Математички факултет
Универзитет у Београду

Тема мастер рада:

Електронске лекције о типовима података, наредбама и изразима за трећи разред средње школе у програмском језику C#

Студент: Милош Вучић 1052/2012
Ментор: доц. др Мирослав Марић

Београд,
2013.

Увод

Програмирање (engl. programming) је вештина помоћу које корисник ствара и извршава алгоритме користећи одређене програмске језике да би направио рачунарски програм. Програмирање садржи елементе уметности, науке, математике и конструисања. Сваки програмер пише програмски код у неком програмском језику. Различити програмски језици подржавају различите стилове програмирања, као и различите нивое знања, умећа и детаља које програмер треба поседовати [11].

Свако програмирање подразумева логику. Било који проблем може се свести на некакав математички модел. Када пишемо програм ми се тада служимо унапред одређеном синтаксом, помоћу које правимо математичке моделе које касније можемо примењивати на било који проблем тог типа. У овом раду ћемо се упознати са делом програмског језика *C#*.

C# (C-sharp) је један од млађих програмских језика. Настао је 2002. године као саставни део пакета Microsoft .NET Framework. C# је објектно оријентисан програмски језик као и већина модерних виших програмских језика (C++, Java итд). Језик је опште примене и намењен је изради апликација за .NET Framework платформу.

Такође, C# је један од програма заснованих на прозорима. Да би се управљало програмом заснованим на прозорима који има развојно окружење, мора се знати језик којим се тај програм служи. Због тога је тренутно најважније питање: који програмски језик треба одабрати, као и шта и колико ће се добити уколико би исти савладали. У циљу добијања одговора на ово питање упоређиваће се особине програмских језика: Delphi, Java i C#. Упоређивање особина биће приказане кроз табелу 1.

Осим напредних особина програмског језика C# које се могу видети у табели, оно што га издваја од других програмских језика су и: једноставност коришћења као и већа употребљивост и практичност (у односу на друге програмске језике) у даљем раду и будућности.

C# као објектно-оријентисани језик омогућава брз развој широког спектра апликација заснованих на Microsoft.NET платформи па је и то једна од предности у односу на остале програмске језике. Microsoft.NET је платформа чији је циљ да се скрати време потребно за развој, на тај начин што ће програмере ослободити брига око расподеле меморије, безбедности типова, провере граничних вредности индекса у низовима итд., а више пажње посветити развоју саме апликацији. Узимајући у обзир ове чињенице C# се бира као програмски језик са којим ће се радити.

Табела 1. *Упоређивање особина програмских језика: Delphi, Java і C#*

Особине	Delphi	Java	C#
1. 64 - битни компајлер	нема	*има*	*има*
2. Сакупљач отпадака - garbage collector	нема	*има*	*има*
3. Енумератори - групе константи	нема	нема	*има*
4. Делегати	нема	нема	*има*
5. Индексери - специјалне синтаксе за преклапање оператора	нема	нема	*има*
6. Прави правоугаони низови	нема	нема	*има*
7. Петља foreach	нема	нема	*има*
8. Linq библиотека	нема	нема	*има*
9. Аутоматско утврђивање типова var	нема	нема	*има*
10. Комплексни и високо прецизни децимални бројеви	нема	нема	*има*
У табели су приказане само неке од напредних особина програмског језика C#. Оно што га заиста чини језиком будућности јесте способност да иде у корак са временом и новим технологијама.			

Прецизније, у овом раду ће бити опширнијег упознавања са типовима података у C# као и наредбама и изразима у којима се ти типови појављују.
Овај рад представља део електронског курса који се може наћи на следећој веб адреси:

http://www.edusoft.math.rs/csharp/

У овом електронском курсу, детаљно и кроз мноштво примера, приказан је програмски језик C# који се по плану и програму обрађује у трећем разреду гимназије у оквиру наставног предмета рачунарство и информатика.
Доступност оваквог једног садржаја је велика помоћ у раду и учењу. Осим тога што је бесплатан, испуњен је интересантним примерима из живота који су приказани на сликовит начин. Разне анимације и слике описују и пружају јаснију слику за многе елементе што олакшава разумевање самог текста. Ово је заправо приказ једног наставног садржаја на

модеран, оригиналан и занимљив начин чиме ће корисник бити подстакнут на веће размишљање и креативност у раду.
Посећивањем веб адресе

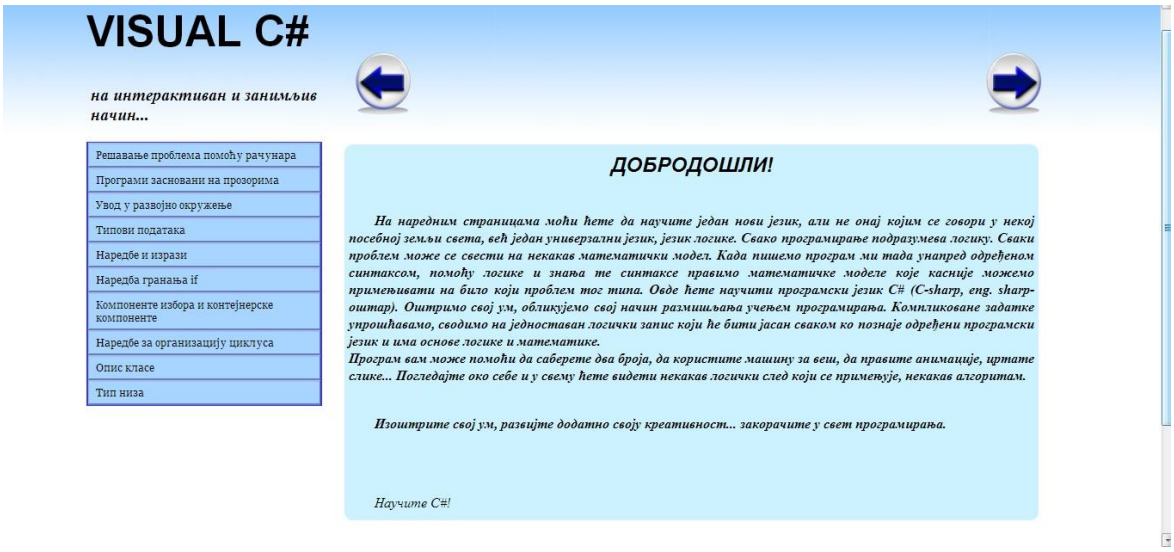
http://www.edusoft.math.rs/csharp/

на почетној страни могу се одабрати три верзије програмског језика *C#* (Слика 1).

Слика 1. Почетна страна електронског курса

То су Visual Studio 2005, Visual Studio 2010 и Visual Studio 2012. Свака од тих верзија је детаљно обрађена кроз целине које прате наставни план и програм наставног предмета рачунарство и информатика.

Након одабира верзије, корисник у менију са леве стране може видети које су све то тематске целине обухваћене овим електронским курсом (слика 2). У свакој верзији су обрађене исте тематске целине али се слике и кодови у тим верзијама разликују па је због тога постојала потреба да се свака верзија посебно обради.
У овом електронском курсу програмски језик *C#* је обрађен кроз десет тематских целина а свака од тих целина у себи садржи мање подтеме како би кориснику била омогућена лепша прегледност и једноставнији избор жељене области.



Слика 2. Прозор након одабира жељене верзије

Кликом на одређену тему, појављују се поднаслови тема којима се може приступити кликом на њих (слика 3).

Слика 3. Прозор који се приказује након одабира Целобројног типа

При састављању програма и редоследу тематских целина водило се рачуна о обезбеђивању поступности у остваривању циљева, а то су стицање знања, овладавање

вештинама као и унапређивање информатичке писмености, па је сваки од ових електронских курсева подељен у десет целина а те целине су:

1. Решавање проблема помоћу рачунара

У овој целини корисник се упознаје са: основама алгоритмизације, трансформацијама проблема на облик погодан за решавање на рачунару, програмским језицима и њиховим синтаксама и семантикама.

2. Програми засновани на прозорима

У овој целини корисник се упознаје са: основним карактеристикама програма заснованих на прозорима, елементима графичког корисничког интерфејса (Graphical User Interface), програмима руковођеним догађајима (догађаји, извори догађаја и обрада догађаја).

3. Увод у развојно окружење програмског језика

У овој целини корисник се упознаје са: почетком рада и управљањем развојним окружењима, празним пројектом, чувањем и отварањем пројекта, формама и подешавањима њених својстава, додавањем компоненти форми, компонентама у жижи као и са једноставним компонентама: натпис (Label), оквир за уношење и приказивање текста, дугме (Button), часовник (Timer) и оквир за графички објекат. Поред овога корисник се упознаје и са својствима компоненти и њиховим подешавањем и са догађајима компоненти и обрадом тих догађаја.

4. Типови података

У овој целини корисник се упознаје са: целобројним типом (опсегом целобројног типа, аритметичким операцијама, операцијама поређења и стандардним функцијама дефинисаним на целобројном типу, приоритету операција), реалним типом (опсегом реалног типа, аритметичким операцијама и стандардним функцијама дефинисаним на реалном типу), логичким типом, знаковним типом, низовним типом, стринг типом и основним функцијама за рад са стринговима, набројивим типом, класама и методама класе.

5. Наредбе и изрази

У овој целини корисник се упознаје са: синтаксом и семантиком израза, аритметичким изразима, логичким изразима, наредбама доделе, конверзијом типова података, уношењем и приказивањем података, алгоритмом за размену вредности две променљиве.

6. Наредба гранања if

У овој целини корисник се упознаје са: синтаксом наредбе if, са алгоритмима за:

- одређивање минимума/максимума два/три броја;
- уређивање два/три броја у монотоно неопадајућем/нерастућем поретку;

- одређивање сутрашњег и јучерашњег датума у односу на данашњи или дати дан;
- приказ назива дана у недељи на основу учитаног редног броја дана.

7. Компоненте избора и контејнерске компоненте

У овој целини корисник се упознаје са: компонентама избора (радио-дугме - RadioButton, оквир за потврду - CheckBox, оквир с листом - ListBox, комбиновани оквир - ComboBox) и контејнерским компонентама (оквир за групу - GroupBox, плоча - Panel).

8. Наредбе за организацију циклуса

У овој целини корисник се упознаје са: синтаксама наредби за организацију циклуса, применама наредби break и continue у циклусима, алгоритмима за:

- табелирање вредности функција;
- израчунавање сума и производа;
- испитивање својстава целих бројева.

9. Опис класе

У овој целини корисник се упознаје са: описом методе класе (функције и процедуре), формалним параметрима методе, телом методе, синтаксом позива методе, стварним параметрима методе, локалним променљивама методе, пољем класе.

10. Тип низа

У овој целини корисник се упознаје са: једнодимензионалним низовима, алгоритмима са низовима:

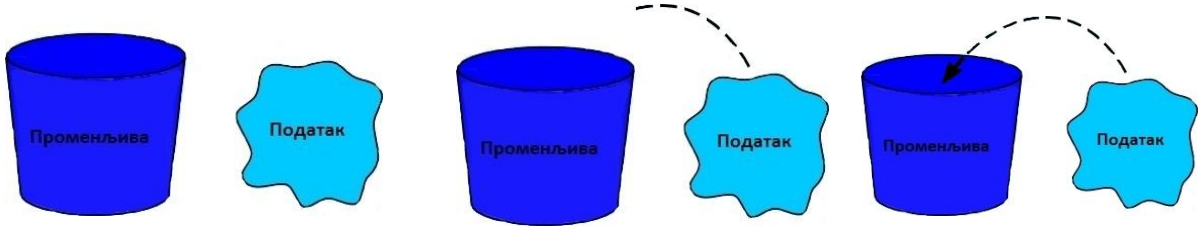
- основне операције са низовима;
- израчунавање просечне вредности;
- израчунавање минималне/максималне вредности низа;
- претраживање у низу;
- сортирање низа,

дводимензионалним низовима, алгоритмима за израчунавање и трансформације на табели и њеним деловима, визуелним компонентама за табеларним приказом текста.

Свака од ових тематских целина прожета је мноштвом примера и интересантним анимацијама које су прилагођене интересовањима и могућностима ученика овог узраста. Као што је већ напоменуто, овај рад је заправо папирна верзија дела електронског курса који се односи на целине о *типовима података, наредбама и изразима*.

Промењљиве и оператори

Промењљиве су места у меморији у којима подаци могу привремено да се чувају да би програм могао да их користи (слика 4). Оне имају *тип, име* и *вредност*. Вредност промењљиве може да се мења за време извршавања програма, због чега је и добила назив – промењљива. Да би уопште могла да се употреби, промењљива најпре мора да се декларише: њен тип мора да се наведе и мора јој се дати име.
Тип промењљиве дефинише дозвољени распон вредности које та промењљива може да садржи, као и операције које могу да се обављају на датој промењљивој.



Слика 4. Чување податка у промењљивој

Типом података дефинисан је :
- начин регистровања података у меморији
- скуп могућих вредности тих података
- скуп могућих акција над подацима

Име промењљиве гради се на основу следећег правила :
Име је низ слова (великих и малих) енглеске абецеде, знака за подвлачење (_) и цифара. Име мора почети словом или знаком за подвлачење. Према томе, примери имена промењљивих су NazivPredmeta, ucenik1, Rezultat, radni_dan, _ime, ... док godisnje-doba, 1dan, ... нису и не могу бити имена промењљивих. Више о промењљивој се може наћи у књизи [2].

Напомена
Важно је напоменути да C# разликује велика и мала слова тако да имена промењљивих GodisnjeDoba и godisnjeDoba нису иста.

Именовање података у апликацијама и дефинисање типа података којем припадају постиже се декларацијом промењљивих на следећи начин:

<ime tipa> <ime promenljive1>, …, <ime promenljiveN> ;

Пример 1.1 Декларација променљиве у програмском језику C#.

```
int a, b;
char p, q;
```

Оператори

У овом поглављу биће речи о одређеним врстама оператора и чему они служе. То су оператори додељивања, аритметички оператори и оператори поређења. Више о операторима се може наћи у књигама [4], [8].

Оператор додељивања поставља вредност променљиве (табела 2).

Табела 2. *Оператор додељивања и његов опис*

Оператор додељивања	
Оператор	**Опис**
=	Додељивање вредности променљивој

Пример 2.1 Додељивање вредности променљивој коришћењем оператора додељивања.

```
//Navodi se tip i ime promenljivih koje se koriste
int a;
int b;
//Promenljivoj a se dodeljuje vrednost 5, a promenljivoj b
//vrednost 13
a = 5;
b = 13;
```

Аритметички оператори извршавају аритметичке операције (табела 3).

Табела 3. *Аритметички оператори и њихов опис*

Аритметички оператори	
Оператор	**Опис**
+	Сабирање
-	Одузимање
*	Множење
/	Дељење
%	Остатак при дељењу

Пример 2.2 Употреба оператора сабирања (+), одузимања (-), множења (*) и дељења (/).

```
int a = 1 + 2;
int b = 3 - 2;
int c = 2 * 5;
int d = 10 / 3;
```

Оператори поређења користе се за упоређивање односа између вредности двеју променљивих и враћају вредност true или false (тачно или нетачно) (табела 4).

Табела 4. *Оператори поређења и њихов опис*

Оператори поређења		
Оператор	**Опис**	**Повратна вредност**
==	поређење вредности	a == b - има вредност true ако су a и b једнаке, false иначе
!=	неједнако, различито	a != b - има вредност true ако је a различито од b, false иначе
>	веће од	a > b - има вредност true ако је a веће од b, false иначе
<	мање од	a < b - има вредност true ако је a мање од b, false иначе
>=	веће од или једнако	a >= b - има вредност true ако је a веће од или једнако b, false иначе
<=	мање од или једнао	a <= b - има вредност true ако је a мање од или једнао b, false иначе

Напомена
Водити рачуна о разлици између оператора (==) који пореди две вредности и оператора доделе (=) који врши додељивање вредности.

Пример 2.3 Коју повратну вредност имају следећи изрази?

```
10 < 3 //Postavlja se pitanje, da li je 10 manje od 3? Kako ovo nije
       //ispunjeno ovaj izraz vraća false

10 <= 3 //Postavlja se pitanje, da li je 10 manje od ili jednako 3? Kako ovo
        //nije ispunjeno ovaj izraz vraća false

10 != 3 //Postavlja se pitanje, da li je 10 različito od 3? Kako je ovo
        //ispunjeno ovaj izraz vraća true

10 > 3 //Postavlja se pitanje, da li je 10 veće od 3? Kako je ovo ispunjeno
       //ovaj izraz vraća true

10 >= 3 //Postavlja se pitanje, da li je 10 veće od ili jednako 3? Kako je
        //ovo ispunjeno ovaj izraz vraća true
```

Више примера сличних примеру 2.3 се могу наћи у књизи [3].

Приоритет оператора

Када се пише израз који садржи више од једног оператора, компајлер користи скуп правила која одређују који оператор треба применити први, други, трећи и тако даље док се не израчуна цео израз. Сваки оператор има уграђени проротет, или првенство, које компајлер користи да би одредио који оператор треба следеће применити.

```
int mojBroj = 2 + 5 * 10;
```

На изразу a + b * c се може уочити коришћење правила приоритета оператора. Како оператор множења (*) има већи приоритет, или веће првенство, од оператора сабирања (+), десна страна израза се израчунава на следећи начин: 5 * 10 = 50, 2 + 50 = 52. Због тога се промењливој mojBroj додељује вредност 52. Погрешно би било када би се прво израчунало 2 + 5 = 7 па затим 7 * 10 = 70 и промењливој mojBroj доделила вредност 70. Међутим, приоритет оператора се може прегазити коришћењем заграда. На пример:

```
int mojBroj = (2 + 5) * 10;
```

Оператор множења (*) и дељења (/) имају исти приоритет, па се оператори извршавају слева надесно.

```
int mojBroj = 2 * 20 / 5 ;
```

Тако се прво израчунава 2 * 20 = 40, а затим 40 / 5 = 8. Због тога се променљивој mojBroj додељује вредност 8.

У C# има велики број оператора и ови оператори се могу груписати у категорије које имају исти приоритет. Табела 5 говори о редоследу првенства оператора - прво је показана категорија са највећим приоритетом. У овој табели приказане су категорије оператора и оператори у оквиру сваке категорије који су сложени по редоследу првенства.
Када се операнд стави између два оператора са истим приоритетом, редослед операције која ће се извршити одређује асоцијативност оператора. Са изузетком додељивања оператора, оператори који користе два операнда (нпр. оператор сабирања) су лево асоцијативни - што значи да се операције извршавају слева надесно. Оператори додељивања и троструки оператори су десно асоцијативни - што значи да се операције извршавају сдесна налево.

Табела 5. *Приоритет оператора*

Приоритет оператора	
Категорија	**Оператори**
Основна	Група : (x)
	Приступ пољу : x.y
	Позив методе : f(x)
	Приступ индексу : a[x]
	Инкрементирање са суфиксом : x++
	Декрементирање са суфиксом : x--
	Нови објекат : new
	Get type : typeof
	Get size: sizeof коришћење граничне провере : checked
	Без граничне провере : unchecked
Унарна	Позитивна вредност: +
	Негативна вредност: -
	Логичко Boolean НЕ : !
	НЕ на нивоу бита: -
	Инкрементирање са префиксом : x++

	Декрементирање са префиксом : x--
	Конверзија: (type)
Вишеструки	Множење: *
	Дељење: /
	Остатак при дељењу: %
Додатни	Сабирање: +
	Одузимње: -
Померање на нивоу бита	Лево померање
	Десно померање
Релациони	Мање од : <
	Веће од : >
	Мање од или једнако : <=
	Веће од или једнако : >=
	Компатабилност типа : is
	Компатибилнст са оператором за конверзију : as
Једнакост	Једнако : =
	Неједнако : !=

Типови података

Подаци су објекти који се обрађују у програмима. Основно обележје сваког податка је његов тип. Типови података одређују могуће вредности података и могуће операције које могу да се изводе над тим подацима. C# је строго типизиран језик, што значи да се мора доделити тип података сваком меморијском елементу који користимо у програму. С обзиром да је C# модеран језик, он има велики број уграђених типова који долазе спремни за употребу у нашим сопственим програмима.

Целобројни тип

Како једначина x + a = b нема увек решење у скупу N (прецизно: када је a $\geq$ b), скуп N се зато проширује у скуп целих бројева Z = {... , -2, -1,0,1,2,3, ...}. Скуп Z је неограничен здесна и слева јер не постоји ни број од којега су сви цели бројеви мањи, ни број од којега су сви цели бројеви већи. Као и скуп природних бројева, скуп Z је затворен за операције сабирања и множења. То значи да је збир и производ било која два цела броја такође цео број. Међутим, за разлику од природних бројева, скуп целих бројева је затворен и за одузимање. Ово не важи и за дељење, јер количник два цела броја не мора да буде цео број (на пример: 1 подељено са 2).

Као и у математици, и у C# целобројни тип ће имати нека ограничења. Постоји осам целобројних типива који се могу користити да би се представили цели бројеви. У Табели 6 приказани су ови целобројни типови. У њој су приказани C# типови, број бајтова које користе и опсег вредности које се могу меморисати за тај тип.

Табела 6. *Целобројни типови за представљање целих бројева*

C# типови	Коришћено бајтова	Вредности
sbyte	1	-128 до 127 (означен)
byte	1	0 до 255 (неозначен)
short	2	-32,768 до 32,767 (означен)
ushort	2	0 до 65,535 (неозначен)
int	4	-2,147,483,647 до 2,147,483,647 (означен)
uint	4	0 до 4,294,967,295 (неозначен)
long	8	-9,223,372,036,854,775,808 до 9,223,372,036,854,775,808 (означен)
ulong	8	0 до 18,446,744,073,709,551,615 (неозначен)

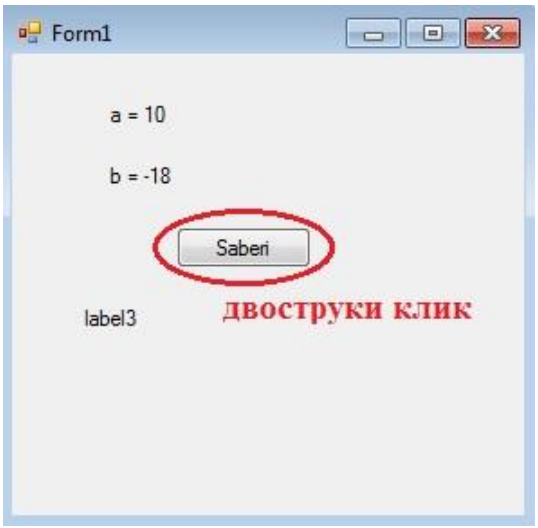
Напомена

Треба се постарати да се одговарајући тип изабере зависно од тога да ли је целобројна вредност означена или неозначена и у зависности од опсега који треба да се меморише.

До сада се видело како се врши додела вредности променљивој, које аритметичке операције се смеју над њом вршити, који оператори поређења постоје и којег су протитета па се стога могу направити први програми.

Пример 3.1 Написати програм који сабира два цела броја и њихов збир исписује у лабели.

Решење : Треба декларисати променљиве a и b које су типа int. Затим креирати и трећу променљиву (rezultat) у коју се смешта збир та два броја. Што се тиче компоненти, користе се три лабеле и једно дугме. Две лабеле се преименују у „a" и у „b" а у трећој ће се исписати резултат. Такође име дугмета се преименује у „Saberi" (слика 5).

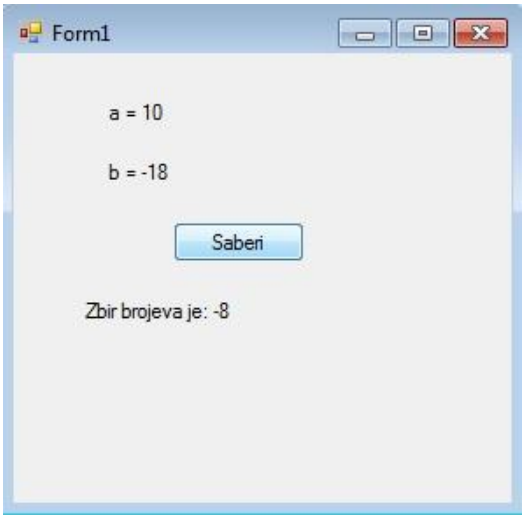


Слика 5. Изглед форме пре покретања програма

Алгоритам 1. Збир два цела броја

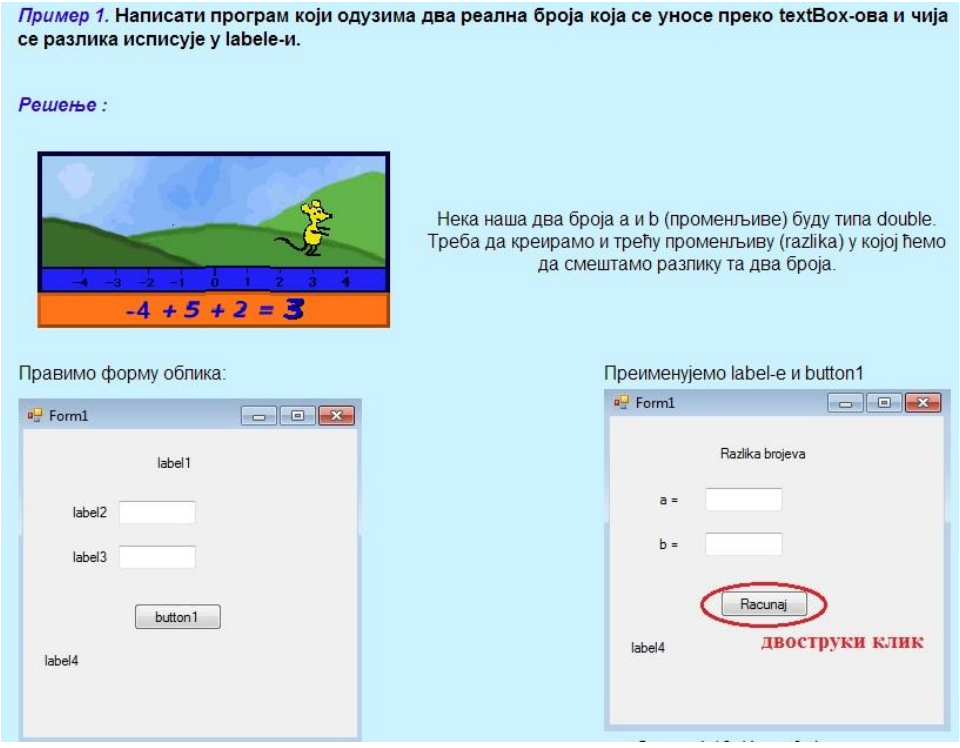
```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koriste
    int a, b;
    //Promenljiva u koju se smesta rezultat
    int rezultat;
    a = 10;
    b = -18;
```

```
    rezultat = a + b;
    label3.Text = "Zbir je :"  + rezultat;
}
```



Слика 6. Изглед форме након покретања програма

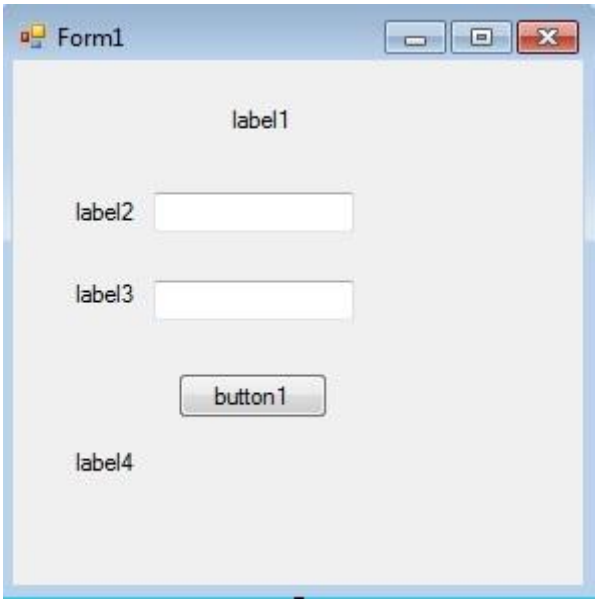
Сличан пример који илуструје одузимање два реална броја приказан је у електронском курсу (слика 7).



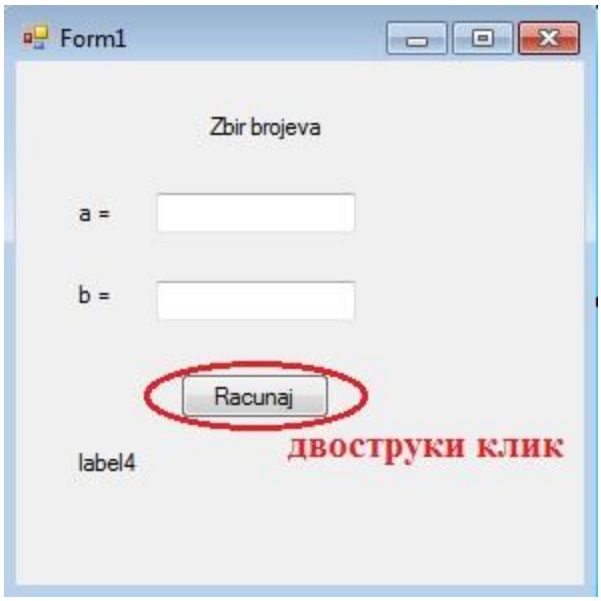
Слика 7. Изглед примера за разлику два цела броја у електронском курсу

Пример 3.2 Написати програм који сабира два цела броја која се уносе преко textBox-ова и чији се збир исписује у labele-и.

Решење: У претходном примеру приказан је праграм у коме су се сабирци уносили директно у коду програма, овде ће се уносити преко textBox-ова. Треба декларисати промењљиве a и b које су типа int. Затим треба да се креира и трећа променљива (rezultat) у коју се смешта збир та два броја. Овај пример се разликује од претходног јер кориснику треба да буде омогућено да вредности сабирака унесе преко textBox-ова. Лабеле и дугме се могу преименовати као на слици 9.



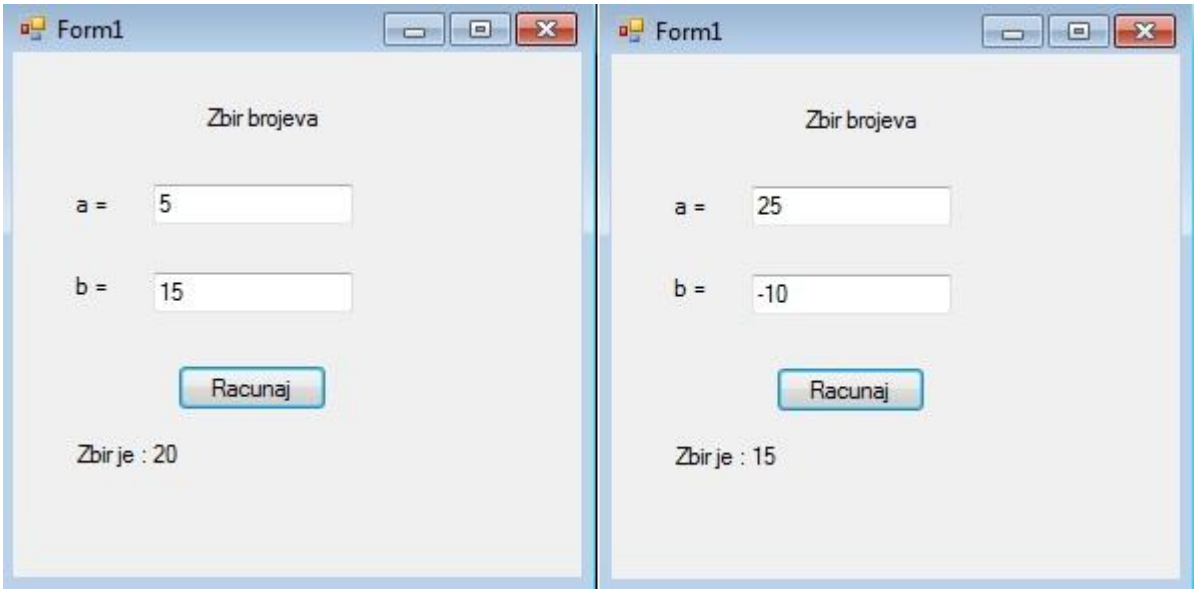
Слика 8. Изглед почетне форме



Слика 9. Изглед форме након преименовања компоненти

Алгоритам 2. Збир два цела броја која се уносе преко textbox-а

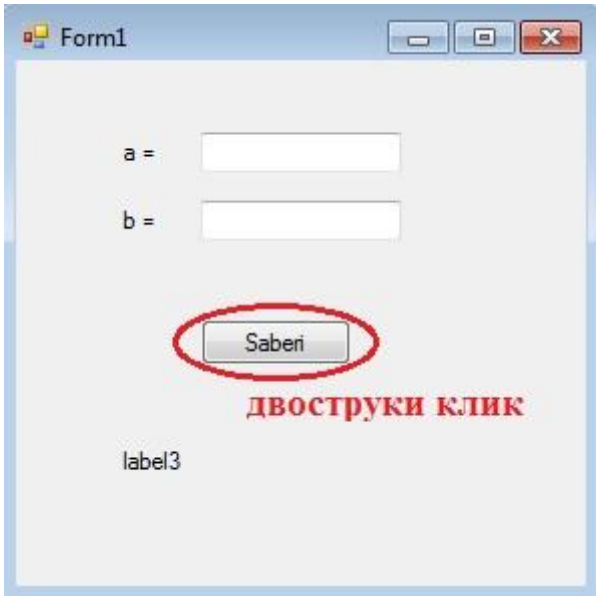
```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koriste
    int a, b;
    //Promenljiva u koju se smesta rezultat
    int rezultat;
    a = int.Parse(textBox1.Text);
    b = int.Parse(textBox2.Text);
    rezultat = a + b;
    label4.Text = "Zbir je : "  + rezultat;
}
```



Слика 10. Изглед форме након покретања програма

Пример 3.3 Написати програм који сабира два броја из сегмента [-128,127] која корисник унесе преко textBox-ова.

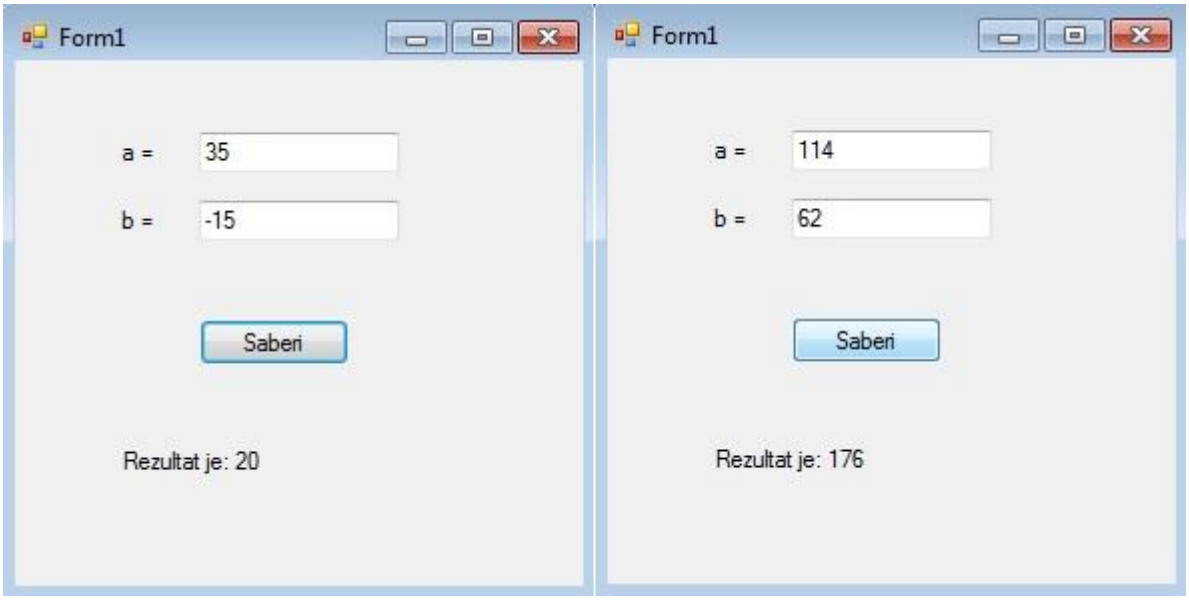
Решење: Направити две промењиве a и b у које се смешта оно што корисник унесе преко textBox-ова. Као и у претходном примеру треба направити и трећу промењиву (rezultat) у коју се смешта њихов збир. Ради боље читљивости програма за имена промењивих треба користити осмишљена имена, односно имена која асоцирају на врсту информације која се у њима чува. Поред овога треба промењивој увек додељивати најпогоднији тип. Пошто се овде ради о бројевима који припадају одговарајућем сегменту, типови промењивих a и b ће бити sbyte. Остало је још видети којег ће типа бити промењива rezultat. Како се ради о збиру два броја као резултат ће се добити број који не мора бити из сегмента датог сегмента, зато се за њу узима тип int. Лабеле и дугме се могу преименовати као на слици 11.



Слика 11. Изглед форме пре покретања програма

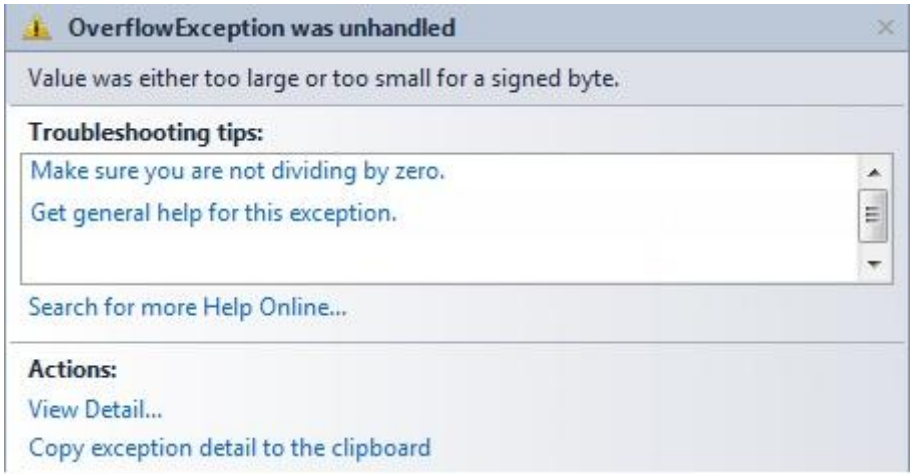
Алгоритам 3. Збир бројева из сегмента [-128,127]

```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljivoj a se dodeljuje tip sbyte kao i promeljivoj b
    sbyte a;
    sbyte b;
    //Pravi se i treca promenljiva rezultat u koju se
    //smesta zbir prve dve i na osnovu prethodnog
    //teksta dodeljuje joj se tip int
    int rezultat;

    //Vrednost koju je korisnik uneo preko textBox-a u
    //promenljivu a
    a = sbyte.Parse(textBoz1.Text);
    //Isti postupak se primenjuje za promenljivu b
    b = sbyte.Parse(textBox2.Text);

    //U promenljivoj rezultat se unosi vrednost zbira a i b
    rezultat = a + b ;
    //Ispisivanje rezultata u labelu
    label3.Text = "Zbir je : "  + rezultat;
}
```



Слика 12. Изглед форме након покретања програма

Поставља се питање шта би се десило да се некој променљивој доделила вредност која излази из опсега који одговара типу sbyte а то је [-128,127] ? Нека је то нпр. вредност 154. У том случају програм би се побунио и избацио грешку (слика 13) која говори да је унешена вредност за променљиву изван одговарајућег интервала за тип byte.

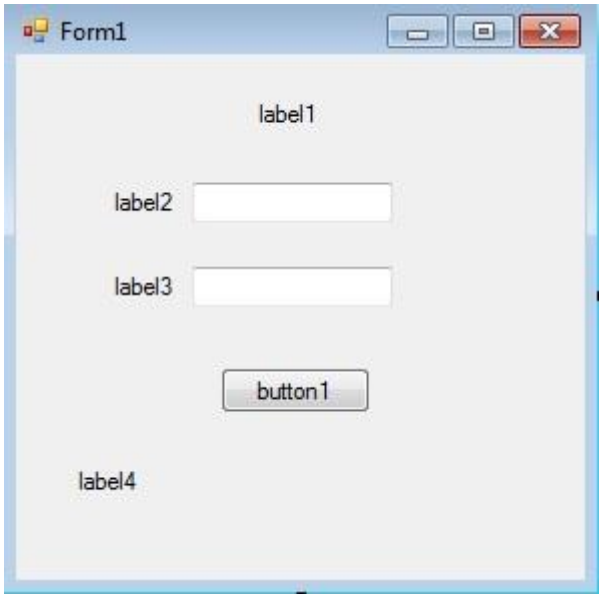


Слика 13. Изглед прозора који се појављује када настане грешка

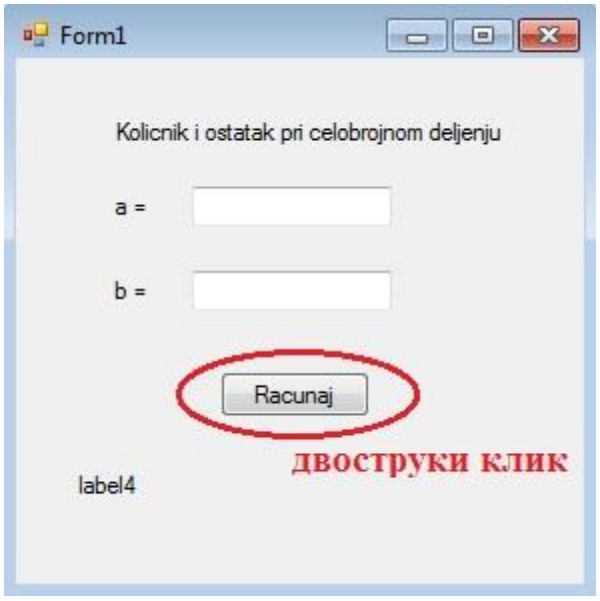
Као што је на почетку речено, треба се постарати да се изабере одговарајући тип зависно од тога да ли је целобројна вредност означена или неозначена и у зависности од опсега који треба да се меморише.

Пример 3.4 Саставити програм који одређује количник и остатак при целобројном дељењу два цела броја унета преко textBox-ова.

Решење: Треба декларисати промењљиве a и b које су типа int. Затим креирати промењљиву kolicnik у коју се смешта количник та два броја као и промењљиву ostatak у коју ће се сместити остатак при целобројном дељењу та два броја. Што се тиче компоненти, користе се четири лабеле и једно дугме. Три лабеле се преименују док ће се у четвртој исписати резултат. Такође име дугмета се преименује у „Racunaj". Лабеле и дугме се могу преименовати као на слици 15.



Слика 14. Изглед почетне форме

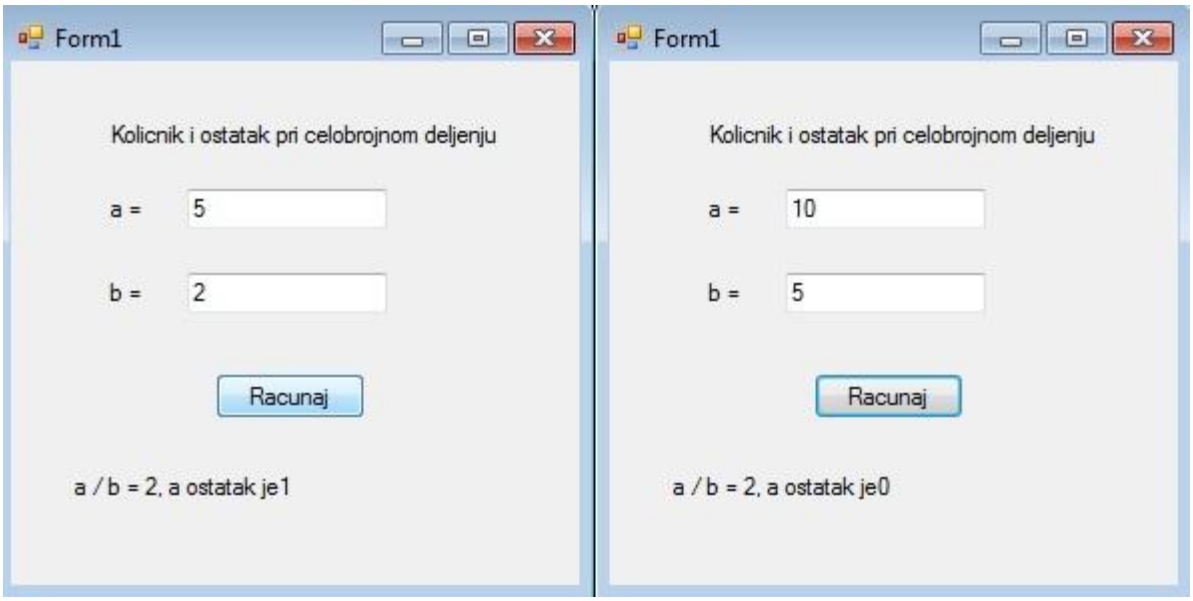


Слика 15. Изглед форме након преименовања компоненти

Алгоритам 4. Количник и остатак при целобројном дељењу два цела броја

```
private void button1_Click(object sender, EventArgs e)
{
    //Uzima se vrednost za promenljivu a koju je korisnik uneo
    int a = int.Parse(textBox1.Text) ;
    //Uzima se vrednost za promenljivu b koju je korisnik uneo
    int b = int.Parse(textBox2.Text) ;
    //Deklarise se promenljiva ostatak
    int ostatak;
    //Racuna se ostatak
    ostatak = a % b;
    //Deklarise se promenljiva kolicnik
    int kolicnik;
```

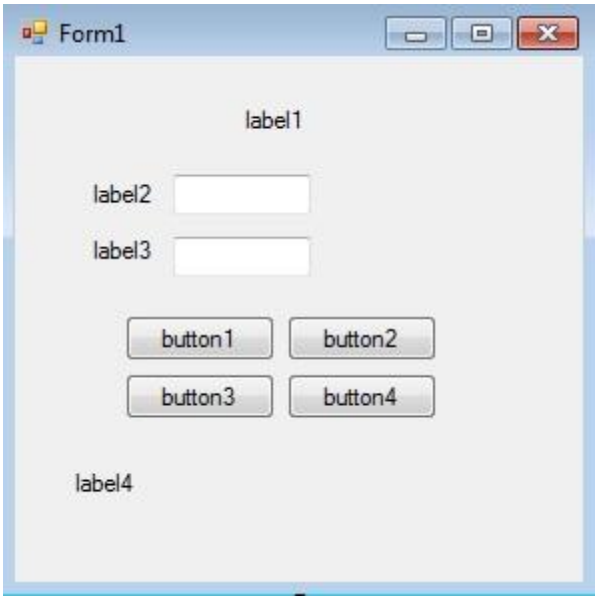
```
    //Racuna se kolicnik
    kolicnik = a / b;
    //Ispisuje se resenje
    label3.Text = "a / b = " + kolicnik +  ", a ostatak je" +
ostatak;

}
```



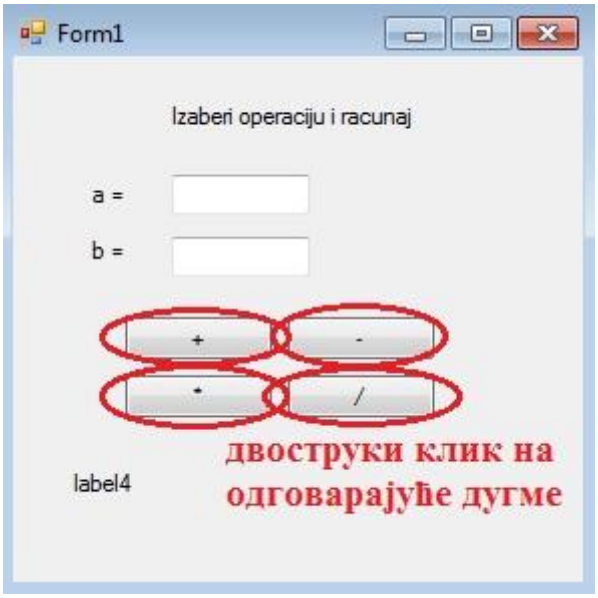
Слика 16. Изглед форме након покретања програма

Пример 3.5 Саставити програм који за унета два броја преко textBox-ова, рачуна збир, разлику, производ или количник, у зависности од тога коју операцију корисник одабере кликом на дугме (за сваку операцију је одговарајуће дугме).

Решење: За решавање овог задатка потребне су четири лабеле, четири дугмета и два textBox-а. Треба креирати две промењиве a и b у које ће се користећи конвертовање сместити оно што корисник унесе у textBox1 и textBox2. Креира се промењива rezultat у коју ће се у зависности од тога на које се дугме кликне смештати збир, разлика, производ или количник унетих бројева. Неке од лабела и дугмади се могу преименовати као на слици 18.



*Слика 17.*Изглед почетне форме



Слика 18. Изглед форме након преименовања компоненти

Алгоритам 5. Збир, разлика, производ или количник два цела броја

```
    //kod za zbir
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koriste
    int a, b;
    //Promenljiva u koju se smesta rezultat
    int rezultat;
    a = int.Parse(textBox1.Text);
    b = int.Parse(textBox2.Text);
    rezultat = a + b;
    label4.Text = "Zbir je : "  + rezultat;
}

    //kod za razliku
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koriste
    int a, b;
    //Promenljiva u koju se smesta rezultat
    int rezultat;
    a = int.Parse(textBox1.Text);
    b = int.Parse(textBox2.Text);
    rezultat = a - b;
```

```
    label4.Text = "Razlika je : "  + rezultat;
}

    //kod za proizvod
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koristimo
    int a, b;
    //Promenljiva u koju se smesta rezultat
    int rezultat;
    a = int.Parse(textBox1.Text);
    b = int.Parse(textBox2.Text);
    rezultat = a * b;
    label4.Text = "Proizvod je : "  + rezultat;
}

    //kod za kolicnik
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koriste
    int a, b;
    //Promenljiva u koju se smesta rezultat
    int rezultat;
    a = int.Parse(textBox1.Text);
    b = int.Parse(textBox2.Text);
    rezultat = a / b;
    label4.Text = "Kolicnik je : "  + rezultat;
}
```

Слика 19. Изглед форме након покретања програма

Реални тип

У математици се скуп реалних бројева означава са R. Реални бројеви су сви рационални и ирационалани бројеви. Скуп реалних бројева R је бесконачан и непребројив. Како је скуп R у математици настао због потребе да се прошире цели бројеви тако је и у C# било потребно увести тип података који ће представљати нешто што је паралелно са реалним бројевима. Због тога у C# - у постоје типови података са покретним зарезом који се могу користити да би се представили такви бројеви. У C# - у постоје три таква типа, то су: float, double и decimal (Табела 7). Избор одговарајућег типа зависи од вредности и броја значајних цифара који је потребан да би се меморисао број у покретном зарезу.

Табела 7. *Типови података за представљање реалних бројева*

Типови за покретни зарез		
C# тип	**Коришћено бајтова**	**Вредности**
float	4	Приближно +/- $1.5*10^{-45}$ до приближно +/- $3.4*10^{38}$ са седам значајних цифара
double	8	Приближно +/- $1.5*10^{-324}$ до приближно +/- $1.7*10^{308}$ са 15 до 16 значајних цифара
decimal	12	Приближно +/- $1.0*10^{-28}$ до приближно +/- $7.9*10^{28}$ са 28значајних цифара

Иако тип decimal подржава мањи опсег бројева, он је у неким случајевима пожељнији јер никада неће доћи до грешака у заокруживању које се могу јавити код типова float и double. Промељива типа decimal може да меморише број са тачношћу од 28 цифара.

Напомена

Када се конкретан број додељује променљивој која је типа float, на крају се мора додати знак f или F.

Пример 4.1 Додељивање конкретног броја променљивој која је типа float и типа double.

Када се конкретан број додељује типу float

```
float mojBroj = 1.2f;
```

Када се конкретан број додељује типу double, на крају се може додати знак d или D али ово није неопходно.

```
double mojBroj = 1234.5678d;
double mojBroj = 1234.5678;
```

Треба обратити пажњу да при додељивању конкретног броја типу double или float се може користити и експоненцијална нотација.

Пример 4.2 Коришћење експоненцијалне конотације

```
double mojBroj1 = 3.6e+5;  // = 360000
double mojBroj2 = 1.2e-2;  // = 0,012
```

Број 3.6e+5 користи експоненцијалну нотацију да би представио број 3.6 * 10 на пети и исти је као и 360 000. Слично, број 1.2e-2 је 1.2 * 10 на минус други што је исто као и 0,012 .

Напомена

При додељивању конкретног броја променљивој која је типа decimal, на крају се мора додати знак m или M.

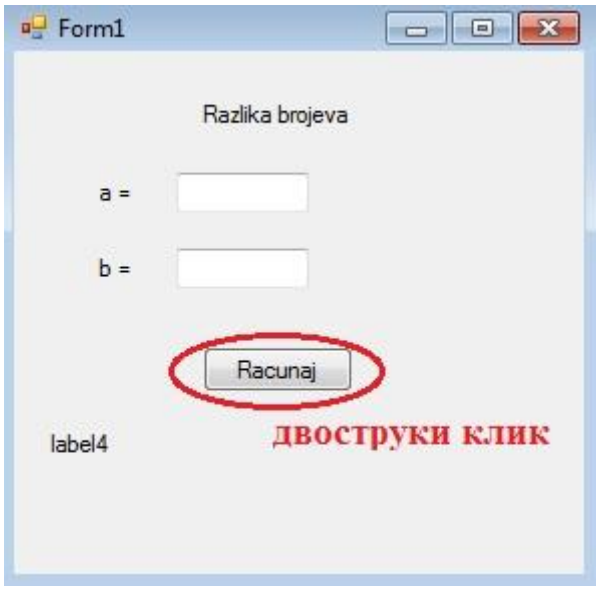
Пример 4.3 Додељивање конкретног броја променљивој која је типа float и типа double.

```
decimal  mojBroj3 = 34356.234567m ;
```

Пример 4.4 Написати програм који одузима два реална броја која се уносе преко textBox-ова и чија се разлика исписује у labele-и.

Решење: Треба декларисати промењљиве a и b које су типа double. Затим треба креирати и трећу променљиву (razlika) у коју се смешта разлика та два броја. Што се тиче компоненти, користе се четири лабеле, једно дугме и два textBox-а. Неке од лабела и дугме се могу преименовати својевољно или као на слици 21.

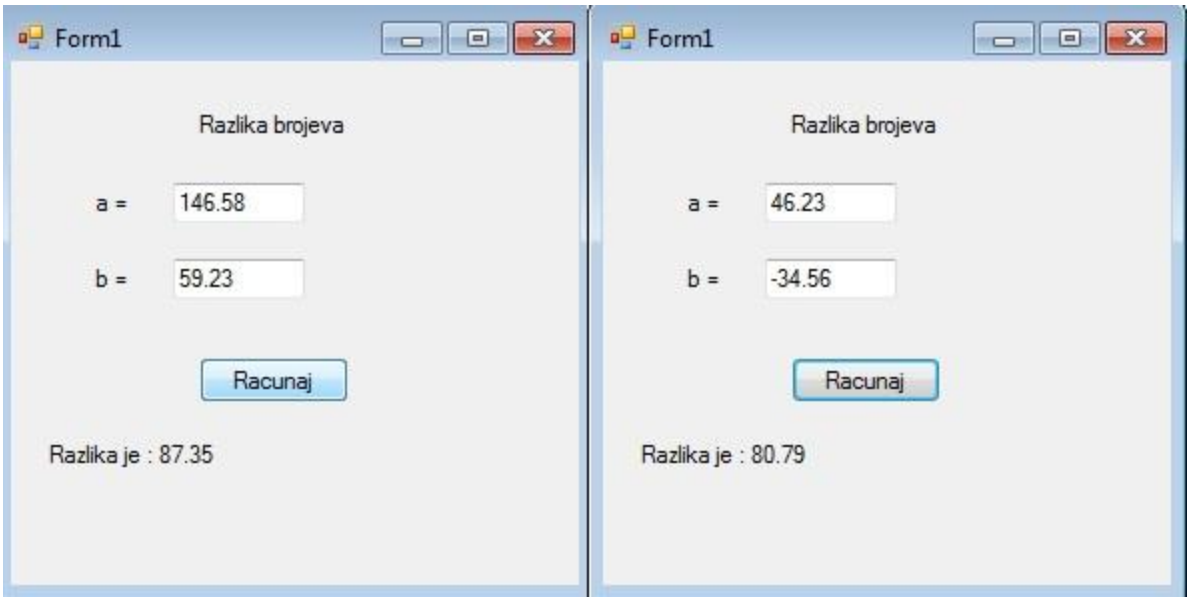
Слика 20. Изглед почетне форме



Слика 21. Изглед форме након преименовања компоненти

Алгоритам 6. Одузимање два реална броја

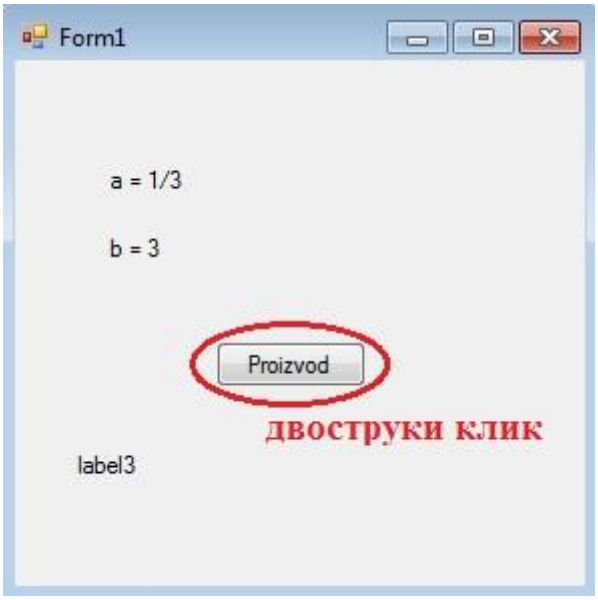
```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje ce koriste
    double a, b;
    //Promenljiva u koju ce se smestati rezultat
    double rezultat;
    a = double.Parse(textBox1.Text);
    b = double.Parse(textBox2.Text);
    rezultat = a - b;
    label4.Text = "Razlika je : "  + rezultat;
}
```



Слика 22. Изглед форме након покретања програма

Пример 4.5 Направити програм који кликом на дугме исписује у лабели производ бројева 1/3 и 3.

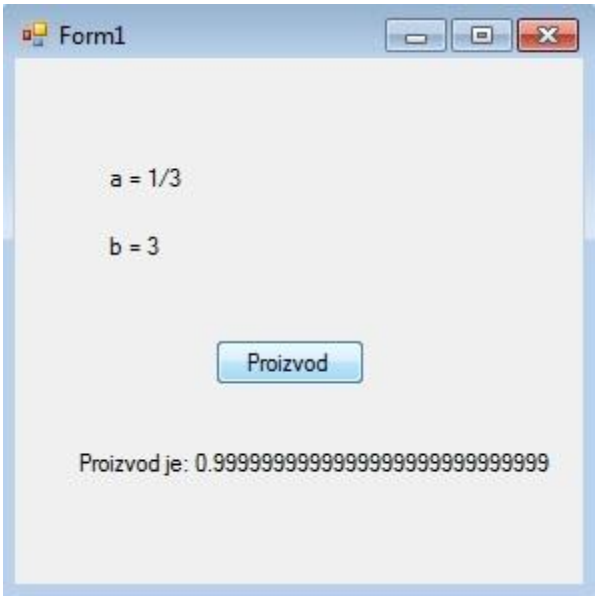
Решење : При решавању овог примера, користе се три променљиве. Нека су то a, b и proizvod. Свака од њих ће бити типа decimal. Треба водити рачуна о томе да када се користи тип decimal, на крају се мора додати m или M. Користиће се три лабеле и једно дугме које се могу преименовати као на слици 23.



Слика 23. Изглед форме пре покретања програма

Алгоритам 7. (1/3) * 3 није једнако 1

```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljivoj a se dodeljuje tip float kao i promeljivoj b
    decimal a;
    decimal b;
    //Pravi se i treca promenljiva proizvod u koju se
    //smesta proizvod prve dve
    decimal proizvod;


    //Vrednost promenljive a se postavlja na 1/3
     a = 1/3m;

    //Vrednost promenljive b se postavlja na 1
     b = 3m;

    //U promenljivoj proizvod se unosi proizvod
    //promenljivih a i b
    proizvod = a * b;
    //Ispisivanje rezultat u labelu
    label3.Text = "Proizvod je : "  + proizvod;
}
```

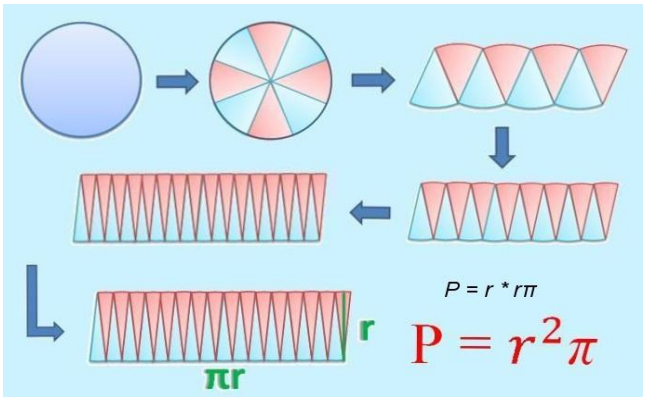


Слика 24. Изглед форме након покретања програма

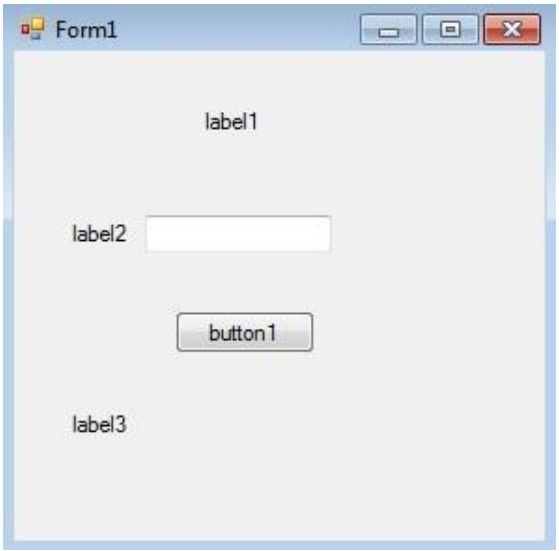
Као решење добија се број који после зареза има 28 цифара броја 9. Зашто? Ако се погледа табела 7 види се да тип decimal има 28 значајних цифара. Овај пример је показао да оно што је тачно у математици не мора бити тачно и у програмирању, на пример (1/3) * 3 није једнако 1.

Пример 4.6 Написати програм који рачуна површину круга за полупречник који се задаје из textBox-а, при чему се користи да π има приближну вредност 3,14.

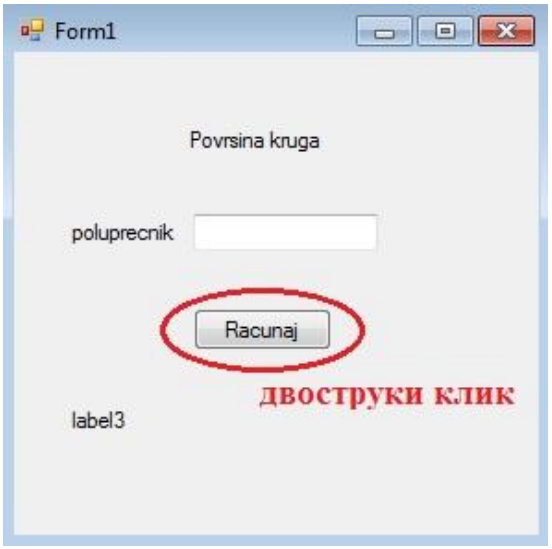
Решење : За решавање овог задатка потребне су три лабеле, једно дугме и један textBox. Креира се променљива r у којој се након конвертовања смешта оно што корисник унесе у textBox1, а променљива Pi се поставља на вредност 3,14. Кликом на дугме у лабели се исписује тражени резултат. Лабеле и дугме се могу преименовати као на слици 27.



Слика 25. Долажење до формуле за површину круга



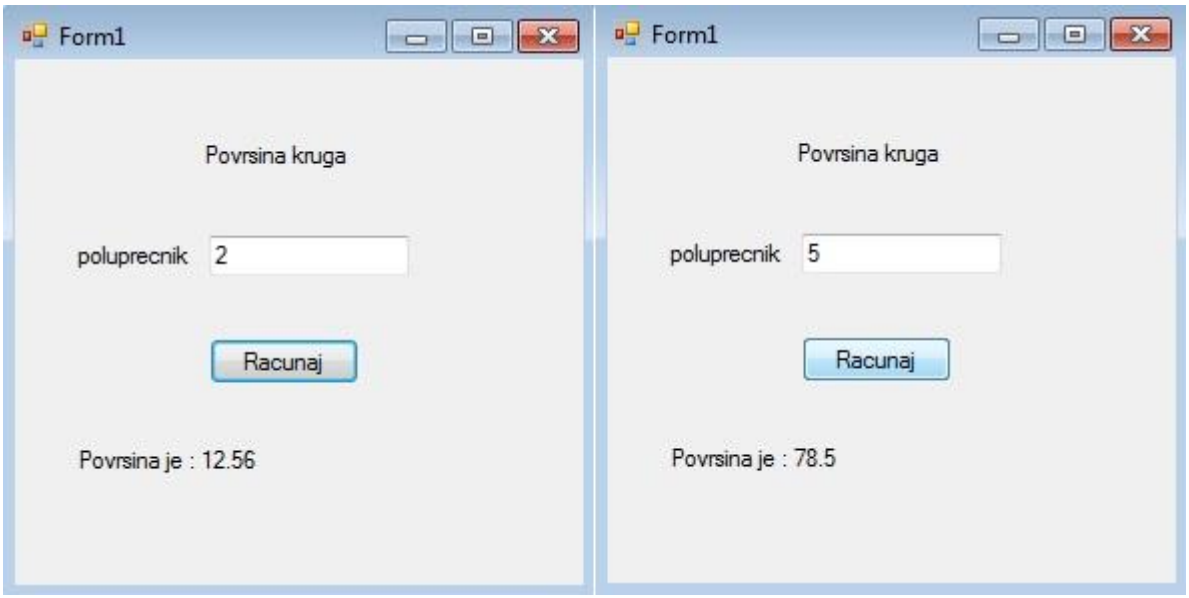
*Слика 26.*Изглед почетне форме



Слика 27. Изглед форме након преименовања компоненти

Алгоритам 8. Површина круга за полупречник који се задаје из textBox-а

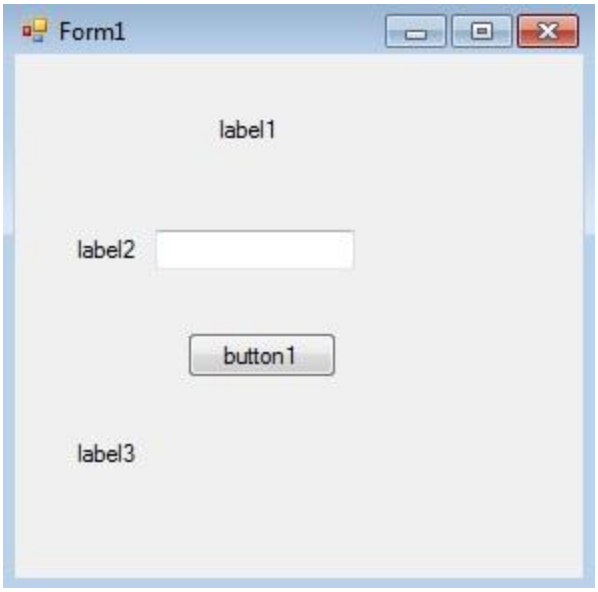
```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koriste
    double r;
    double Pi = 3.14;
    //Promenljiva u koju se smesta rezultat
    double povrsina;
    r = double.Parse(textBox1.Text);
    povrsina = r * r * Pi;
    label3.Text = "Povrsina je : "  + povrsina;
}
```



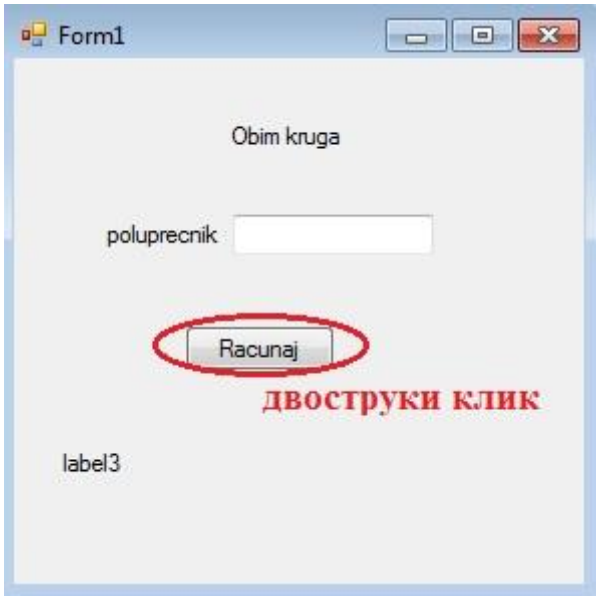
Слика 28. Изглед форме након покретања програма

Пример 4.7 Написати програм који рачуна обим круга за полупречник који се задаје из textBox-а, при чему се користи да π има приближну вредност 3,14.

Решење : За решавање овог задатка потребне су три лабеле, једно дугме и један textBox. Као и у претходном задатку креира се промењљива r у коју се смешта оно што корисник унесе у textBox а вредност промењљиве Pi поставља се на 3,14. Креира се и промењљива obim у коју се кликом на дугме смешта израчунат обим круга за задато r. Лабеле и дугме се могу преименовати као на слици 30.



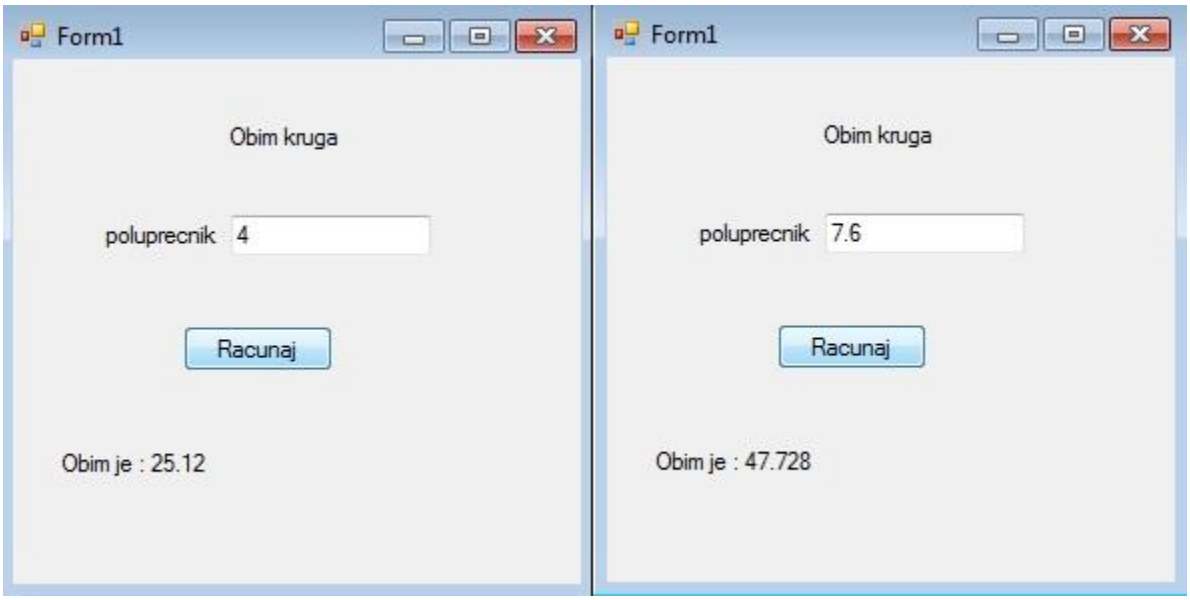
Слика 29. Изглед почетне форме



Слика 30. Изглед форме након преименовања компоненти

Алгоритам 9. Обим круга за полупречник који се задаје из textBox-а

```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljive koje se koristie
    double r;
    double Pi = 3.14;
    //Promenljiva u koju se smesta rezultat
    double obim;
    r = double.Parse(textBox1.Text);
    obim = 2 * r * Pi;
    label3.Text = "Obim je : "  + obim;
}
```



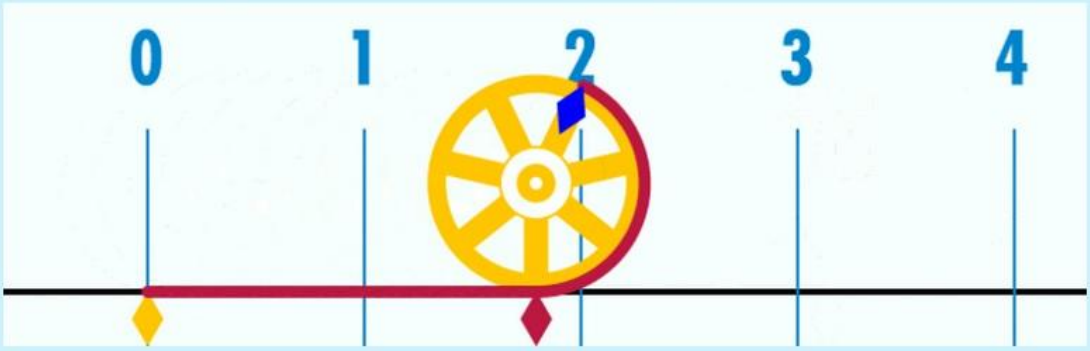
Слика 31. Изглед форме након покретања програма

На слици 32 се види како је пример 4.7 представљен на електронском курсу.

Пример 4. **Написати програм који рачуна обим круга за полупречник који се задаје из textBox-а, при чему је Pi~3.14.**

Решење :

За решавање овог задатке биће нам потребне три лабеле, једно дугме и један textBox. Као и у претходном задатку креирамо променљиву r у коју смештамо оно што корисник унесе у textBox а вредност променљиве Pi постављамо на 3,14. Креирамо и променљиву obim у коју кликом на дугме смештамо израчунат обим круга за задато r.



Слика 32. Изглед примера за обим круга у електронском курсу

Логички тип

Логички типови података служе за представљање резултата логичких исказа који могу бити тачни или нетачни. Логички тип је bool и може да има само две вредности: true (тачно) и false (нетачно) и заузима један бајт.

Табела 8. *Логички тип и његове вредности*

Логички тип		
C# тип	**Коришћено бајтова**	**Вредности**
bool	1	true или false

Пример 5.1 Додељивање типа bool променљивој

```
bool iskaz = true;
```

Већи број примера везаних за логички тип се може пронаћи на веб адреси [15].

Знаковни тип

Знаковни типови података служе за представљање слова, цифара и специјалних знакова који се могу јавити у разним текстовима.
Знаковни тип података char представља 16-битне UNICODE знакове (UNICODE је стандард за електронско кодирање већине светских писаних језика).

Табела 9. *Знаковни тип и његове вредности*

Знаковни тип		
C# тип	**Коришћено бајтова**	**Вредности**
char	2	16-битни UNICODE знак

Пример 6.1 Додељивање знака типу char између једноструких навода.

```
char mojZnak = 'A';
```

Типу char може се доделити и escape знакови. У табели 10 приказани су escape знакови које дозвољава C#, као и њихова значења.

Табела 10. *Escape знакови које дозвољава C#*

Escape знакови	
Escape знак	**Опис**
\ ‘	Једноструки наводник
\ “	Двоструки наводник
\ \	Усправна коса црта
\ 0	Нула
\ a	упозорење
\ b	Померање за једно место уназад
\ f	Команда која даје инструкције штампачу да отпусти тренутни лист
\ n	Нова линија
\ r	Знак који враће курсор или главу штампача на почетак линије
\ t	Хоризонтално увлачење
\ v	Вертикално увлачење

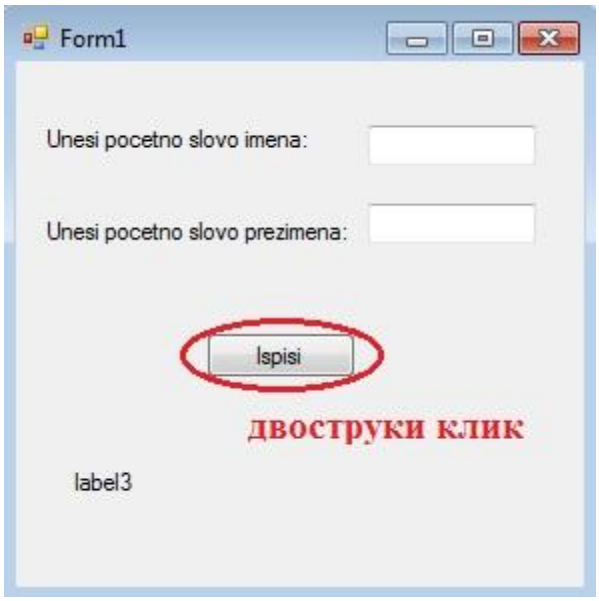
Пример 6.2 Додељивање знака за хоризонтално увачење променљивој mojZnak.

```
char mojZnak = '\ t';
```

Пример 6.3 Стаставити програм који за унета почетна слова имена и презимена корисника преко textBox–ова, кликом на дугме враћа његове иницијале.

Решење : Треба направити две променљиве ime и prezime у које ће се смештати слова која корисник унесе. Како се ради о словима, њихов тип ће бити char. За решавање овог задатка потребне су и три лабеле, два textBox–а и једно дугме. Неке од њих се могу својевољно преименовати или као на слици 33. У променљиву ime смешта се вредност из првог textBox–а, у који корисник уноси прво слово свог имена, а у променљиву prezime

смешта се вредност из другог textBox–а, у који корисник уноси почетно слово свог презимена.



Слика 33. Изглед форме пре покретања програма

Алгоритам 10. Исписивање иницијала имена и презимена

```
private void button1_Click(object sender,  EventArgs e)
{
    //Promenljivoj ime dodeljuje se tip char kao i promeljivoj
    //prezime
    char ime;
    char prezime;

    //Vrednost koju je korisnik uneo u textBox smesta se u
    //promenljivu ime koja je tipa char
    //ali pre toga se mora izvrsiti konvertovanje. Isto
    // se ponovi za promenljivu prezime .
    ime = char.Parse(textBox1.Text);
    prezime = char.Parse(textBox2.Text);

    //Ispisivanje promenljivih ime i prezime u labelu
    label3.Text = "Vasi inicijali su: "  + ime + " " + prezime;
}
```

Слика 34. Изглед форме након покретања програма

Већи број примера везаних за тип char се може наћи на веб адреси [13].

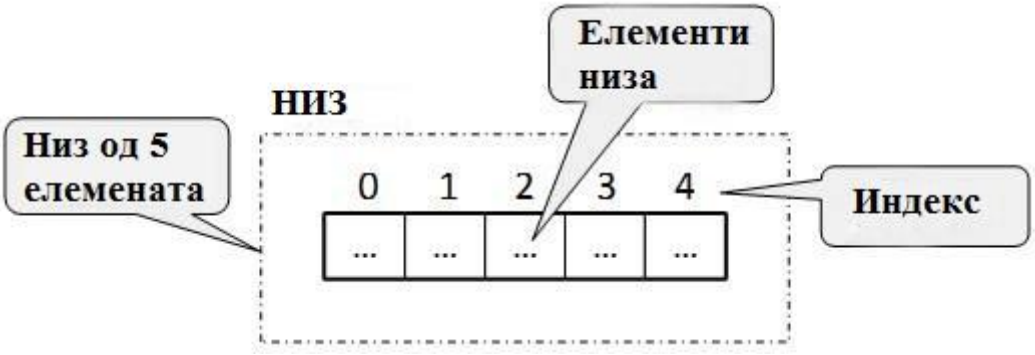
Низовни тип

Огроман је број проблема који би се употребом простих променљивих врло тешко решио или се уопште не би могао решити. Често је потребно у програму декларисати и више хиљада променљивих. Увођење толико променљивих на уобичајан начин практично је неизводљив. Због тога се у програмским језицима уводи појам *низа*, тј. *низовни тип***.**

Низовни тип описује ограничен уређен скуп променљивих истог типа, које се називају компоненте низа. Један такав низ је представљен на слици 35. Он представља скуп променљивих истог типа чије су компоненте троуглови.

Слика 35. Пример низа троуглова

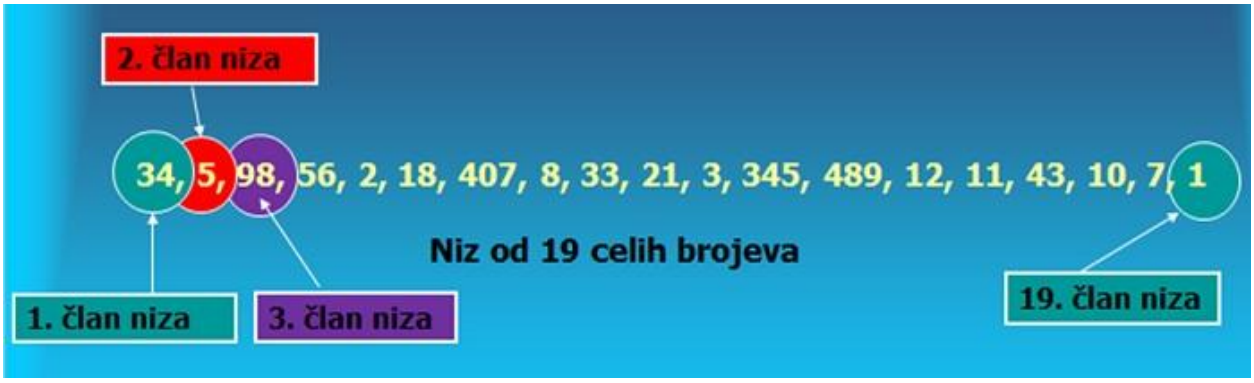
Сваки податак у низу се назива *елементом* низа. Сваки елемент има свој *индекс*, односно објекат преко којег приступамо том елементу у низу.



Слика 36. Основни елементи низа

Број елемената у низу представља његову дужину. Тако је дужина низа са слике 36 једнака 5. Елементи низа нумерисани су са 0, 1, 2, 3, 4. Ови бројеви представљају индексе елемената низа.

Низ може бити једнодимензионалан (када се једноставно назива низ), дводимензионалан (када се назива матрицом, због аналогије са истоименим математичким појмом) и вишедимензионалан. Код једнодимензионалног низа, димензија се поистовећује са дужином низа, нпр. каже се да је низ димензије n. Код вишедимензионалних низова не постоји појам дужине, него се увек каже да је низ димензија m x n x ... x z или (m, n, ..., z).
За низ је битно познавати његову димензију да би се могао исправно индексирати односно приступати његовим елементима.
Елементима низа се приступа преко заједничког имена (назив низовне променљиве) и фиксног броја индекса. Стога се низ може посматрати као група индексираних променљивих истог типа са заједничким именом.



Слика 37. Једнодимензиони низ чији су елементи цели бројеви

Сваки елемент низа има онолико индекса колико сам низ има димензија. Тако елемент једнодимензионалног низа има један индекс, а n-димензионални низ има n индекса. Детаљније о низовима у књизи [5] и на веб адреси [17].
У случају једнодимензионалних низова (слика 37), дужина низа се поистовећује са бројем његових елемената, а индекс елемента представља редни број елемента у низу, при чему је први индекс нула.

Декларација низова у C#-у:

<тип елемента> <назив_променљиве>[<величина_низа>];

Пример 7.1 Декларација низа mojNIz од 6 целобројних елемената.

```
int mojNiz[6];
```

У примеру променљива mojNiz је низ који је типа (int []), односно низ целих бројева који има шест елемената. У C#-у креирање низа односно алокација меморије се врши помоћу речи new.

```
int[] mojNiz = new int[6];
```

Овим се креира низ од 6 елемената који су цели бројеви. Пре коришћења елемената низа, њима је потребно доделити неку вредност. Креирањем низа те вредности биће једнаке нули. Почетне вредности елемената можемо доделити на различите начине. Један од њих је:

```
int[] mojNiz = {1, 2, 3, 4, 5, 6};
```

Пример 7.2 Креирање и иницијализација низа које је извршено истовремено.

```
mojNiz[0]=1;
mojNiz[1]=2;
    .
    .
    .
```

Пример 7.3 Декларација дводимензионалних низова.

```
int[,] intMatrica;
float[,] floatMatrica;
String[,,] strNiz;
```

У примеру 7.3 прва два низа су дводимензионална док је трећи тродимензионалан. Креирање вишедимензионланих низова врши се такође помоћу речи new.

Пример 7.4 Креирање тродимензионалног низа.

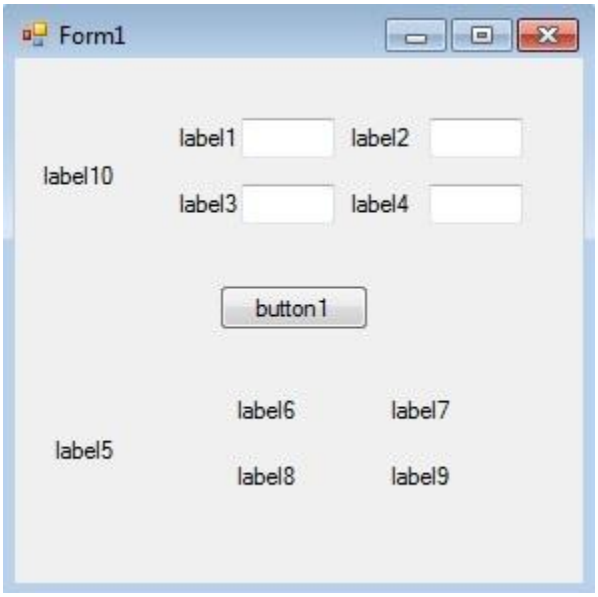
```
int[,] intMatrica =
{
      //Red 0 vrednosti
      {1, 2, 3, 4},
      //Red 1 vrednosti
      {5, 6, 7, 8},
      //Red 2 vrednosti
      {9, 10, 11, 12},
};

      //Matrica je dimenzija 3 x 4 (3 reda, 4 kolone)
```

Елементима низа се приступа користећи сличну синтаксу као и код декларације, користећи оператор средње заграде:

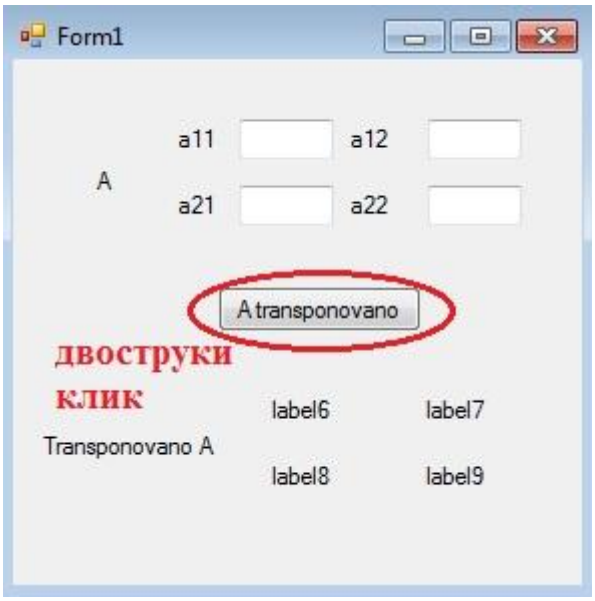
```
//Deklaracija jednog niz
int niz[20];
//Deklaracija jedne pomocne promenljive
int x;
//Dodeljivanje broja 17 prvom elementu niza
niz[0] = 17;
//Dodeljivanje promenljivoj x vrednost prvog elementa niza
x = niz[0];
```

Пример 7.5 Направити програм који за унете елементе матрице А димензије 2x2, израчунава А транспоновано.

Решење:
За решавање овог задатка биће потребно десет лабела, једно дугме и четири textBoxa-а. Како матрица А има четири елемента, треба направити низ реалних бројева у који ће се ти елементи смештати. Лабеле и дугме се могу преименовати као на слици 39.



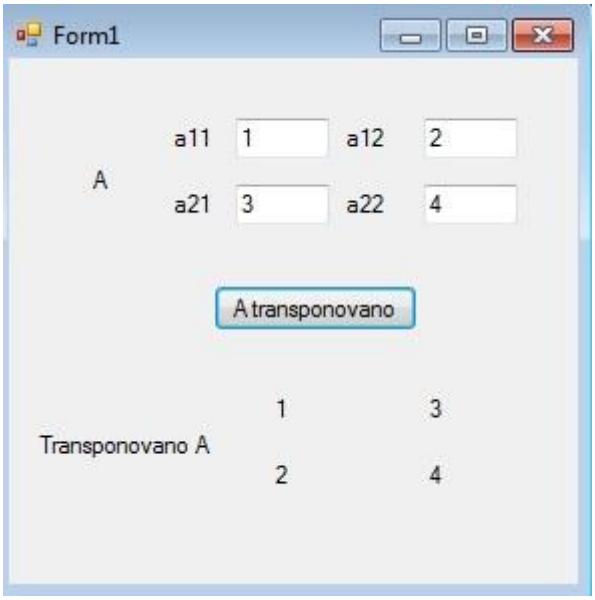
Слика 38. Изглед почетне форме



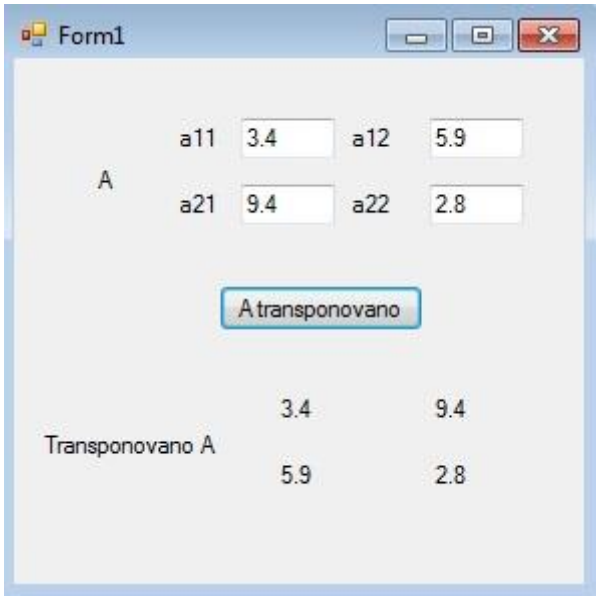
Слика 39. Изглед форме након преименовања компоненти

Алгоритам 11. За унете елементе матрице A димензије 2x2 , рачуна A транспоновано

```
float[] clanovi = new float[4];

private void button1_Click(object sender, EventArgs e)
{
    clanovi[0] = float.Parse(textBox1.Text);
    clanovi[1] = float.Parse(textBox2.Text);
    clanovi[2] = float.Parse(textBox3.Text);
    clanovi[3] = float.Parse(textBox4.Text);

    label6.Text = " "  + clanovi[0];
    label7.Text = " "  + clanovi[2];
    label8.Text = " "  + clanovi[1];
    label9.Text = " "  + clanovi[3];
}
```



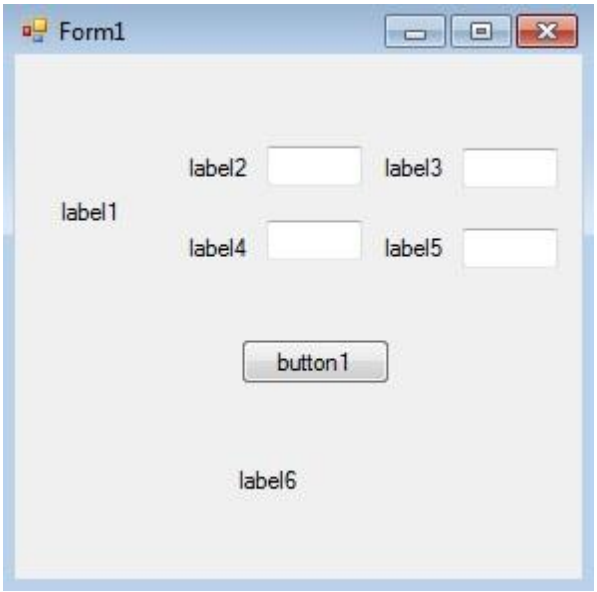
Слика 40. Изглед форме након покретања програма



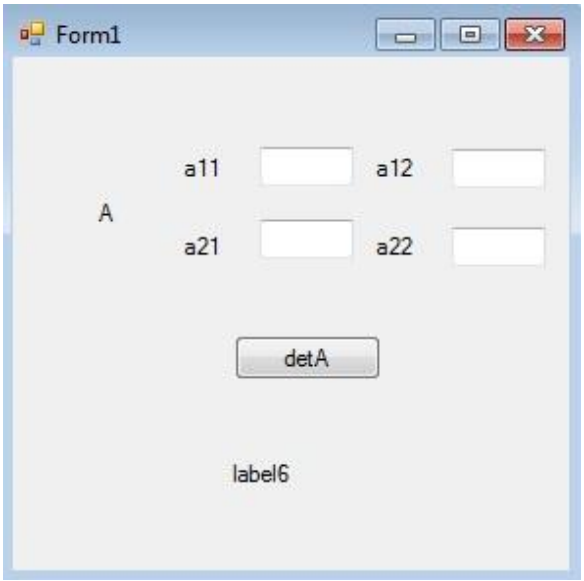
Слика 41. Изглед форме након покретања програма

Пример 7.6 Направити програм који за унете елементе матрице A димензије 2x2, рачуна детерминанту матрице A.

Решење: За решавање овог задатка биће потребно шест лабела, једно дугме и четири textBox-а. Како матрица A има четири елемента треба направити низ реалних бројева у који ће се ти елементи смештати. Лабеле и дугме се могу преименовати као на слици 43.



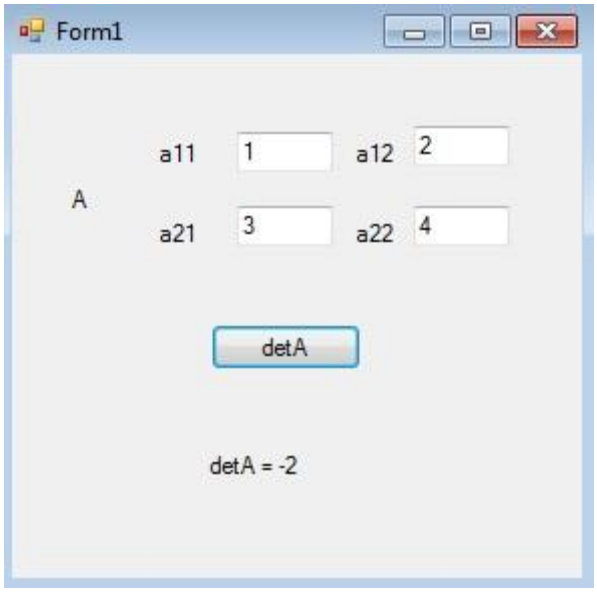
*Слика 42.*Изглед почетне форме



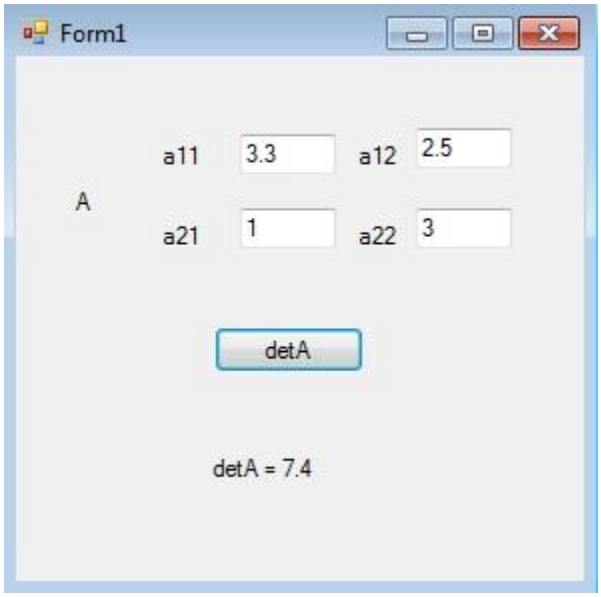
Слика 43. Изглед форме након преименовања компоненти

Алгоритам 12. За унете елементе матрице A димензије 2x2, рачуна детерминанту матрице A

```
float[] clanovi = new float[4];
float det;

private void button1_Click(object sender, EventArgs e)
{
   //Clanovima niza se dodeljuju vrednosti unete u textBox-ove
    clanovi[0] = float.Parse(textBox1.Text);
    clanovi[1] = float.Parse(textBox2.Text);
    clanovi[2] = float.Parse(textBox3.Text);
    clanovi[3] = float.Parse(textBox4.Text);
   //U promenljivu det se smesta vrednost determinante matrice A
    det = clanovi[0]*clanovi[3] - clanovi[1]*clanovi[2];
    label6.Text = "detA = "  + det;
}
```

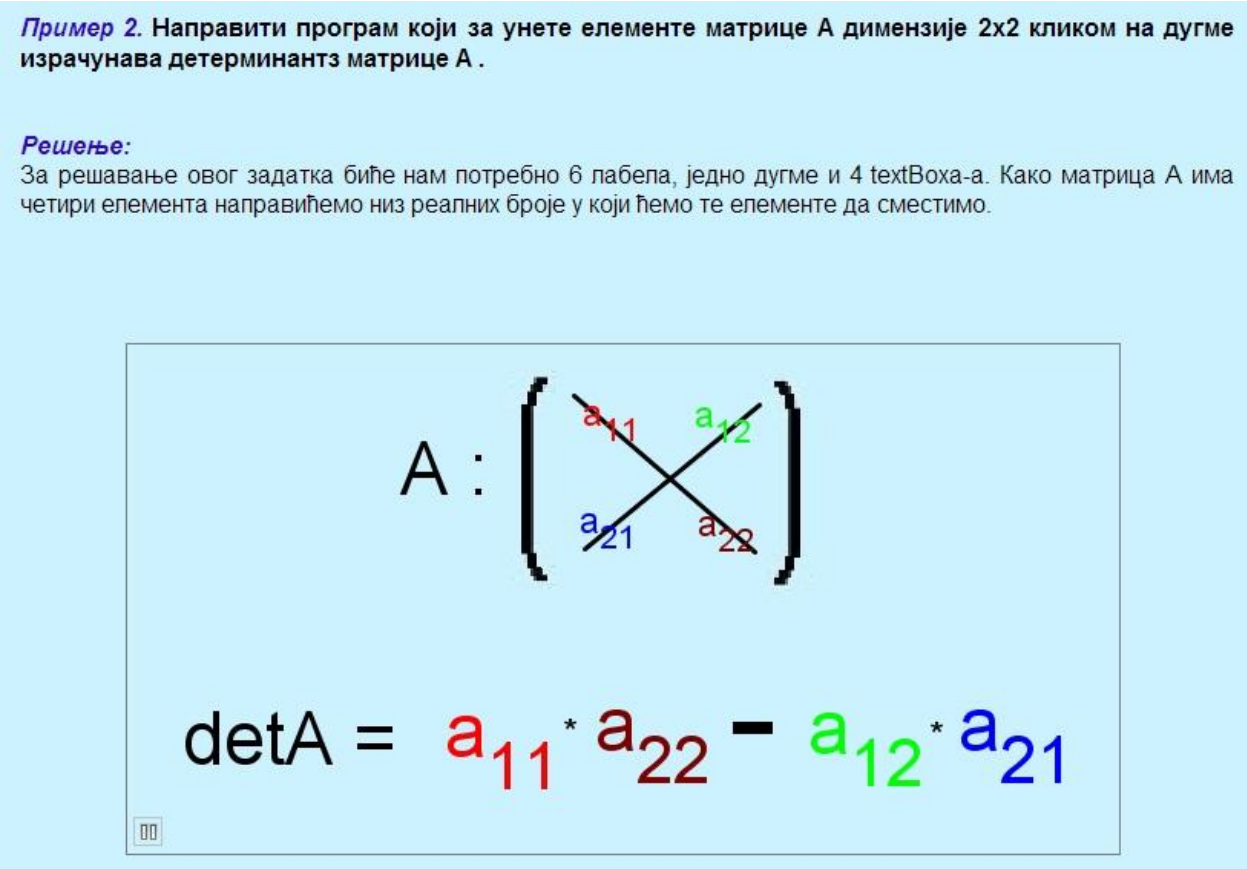


Слика 44. Изглед форме након покретања програма



Слика 45. Изглед форме након покретања програма

У електронском курсу се може видети анимација за рачунање детерминанте матрице која може помоћи у идеји за прављење овог програма (слика 46).



Слика 46. Изглед прозора у електронском курсу на коме је анимација за рачунање детерминанте матрице

Стринг тип

Превод речи string на наш језик био би *ниска*. У програмирању и другим гранама математике, ниска (string) је уређени низ симбола. Стрингови су дакле, низови UNICODE знакова. Стринг је погодан за карактер-по-карактер обрађивање и по томе се разликује од класичних низова. У овом контексту, стринг не мора репрезентовати текст. За промењљиву која је декларисана типом string, обично се алоцира довољно меморије да се „ускладишти" одређена количина симбола. Технички, ниске су представљене као низови карактера на чији се десни крај дописује карактер '\0' (тзв. терминална нула). Из овога разлога, ниске у C#-у се називају ниске терминисане нулом. Последица овога је да не постоји ограничење за дужину ниске у C#-у, али да је неопходно проћи кроз целу ниску како би се одредила њена дужина. Неке ниске су већ унапред дефинисане, као и њихова вредност и значење. Ниске у програмском језику C# се наводе између двоструких наводника.

```
string niska = " Zdravo svete ! ";
```

Промењљива типа String се може затим исписати у конзоли коришћењем следећег позива методе који прихвата string као улаз:

```
System.Console.WriteLine(niska);
```

String се може креирати и од других мањих стрингова, коришћењем оператора +.

Пример 8.1 Креирање стрингова помоћу оператора +.

```
string niska1 = " Zdravo svete ! ";
string niska2 = " iz C# ! ";
string niska3;
niska3 = niska1 + niska1;
System.Console.WriteLine(niska3);
```

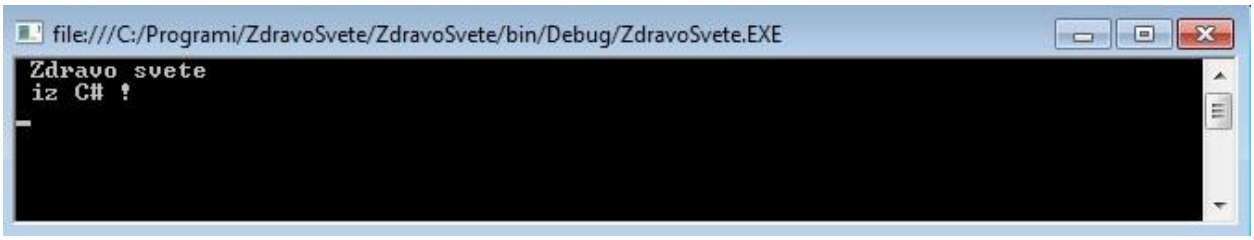
У промењљиву типа стринг се такође могу уградити и escape знакови. Ради подсећања који су то escape знакови које дозвољава C# треба погледати табелу 10.

Пример 8.2 Направи програм који у конзоли исписује реченицу „Здраво свете из C#!" користећи escape знакове.

Решење:

Алгоритам 12. Исписивање реченице „ Здраво свете из C#!" у конзоли

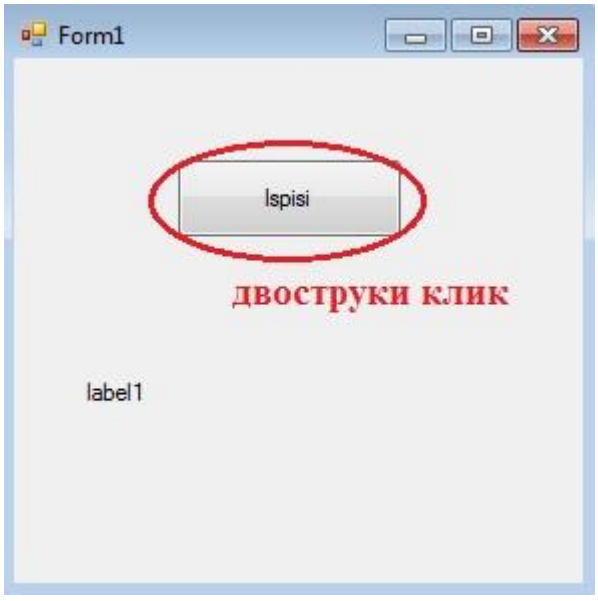
```
static void Main(string[] args)
{
    string niska = " Zdravo svete  " + " \n iz C# ! ";
    System.Console.WriteLine( niska );
}
```



Слика 47. Изглед конзоле након покретања програма

Пример 8.3 Написати програм који притиском на дугме исписује у лабелу текст из ниске унете у коду програма.

Решење : За решавање овог задатка биће потребна једна лабела и једно дугме. Име дугмета се може преименовати у „Ispisi" као на слици 48.



Слика 48. Изглед форме пре покретања програма

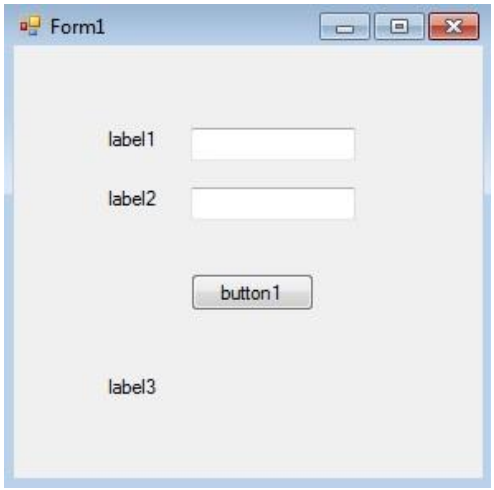
Алгоритам 12. Исписивање текста из ниске

```
private void button1_Click(object sender,  EventArgs e)
{
    string niska = " Zdravo svete  " + " \n iz C# ! ";
    label1.Text = niska;
}
```

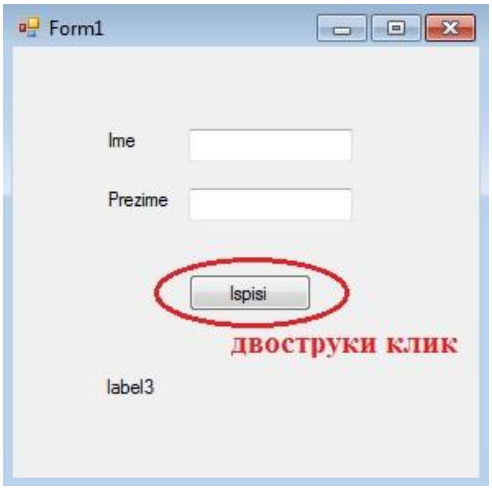
Слика 49. Изглед форме након покретања програма

Пример 8.4 Направити програм који кликом на дугме исписује име и презиме у label – и, које корисник унесе са тастатуре у textBox - ове.

Решење: За решавање овог задатка биће потребне две променљиве типа string. Једна се назива ime а друга prezime. Од компоненти користиће се три лабеле, два texBox-а и једно дугме. Компоненте се могу преименовати као на слици 51. Креирају се променљиве ime и prezime које су типа string и у њих се смешта оно што корисник унесе у texBox-ове. Кликом на дугме у лабели се исписују име и презиме које је корисник унео.



Слика 50. Изглед почетне форме

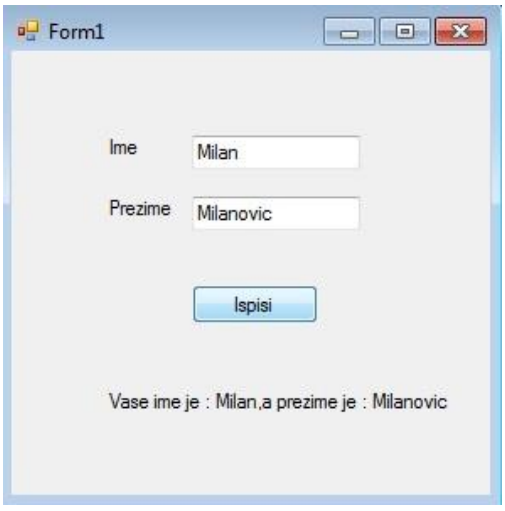


Слика 51. Изглед форме након преименовања компоненти

Алгоритам 12. Исписивање имена и презимена унешених преко textbox-а

```
    //Kreiraju se promenljive ime i prezime tipa String
    String ime ;
    String prezime ;
private void button1_Click(object sender, EventArgs e)
{
    //U promenljive ime i string smesta se ono sto je korisnik
    //uneo u textBox1 i textBox2
    ime = textBox1.Text ;
    prezime = textBox2.Text ;
    label3.Text ="Vase ime je :" + ime +",a prezime je : "+
    prezime;
}
```

Слика 52. Изглед форме након покретања програма



Слика 53. Изглед форме након покретања програма

C# има богат скуп функција за манипулисањем стринговима тј. у C#-у су уграђене методе за рад са типом стринг. У следећим примерима биће илустрована нека својства и неке од метода за рад са стринговима.

Коришћење својства lenght

Својство lenght се користи да би се добио број знакова у стрингу. Својство lenght враћа вредност типа int. У следећем примеру се показује на који начин се може видети колики је број знакова у стрингу. Направи се једна променљива recenica у којој треба исписати „Бити или не бити!". Треба да се у конзоли испише колико та реченица садржи знакова.

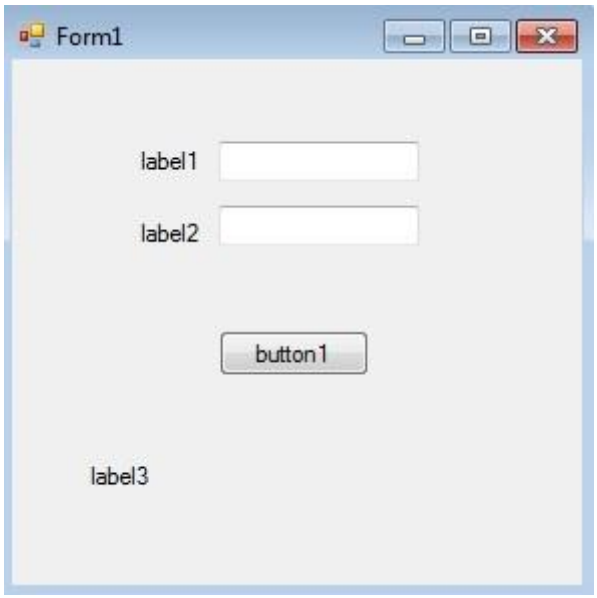
Алгоритам 13. Исписивање броја знакова у реченици

```
static void Main(string[] args)
{
    //Promenljivu recenica koja je tipa string se postavlja na
    //zeljenu recenicu Biti ili ne biti !
    string recenica = "Biti ili ne biti!";

    //Treba da se u konzoli ispise broj znakova u toj
    //recenici.
    //To se postize koriscenjem svojstva length na sledeci
    //nacin :
    System.Console.WriteLine(recenica.Length);

}
```

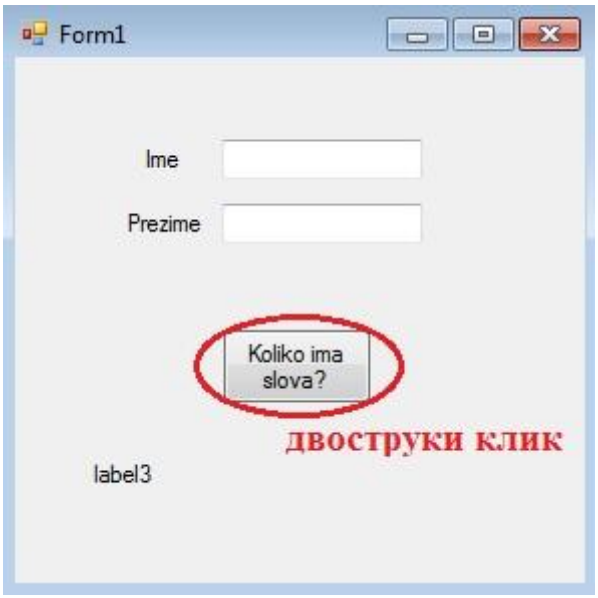
Покретањем овог кода, у конзоли ће се појавити исписан број 17. Програм је избројао све знакове у реченици "Бити или не бити!" у којој се поред слова као знакови рачунају и белине и знаци интерпукције. Зато је 17 број занкова у реченици.

Пример 8.5 Направити програм који кликом на дугме рачуна број слова у имену и презимену које корисник унесе са тастатуре преко textBox - ова и исписује га у лабели.

Решење :
За решавање овог задатка потребне су две променљиве типа string. Једна ће се назвати ime, а друга prezime. У трећу променљиву duzina смешта се број слова у имену и презимену које је корисник унео користећи својство length. Компоненте које се користе се могу преименовати као на слици 55.



Слика 54. Изглед почетне форме

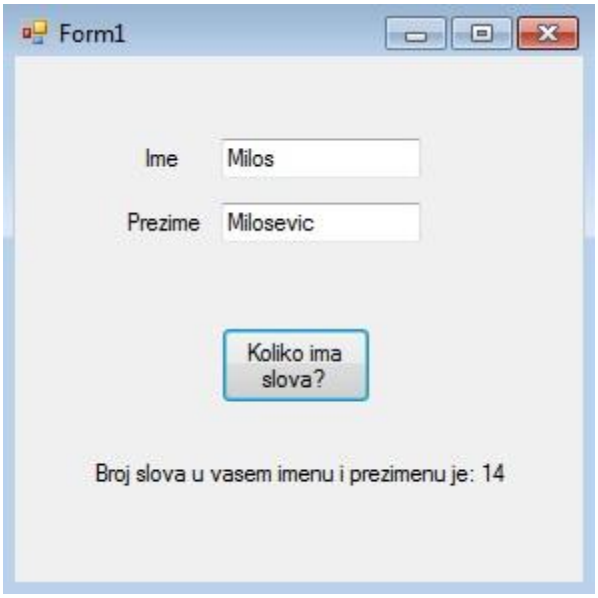


Слика 55. Изглед форме након преименовања компоненти

Алгоритам 13. Рачунање броја слова у имену и презимену унесених преко textbox-а

```
    //Kreira se promenljiva ime i prezima tipa String i
    //promenljiva duzina tipa int
    String ime;
    String prezime;
    int duzina;
private void button1_Click(object sender, EventArgs e)
{
    //U promenljive ime i prezime smesta se ono sto je korisnik
    //uneo u textBox1 i textBox2
    ime = textBox1.Text;
    prezime = textBox2.Text;
    //U promenljivu duzina smesta se broj karaktera u
    //promenljivoj ime i broj karaktera
    //u promenljivoj prezime koristeci svojstvo Length
    duzina = duzina = ime.Length + prezime.Length;
    label3.Text ="Broj slova u vasem imenu i prezimenu je: " +
duzina;
}
```

Слика 56. Изглед форме након покретања програма



Слика 57. Изглед форме након покретања програма

Поређење два стринга коришћењем методе Comapre()

Метода Comapre() може се користити за поређење знакова смештених у два стрига. Метода Comapre() враћа вредност типа int који указује на то да ли је први стринг већи, једнак или мањи од другог стринга. На пример стринг ,,abc" алфабетски је мањи од стринга ,,bbc" јер при упоређивању првог карактера у стринговима карактер ,,a" првог стринга је алфабетски мањи од првог карактера ,,б" другог стринга, па је самим тим стринг ,,abc" мањи од стринга ,,bbc".
Поређење се врши алфабетски над словима и нумерички над било којим бројем који се појави у стрингу. Метода Comapre() користи следеће правила за одређивање вредности, типа int, која се враћа:

1) Ако је први стринг већи од другог стринга, враћа се 1;
2) Ако је први стринг једнак другом стринга, враћа се 0;
3) Ако је први стринг мањи од другог стринга, враћа се -1.

Најједноставнија верзија методе Comapre() прихвата два параметра типа string и користи следећу синтаксу :

String.Compare(niska1 , niska 2);

где су niska1 и niska2 стрингови који се желе упоредити.

Пример 8.6 Коришћење методе Comapre()

```
static void Main(string[] args)
{
    //Uporedjuju se dva stringa, string "bbc" sa stringom
    //"abc"
    //S ozirom da je "bbc" alfabetski veci od stringa "abc",
    //metoda Comapre() vraca vrednost 1
    //koja se zatim smesta u promenljivu rezultat koja je tipa
    //int
    int rezultat;
    rezultat = String.Compare("abc", "bbc" );
}
```

Коришћење малих и великих слова при поређењу стрингова

Још једна верзија методе Compare() поред две променљиве типа String, прихвата параметар типа bool који специфицира да ли се у поређењу жели користити и величина слова. Ова верзија методе Compare() користи следећу синтаксу:

String.Compare(niska1 , niska 2 , ignorisiSlova);

ignorisiSlova је променљива типа bool, којом се назначује да ли се у поређењу користи и величина слова стрингова niska1 и niska 2. Ако се вредност променљиве ignorisiSlova постави на true, онда се величина слова два стринга не разматра при поређењу (ово је подразумевана вредност). Ако се ignorisiSlova постави на false, онда се при поређењу разматра величина слова.
У примеру 8.7 метода Compare() враћа 0, јер се није узимала величина слова при поређењу стрингова „bbc" и „BBC".

Пример 8.7 Коришћење методе Compare() без узимања у обзир величине слова при поређењу.

```
static void Main(string[] args)
{
    //Uporedjuju se dva stringa, string "BBC" sa stringom "bbc"
    //ali se pri tom ne vodi racuna o velicini slova.
    //Koristi se metoda Compare() koja
    //prima 3 parematrea , od koga su prva dva tipa string, a
    //trci je tipa bool i odredjuje
    //da li ce se velicina slova uzimati u obzir ili ne
    //Vrednost koju metoda Compare() vraca, smesta se u
    //promenljivu rezultat tipa int
    int rezultat ;
    rezultat = String.Compare("bbc", "BBC" ,true);
    //Vrednost promenljive rezultat je 0 jer je odluceno da se
    //u poredjenje ne ukljucuje
    //velicina slova
}
```

Пример 8.8 илуструје методу Compare(), враћа -1, јер када се не игнорише величина слова „bbc" је мање од „BBC".

Пример 8.8 Илустрација методе Compare() при игнорисању величине слова.

```
static void Main(string[] args)
{
    int rezultat ;
    rezultat = String.Compare("bbc", "BBC" ,false);
    //Vrednost promenljive rezultata je -1 jer je odluceno da se
    //u poredjenje ukljuci velicina slova

}
```

Конверзија величине слова стринга коришћењем метода ToUpper() и ToLower()

Методе ToUpper() и ToLower() могу се користити да би се конвертовала величина слова у стрингу. Метода ToUpper() враћа стринг са свим великим словима, а метода ToLower() враћа стринг са свим малим словима. Обе методе враћају нови стринг. Најједноставније верзије ових метода користе следећу синтаксу:

niska1.ToUpper()
niska2.ToLower()

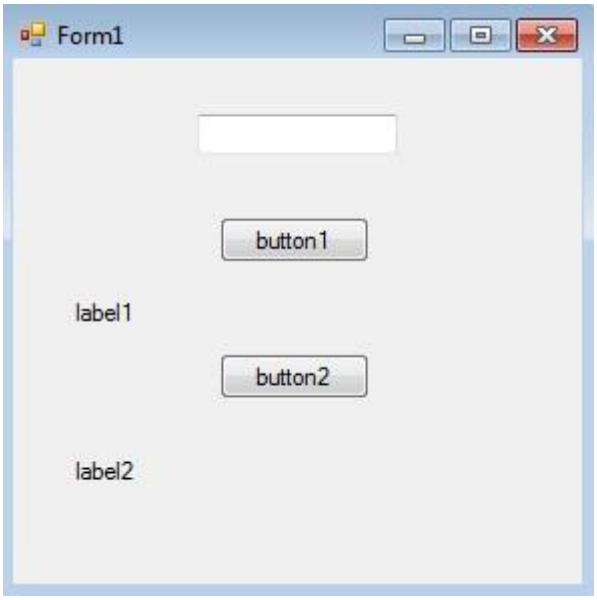
Пример 8.9 Коришћење методе ToUpper() и ToLower().

```
//U promenljivu novaNiska1 smesta se sve iz niska1 samo
//napisano VELIKIM slovima
String novaNiska1 = niska1.ToUpper();
//U promenljivu novaNiska2 smesta se sve iz niska2 samo
//napisano malim slovima
String novaNiska2 = niska1.ToLower();
```

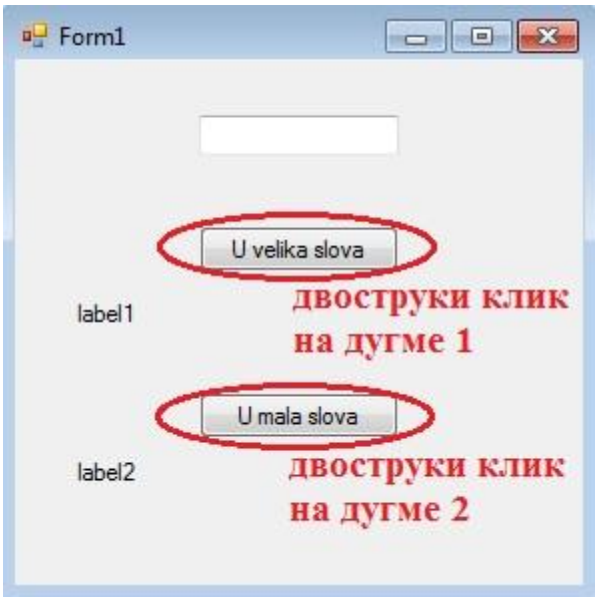
Ако је промељива niska1 била постављена на „Biti ili ne biti" онда предходне линије кода дају нове променљиве novaNiska1 и novaNiska2 које су постављене на следеће стрингове респективно: „BITI ILI NE BITI", „biti ili ne biti".

Пример 8.10 Направити програм који кликом на дугме, реченицу која је унета у textBox исисује малим словима у лабели. У истој форми креирати и друго дугме које када се кликне на њега исписује реченицу у којој су сва слова велика.

Решење: За решавање овог задатка биће потребна променљива типа string у коју ће се сместити реченица коју корисник унесе у textBox. Конвертовање слова у мала или у велика постиже се коришћењем метода ToUpper() и ToLower(). Компоненте се могу преименовати као на слици 59.



Слика 58. Изглед почетне форме



Слика 59. Изглед форме након преименовања компоненти

Алгоритам 14. Унету реченицу исписује малим и великим словима

```
    //Kreira se promenljiva recenica tipa String
    String recenica;

private void button1_Click(object sender, EventArgs e)
{
    //U promenljive recenica smesta se ono sto je korisnik uneo
    //u textBox1
    recenica = textBox1.Text;
    //U label1 se ispisuje recenica velikim slovima
    label1.Text = recenica.ToUpper();

}
private void button2_Click(object sender, EventArgs e)
{
    //U label2 se ispisuje recenica malim slovima
    label2.Text = recenica.ToLower();
}
```

Слика 60. Изглед форме након покретања програма

Слика 61. Изглед форме након покретања програма

Повезивање сринговa коришћењем методе Concat()

Метода Concat() се може користити за повезивање стрингова. Метода Concat() враћа нови стринг који се добија тако што на крају претходног стринга додаје сваки нови обезбеђени стринг. Њена најједноставнија верзија прихвата два стринга коришћењем следеће синтаксе:

String.Concat(niska1, niska2)

niska1 и niska2 су стрингови који се желе повезати.
У примеру 8.11 коришћена је ова верзија методе Concat() како би се повезала два стринга, „Dobar" и „dan", при чему се стринг који се враћа смешта у промењиву novaNiska.

Пример 8.11 Повезивање два стринга коришћењем методе Concat().

```
//Pravi se nova promenljiva novaNiska u koju se smesta ono sto
//se zeli povezati
string novaNiska = String.Concat("Dobar ", "dan");
```

Стринг novaNiska биће постављен на „Dobar dan".

Методи Concat() може се проследити било који број стирнгова које се жели повезати и следећој верзији методе Concat( ) прослеђена су четири стринга:

```
string novaNiska1 = String.Concat("Dobar ", "dan", " svima", "
!");
```

Стринг novaNiska1 биће постављен на „Dobar dan svima!".

Повезивање сринговa коришћењем оператора сабирања

За повезивање стрингова може се користити оператор сабирања (+).

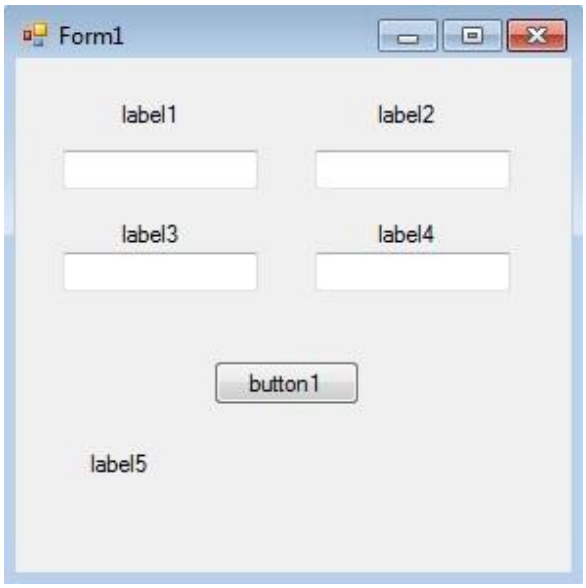
Пример 8.12 Повезивање стрингова коришћењем операторта сабирања (+).

```
    string novaNiska = "Dobar" + " dan" + " svima" + " !";
```

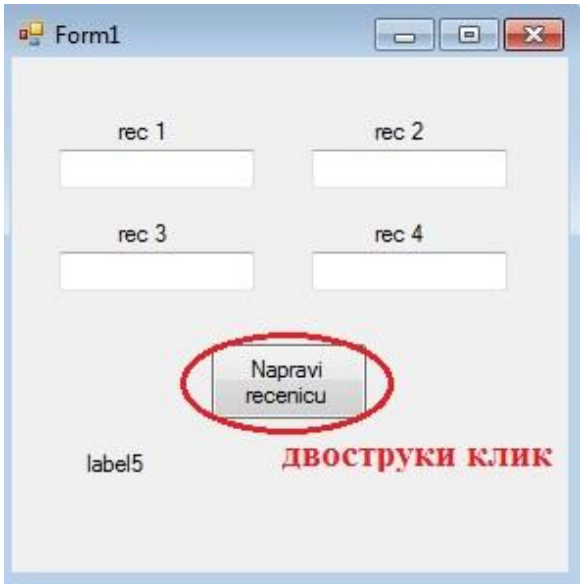
Стринг novaNiska биће постављен на „Dobar dan svima!".

Пример 8.13 Направити програм који кликом на дугме повезује четири речи у реченицу које корисник унесе са тастатуре у textBox – ове, а затим ту реченицу исписује у лабели.

Решење: За решавање овог задатка биће потребна четири textBox-а, пет лабела и једно дугме. Треба направити пет променљивих: rec1, rec2, rec3, rec4 и recenica типа String у које ће се смештати оно што корисник унесе у textBox-ове, а у променљиву recenica ће се смештати реченица направљена од садржаја смештеног у променљиве rec1, rec2, rec3 и rec4. За повезивање променљивих користи се метод Concat(). Компоненте се могу преименовати као на слици 63.



*Слика 62.*Изглед почетне форме



Слика 63. Изглед форме након преименовања компоненти

Алгоритам 15. Повезивање речи у реченицу

```
    //Kreira se promenljiva rec1, rec2, rec3, rec4 i recenica
    //tipa String
    String rec1, rec2, rec3, rec4, recenica;

private void button1_Click(object sender, EventArgs e)
{
    //U promenljive rec1, rec2, rec3, rec4 i recenica se
    //smesta ono sto je korisnik uneo u
    //odgovarajuce textBox - ove
    rec1 = textBox1.Text;
    rec2 = textBox2.Text;
```

```
    rec3 = textBox3.Text;
    rec4 = textBox4.Text;
    //U promenljivu recenica se koristeci metod Concat(),
    //smestaju niske rec1, rec2, rec3 i rec4
    recenica = String.Concat(rec1, rec2, rec3, rec4);
    label5.Text = recenica;
}
```

Слика 64. Изглед форме након покретања програма

Слика 65. Изглед форме након покретања програма

Копирање стрингова коришћењем методе Copy()

Метода Copy() се користи да би се ископирао неки жељени стринг. Она користи следећу синтаксу :

String.Copy(niska1)

при чему је niska1 стринг који се жели ископирати.

Пример 8.14 приказује коришћење методе Copy() за копирање стринга niska1 у стринг niska2.

Пример 8.14 Копирање једне ниске у другу коришћењем мотоле Copy().

```
string niska2 = String.Copy(niska1);
```

Копирање стрингова коришћењем оператора додељивања

Стрингови се могу копирати коришћењем оператора додељивања (=) којим се копира један стринг у други.

Пример 8.15 Копирање стрингова коришћењем оператора =.

```
string niska1 = "Zdravo svete !";
string niska2 = niska1;
```

Стринг niska2 такође ће бити постављен на "Zdravo svete!" .

Провера да ли су два стринга једнака коришћењем методе Equals()

Метода Equals() се користи да би се проверило да ли су два стринга једнака. Метода Equals() враћа вредност типа bool и има статичку верзију и верзију истанце. Статична верзија методе Equals() може се позвати преко класе String и она прихвата два стринг параметра, за које се проверава да ли су исти. Синтакса статичне методе Equals() је:

String.Equals(niska1, niska2);

где су niska1 и niska2 два стринга која се пореде.
У примеру 8.16 метода Equals() враћа true јер су два стринга која се пореде једнака.

Пример 8.16 Поређење два стринга помоћу методе Equals().

```
//Proverava se da li string "abc" jednak stringu "abc"
bool rezulataPoredjenja = String.Equals("abc","abc");
```

Верзија инстанце методе Equals() позива се коришћењем стварног стринга и она пореди стринг са обезбеђеним стринг параметром. Синтакса ове верзије Equals() је:

niska1.Equals(niska2)

niska1 и niska2 су стрингови који се пореде.

Провера да ли су два стринга једнака коришћењем оператора (==)

Да би се проверило да ли су два стринга једнака може се користити оператор једнакости (==).

Пример 8.17 Употреба оператора (==) при поређењу два стринга.

```
//Proverava se da li je string niska1 jednak stringu niska2
string niska1 = "Zdravo";
string niska2 = "svete";
bool rezulataPoredjenja = niska1 == niska2;
```

У овом примеру променљива rezulataPoredjenja добија вредност false, јер је садржај променљивих niska1 и niska2 различит.

Извлачење подстринга из стринга коришћењем методе Substring()

Метода Substring() може се користити да би се из стринга извукао подстринг. Метода Substring() враћа стринг. Најједноставнија верзија методе Substring() враћа подстринг који почиње од специфичног индекса и користи следећу синтаксу

niska1.Substring(index)

где је index вредност типа int која специфира позицију знака од које треба почети са читањем стринга niska1.

У следећем изразу користи се верзија методе Substring(), која враћа подстринг стринга niska1, почев од индекса 5:

```
    string niska2 = niska1.Substring(5);
```

С обзиром да је променљива niska1 постављена на „Biti ili ne biti", променљива niska2 се поставља на „ili ne biti".

Друга верзија методе Substring(), за специфицирање броја знакова које треба прочитати користи још један параметар типа int и користи следећу синтаксу:

niska1.Substring(index, brojKaraktera);

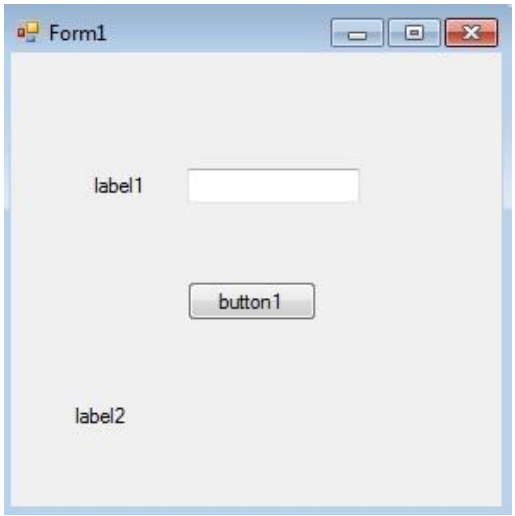
У следећем изразу користи се ова верзија методе Substring():

```
string niska3 = niska1.Substring(5,3);
```

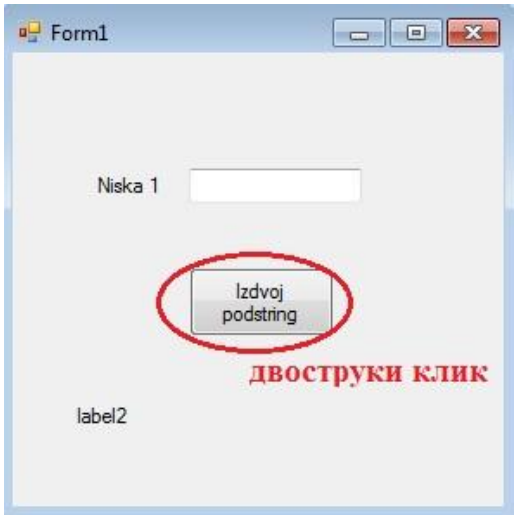
Како је niska1 постављена на „Biti ili ne biti" тада ће променљива niska3 бити постављена на „ili" .

Пример 8.18 Направити програм који када се кликне на дугме извлачи из ниске унешене преко textBox-а све преостале карактере почевши од оног са индексом 4.

Решење : За решавање овог задатка биће потребан један textBox, две лабеле и једно дугме. Треба направити променљиву niska1 типа String у коју ће се сместити оно што корисник унесе у textBox и променљиву niska2 у коју ће се сместити подстринг из променљиве niska1. За извлачење подстринга из стринга niska1 користи се метод Substring(). Компоненте се могу преименовати као на слици 67.



*Слика 66.*Изглед почетне форме



Слика 67. Изглед форме након преименовања компоненти

Алгоритам 16. Извлачење преосталих карактера из задате ниске почевши од оног са индексом 4

```
    //Kreiraju se promenljive niska1 i niska2 tipa String
    String niska1, niska2;
private void button1_Click(object sender, EventArgs e)
{
    //U promenljivu niska1 se smesta ono sto korisnik unese u
    //textBox
    niska1 = textBox1.Text;
    //U promenljivu niska2 se smesta deo niske niska1 metodom
    //Substring()
    niska2 = niska1.Substring(4);
    label2.Text = niska2;
}
```

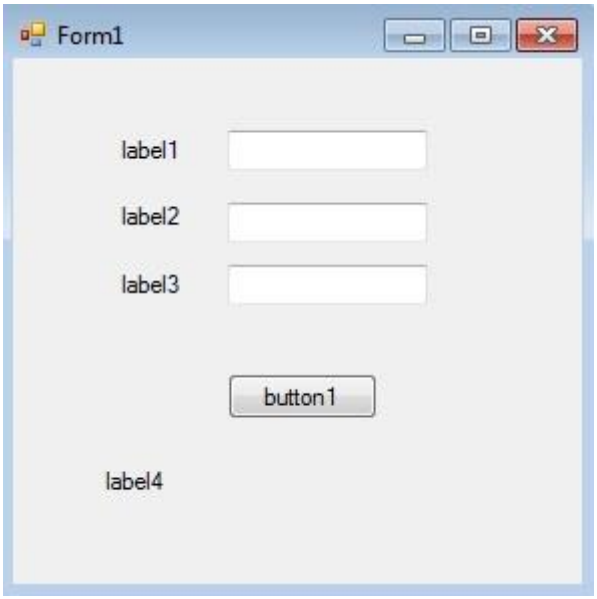
Слика 68. Изглед форме након покретања програма

Слика 69. Изглед форме након покретања програма

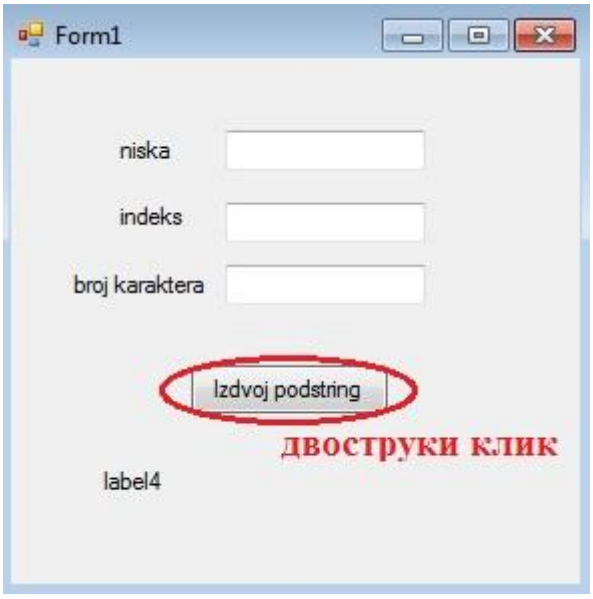
Пример 8.19 Направити програм који кликом на дугме извлачи из ниске унешене преко textBox-а карактере почевши од оног са индексом који корисник унесе у други textBox. Омогућити кориснику да уносе и број знакова које треба издвојити из задатог стринга.

Решење: За решавање овог задатка биће потребна три textBox-а, четири лабеле и једно дугме. Треба направити променљиву niska1 типа String у коју ће се сместити оно што

корисник унесе у textBox1 и промењљиву index у коју ће се сместити индекс који корисник унесе у textBox2 и промењљиву brojKaraktera у коју ће се сместити оно што корисник унесе у textBox3. Кликом на дугме, новодобијена ниска ће се исписати у лабелу. Компоненте се могу преименовати као на слици 71.



*Слика 70.*Изглед почетне форме



Слика 71. Изглед форме након преименовања компоненти

Алгоритам 17. Извлачење преосталих карактера из задате ниске почевши од оног са индексом унешеним преко textbox-а

```
    //Kreiraju se promenljive niska i podstring tipa String i
    //promenljive index i brojKaraktera tipa int
    String niska;
    String podstring;
    int index;
    int brojKaraktera;

private void button1_Click(object sender, EventArgs e)
{
    //U promenljivu niska se smesta ono sto korisnik unese u
    //texbox1
    niska = textBox1.Text;
    //U promenljivu index se smesta ono sto korisnik unese u
    //textBox2 ali
    //to se prvo mora konvertovati u odgovarajuci tip sto je u
    //ovom slucaju int
```

```
    index = int.Parse(textBox2.Text);
    // Slicno se uradi i za promenljivu brojKaraktera
    brojKaraktera = int.Parse(textBox3.Text);


    // Koristeci metod Substring() u promenljivu podstring
    // se smesta string koji se dobija izvlacenjem dela stringa
    // iz promenljive niska
    podstring = niska.Substring(index, brojKaraktera);
    //Ono sto se dobija u promenljivoj podstring, ispisuje se u
    //lebeli
    label4.Text = podstring;
}
```

Слика 72. Изглед форме након покретања програма

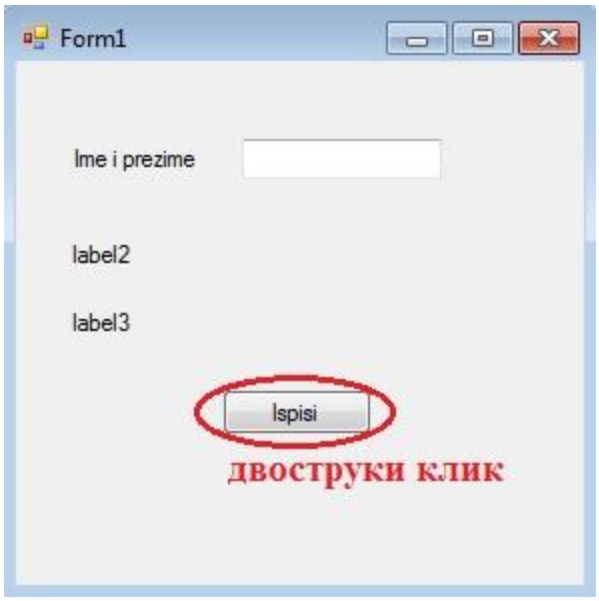
Слика 73. Изглед форме након покретања програма

Метода indexOf()

Метода indexOf() је још једна од метода из класе String. Она враћа индекс првог појављивања специфицираног подстринга знакова унутар стринга. Индекси почињу од нуле, а подстринг је заправо део оригиналног стринга. Тип повратне вредности који враћа ова метода је int.

Пример 8.20 Написати програм који кликом на дугме, за унето име и презиме преко textBox–а издваја име и презиме засебно и исписује их у посебне labele-е.

Решење: За решавање овог задатка биће потребне три лабеле, један textBox и једно дугме. Преко textBox-а се уносе име и презиме, а затим се помоћу методе indexOf() и Substring() издваја засебно име и презиме и смешта у одговарајуће лабеле. Компоненте се могу преименовати као на слици 74.

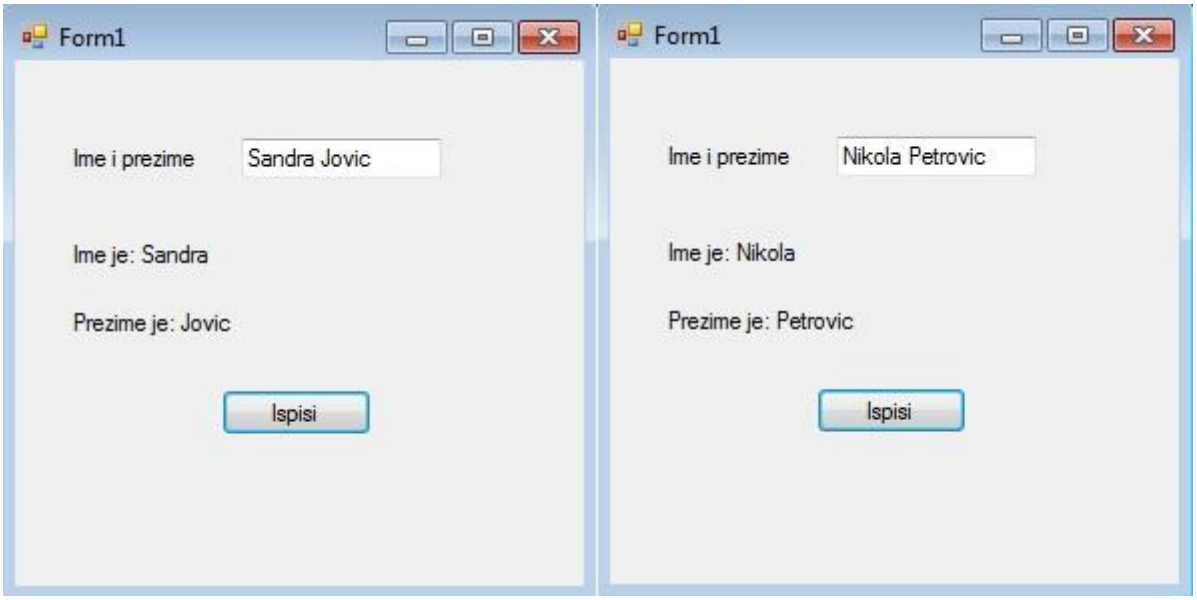


Слика 74. Изглед форме пре покретања програма

Алгоритам 18. Издвајање унетог имена и презимена у засебне лабеле

```
private void button1_Click(object sender, EventArgs e)
{
    //Kreiraju se promenljive ime i prezime tipa string i
    //pozicija,koja je tipa int. U promenljivu praznina koja
    //je tipa char, smesta se jedna belina
    string imeIprezime = textBox1.Text;
    string ime, prezime;
    char praznina = ' ';
    int pozicija;
    //Ovo ce pisati u labelama pre nego sto se u njih smesti ime
    //i prezime
    label2.Text = "Ime je: ";
    label3.Text = "Prezime je: ";
```

```
    //Trazi se pozicija praznine izmedju imena i prezimena
    //pomocu metoda IndexOf()
    pozicija = imeIprezime.IndexOf(praznina);
    if (pozicija < 0)
        //Ispisuje se poruka o neispravnom unosu
        MessageBox.Show("Neispravan unos!");
    //Ukoliko je unos ispravan
    else
    {
        //U promenljivu ime smesta se pomocu metode Substring(),
        //izdvojen podstring, odnosno ime
        ime = imeIprezime.Substring(0,pozicija);
        //U promenljivu prezime smesta se pomocu metode
        //Substring(), izdvojen podstring, odnosno prezime
        prezime = imeIprezime.Substring(pozicija+1);
        //Ispisivanje ime i prezime u zasebne labele
        label2.Text = label2.Text + ime;
        label3.Text = label3.Text + prezime;
    }
}
```



Слика 75. Изглед форме након покретања програма

Класа String садржи још много метода а неке од њих су наведене у табели 12. Више о методама класе String се може видети на веб адреси [16].

Табела 11. *Неке од метода класе String*

Метода	Тип који се враћа	Опис
LastIndexOf()	int	Враћа индекс последњег појављивања подстринга знакова у стрингу.
Remove()	string	Уклања специфицирани број знакова из стринга стартујући од задатог индекса у стрингу.
Replace()	string	Замењује сва појављивања обезбеђеног подстринга знакова другим под стрингом или знаком.
Split()	string[]	Дели стринг на низ стрингова коришћењем обезбеђеног сепаратора Затим се враћа низ стрингова.
Trim()	string	Уклања бланко знаке или специфициране знакове са почетка и краја стринга.

Набројиви тип

Набрајање омогућава да се креира скуп константи на које се може референцирати у програму. Набрајање је вредносни тип. На пример, уколико се пише програм који ће се користити у класи астрономије и да програм треба да се референцира на редослед планета у соларном систему у односу на Сунце. Редослед је: Меркур, Венера, Земља, Марс, Јупитер, Сатурн, Нептун, Плутон.

Набрајање се декларише коришћењем кључне речи enum. У следећој поједностављеној синтакси приказано је како се креира набрајање:

enum ime_nabrajanja { lista-konstanti }

ime_nabrajanja је име које се додељује набрајању. По конвенцији, прво слово имена набрајања је велико.
lista_konstanti је листа константи које се налазе у набрајању. Детаљније о набројивом типу у књизи [1].
У примеру 9.1 декларисано је набрајање названо Planete.

Слика 76. Распоред планета у Сунчевом систему

Пример 9.1 Декларација набрајања названо Planete.

```
enum Planete
{
    Merkur,
    Venera,
    Zemlja,
    Mars,
    Jupiter,
    Saturn,
    Uran,
    Neptun,
    Pluton
}
```

Као што се може видети, набрајањем Planete дефинише се девет константи, при чему свака константа представља позицију планете у односу на Сунце. Прва константа, Merkur, има подразумевану (default) вредност 0.
Остале константе имају за 1 већу вредност у односу на предходну вредност, тако да је константа Venera постављена на 2, константа Zemlja на 3, и тако даље до константе Pluton, која је постављена на 8. Подразумевани тип за константе у набрајању је int.

Константе у набрајању се могу и иницијализовати, као што је приказано следећим примером 9.2:

Пример 9.2 Иницијализација константи у набрајању названо Planete

```
enum Planete
{
    Merkur = 1,
    Venera,
    Zemlja,
    Mars,
    Jupiter,
    Saturn,
    Uran,
    Neptun,
    Pluton
}
```

Први упис поставља константу Merkur на вредност 1. Као и раније, друге константе се постављају на вредност за 1 већу у односу на предходну вредност, тако да је константа

Venera постављена на 2, константа Zemlja на 3, и тако даље све до константе Pluton, која је постављена на 9.
Да би се приступило елементу, користи се нотација тачке. На пример, да би се приступило елементу Zemlja, користилио би се Planete.Zemlja. У следећем изразу приказана је позиција Земље у односу на Сунце:

```
System.Console.WriteLine(     "Pozicija     Zemlje     =     "     +
(int)Planete.Zemlja);
```

Обраћа се пажња на употребу оператора за конверзију, да би се добила вредност на коју је Земља постављена у набрајању. Овом наредбом приказан је следећи израз:

Pozicija Zemlje = 3;

Ако се не уради конверзија оператором за конверзију, онда ће Planete.Zemlja из претходне наредбе вратити стринг ,,Zemlja". Већи број примера везаних за набројив тип се може наћи у књигама [6],[7] и [9].

Специфицирање вредности у набрајању

Може се специфицирати вредност константи у набрајању. На пример, следећа енумерација, названа PeriodicnostPlaneta, дефинише орбиталне периоде (време које је потребно планети да обиђе Сунце), за прве четири планете; орбитални периоди су изражени у данима.

Пример 9.3 Набрајања названо PeriodicnostPlaneta

```
enum PeriodicnostPlaneta
{
    Merkur = 88,
    Venera = 225,
    Zemlja = 365,
    Mars = 687,
}
```

Специфицирање основног типа у набрајању

Као што је раније поменуто, подразумевана вредност за тип енумерације је int. Тип који користи енумерација познат је као *основни тип*. Као основни тип енумерације може се користити било који целобројни тип. Да би се поставио основни тип треба ставити тип након имена енумерације и додати две тачке испред типа. У примеру 9.4 коришћен је тип long као основни тип енумерације.

Пример 9.4 Специфицирање основног типа у набрајању

```
enum PeriodicnostPlaneta: long
{
    Merkur = 88,
    Venera = 225,
    Zemlja = 365,
    Mars = 687,
}
```

Напомена
Иста константа се не може користити у декларацијама различитих набројивих типова података. На пример, дефиниције типова: enum radniDani { pon, uto, sre, cet, pet }; enum vikend { pet, sub, ned }; су НЕКОРЕКТНЕ, јер се константа pet појављује у дефиницији оба типа.

Ако се треба добити вредности промнљиве типа enum у облику стринга може се користити следећа линија кода:

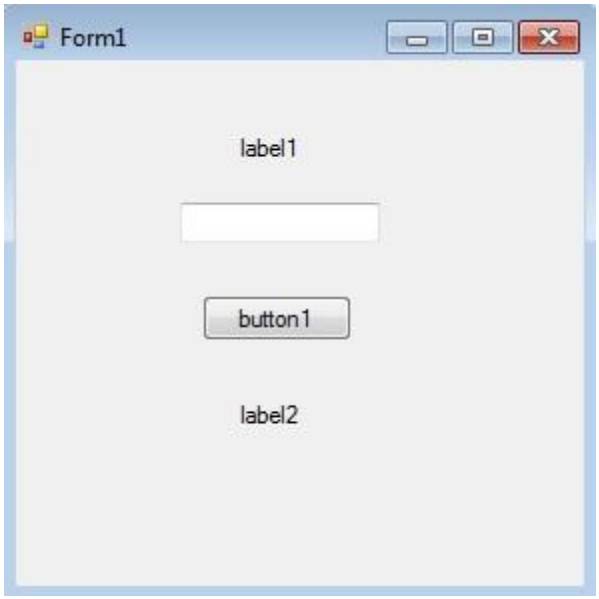
```
(( PeriodicnostPlaneta)88).ToString();
```

Ова линија кода би за повратну вредност имала стринг „Merkur", уколико би се користила енумерација као у примеру 9.4.

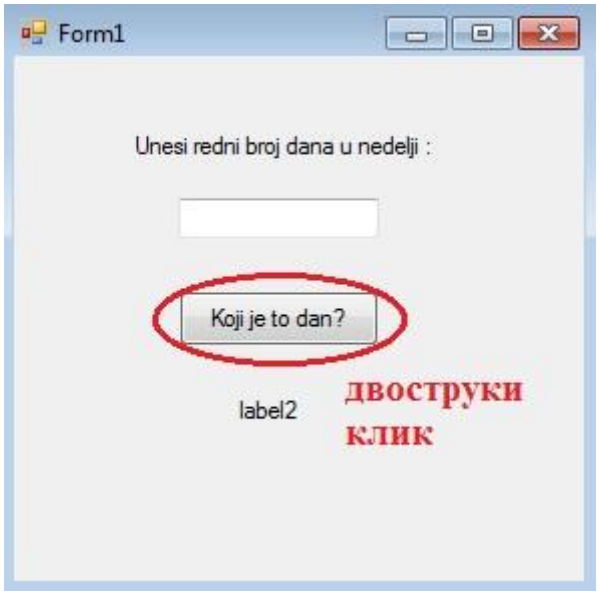
На примеру 9.5 биће приказано коришћење набројивог типа.

Пример 9.5 Направити програм који за редни број дана у недељи задат преко textBox-а, исписује његово име.

Решење: Користећи набројиви тип треба направити енумерацију Dani која ће садржати елементе: ponedeljak, utorak, sreda, cetvrtak, petak, subota и nedelja, који ће редом имати вредности 1, 2, 3, 4, 5, 6, 7. Од компоненти користе се две лабеле, један texBox и једно дугме који се могу преименовати као на слици 78.



*Слика 77.*Изглед почетне форме



Слика 78. Изглед форме након преименовања компоненти

Алгоритам 18. За задати редни број дана у недељи исписује његово име

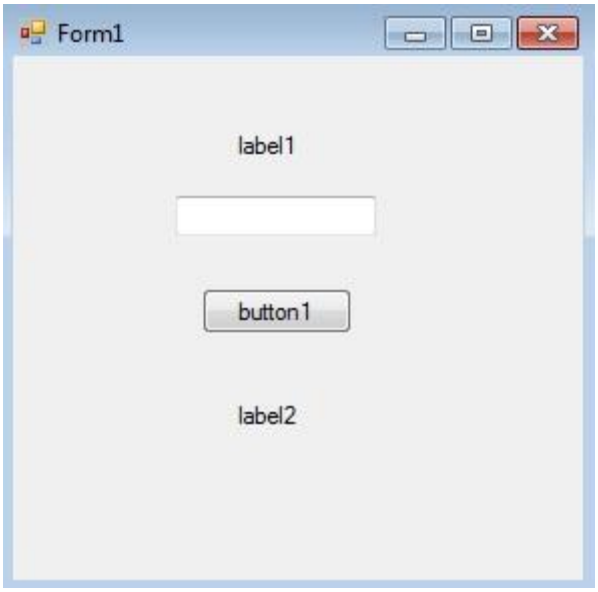
```
    //Kreira se promenljiva k tipa int
    int k;
    //Kreira se enumeracija Dani i svakom elementu dodaje
    //se vrednost
enum Dani
{
    Ponedeljak = 1,
    Utorak = 2,
    Sreda = 3,
    Cetvrtak = 4,
    Petak = 5,
    Subota = 6,
    Nedelja = 7
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    //Uzima se vrednost za promenljivu k koju je korisnik uneo u
    //textBox1
    k = int.Parse(textBox1.Text);
    //U lebeli2 ispisuje se resenje ispisivanjem naziva
    //vrednosti naseg nabrojivog tipa Dani
    label2.Text =k + ". dan u nedelji je : " +
((Dani)k).ToString();
}
```

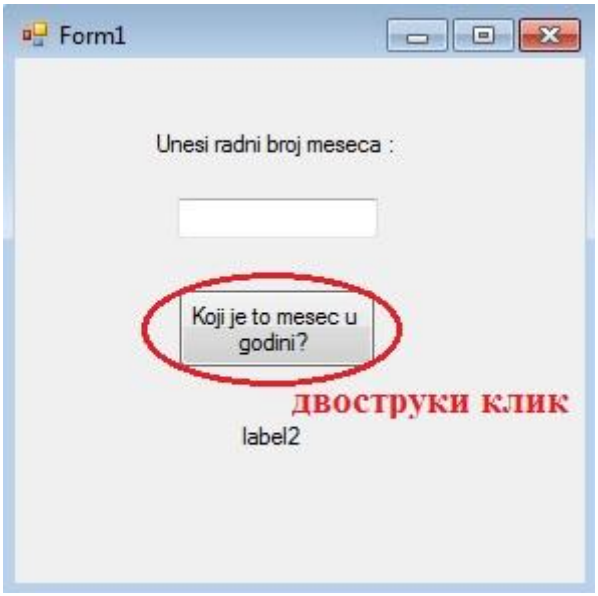
Слика 79. Изглед форме након покретања програма

Пример 9.6 Направити програм који кликом на дугме за редни број месеца у години задат преко textBox-а, исписује његово име у лабели .

Решење: Користећи набројиви тип треба направити енумерацију Mesec која ће садржати елементе : Januar, Februar, Mart, April, Maj, Jun, Jul, Avgust, Septembar, Oktobar, Novembar и Decembar, који ће редом имати вредности 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. Од компоненти се користе две лабеле, један texBox и једно дугме који се могу преименовати као на слици 81.



Слика 80. Изглед почетне форме



Слика 81. Изглед форме након преименовања компоненти

Алгоритам 19. За задати редни број месеца у години исписује његово име

```
    //Kreira se promenljiva k tipa int
    int k;
    //Kreira se enumeracija Mesec i svakom elementu se dodaje
    //vrednost
enum Mesec
{

    Januar = 1,
    Februar,
    Mart,
    April,
    Maj,
    Jun,
    Jul,
    Avgust,
    Septembar,
    Oktobar,
    Novembar,
    Decembar

}
```

```
private void button1_Click(object sender, EventArgs e)
{
    //Uzima se vrednost za promenljivu k koju je korisnik uneo
    //preko textBox1
    k = int.Parse(textBox1.Text);
    //U lebeli2 ispisuje se resenje ispisivanjem naziva
    //vrednosti naseg nabrojivog tipa Mesec
    label2.Text =k + ". mesec u godini je : " +
((Mesec)k).ToString();

}
```

Слика 82. Изглед форме након покретања програма

Класе и методе класе

Класа представља уопштени шаблон на основу кога се израђују објекти. Објекат је нека променљива која је састављена од података који садрже информације. На пример, уколико се животиње посматрају као објекти, онда ће њихова класа бити Zivotinja. Животиње се разликују по врсти, боји, начину оглашавања ... .
На пример: мачке, пси, патке... Мачке се међусобно разликују по врсти и боји крзна, пси такође, а једни од других се могу разликовати и по начину оглашавања. Заједничке карактеристике животиња су врста, боја крзна или перја. Свакој животињи се може придружити нека од ових особина, односно нека њихова заједничка својства. Ове карактеристике се називају *атрибути*.
Класа се дефинише навођењем речи *class* иза које следи идентификатор класе (име класе). Затим се у витичастим заградама {} дефинишу чланови класе (атрибути и методи).

Пример 10.1 Дефинисање класе.

```
class Ime_klase
{
      Clanovi (promenljive i metode)
}
```

На пример, класа ученик се може интуитивно дефинисати на следећи начин:

Пример 10.2 Дефинисање класе ученик.

```
public class Ucenik
{
    //Navode se atributi koje ce klasa Ucenik da ima
    String ime, prezime;
    char odeljenje;
    int razred;
}
```

Напомена

Атрибути описују одређену особину објекта (име, презиме, разред, одељење). Најчешће различити објекти исте класе имају различите вредности атрибута. Често се каже да вредности атрибута дефинишу стање објекта. При опису атрибута мора се навести тип коме тај атрибут припада (целобројни, реални, знаковни...) и име атрибута. При томе, наведени тип је претходно дефинисан (уграђен у систем или дефинисан од стране програмера).

Приступ атрибутима објекта у програмском језику C# реализује се навођењем имена објекта, тачке (‘.’) и имена атрибута. На пример, ако је X објекат класе Ucenik, атрибуту razred приступа се са X.razred.

У класи Zivotinja радње које ће животиња обављати зависи од тога које особине има та животиња. Радње које животиња обавља су заправо функције. Функције које се врше над атрибутима класе називају се *методи класе*. Метод који се може дефинисати за животињу може бити начин оглашавања. Свака животиња одређене врсте има јединствен начин оглашавања.

Методе дефинишу операције које се могу применити над објектима класе,и обично се примењују над променљивим из класе.

Пример 10.3 Класа Zivotinja са конструктором и методом за оглашавање.

```
class Zivotinja
{
    //Ovo su atributi klase Zivotinja.
    public String vrsta;
    public String ime;

    //Ovo je konstruktor. Mora se napraviti zivotinja da bi se
    //na nju mogle primenjivati metode.
public Zivotinja(string ime, string vrsta)
{
    this.ime = ime;
    this.vrsta = vrsta;

}

    //Ovo je metod klase Zivotinja. Kao argument uzima vrstu
    //zivotinje.
public string predstavljanje(string vrsta)
{
    return "Ja sam " + vrsta;
}

}
```

Под дефиницијом класе подразумева се навођење свих чланова класе. Кад је класа дефинисана, могу се дефинисати и њени објекти. Дефиниција класе у потпуности одређује класу. У примеру 10.3 се потпуно одређује животиња: врста животиње, њено име и њено представљање (то је метод).

Пре позива метода везаног за класу неопходно је помоћу конструктора направити одређени објекат на који ће се тај метод применити. Дакле, прво треба направити животињу, то ће бити објекат типа Zivotinja.

Пример 10.4 Коришћење конструктора.

```
//pravimo zivotinju pomocu konstruktora iz klase
Zivotinja zivotinja = new Zivotinja(ime, vrsta);
```

Уколико у класи није дефинисан конструктор, користи се подразумевани конструктор који увек постоји, а потом се додељује вредност одређеним атрибутима.

```
//pomocu podrazumevanog konstruktora pravimo zivotinju
Zivotinja zivotinja = new Zivotinja();
//dodeljujemo joj ime
zivotinja.ime = "Pera";
```

```
//dodeljujemo mu vrstu
zivotinja.vrsta = "macka";
```

Одређени метод се позива помоћу оператора тачка.

```
ljubimac.predstavljanje(vrsta);
```

Дакле, метод класе је именовани блок наредби који се састоји из заглавља и тела метода. У заглављу се наводи повратни тип (ако метод не производи вредност коју враћа, наводимо резервисану реч void), затим име метода за којим следи у малим заградама списак параметара метода. За сваки параметар наводи се тип коме тај параметар припада као и име параметра. После заглавља у витичастим заградама наводи се тело метода које се састоји из одговарајућих наредби програмског језика C#. Метод би имао следећу синтаксу:

```
povratni tip  ime_metode (lista parametara koje metoda koristi)
{
    telo metode
}
```

Сви атрибути конкретног објекта доступни су и видљиви свим методима тог објекта и трају док траје објекат. За разлику од њих, променљиве декларисане у оквиру неког од метода (локалне променљиве), настају извршавањем тог метода. Видљиве су само у оквиру њега и гасе се завршетком тог метода. Зато ако је потребно да се користе вредности у оквиру више метода једне класе (или у више извршавања једног метода), треба их дефинисати као атрибут класе.

Позив метода објекта у програмском језику C# реализује се:

imeObjekta.imeMetoda(lista vrednosti parametara)

Напомен

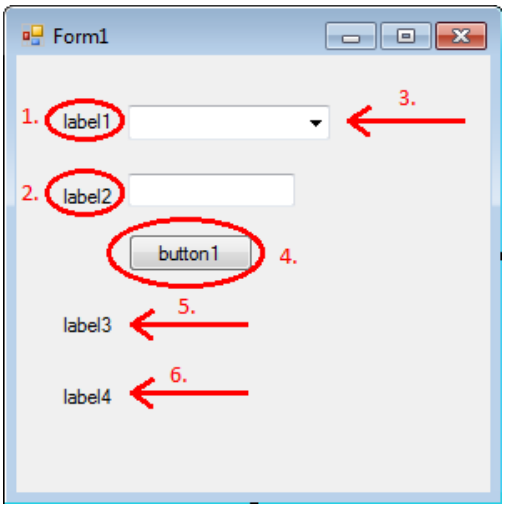
Класа је скуп објеката који имају заједничку структуру и понашање. Класа представља нови, комплексни тип података. Заправо, сви типови података: Integer, Boolean, Float, Double, String и тако даље, представљају посебне класе које у себи имају дефинисане одређене особине и методе, тј. операције које се над њима могу обављати. Имена класа пишу се великим почетним словом.

На примеру 10.5 следи објашњење како се креира нова класа у C#. Више о класама и методама класе се може наћи на веб адресама [13],[14] или у књизи [1].

Пример 10.5 Направити класу Zivotinja која ће имати две класне променљиве - ime и vrsta. Унутар класе креирати две методе: predstavljanje (која ће саопштавати која животиња је у питању) и методу oglasavanje. Затим, написати програм у коме ће се креирати објекти класе и позивати методе.

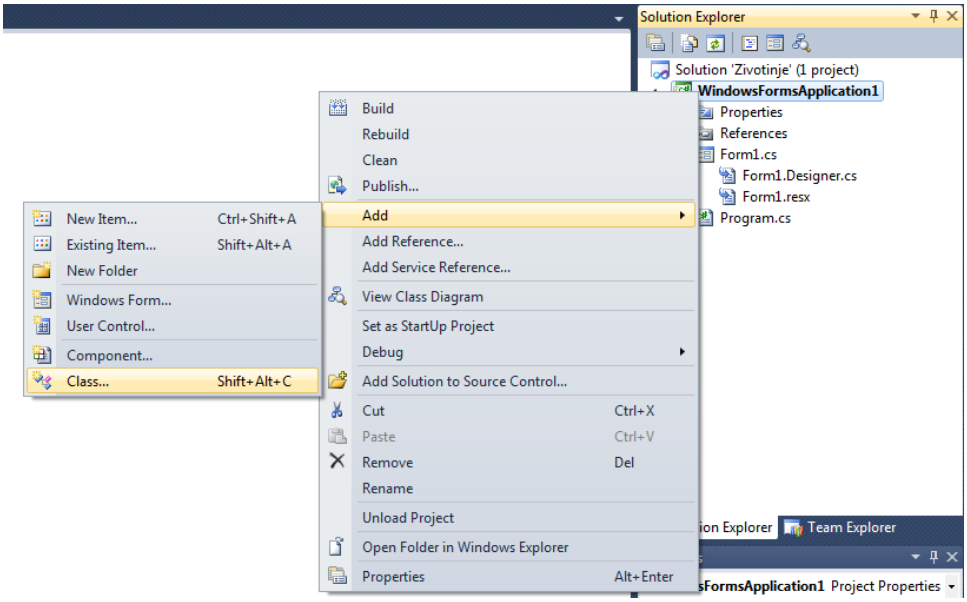
Решење: Креирати нов пројекат под именом Zivotinje.

1. label1 - преименовати у Zivotinja
2. label2 - преименовати у ime
3. comboBox1 - додати неколико домаћих животиња
4. button1 - преименовати дугме у Predstavi se
5. label3 – лабела у којој ће се исписати врста љубимца
6. label4 – лабела у којој ће се исписати на који начин се љубимац оглашава



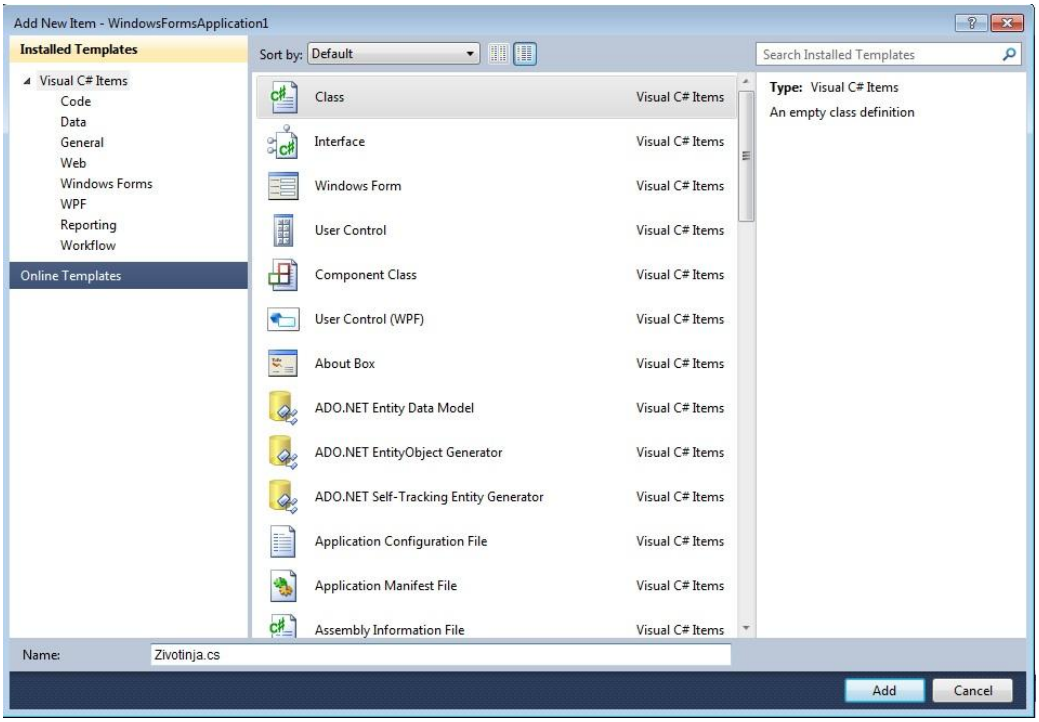
Слика 83. Изглед форме пре покретања програма

Сада треба креирати класу Zivotinja. Она мора бити креирана унутар пројекта на коме радимо. То се постиже на следећи начин. Унутар радног окружења, у горњем десном углу налази се Solution Explorer. Десним кликом на име пројекта на коме се ради, отвара се мени са опцијама. Треба изабрати опцију Add Class као на слици 84.



Слика 84. Поступак прављења нове класе

Сада треба додати име класи и након тога, притиском на дугме Add, креира се класа и отвара се прозор где се пише код за класу.



Слика 85. Поступак прављења нове класе

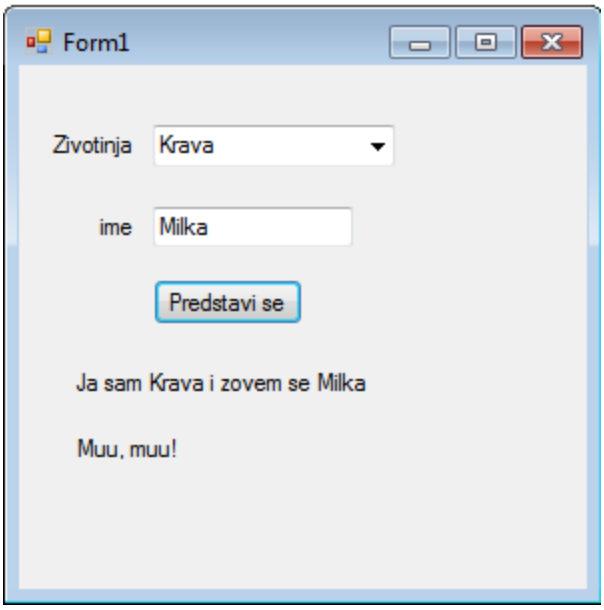
Алгоритам 20. Класа Zivotinja и методи класе

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace WindowsFormsApplication1
{
    class Zivotinja
    {
        public String ime;
        public String vrsta;

        public Zivotinja(string ime, string vrsta)
        {
            this.ime = ime;
            this.vrsta = vrsta;
        }

        public string predstavljanje(string ime, string vrsta)
        {
            return "Ja sam " + vrsta + " i zovem se " + ime;
        }

        public string oglasavanje(string vrsta)
        {
            if (vrsta.Equals("Macka"))
                return "Mijau, mijau!";
            else if (vrsta.Equals("Pas"))
                return "Av,av!";
            else if (vrsta.Equals("Kokoska"))
                return "Ko, ko!";
            else if (vrsta.Equals("Krava"))
                return "Muu, muu!";
            else if (vrsta.Equals("Ovca"))
                return "Bee, bee!";
            else if (vrsta.Equals("Patka"))
                return "Kva, kva!";
            else
                return "Niste izabrali zivotinju!";


        }
    }
}
```

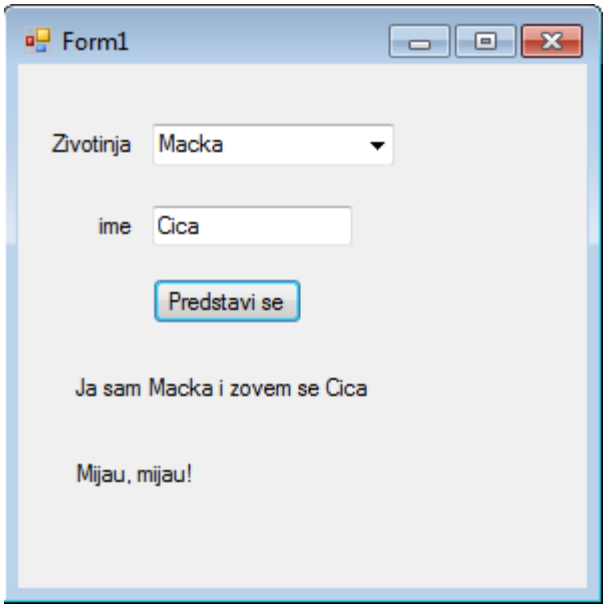
Креирана је класа Zivotinja са методама predstavljanje и oglasavanje. Треба написати програм који ће се извршавати кликом на дугме. Двокликом на дугме Predstavi se отвара се прозор где се пише одговарајући код.

Алгоритам 21. Коришћење класе Zivotinja и њених метода

```
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 WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string ime, vrsta;
            ime = textBox1.Text;
            vrsta = comboBox1.SelectedItem.ToString();

            Zivotinja z = new Zivotinja(ime, vrsta);
            label3.Text = z.predstavljanje(ime, vrsta);
            label4.Text = z.oglasavanje(vrsta);

        }

        private void Form1_Load(object sender, EventArgs e)
        {

            comboBox1.SelectedIndex = 0;
        }
    }
}
```



Слика 86. Изглед форме након покретања програма



Слика 87. Изглед форме након покретања програма

C# има и већ посојеће, тј. дефинисане класе чије методе се могу одмах користити.

Класа System.Math

System.Math класа нуди многе методе које се могу користити при израчунавању тригонометријских, логаритамских и других математичких прорачуна.
На пример, метод Pow**,** System.Math класе, може се користити за степеновање неког броја. Такође System.Math класа садржи и бројне већ дефинисане константе које се могу директно користити, као на пример константу PI.

Пример 10.5.1 Израчунавање површине круга полупречника r помоћ System.Math класе и њених већ постојећих метода и константи.

```
double povrsina = Math.Pow(r,2)*Math.PI;
```

Први аргумент методе Pow је полупречник, тј. оно што се степенује, а други, број на који се степенује полупречник.

Константа PI се такође налази дефинисана у класи Math и може се користити позивањем Math.PI. Следећи метод који се често користи јесте метод *Sqrt*. То је метод који се користи за рачунање корена неког броја или неког израза.

Пример 10.5.2 Израчунавање хипотенузе правоуглог троугла, ако су нам већ познате његове катете a и b.

```
double hipotenuza = Math.Sqrt(a*a+b*b);
```

Могла се користити и већ од малопре позната метода Pow

```
double hipotenuza = Math.Sqrt(Math.Pow(a,2)+Math.Pow(b,2));
```

Abs метод враћа апсолутну вредност броја (позитивну вредност броја независно од његовог знак). Повратна вредност методе Math.Abs(-3) у примеру 10.5.3 враћа апсолутну вредност броја -3, што је 3.

Пример 10.5.3 Коришћење методе Abs

```
double apsolutna_vrednost = Math.Abs(-3);
```

Метод *Round* заокружује број на најближи цео број. На пример, следећи код се користи за заокруживање броја 5,49999 на најближи цео број. Резултат ће бити 5.

Пример 10.5.4 Коришћење методе Round

```
double ceo_broj = Math.Round(-3);
```

Ако се користи Round да се заокружи број који је тачно на пола између два броја, као што је 5.5, Round увек враћа паран број најближи броју. Дакле, Math.Round(5.5) ће вратити 6, али Math.Round(8.5) ће вратити 8.

Math.Max и *Math.Min* враћају већи и мањи од два броја, респективно. Они подржавају скоро све типове бројева као параметре.

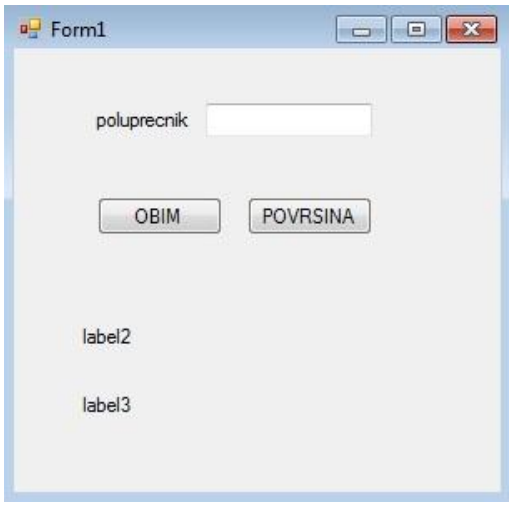
Пример 10.5.4 Коришћење метода Math.Max и Math.Min

```
//Maksimum ce biti 5
int maksimum = Math.Max(5,2);
//Minimum ce biti 2.3
float minimum = Math.Min(7.6,2.3);
```

Више о класи System.Math се може наћи на веб адресама [10],[12].

Пример 10.6 Направити класу круг и методе које рачунају обим и површину круга.

Решење : За решавање овог задатка потребне су три лабеле, један textBox и два дугмета. Компоненте се могу преименовати као на слици 88.

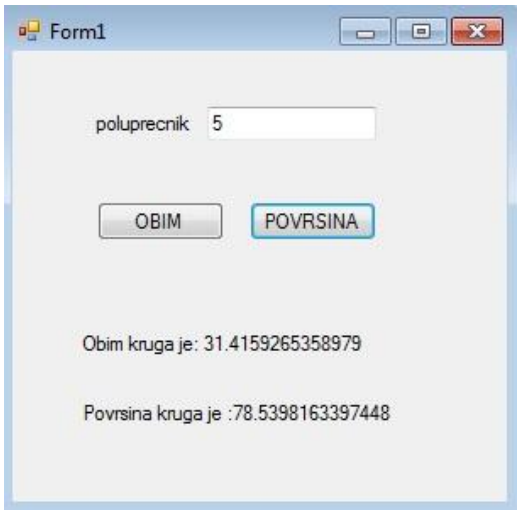


Слика 88. Изглед форме пре покретања програма

Алгоритам 22. Класа круг и методе које рачунају обим и површину круга

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WindowsApplication1
{
   public partial class Form1 : Form
   {
         public Form1()

        {
            InitializeComponent();


        }
    //Definise se klasa krug
public class Krug
{
    double r;
```

```
    //Metoda koja racuna povrsinu kruga
    public double Povrsina(double r)
    {
        return Math.Pow(r,2)* Math.PI;
    }
        //Metoda koja racuna obim kruga
        public double Obim( double r)
        {
            return 2 * r * Math.PI;
        }
         }
private void button1_Click( object sender, EventArgs e)
{
    double r = double.Parse(textBox1.Text);
    //Kreira se primerak klase
    Krug k = new Krug();
    //Poziva se metoda Obim
    double obim = k.Obim(r);
    label2.Text = "Obim kruga je: " +obim;
}
private void button2_Click(object sender, EventArgs e)
{
    double r = double.Parse(textBox1.Text);
    Krug k1 = new Krug();
    label3.Text = "Povrsina kruga je :" + k1.Povrsina(r);
}
    }
}
```



Слика 89. Изглед форме након покретања програма

Наредбе и изрази

У току обраде информација често се долази у ситуацију да се на податке примењују различите операције којима се стварају нове вредности (резултати), који се даље обрађују применом нових операција коначно много пута. За то су корисне одговарајуће наредбе и изрази и зато ће сада бити речи о томе шта представљају изрази, а шта наредбе у програмском језику C#.

Низ променљивих, константи и позива метода међусобно одвојених операцијским знацима се називају *изразом*. Изрази могу бити *прости* (само константа, променљива или позив метода) или *сложени* са произвољним бројем оператора и елемената израза одвојених заградама.

Дефиниција израза:

1) константе, променљиве и позиви метода су изрази;
2) ако су А и Б изрази, а о оператор који означава бинарну операцију, тада је А о Б израз;
3) ако је А израз и ◊ оператор који означава унарну операцију, тада је ◊А израз;
4) ако је А израз, тада је (А) израз;
5) израз можемо добити само коначном применом правила 1), 2), 3) и 4).

Изрази који се могу наћи у оквиру програма који пишемо, без обзира да ли су аритметички или логички, пишу се у складу са установљеним правилима у математици и синтаксом програмског језика:

1. Свака отворена заграда мора бити затворена.
2. Два знака операције не могу се наћи један поред другог.
3. Израз не може стајати самостално у програму.
4. Вредност израза израчунава се аутоматски доделом вредности параметрима.

Наредба у C# језику је једна инструкција програмског кода која се завршава са знаком ";". Наредбе се у C# извршавају следно, једна за другом. Наредба може бити *проста наредба* уколико се састоји само из израза за којим следи карактер " ; ", а може бити и *сложена наредба* (блок) која се добије када се више простих наредби групише навођењем витичастих заграда. *Празна наредба* има само карактер " ; " или евентуално коментар.

Пример 11.1 Просте и сложене наредбе.

```
//Prosta naredba
int duzina;
duzina = 15;
int x, y, z;
z = x + y;
```

```
{
   //Slozena naredba
   int x = 3;
   int y = 5;
   x = x + y;
   y = y - 1;
}
```

Често се користи прорећивање да би код био читљивији. На пример, додавање празних линија или увлачење низа повезаних инструкција, може помоћи да се види како је програм структуиран. У C#-у, ови додатни бланко знаци, увлачења коришћењем тастера Tab са тастатуре и празне линије познати су као празан простор јер ти знакови представљају само додатни „простор“ у наредбама.

Пример 11.2 Наредбе без додатног празног простора.

```
int vrednost = 3;
System.Console.Writeline("vrednost =" + vrednost);
```

Пример 11.3 Наредбе са додатним празним простором.

```
int vrednost  =  3;
System.Console.Writeline( "vrednost ="  +  vrednost );
```

C# обично игнорише празан простор у коду, међутим не игнорише га када је он део стринга као у изразу:

```
System.Console.Writeline("vrednost  =  " + vrednost );
```

У овом случају, код под знацима навода садржи празан простор који формира део стринга и због тога се не игнорише.

Савет
Празан простор треба користити на прави начин, да би код био читљивији. Превише или премало празног простора учиниће програм тежим за разумевање.

Као што је већ речено, блокови се користе да би се груписале наредбе. Блок почиње отвореном великом заградом ({) и завршава се затвореном великом заградом (}). У следећем примеру приказан је блок. Две наредбе унутар блока су увучене за два бланко знака, што помаже да се види да се наредбе налазе унутар блока као на примеру 11.4.

Пример 11.4 Блок са две наредбе.

```
{
   int vrednost = 3;
   System.Console.Writeline("vrednost = " + vrednost);
}
```

Савет
Добра програмерска пракса је да се блокови користе чак и када је у питању само једна наредба. На овај начин, ако је потребно додати још једну наредбу, већ постоји блок на правом месту. Да блок није постојао, може се направити грешка додавањем нове наредбе, а не додавањем заграда за блок.

Додавање коментара

Својим програмима могуће је додати коментаре, ради описивања кода, чинећи га разумљивијим. Суштина је да би требало додавати коментаре ради бољег разумевања, али не и да треба коментарисати сваку линију. Траба разумно користити коментаре, као и празан простор.

Синтакса и семантика израза

Синтакса израза представља правила помоћу којих се гради и проверава коректност израза у самом програму. Програмски језици су тако конструисани да рачунар може лако да уочава и упозорава на синтаксне грешке у програму.

На пример, ако у аритметичком изразу нису затворене заграде, ако није стављена ";" на крају наредбе или није затворена витичаста заграда на крају блока наредби, приликом покретања програма јавиће се грешке.

Такође, до синтаксне грешке може доћи ако се оператор %, који даје остатак при дељењу, примени на операнде реалног типа, из разлога што се он може примењивати искључиво на операнде целобројног типа.

Семантика израза одређује значење израза у програму, односно придружује значење синтаксно исправним изразима и описује понашање рачунара током њиховог извршавања. Неки аспекти семантичке коректности могу се проверити током превођења програма (на пример, да су све променљиве које се користе у изразима дефинисане и да су одговарајућег типа), док се неки аспекти могу проверити тек у фази извршавања програма (на пример, да не долази до дељења нулом у неком изразу). Семантичке грешке углавном настају као последица грешке у алгоритму.

Применом оператора на два неодговарајућа операнда, може доћи до семантичке грешке јер семантичка правила одређују ограничења на типовима података.
Честа грешка је коришћење оператора „=" уместо оператора „==", када желимо да испитамо да ли су неке две вредности једнаке. Тако израз *x = y* не представља исто што и израз *x == y*, јер у првом изразу се променљивој *x* додељује вредност променљиве *y*, а у другом се пита да ли је *x* једнако са *y*.
Сличне грешке се праве код оператора & и &&, као и код оператора | и || , па је неопходно уочити разлику између логичких оператора (&&, ||) и битовских оператора (&, |).

Напомена
Дакле, треба обратити пажњу да: = није исто што и ==

Аритметички изрази

Аритметички изрази служе за обраду нумеричких података и као резултат дају број. Сабирање, одузимање, множење, дељење, одређивање остатка при дељењу два цела броја су примери аритметичких израза.

Аритметички изрази могу бити:

1. Једноставни - састоје се само од једног операнда
2. Сложени – састоје се од два или више операнада повезаних аритметичким операторима

Операнди могу бити:

- целобројне и реалне константе
- целобројне и реалне промењиве
- елемент низа
- позив целобројних и реалних функције
- аритметички изрази у заградама

Већ је речено да су операнди у аритметичким изразима међусобно повезани операторима и да су у питању аритметички оператори. Треба се подсетити аритметичких оператора и њихових функција, као и поделе на унарне и бинарне. Унарни оператори раде са једним операндом, а бинарни раде са два операнда.

Табела 12. *Аритметички оператори и њихове функције*

Аритметички оператори	
бинарни	
+	Сабирање
-	Одузимање
*	Множење
/	Дељење
%	Остатак при дељењу првог операнда другим
унарни	
+	Промена знака операнда или израза
-	

Аритметички оператори су дефинисани за све нумеричке типове података.
Оператори **+ , - , * , /** представљају основне аритметичке операције сабирања, одузимања, множења и дељења..

Оператор дељења означава различите операције у зависности од типа својих операнада. То нам показује пример 12.1.

Пример 12.1 Колика је вредност количника:
а) 9 / 5
б) 9.0 / 5.0
в) 9.0 / 5?

Решење:
а) Израз 9 / 5 има вредност 1, што значи да када се оператор дељења примењује на две целобројне вредности, као резултат се добија цео број (цео део количника). Тада се каже да је у питању целобројно дељење**.**

б) У овом изразу се примењује реално дељење јер су операнди реалног типа, па израз 9.0 / 5.0 има вредност 1.8.
в) Пошто је у изразу 9.0 / 5 бар један операнд реалног типа, примењује се реално дељење, па овај израз има вредност 1.8**.**

Оператор % је могуће применити искључиво на операндима целобројног типа. Користи се за одређивање остатка при дељењу првог операнда другим, као на пример, 5 % 2 = 1, 16 % 4 = 0.

Оператори инкрементирања (увећавања за 1) ++ и декрементирања (умањивања за 1) су унарни оператори које се могу употребити у префиксном (*++n* и *--n*) или у постфиксном облику (n++ и n--)
Разлика између ова два облика је у томе што, на пример, ++n увећава променљиву n пре него што се њена вредност користи, а n++ увећава n након што се њена вредност користи. Тако се x = ++n; разликује од x = n++; што се показује на примеру 12.2.

Пример 12.2 Израчунати вредност променљивих након извршавања следећих наредби:
а)

```
int n, x;
n = 3;
x = ++n - 2;
```

б)

```
int n, x;
n = 3;
x = n++ - 2;
```

Решење:
а) Овде долази до префиксног коришћења оператора инкрементирања, тако да се у наредби x = ++n - 2; прво увећа променљива n на 4, а затим израчуна вредност израза (x = 4 - 2) и променљива x добије вредност 2. Резултат је n = 4, x = 2.

б) Овде долази до постфиксног коришћења оператора инкрементирања, тако да се у наредби x = n++ - 2; прво користи вредност променљиве n, па је вредност променљиве x = 3 - 2, дакле x=1, а затим увећа вредност променљиве n на 4. Резултат је n = 4, x = 1.

Треба се подсетити приоритета аритметичких оператора о којима је било речи у поглављу о операторима. Укратко: унарни оператори **+, -, ++,** -- су највишег приоритета. Оператори ***, /** и **%** су вишег приоритета од оператора **+** и **-**.
Сви бинарни аритметички оператори су лево асоцијативни, а унарни су десно асоцијативни.

Редослед рачунања вредности аритметичких израза одређен је:

- употребом заграда;
- приоритетом извршавања аритметичких операција;
- положајем оператора унутар израза;
- у изразу се рачунају прво подизрази у заградама;

Пример 12.3 Израчунати вредност следећих израза:

1. 15 / 2.0 + 4 - 2
2. 12.0 / 3.0 - 2.5 + 8 % 5
3. 16 / 3 - 16 % 3
4. 4 % 2 * 5 + 4
5. 4 + 24 % (2 * 3)
6. 4 % 7 + 7 % 4
7. 7 % 2 + 3 / 3 - 2
8. 3 * 5 / 2 % 5 + 1
9. 24 / 3 * 12 % 5 + 5

Решење:
1. 15 / 2.0 + 4 - 2 = 7.5 + 4 - 2 = 11.5 - 2 = 9.5

2. 12.0 / 3.0 - 2.5 + 8 % 5 = 4.0 - 2.5 + 8 % 5 = 1.5 + 8 % 5 = 1.5 + 3 = 4.5
3. 16 / 3 - 16 % 3 = 5 - 1 = 4
4. 4 % 2 * 5 + 4 = 0 * 5 + 4 = 0 + 4 = 4
5. 4 + 24 % (2 * 3) = 4 + 24 % 6 = 4 + 0 = 4
6. 4 % 7 + 7 % 4 = 4 + 3 = 7
7. 7 % 2 + 3 / 3 - 2 = 1 + 1 - 2 = 2 - 2 = 0
8. 3 * 5 / 2 % 5 + 1 = 15 / 2 % 5 + 1 = 7 % 5 + 1 = 2 + 1 = 3
9. 24 / 3 * 12 % 5 + 5 = 8 * 12 % 5 + 5 = 96 % 5 + 5 = 1 + 5 = 6

Пример 12.4 Израчунати вредност промењљивих x и y након извршавања следећег дела кода:

```
int x, y;
int a = 0;
int b = 0;

a++;
++b;

x = ++a;
y = b++;
```

Из коментара датог кода можемо доћи до решења:

```
//Uvecava vrednost promenljive a za 1, tj. a = 1
a++;
//Uvecava vrednost promenljive b za 1, tj. b = 1
++b;
/* Prefiksni operator uvecava promenjivu a, a promenljiva x
dobija vrednost jednaku uvecanoj vrednosti promenljive a, a = 2,
x = 2 */
x = ++a;
/* Postfiksni operator uvecava promenjivu b, a promenljiva y
dobija vrednost jednaku staroj (neuvecanoj) vrednosti
promenljive b, b = 2, y = 1 */
y = b++;
```

Вредности промењљивих после извршења наредби: x = 2, y = 1.

Логички изрази

Логички изрази служе за поређење различитих података и за друге логичке операције. То су изрази састављени од логичких операнада повезаних логичким операторима. *Логички операнди* су релацијски изрази или логички изрази.

Логички оператори који се користе у језику C# су:

- оператор конјункције - && (and) који проверава да ли су оба израза тачна;
- оператор дисјункције - || (or) који проверава да ли је макар један израз тачан;
- оператор негације - ! (not) који има истиниту вредност само када је израз нетачан.

Приоритет логичких оператора:
Логички оператори имају већи приоритет од релационих оператора (<, >, <=, >=, ==, !=). Највиши приоритет од логичких оператора има оператор !, затим && и на крају ||.

Дефинисање логичких израза

1. Логичке константе *true* и *false* су логички изрази;
2. Логичке променљиве (типа Boolean) су логички изрази;
3. Релацијски изрази су логички изрази;
4. Позиви логичких функција су логички изрази;
5. Ако су L1 и L2 логички изрази, тада су логички изрази и (L1 && L2), (L1 || L2), (!L1), (!L2).

Унарни логички оператор ! негира логичку вредност на коју се примењује. Израз !L1 има вредност true уколико L1 има вредност false и обрнуто (Табела 13).

Табела 13. *Унарни логички оператор*

x	!x
true	false
false	true

Израз облика L1 && L2 има вредност true само ако обе логичке вредности имају вредност true, у супротном, вредност израза је false.

Табела 14. *Израз облика L1 && L2*

&&	false	**true**
false	false	false
true	false	**true**

Израз облика L1 || L2 има вредност false само ако обе логичке вредности имају вредност false, у супротном, вредност израза је true.

Табела 15. *Израз облика L1 || L2*

||	**false**	true
false	**false**	true
true	true	true

У израчунавању вредности логичких израза користи се *лењо израчунавање*. Његова основна карактеристика јесте израчунавање само оног што је неопходно.

На пример, приликом израчунавања вредности израза (2 < 1) && (x++ > 10), биће израчунато да је вредност подизраза (2 < 1) false, па је и вредност целог израза false (због својства логичког &&). Зато нема потребе израчунавати вредност подизраза x++, па ће вредност променљиве x остати непромењена након израчунавања вредности наведеног израза.
С друге стране, након израчунавања вредности израза (1 < 2) && (x++ > 10), вредност променљиве x ће бити промењена (увећана за 1) јер подизраз (1 < 2) има вредност true, па је потребно израчунати и вредност другог операнда. У изразима у којима се јавља логички оператор &&, уколико је вредност првог операнда једнака true, онда се израчунава и вредност другог операнда.
У изразима у којима се јавља логички оператор ||, уколико је вредност првог операнда једнака true, онда се не израчунава вредност другог операнда.
Уколико је вредност првог операнда једнака false, онда се израчунава и вредност другог операнда.
На пример, након израчунавања вредности израза (1 < 2) || (x++ > 10), не мења се вредност променљиве x јер подизраз (1 < 2) има вредност true, па се вредност другог операнда не рачуна, а након (2 < 1) || (x++ > 10), мења се (увећава за 1) вредност променљиве x јер је вредност подизраза (2 < 1) једнака false, па се израчунава и вредност другог операнда.

Пример 13.1 За x = 5, y = 2, одредити вредност следећих израза:

1. (x * y != 0) && (y > x)
2. (x % y > 0) || (y > x)
3. (x % 3 == 0) && ((x + y) > = 0)
4. (y % 2 == 0) || (x / 3 != 0

Решење:
1. (x * y != 0) && (y > x) = (5 * 2 != 0) && (2 > 5) = (10 != 0) && false = true && false = false.
2. (x % y > 0) || (y > x) = (5 % 2 > 0) || (2 > 5) = (1 > 0) || false = true || false = true
3. Како је 5 % 3 = 2, а 2 != 0, то је вредност подизраза (x % 3 == 0) false, може се закључити да је вредност целог логичког израза (x % 3 == 0) && ((x + y) > = 0) false, без рачунања вредности другог операнда, на основу лењог израчунавања.
4. Како је вредност подизраза y + 2 == 4 true (2 + 2 = 4, а 4 == 4), може се закључити, без рачунања вредности другог операнда, да наш израз (y + 2 == 4) || (x / 3 != 0) има вредност true, на основу лењог израчунавања.

Пример 13.2 За i = 1, j = 2, k = 4, одредити вредност следећих израза:

1. (i < j) && (k != i)
2. ((i + j) >= k) || (i == 2)
3. !(j < k)
4. ((k - j) > i) && (k == 4)

Решење:
1. (i < j) && (k != i) = (1 < 2) && (4 != 1) = true && true = true
2. ((i + j) >= k) || (i == 2) = ((1 + 2) >= 4) || (1 == 2) = (3 >= 4) || false = false || false = false
3. !(j < k) = ! (2 < 4) = ! true = false
4. ((k - j) > i) && (k == 4) = ((4 - 2) > 1) && (4 == 4) = (2 > 1) && true = true && true = true

Пример 13.3 Записати у облику логичких израза:

1. x ∈ [0, 1];
2. x ∈ [-1, 1] U [2, 5];
3. Бар један од целих бројева x, y, z је позитиван;
4. Сва три броја x, y, z су позитивна;

Решење:

1. (x >= 0) && (x <= 1)
2. (x >= -1) && (x <= 1) || (x >= 2) && (x <= 5)
3. (x > 0) || (y > 0) || (z > 0)
4. (x > 0) && (y > 0) && (z > 0)

Пример 13.4 Саставити програм који одређује истиносну вредност формуле (p || q) || r, где су p, q и r искази дате истиносне вредности.

Решење:

Алгоритам 23. Истинитосна вредност формуле (p || q) || r

```
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 Zadatak1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

private void button1_Click(object sender, EventArgs e)
{
    label1.Text = "p = ";
    //Uzima se vrednost za promenljivu p koju je
    //korisnik uneo
    Boolean p = Boolean.Parse(textBox1.Text);
    label2.Text = "q = ";
    //Uzima se vrednost za promenljivu q koju je
    //korisnik uneo
    Boolean q = Boolean.Parse(textBox2.Text);
    label3.Text = "r = ";
    //Uzima se vrednost za promenljivu r koju je
```

```
    //korisnik uneo
    Boolean r = Boolean.Parse(textBox3.Text);
    //po formuli se racuna i ispisuje resenje
    label4.Text = "(p || q) || r = " + ((p || q) || r);
      }
   }
}
```

Наредба доделе

При упознавању са наредбом доделе важно је објаснити доделу облика x = x + 1; која може да буде збуњујућа због сличности са математичком једначином која нема решења. Зато је потребно указати на разлику између знака једнакости који се користи у саставу наредбе "доделе" вредности („=") и знака једнакости који се користи за означавање релације „једнако" („==").
У C# наредба x = x + 1; представља једну наредбу доделе, јер се променљивој x додељује њена вредност увећана за 1 и то је сада нова вредност променљиве x.
Знак једнакости „=", који учествује у овој наредби називамо оператором доделе. То је оператор најнижег приоритета, а асоцијативност је здесна налево.

Општа синтакса наредбе доделе:

< променљива > = < израз >

На левој страни је име променљиве којој се додељује вредност, а на десној страни је конкретна вредност, израз или функција чија се вредност додељује променљивој. Променљива и вредност се морају слагати по типу.

Наредба доделе се извршава у два корака:

1. прво се одређује вредност израза са десне стране оператора доделе;
2. израчуната вредност се додељује променљивој са леве стране оператора доделе, односно, добијена вредност се уписује у меморијску локацију која је резервисана за чување те променљиве;

Посебно треба истаћи да свака наредба доделе вредности променљивој поништава њен претходни садржај, јер променљива може чувати само једну вредност.

Пример 14.1 Наредбе доделе.

```
int x = 3;
int x = x + 5;
a = b * 2 + 26;
p = (a > 10) && (a < 30);
s = "Ja se zovem " + ime + " " + prezime;
```

Приметно је да се у другом изразу променљива *x* појављује, како у изразу са десне стране, тако и на месту променљиве којој се додељује вредност. Наредбу доделе облика:

< променљива > = < променљива > < оп > < израза >

где је <променљива> на левој и десној страни иста, а <оп> један од оператора +, -, *, / или % , можемо записати коришћењем специфичних (сложених) оператора доделе <оп>= на следећи начин:

< променљива > < оп > = < израза >

На пример, израз x = x + 3 * (y - 6) + 5 је еквивалентан са изразом x + = 3 * (y - 6) + 5 , а израз a = a * (b + c) са a* = b + c.

Пример 14.2 Написати наредбу која променљивој:

а) x додељује вредност -77.7
б) n додељује збир вредности променљиве n и броја 5
в) s додељује аритметичку средину реалних бројева a,b и c
г) c додељује дужину хипотенузе правоуглог троугла на основу дужина катета a и b
д) r додељује дужину растојања између тачака са координатама (x, y) и (x, y)

Решење:
а) x = -77.7;
б) n = n + 5;
в) s = (a + b + c)/3;
г) c = sqrt(sqr(a) + sqr(b));
д) r = sqrt(sqr(x - x) + sqr(y - y));

Пример 14.3 Шта ће бити вредност променљивих након извршавања следећих наредби:

```
int a = 4;
int b = 7;
a = 2 * a - b;
b = 7 * a - b;
a = a - a;
```

Решење:
Из наредбе a = 2 * a - b; добија се да је a = 1, а из наредбе b = 7 * a - b; добија се да је b = 0 и из последње наредбе a = a - a; добија се да је и a = 0.
Дакле, након извршења ових наредби, вредност обе променљиве ће бити 0.

Конверзија типова података

Сваки израз даје неку вредност чији тип зависи од типова чланова израза. Када су у неком изразу сви чланови истог типа, тада је и вредност израза тог типа.

Тако је на пример:
1. за декларацију: float y = 5, x = 2; израз y / x даје реалну вредност 2.5 јер су обе променљиве типа float,
2. за декларацију: int y = 5, x = 2**;**
израз y / x даје целобројну вредност 2 (одбацује се остатак дељења) јер су обе променљиве типа int.

Врло често се долази у ситуацију да треба извести аритметичке операције над подацима различитог типа (int, float), па је неопходно податке превести у одговарајући облик. Такође, додела вредности променљивој може се извршити само ако је вредност истог типа као променљива или се може превести у одговарајући тип.
Превођење података из једног у други тип може се у програмском језику C# вршити аутоматски (*имплицитна конверзија*) или под контролом програмера (*експлицитна конверзија*).

Имплицитна конверзија

Разликујемо две ситуације у којима се примењује имплицитна конверзија:

1. при израчунавању вредности израза компајлер аутоматски усклађује типове података који се користе у изразу, уколико је то могуће;
2. при додељивању вредности променљивој, вредност се аутоматски конвертује у тип променљиве уколико је то могуће;

Основна идеја имплицитне конверзије је да се не губе информације и да се сачува прецизност података. Постоје правила конверзије по којима компајлер аутоматски извршава имплицитну конверзију.
Имплицитна конверзија се најчешће примењује код нумеричких типова података.

Променљива типа А, чији скуп могућих вредности је подскуп скупа могућих вредности типа Б, може се имплицитно конвертовати у тип Б.
У табели 16 дате су могућности имплицитне конверзије ужег у шири тип.

Табела 16. *Имплицитна конверзија типова у C#*

тип	**Може се имплицитно конвертовати у**
sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
ulong	sbyte, byte, short, ushort, int, uint, long, char
char	ushort, int, uint, long, ulong, float, double, decimal
float	double
ulong	float, double, decimal

Пример 15.1 Израчунати вредност променљиве b након извршавања следећих наредби:

```
double b = 12.45;
int x = 10;
b = b + x;
```

Решење: При рачунању вредности израза b + x долази до имплицитне конверзије податка x у тип double, јер су променљиве b (double) и x (int) различитог типа, па је x = 10.0, а нова вредност променљиве b = 22.45

Пример 15.2 Израчунати вредности променљивих a, c, d након извршавања овог кода.

```
int  x = 15, y = 4, a;
double c, d, z = 15.0;
a = x / y;
c = x / y;
d = z / y;
```

Решење: Пошто су променљиве x и y (int) истог типа, вредност израза x / y је такође int (3), па је вредност променљиве a = 3, а вредност променљиве c = 3.00 (при додели целобројне вредности реалној променљивој долази до имплицитне конверзије).
При израчунавању вредности израза z / y долази до имплицитне конверзије променљиве y у тип double, јер је променљива z типа double, а променљива y типа int, па је резултат израза 3.75. Променљива d је типа double па је њена вредност једнака вредности израза d = 3.75.

Експлицитна конверзија

Експлицитна конверзија се користи код оних конверзија које се не могу имплицитно извршити. Експлицитном конверзијом програмер, додатним кодом, од компајлера захтева тражену конверзију.
Експлицитна конверзија се остварује коришћењем оператора *cast* или коришћењем класе *Convert*.
Дозвољене експлицитне конверзије нумеричких типова података коришћењем оператора cast дате су табелом 17.

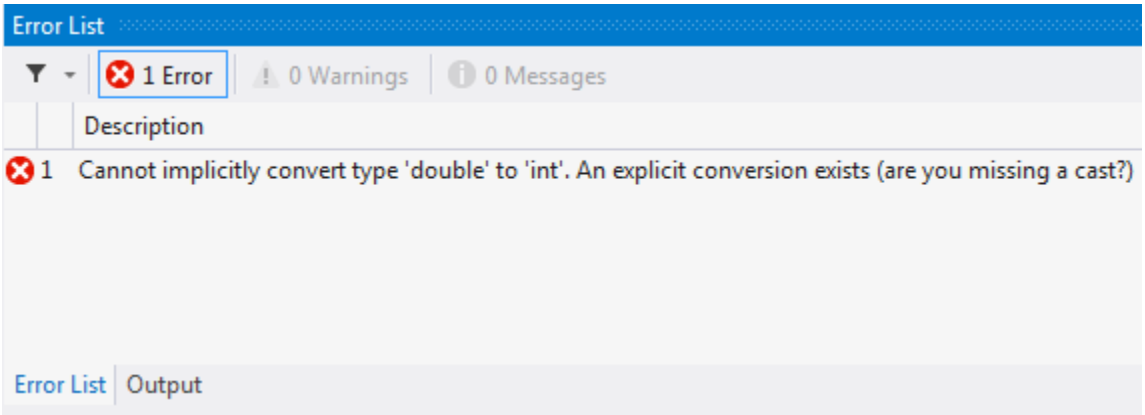
Табела 17. *Експлицитна конверзија типова у C#*

тип	Може се експлицитно конвертовати у
sbyte	byte, ushort, uint, ulong, char
byte	sbyte, char
short	sbyte, byte, ushort, uint, ulong, char
ushort	sbyte, byte, short, char
int	sbyte, byte, short, ushort, uint, ulong, char
uint	sbyte, byte, short, ushort, int, char
long	sbyte, byte, short, ushort, int, uint, ulong, char
ulong	sbyte, byte, short, ushort, int, uint, long, char
char	sbyte, byte, short
float	sbyte, byte, short, ushort, int, uint, long, ulong, char, decimal
double	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, decimal
decimal	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, double

Разматрају се следеће наредбе:

```
int a;
double b = 12.78;
a = b;
```

При покушају превођења овог кода добија се следећа порука да је дошло до грешке, као на слици 90.



Слика 90. Листа грешака које се јављају при извршавању наредби

То значи да се не може имплицитно конвертовати тип double у int. Да би се остварила успешна додела вредности типа double променљивој типа int, неопходно је извршити експлицитну конверзију на следећи начин:

a = (int)b;

па претходне наредбе треба записати у следећем облику:

```
int a;
double b = 12.78;
a =(int)b;
```

По извршавању ових наредби неће доћи до грешке и променљива a сада добија вредност 12. Може се закључити да при конверзији података типа double (float, decimal) у тип int, долази до одбацивања децималног дела податка. Дакле, коришћењем експлицитне конверзије може доћи до губитка информација, односно до сужавања података, када је скуп вредности у који се податак конвертује подскуп скупа вредности податка који конвертујемо. Може се конвертовати само нумерички, знаковни и набројиви тип.

Пример 15.3 Одредити вредности променљиве *x* након извршавања наредби:

```
int a = 11, b = 4;
double x, y;
x = a / b;
y = (double)a / b;
```

Решење:

1) Како су променљиве a и b типа int при рачунању вредности израза a / b не долази до имплицитне конверзије и вредност израза је цео број 2, чиме је дошло до губитка децималног дела резултата (2.75). При додели вредности израза a / b типа int променљивој x типа double, долази до имплицитне конверзије (int у double) и вредност променљиве x је 2.0.

Дакле, x = 2.0.

2) Ако се у изразу a / b употреби оператор cast :

```
x = (double)a / b;
```

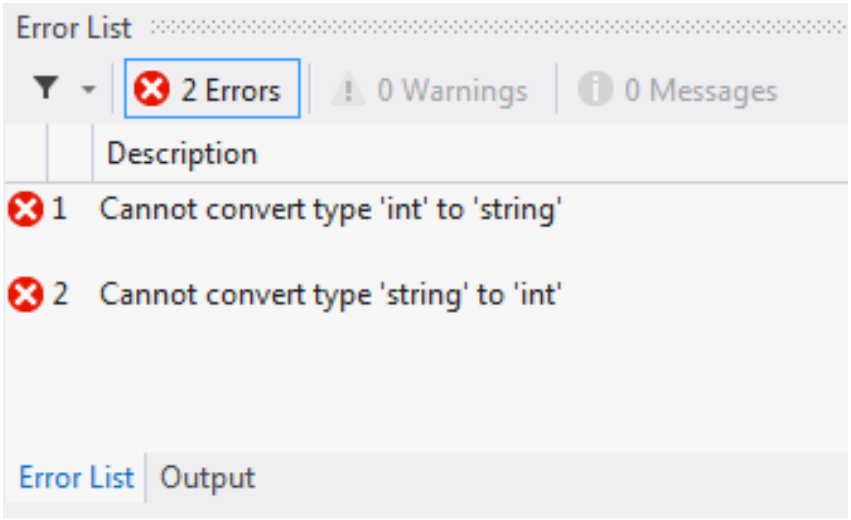
онда се прво извршава експлицитна конверзија вредности промењиве a у тип double. Сада су операнди различитог типа па се врши имплицитна конверзија вредности промењиве b из типа int у тип double. После извршених конверзија, рачуна се вредност израза и додељује промењивој x. По завршетку операције доделе, вредност промењиве x је 2.75.
Дакле, x=2.75.

У примеру 15.3 коришћењем експлицитне конверзије се потиже прецизније рачунање вредности израза.

Ако се покушава превести наредни код:

```
int x = 212;
string s = (string)x;
string p = ”123”;
int y = (int)p;
```

добија се извештај о грешкама као на слици 91 јер наведене експлицитне конверзије није могуће извести оператором cast.



Слика 91. Листа грешака које се јављају при извршавању наредби

Наведене конверзије могуће је обавити коришћењем статичких метода класе Convert којима се подаци основног типа конвертују у други основни тип. Сваки од метода конверзије позива се тако што се наводи име метода, а затим у заградама израз чија се вредност конвертује: Convert.ToInt32(x), Convert.ToDouble(y)...

Коришћењем метода класе Convert претходни код се може записати на следећи начин:

```
int x = 212;
string s = Convert.ToString(x);
string p = ”123”;
int y = Convert.ToInt32(p);
```

Све структуре, којима су представљени основни типови података, садрже метод ToString() којим је омогућено превођење основног типа у string, па тако целобројну променљиву x можемо превести у string на следећи начин:

string s = x.ToString();

Функције за експлицитну конверзију из класе Convert: ToBoolean, ToByte, ToChar, ToDecimal, ToDouble, ToInt16, ToInt32, ToInt64, ToSByte, ToSingle, ToString, ToUInt16, ToUInt32, ToUInt64.

Уношење и приказивање података

C# програми углавном користе улазно/излазне сервисе који су понуђени у .NET Framework библиотеци класа.
У овом поглављу биће речи о уношењу и приказивању података у командној линији, односно у конзоли. За почетак треба објаснити шта је то конзолна апликација.
Конзолна апликација (Console Application) је апликација која се покреће у командном прозору, али нема графички кориснички интерфејс. Комуникација се одвија искључиво са командне линије.
За уношење и приказивање података у конзоли се користи методе класе Console.

WriteLine метода је једна од излазних метода у Console класи. Она приказује стринг који се задаје као параметар на стандардни излазни ток:

System.Console.WriteLine();

Ако се на почетку програма напише директива using System**;** наредба за испис текста је могла изгледати и овако:

Console.WriteLine();

ReadLine metoda je jedna od ulazних метода у Console класе. Користи се за добијање вредности из корисниковог уноса у конзоли.

System.Console.ReadLine(); или Console.ReadLine();

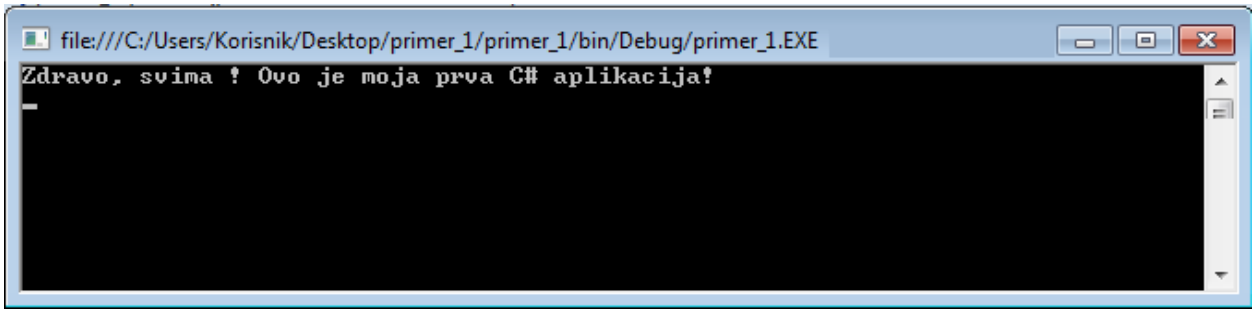
Пример 16.1 Направити конзолну апликацију која на екрану исписује: „Zdravo, svima! Ovo je moja prva C# aplikacija!".

Решење: При покретању Microsoft Visual Studio одабира се New Project, чиме се отвара оквир за дијалог New Project где се одабира Console Application. У пољу Name се даје име конзолној апликацији primer_1, а у пољу Location се изабере место на рачунару где ће се снимити пројекат и када се све то уради, кликне се на дугме OK. На овај начин креирана је једна конзолна апликација. Након тога се отвара прозор за писање кода са већ датим костуром апликације. Линија <static void Main(string[ ] args)> дефинише главну (Main) функцију која се прва покреће приликом извршавања програма.

Алгоритам 24. Исписивање текста у конзоли

```
static void Main(string[] args)
{
    /* Ova linija koda omogucuje ispis teksta.
    Console objekat predstavlja prozor komandne linije, a
    WriteLine je metod ovog objekta kojim se ispisuje tekst,
    odnosno parametar ovog metoda je tekst koji se zeli
    ispisati */
    Console.WriteLine("Zdravo, svima! Ovo je moja prva
    aplikacija!");
    /* Ova linija kod sluzi da se napravi pauza.
    Koristi se metod ReadKey koji ceka da korisnik
    pritisne taster na tastaturi. Bez nje bi se otvorio prozor
    komandne linije, ispisao tekst i prozor bi se odmah zatvorio
    */

    Console.ReadKey();
}
```

Покретање апликације из радног окружења се врши помоћу функцијског тастера F5 на тастатури. Ако је код исправно написан, појављује се прозор конзоле са исписаним текстом, као на слици 92.



Слика 92. Изглед конзоле након покретања програма

Пример 16.2 Направити конзолну апликацију која на екрану исписује име и презиме унете особе, као и место становања, при чему име, презиме и град задајемо преко конзоле

Решење: Као и у примеру 16.1, направи се конзолна апликација која се назива primer_2 и када се отвори прозор са већ датим костуром апликације и главном функцијом Main, у телу ове функције се уписује код који треба да се изврши.

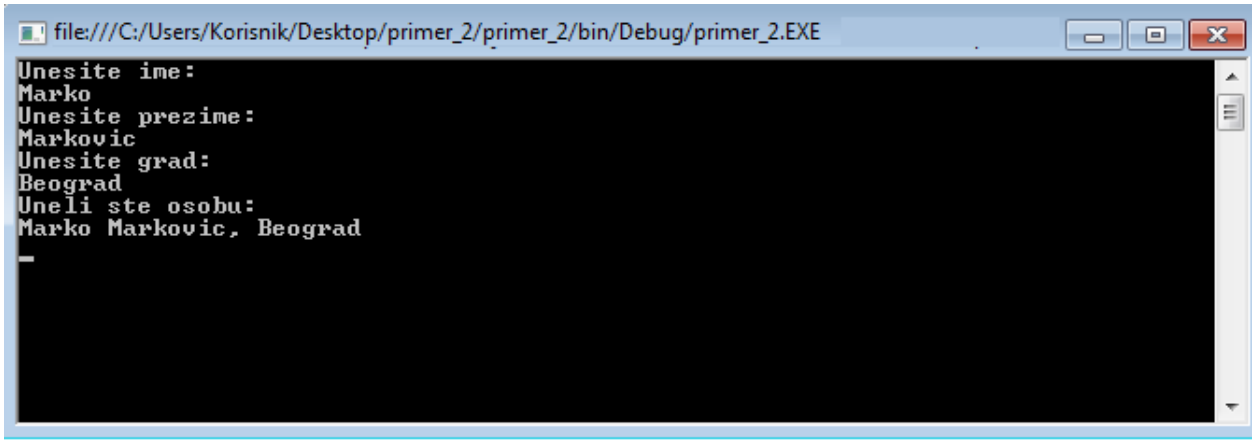
Алгоритам 25. Конзолна апликација која на екрану исписује име и презиме унете особе, као и место становања

```
static void Main(string[] args)
{
    //deklarisanje promenljivih
    string ime;
    string prezime;
    string grad;

    // ispisuje korisniku u komandnoj liniji da unese ime osobe
    Console.WriteLine("Unesite ime:");
    /* Metodom ReadLine ucitavam se ime osobe koje je uneto u
    komandnoj liniji u promenljivu ime tipa string*/
    ime = Console.ReadLine();
    //Ispisuje korisniku u komandnoj liniji da unese prezime
    //osobe
    Console.WriteLine("Unesite prezime:");
    /* Metodom ReadLine ucitava se prezime osobe koje je uneto
    u komandnoj liniji u promenljivu prezime tipa string*/
    prezime = Console.ReadLine();
    //Ispisuje korisniku u komandnoj liniji da unese grad
    Console.WriteLine("Unesite grad: ");
    /* Metodom ReadLine ucitava se grad koji je unet
    u komandnoj liniji u promenljivu grad tipa string*/
    grad = Console.ReadLine();
    Console.WriteLine("Uneli ste osobu: ");
```

```
    //Ispisuje ime i prezime i grad unete osobe u komandnu
    //liniju
    Console.WriteLine(ime + " " + prezime + ", " + grad);
    /* ova linija kod sluzi da se napravi pauza. Koristi se
    metod ReadKey i korisnik treba da pritisne taster na
    tastaturi za kraj programa*/
    Console.ReadKey();

}
```

Када се покрене апликацију помоћу тастера F5 отвара се прозор у коме пише „Unesite ime:". Треба унети име (нпр. Марко) и притиснути Enter на тастатури, након тога се појављује следећа линија у којој пише „Unesite prezime:" и треба унети презиме особе (нпр. Марковић). Када се поново притисне Enter појавиће се линија „Unesite grad:" (нпр. Београд) и на крају када се притисне Enter исписаће се „Uneli ste osobu:" и у следећој линије се појављује име и презиме унете особе и град у којем живи (у нашем примеру Марко Марковић, Београд).

Изглед прозора треба да буде као на слици 93.



Слика 93. Изглед конзоле након покретања програма

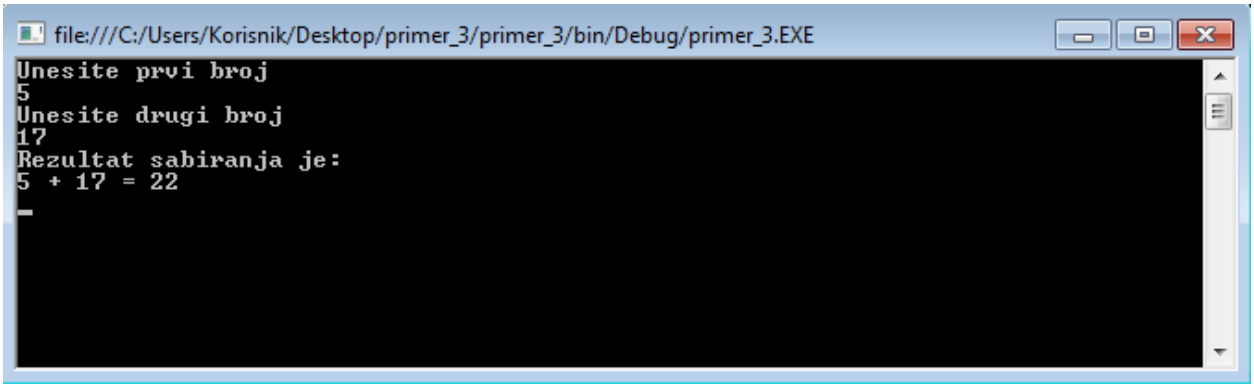
Пример 16.3 Направити конзолну апликацију за унос и сабирање два цела броја.

Решење: Као и у претходним примерима, направи се конзолна апликација у која се назива primer_3 и када се отвори прозор са већ датим костуром апликације и главном функцијом Main, у телу ове функције уписује се код који треба да се изврши.

Алгоритам 26. Конзолна апликација која на екрану исписује која исписује збир два унета броја

```
static void Main(string[] args)
{
    int prvi, drugi, rezultat;
    //U ove promenljive smesta se unos iz konzole
    string string1, string2;
    //ispisuje u konzoli da korisnik unese prvi broj
    Console.WriteLine("Unesite prvi broj");
    //ReadLine metodom dobija se vrednost koju je korisnik uneo
    //u konzoli
    string1 = Console.ReadLine();
    //Metodom int.Parse iz string1 izdvaja se celobrojna
    //vrednost
    prvi = int.Parse(string1);
    //Na isti nacin kao prvi broj, dobija se i drugi broj iz
    //konzole
    Console.WriteLine("Unesite drugi broj");
    string2 = Console.ReadLine();
    drugi = int.Parse(string2);
    //odredjuje se zbir ta dva broja i ispisuje se u konzoli
    rezultat = prvi + drugi;
    Console.WriteLine("Rezultat sabiranja je: ");
    Console.WriteLine(prvi + " + " + drugi + " = " + rezultat);
    Console.ReadKey();
}
```

Када се покрене апликација из радног окружења помоћу тастера F5 отвара се прозор конзоле у коме пише Unesite prvi broj . Уношењем неког броја (нпр. 5) и притискањем Enter на тастатури, појављује се Unesite drugi broj, и тада уносимо и други број (нпр. 17). Када се поново притисне Enter појавиће се тражени резултат сабирања (у примеру то је 22). Изглед прозора треба да буде као на слици 94.



Слика 94. Изглед конзоле након покретања програма

Поред уношења и исписивања података у командној линији, односно конзоли, могуће је и уношење података преко TextBox-а и њихово исписивање у TextBox-у и Label-и.

Унос података у TextBox врши се преко тастатуре, тако што се TextBox прво постави на форму, а онда се у њега укуца тражени податак преко тастатуре. Ако се у TextBox-у жели приказати податак, на пример, вредност променљиве *x*, то се постиже наредбом:

```
textBox1.text = x.toString();
```

Као што су се исписивали подаци у TextBox-у, могуће је исписати податке и на лабелу (Label). Када поставимо Label на форму, следећим наредбама је омогућен испис података на њој.

```
label1.Text = x;
```

На лабели се исписује вредност променљиве *x*, на пример, ако је *x = 5*, исписаће само 5.

```
label1.text = "Vrednost promenljive x je " + x + ".";
```

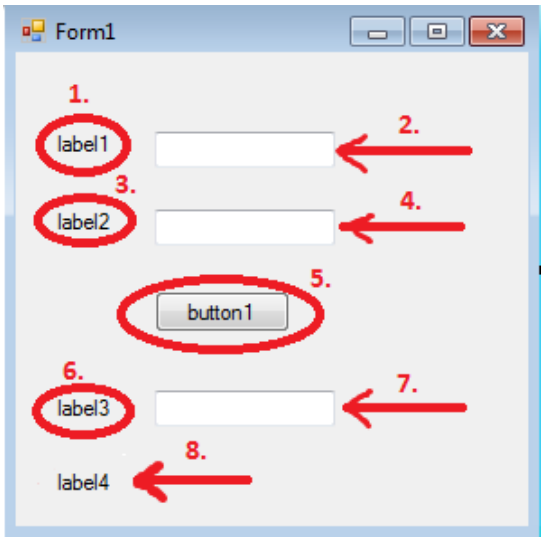
Овом наредбом на лабели се исписује: *Вредност променљиве x је 5.* (ако је x = 5)
Пример 16.4 илуструје унос и приказивање података у TextBox-у і Label-у.

Пример 16.4 Написати програм који кликом на дугме SABERI узима два цела броја које је корисник унео у TextBox-ове, рачуна њихов збир и исписује резултат у TextBox и на Label-у.

Решење:

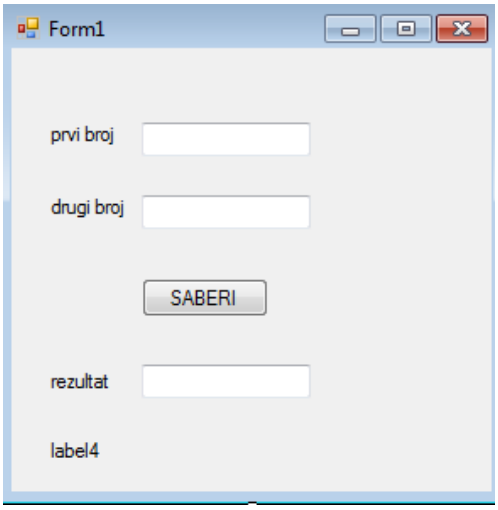
1. label1 – преименује се у назив *prvi broj*
2. textBox1 - у ово поље корисник уноси цео број

3. label2 - преименује се у назив *други broj*
4. textBox2 - у ово поље корисник уноси цео број
5. button1 - преименује се у дугме *SABERI*
6. label3 - преименује се у назив *rezultat*
7. textBox3 - у ово поље ће се исписати резултат рада програма
8. label4 - такође ће се исписати резултат рада програма



Слика 95. Изглед форме пре покретања програма

Када форма има изглед као на слици 96, двоструки клик на дугме *SABERI* и када се отвори нови прозор у телу функције *button1_Click*, уноси се код који треба да се извршава када кликнемо на то дугме.

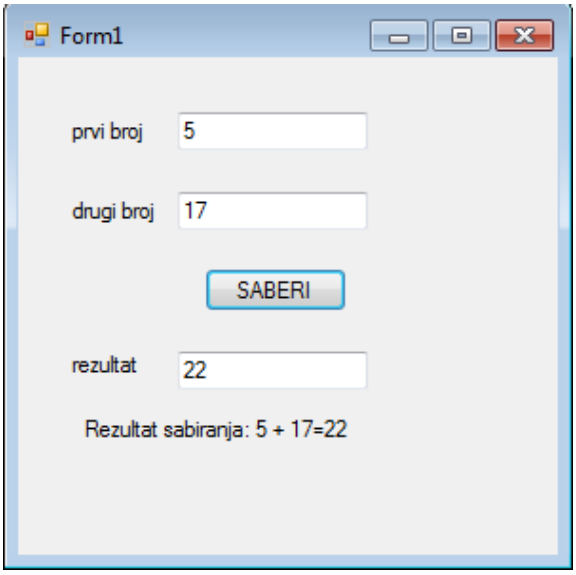


Слика 96. Изглед конзоле након покретања програма

Алгоритам 27. Израчунавање збира два броја

```
private void button1_Click(object sender, EventArgs e)
{
    //deklaracija promenljivih
    int x, y, rez;
    //uzima se vrednost iz prvog textBox-a i prevodimo ga u int
    x = int.Parse(textBox1.Text);
    //uzima se vrednost iz drugog textBox-a i prevodimo ga u int
    y = int.Parse(textBox2.Text);
    //odredjuje se zbir dva uneta broja
    rez = x + y;
    //ispisuje se rezultat u textBox
    textBox3.Text = rez.ToString();
    //ispisuje se "Rezultat sabiranja:" na labelu
    label4.Text = "Rezultat sabiranja: ";
    //ispisuje se rezultat na labelu
    label4.Text = label4.Text + x + " + " + y + "=" + rez;
}
```

При покретању апликације из радног окружења помоћу тастера F5 отвара се форма и потребно је унети бројеве у TextBox-ове и затим притиснути дугме SABERI. Након тога ће се исписати резултат у TextBox и Label. На пример, за унете бројеве 5 и 17, резултат рада програма је приказан на слици 97.



Слика 97. Изглед форме након покретања програма

Закључак

У овам раду су детаљно и кроз мноштво примера описани типови података, наредбе и изрази у програмском језику C#. Видело се зашто су они и колико важни, као и да то спада у основу програмирања и да је без њих бесмислено започети решавање било каквог задатка. Веома је важно да се ученици са овим добро упознају, јер су то чиниоци који се свакодневно користе у програмирању и без чега је програмирање потпуно незамисливо. Још једна битна ствар јесте да се добро обрати пажња које се операције и са којим типовима података могу обављати и какав резулат ће се добити кад се примене жељене операције.

Поред самог упознавања са целобројним и реалним типом и операцијама које се могу над њима обављати, скренута је пажња и на разлике и сличности међу њима.

Што се логичког типа тиче, ово је тип који представља саму срж рачунарства и програмирања, јер се све своди на две основне вредности - true и false. Логика представља нешто на чему се целокупно програмирање заснива па ће мноштво примера у овом раду развити и подстаћи ученика на размишљање.

Низовни тип података је нешто ново, али опет познато ученицима и неопходно је да се добро објасни и научи јер може касније бити веома корисно, нарочито ако се од ученика тражи обрада велике количине података.

Стринг тип је такође битан, јер се у свакодневном животу подједнако често барата са стринговима, тј. нискама, реченицама, као и бројевима. За сваки реалан проблем може се направити програмерски модел који га решава. Ту се ученици упознају и са класама, где могу да праве сопствене типове података и да са њима рукују.

Да би се могле обављати сложеније операције, мора се прво упознати и са наредбама доделе, као и изразима који се појављују у програмирању.

Као што и сам наслов каже, ово су *Електронске лекције о типовима података, наредбамаи изразима,* што значи да су доступне у електронској форми на веб адреси:

http://www.edusoft.math.rs/csharp/

Бројни материјали и разни уџбеници у електронској форми, могу много помоћи у раду и учењу. На модеран, концептуално оригиналан и технолошко иновативан начин могу бити приказани многи наставни садржаји који се обрађују у оквиру теоријске и практичне наставе из предметне области. Сваки материјал оваквог облика може разне садржаје учинити интересантнијим. Интеракција је један од битних особина оваквих материјала. Она помаже у учењу и што бољем разумевању садржаја, подстиче креативност и размишљање корисника. Један од таквих материјала је и овај курс о типовима података, наредбама и изразима. У њему ученик може озбиљније, и можда не тако једноставне

ствари, схватити и научити кроз бројне занимљиве примере из живота. Разне анимације стварају илузију кретања цртежа и на тај начин подстичу на размишљање, и многе појмове који су можда компликованији и апстрактнији за разумевање, могу учинити јаснијим, и то на један веома креативан начин. Овакав рад ће ученицима у школи сигурно олакшати и помоћи у учењу, али и онима који желе да то самостално савладавају. Велика предност оваквих курсева је и у томе што их је могуће користити, дословно речено ,,од куће", приступати им директно са интернета и то бесплатно.

Развој технологије и рачунара помаже и подстиче настанак оваквих елекртонских материјала. Све више су заступљени у многим областима, као помоћно, али и као главно средство при раду и учењу било да је то програмирање, математика, музика, филозофија... Овакав курс ће у свакој области бити само предност и велика помоћ у спознаји разних садржаја.

Литература

[1] Jesse Liberty, *Програмирање на језику C#, превод четвртог издања*, Микро књига, Београд 2007.

[2] Paul Kimel, *C#: напредно програмирање*, Микро књига, Београд *2003.*

[3] Матковић Станка, Вуковић Душа, *Основи програмирања у програмском језику C#,* Завод за уџбенике и наставна средства, Београд 2012.

[4] Brian W. Kernighan, Dennis M. Ritchie, *Програмском језик C*, ЦЕТ, Београд 2003.

[5] John Sharp, *Microsoft Visual C# 2008* , ЦЕТ, Београд 2008.

[6] Laslo Kraus, *Решени задаци из програмског језика C#,* Академска мисао, Београд 2007.

[7] Charles Wright, *C# кроз практичне примере*, Микро књига, Београд 2002.

[8] John Sharp, *Microsoft Visual C# 2012 корак по корак*, ЦЕТ, Београд 2013.

[9] Clovis Tondo, Scott Gimpel, *Програмском језик C# - Решења*, ЦЕТ, Београд 2004

[10] System.Math class in C#, http://www.c-sharpcorner.com

[11] Математички факултет у Београду, http://www.matf.bg.ac.rs

[12] C#, http://www.dotnetperls.com

[13] CSharp tutorial, http://www.java2s.com

[14] Објектно оријентисано програмирање, http://viewsource.rs

[15] C# tutorial, http://www.functionx.com

[16] Microsoft Developer Network, http://msdn.microsoft.com

[17] C# tutorial, http://zetcode.com

Садржај