенация и след това още един пример с оператора Not. Първият пример обединява стойностите на две текстови променливи и визуализира резултата в текстово поле.

```
Dim FirstName As String
Dim LastName As String
FirstName = "Bob"
LastName = "Marley"
TextBox1.Text = FirstName & " " & LastName
```

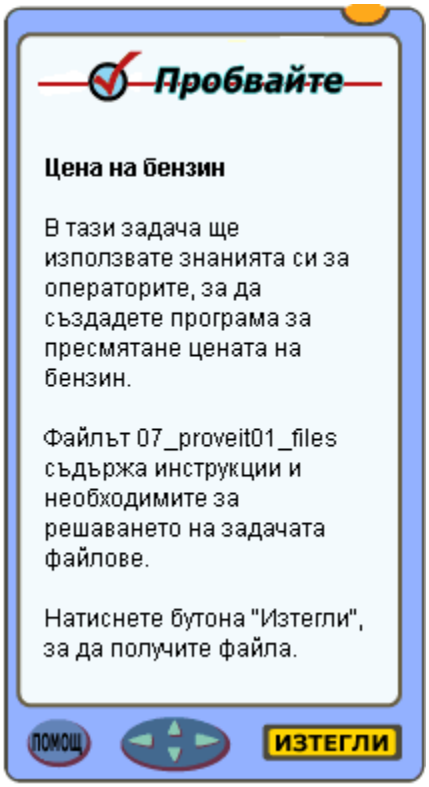
Забелязахте ли как указах на Visual Basic да постави интервал между стойностите на променливите FirstName и LastName? Операторът за конкатенация не вмъква интервали автоматично.

Последният ми пример показва как да използвате оператора Not, за да скриете определен текст, когато друг текст е показан. Запомнете, че операторът Not се използва само с типа Boolean.

```
TextBox1.Visible = True
TextBox2.Visible = Not (TextBox1.Visible)
```

Не съм сигурен защо бихте използвали последния ред с код!

07 Пробвайте 01 Цена на бензин



Можете ли да изчислите стойността на бензина, необходим за следващия етап от пътуването? Докажете го!

1. Създайте форма със свой собствен дизайн.

   Включете следните елементи:
   - Подходящо име на формата.
   - Четири текстови полета, за да покажете следните стойности:
     - Разстояние до следващата дестинация в мили
     - Разход на горивото в галони за една миля
     - Цена на горивото за един галон
     - Обща стойност на изразходваното гориво
   - Четири етикета за обозначаване на текстовите полета.
   - Бутон за пресмятане.

2. След като потребителя въведе разстояние, разход на гориво за една миля и цена на гориво за един галон, програмата ще пресметне стойността на изразходваното гориво, необходимо за достигане на следващата дестинация.

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

Ако програмата работи правилно, я покажете на вашия учител.

Продължение / Обобщение

Форматирайте крайната цена, която трябва да се заплати за изразходваното гориво.

Проучете следния код:

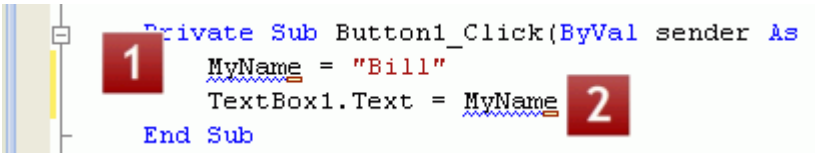
```
X = 123.45
X = FormatCurrency(X)
```

Дебъгване на код

Свършихте чудесна работа! Вече написахте доста код и реализирахте няколко проекти. Може би не всичките ви проекти се компилираха от първия път, но вероятно сте си изградили начин да откривате и отстранявате грешките в кода. Сега ще ви покажа няколко техники, които ще ви помогнат да откривате грешките в кода по-бързо. Процесът на откриване на грешките се нарича дебъгване. Дебъгването е необходима стъпка в създаването на работоспособна програма, която дава правилни резултати. Първо ще ви покажа как Visual Studio ви помага да дебъгвате кода си още докато го пишете.

1. Създайте ново Windows приложение с име DebugView. Добавете към формата едно текстово поле и един бутон. Щракнете двукратно върху бутона, за да създадете обработчик на събитието Button1_Click, и добавете следния код.

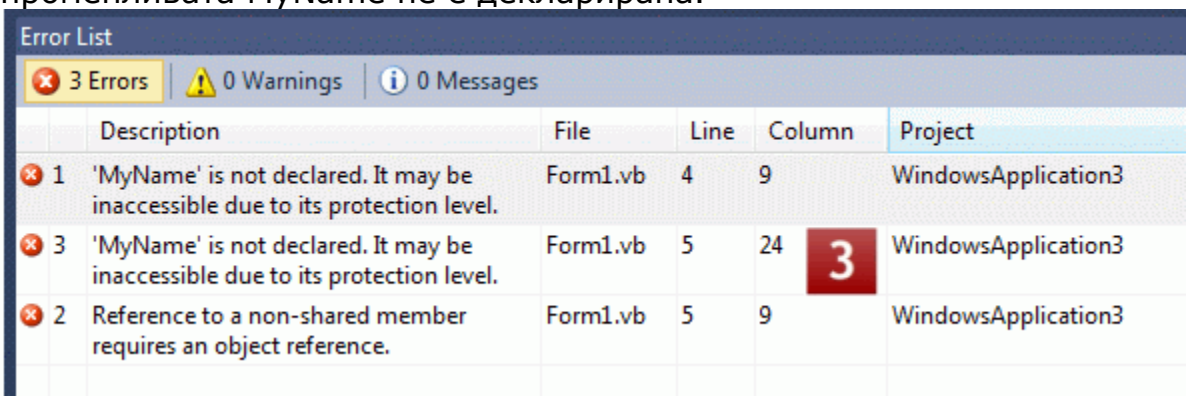
```
MyName = "Bill"
  TextBox1.Text = MyName
```



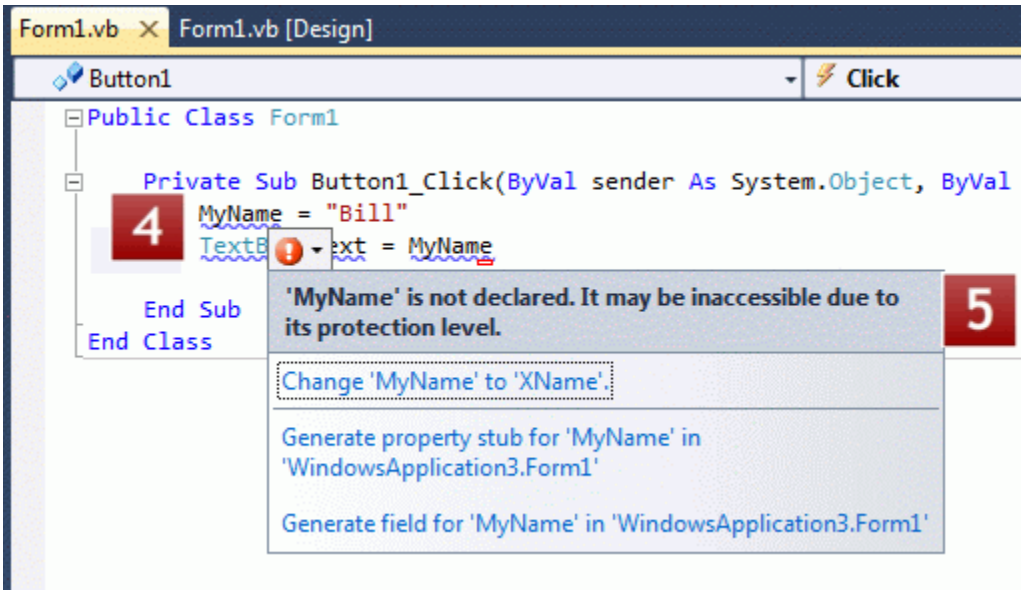
2. Забележете, че още преди да изградите проекта Visual средата подчертава променливата MyName със синя вълнообразна линия и в двата реда от кода. И какво от това? Е, Visual Studio е достатъчно интелигентна среда, за да открие още преди изграждането на проекта, че не сте декларирали променливата MyName. Така преди компилация средата ви предупреждава, че нещо в кода ви не е наред.

3. Изградете и стартирайте проекта. Прозорецът Output ви показва че изграждането е неуспешно. Списъкът с грешки Error List визуализира причините за това. Той притежава два реда, които ви уведомяват, че

променливата MyName не е декларирана.



4. Щракнете двукратно върху първия ред на списъка с грешки Error List –"Name 'MyName' is not declared". Каретката се позиционира в мястото на грешката.



5. Позиционирайте курсора на мишката в края на променливата MyName. Забележете, че се появява подсказващ прозорец с типа на грешката и бутон с удивителен знак, при натискането на който средата предлага възможни решения на грешката. В обработчика на събитието Button1_Click добавете следния код.

```
Dim MyName As String
```

Сините вълнообразни линии изчезват.

6. Изградете и стартирайте проекта. Той се компилира без грешки. Вие успешно дебъгнахте своята програма!

Стъпково изпълнение на кода

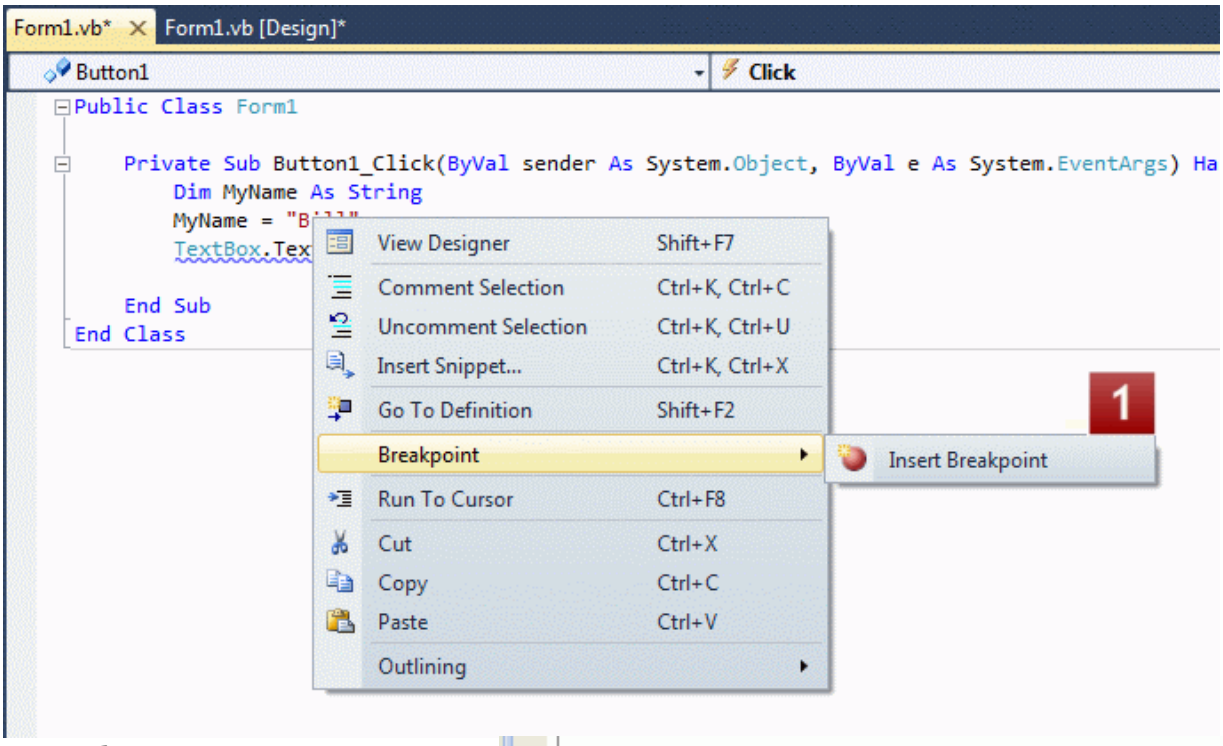
Понякога грешките в кода ви не могат да бъдат открити от Visual Studio. Програмата ви се компилира без грешки, но резултатът или изходът от нея не са точни. В този случай грешките не са синтактични, а се дължат на неправилно структуриране или изпълнение на програмата. Съществуват грешки, които се допускат в дизайн режим, но Visual Studio 2010 не ги регистрира като грешки, тъй като синтаксисът е правилен.

В подобни случаи е удачно да кодът да се проследи стъпка по стъпка, за да се види пътя на неговото изпълнение. Обикновено в кода се поставя "точка на прекъсване", след която изпълнението продължава на стъпки. Точката на прекъсване действа като знак стоп – изпълнението на програмата спира до реда, който е маркиран. За да продължите изпълнението по нататък, трябва да използвате клавиша F11, с който можете да се придвижвате по редовете на кода. Нека ви покажа как става това.

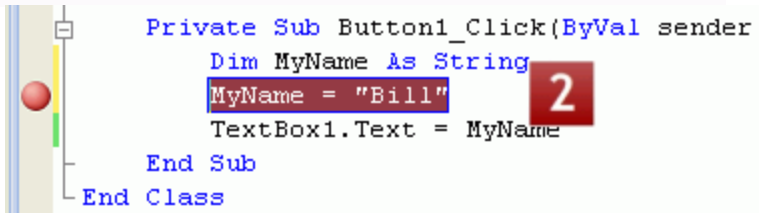
1. В обработчика на събитието Button1_Click открийте следния ред:

```
MyName = "Bill"
```

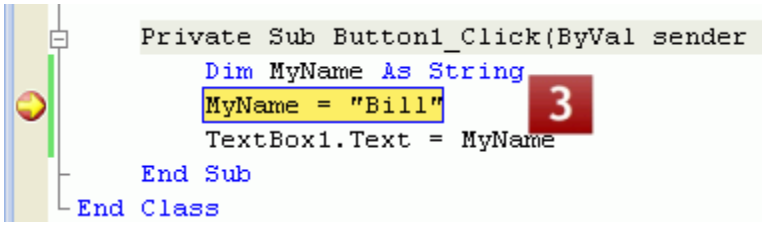
Поставете курсора върху реда. Щракнете с десния бутон на мишката и от появилото се контекстно меню изберете командата "Insert Breakpoint" от менюто "Breakpoint".



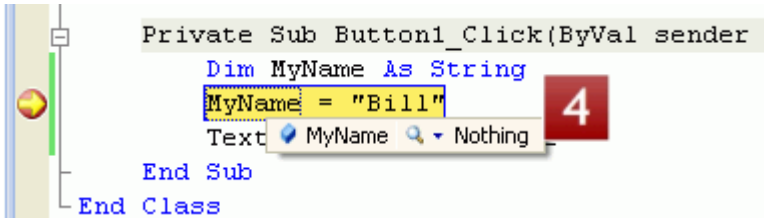
2. Забележете, че редът от кода се маркира с кафяв цвят, а вляво от него се появява червена точка. Изградете и стартирайте програмата. Когато формата се появи на екрана, натиснете бутона.



3. Кодът спира изпълнението си до реда, който сте обозначили с точка на прекъсване (MyName = "Bill"). Този ред все още не е изпълнен. Той се маркира с жълт цвят, а в червената точка вдясно от него се появява жълта стрелка.

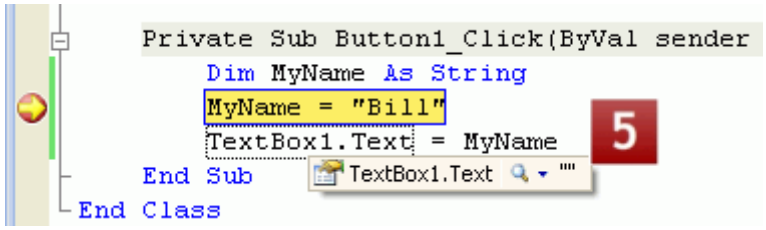


4. Преместете курсора върху променливата MyName. Ще се появи прозорче със стойността на променливата MyName. Променливата няма конкретна стойност



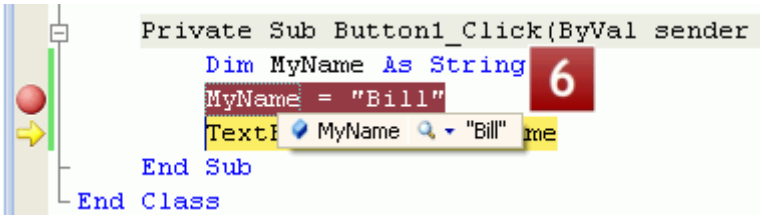
(MyName = Nothing), защото е декларирана, но все още няма присвоена стойност.

5. Преместете курсора върху свойството Text на контрола TextBox1. Ще се появи прозорче със стойността на свойството Text (TextBox1.Text = “”).

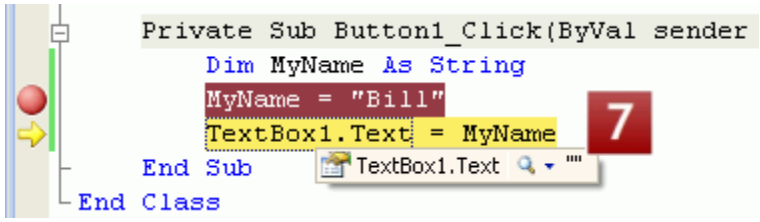


Натиснете клавиша F11. Текущият ред (MyName = "Bill") от кода се изпълнява, след което в жълто се маркира следващия ред.

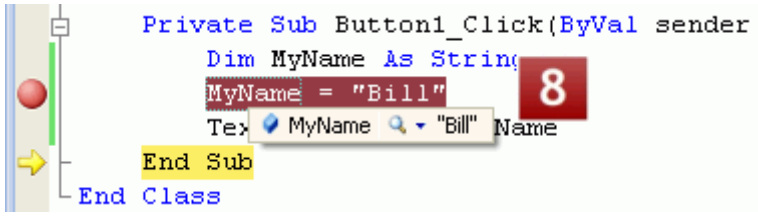
6. Преместете курсора върху променливата MyName. Тъй като променливата вече има зададена стойност, ще се появи прозорче, което показва MyName = “Bill”, т.е. стойността на променливата е “Bill”.



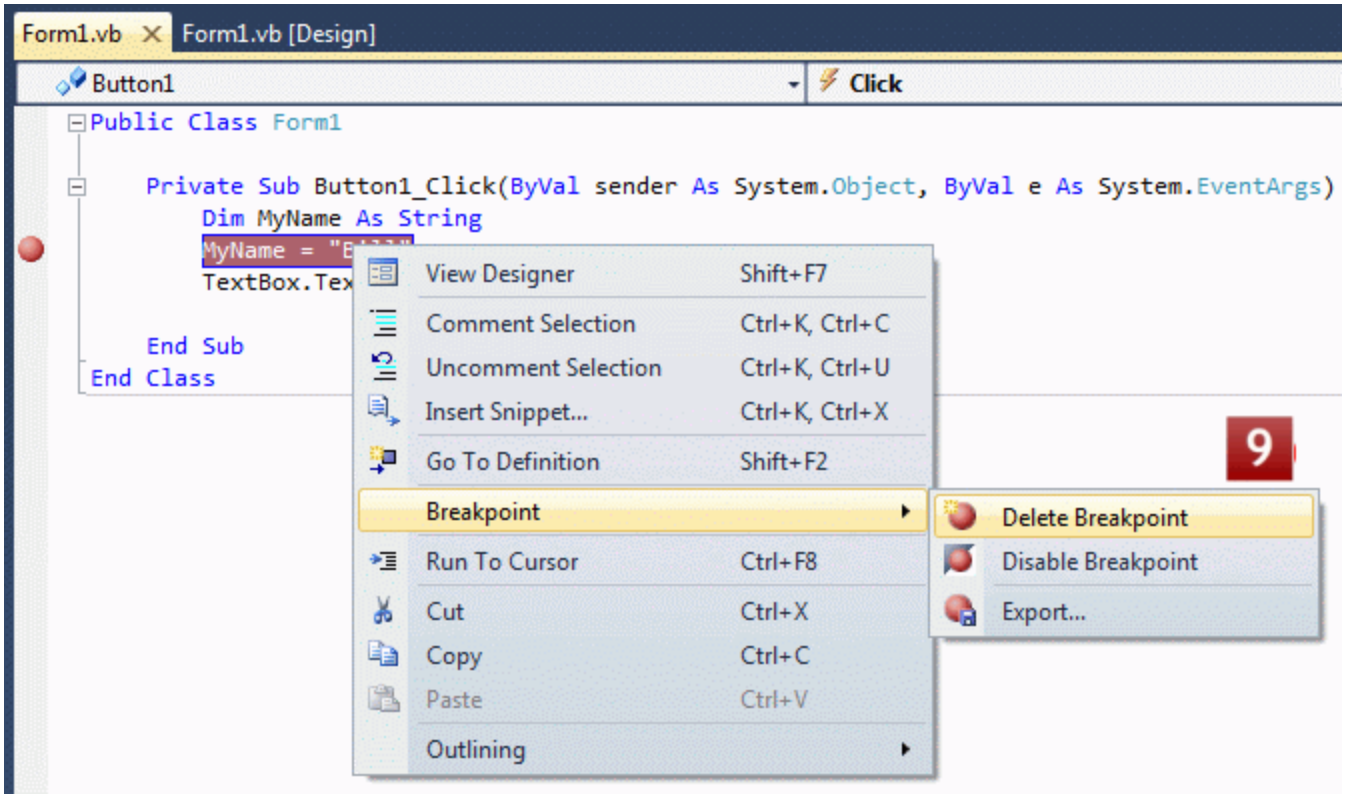
7. Преместете курсора върху свойството Text на контрола TextBox1. Прозорчето със стойността на свойството ще показва Text=””, т.е. стойността на свойството Text не е променена. Натиснете клавиша F11 отново. Текущият ред (`TextBox1.Text = MyName`) от кода се изпълнява, след което в жълто се маркира следващия ред.



8. Преместете курсора върху променливата MyName. Прозорчето със стойността на променливата ще показва MyName = "Bill", тъй като променливата не е променена.

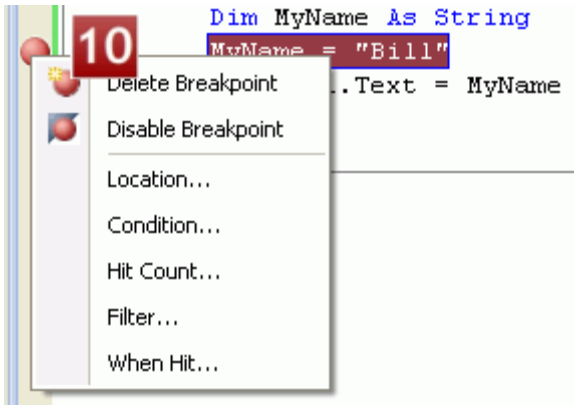


9. Преместете курсора върху свойството Text на контрола TextBox1. Прозорчето със стойността на свойството ще показва TextBox1.Text = "Bill", защото свойството вече е получило стойност от променливата MyName. Натиснете клавиша F11. Преди да завърши изпълнението на кода в обработчика на събитието Button1_Click, формата ще се покаже още веднъж. Щракнете върху знака "X" в горния й десен ъгъл, за да я



затворите.

10. В прозореца с кода щракнете върху реда с точката на прекъсване. Натиснете десния бутон на мишката и от появилото се контекстно меню изберете командата Delete Breakpoint.



Вече завършихте дебъгването. Изпълнението на кода стъпка по стъпка е много удобно. Така можете да проследите пътя, който вашата програма

следва при изпълнението си. Понякога е възможно да откриете, че случайно програмата ви следва грешен път или че използвате неподходяща променлива в конкретен израз. Проследяването на кода ви показва мястото на погрешната стъпка. То ви дава възможност да преглеждате стойностите на променливите и свойствата като премествате курсора на мишката върху тях. Стойностите се обновяват след изпълнението на всеки един ред от кода.

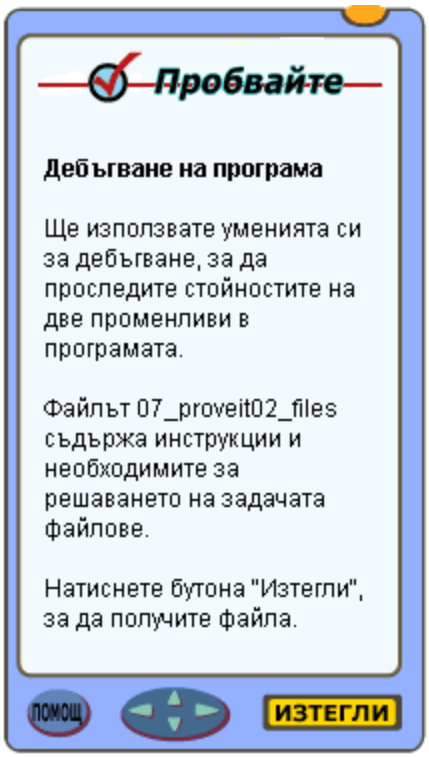
Колкото кодът ви е по-сложен, толкова дебъгването става по-необходимо. Сложният код се дебъгва по-трудно. Скоро ще научите как да управлявате последователността, в която се изпълнява кода, като използвате оператора "if" и операторите за цикъл. Средствата за дебъгване са особено полезни, когато програмата ви изпълнява такива оператори.

Пристигане в София

Ау! Това беше страхотно пътуване! Разговаряхме за толкова много неща. Сега вече трябва да се чувствате сигурни, когато използвате оператори за извършване на математически операции и сливане на символни низове. Дори използвахте Visual Studio средствата за дебъгване, за да отстранявате проблемите в кода си. Мисля, че задачите, които трябва да решите, ще ви се сторят по-забавни и по-предизвикателни. Пробвайте да решите тези, които получих на джобния си компютър. Запазете настроението си докато правите теста, след което Клиф ще ни закара до следващата дестинация.

07 Пробвайте 02 Дебъгване на програма

Нека да проверим уменията ви да дебъгвате програми. Написах програма, която можете да получите от шаблон 02 на секция 07. Това е завършена програма, поради което няма да ви се налага да пишете код. Просто отворете файла Debug Me.sln.



Вашата задача е да дадете отговор на следните три въпроса в текстов файл, в MS Word документ или в стара тетрадка за домашна работа.

1. Каква стойност попълнихте в текстовото поле на формата?

2. Каква е последната стойност на променливата AnswerOne?

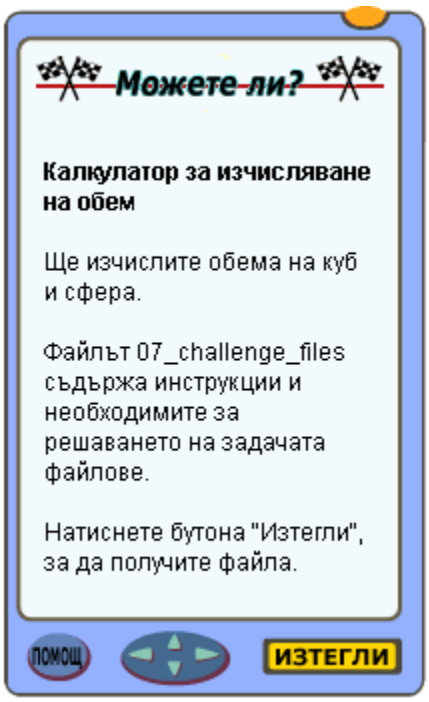
3. Каква е последната стойност на променливата AnswerTwo?

Кодът на програмата съзнателно е структуриран объркващо и сложно така, че отговорите да не са ясни на пръв поглед. Всичко което трябва да направите е да поставите точка за прекъсване изпълнението на програмата в лявото вертикално поле на прозореца с кода, след което да продължите изпълнението на стъпки. За да проверите текущата стойност на определена променлива след като реда, в който се намира е изпълнен, трябва да поставите курсора на мишката върху нея.

Това е много ценно умение, което ще използвате във всички свои бъдещи програми.

Когато получите отговорите, ги покажете на своя учител.

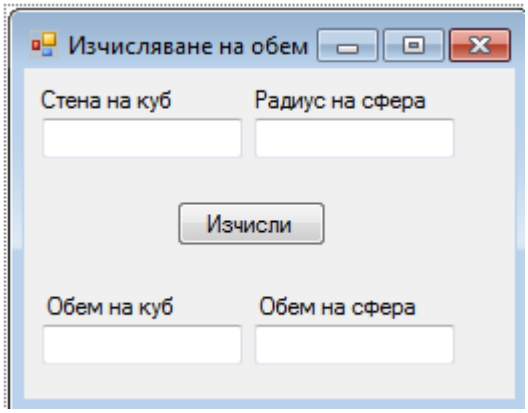
07 Можете ли? Калкулатор за изчисляване на обем



Помните ли кристалните преспапиета, които видяхме в магазина за подаръци вчера? Едното беше с кубична, а другото със сферична форма. Чудех се кое от тях е по-тежко. Тъй като са направени от един и същ материал, по-тежко трябва да е преспапието с по-голям обем.

Нека да напишем програма за пресмятане на обема на куб и сфера.

Създайте форма, която изглежда по следния начин:



Дължината на една от страните на куба ще се попълва в едното текстово поле, а радиуса на сферата в другото.

Ако сте забравили геометричните формули за пресмятане на обем, използвайте следните.

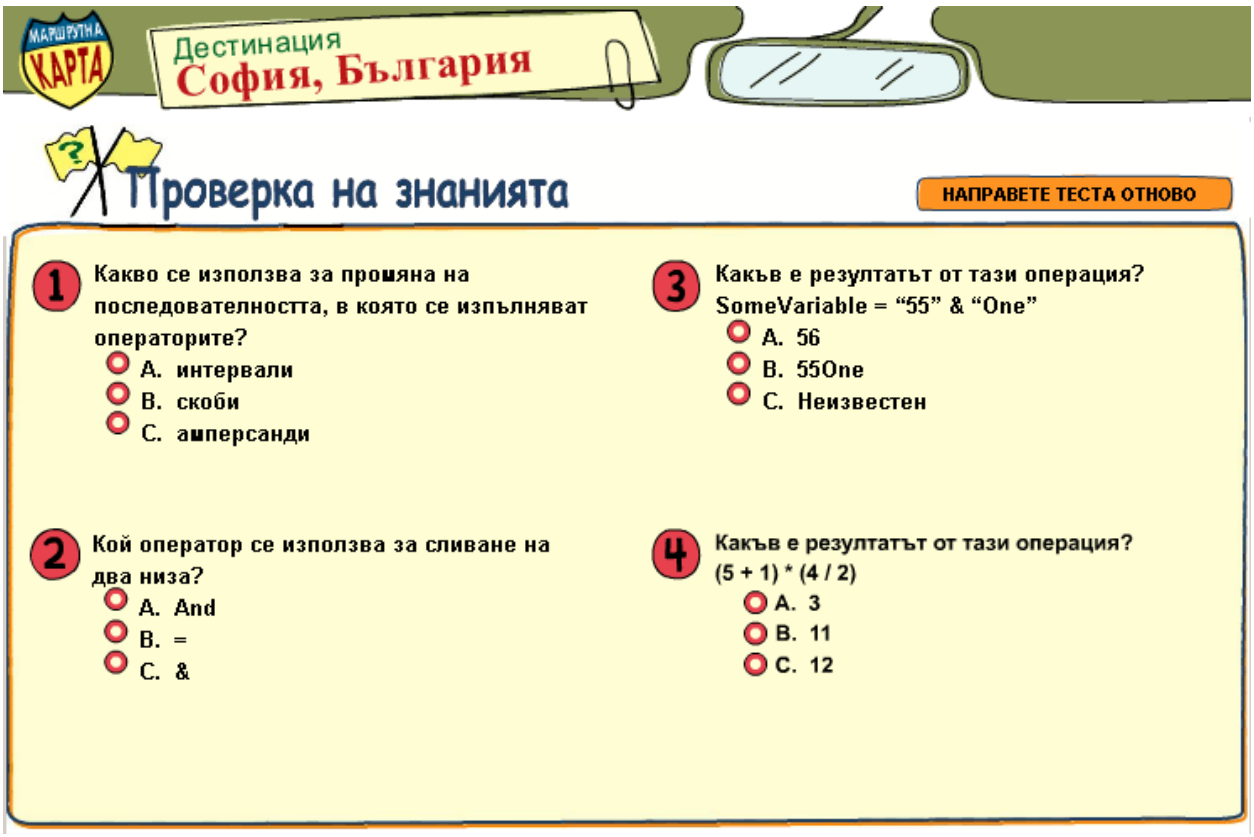
- Обемът на куба се изчислява чрез повдигане дължината на неговата страна на трета степен.
- Обемът на сферата се изчислява като се вземат четири трети от числото Пи и се умножат по радиуса повдигнат на трета степен. Можете да използвате 3.14 като стойност за Пи или Math.PI.

Ако програмата работи правилно, я покажете на вашия учител.

Продължение / Обобщение

Просто за забавление, можете ли да намерите две стойности, които при изчисляването на обема дават един и същ обем?

Ако сте наистина добри в математиката, можете ли да добавите към формата изчисления, които по зададен обем изчисляват страната на куба и радиуса на сферата?


МАРШРУТНА КАРТА
Дестинация
София, България
Проверка на знанията
НАПРАВЕТЕ ТЕСТА ОТНОВО
1
Какво се използва за промяна на последователността, в която се изпълняват операторите?
A. интервали
B. скоби
C. амперсанди
2
Кой оператор се използва за сливане на два низа?
A. And
B. =
C. &
3
Какъв е резултатът от тази операция?
SomeVariable = “55” & “One”
A. 56
B. 55One
C. Неизвестен
4
Какъв е резултатът от тази операция?
(5 + 1) * (4 / 2)
A. 3
B. 11
C. 12