Наредбе и изрази

У току обраде података често долазимо у ситуацију да на податке примењујемо различите операције којима стварамо нове вредности ((међу)резултате), које даље обрађујемо применом нових операција коначно много пута, док не добијемо коначне резултате.

За то су нам корисне одговарајуће наредбе и изрази. У овој теми ћемо се упознати изразима и наредбама у програмском језику C#.

Изразе добијамо комбиновањем променљивих, константи и оператора.

Изрази могу бити **прости** (само константа или променљива) или **сложени** са произвољним бројем оператора и елемената израза одвојених заградама. Сваки израз има свој тип и своју вредност. Тип израза зависи од типова подизраза који га чине, као и од оператора којим се ти подизрази повезују.

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.

Примери израза:

123

’a’

a + 4 * b

(x == 0) && (y == 0)

“Ja se zovem ” + ime + “ ” + prezime

x > y + 12

(12 + x * (a + 3)) / (b - 3) + 1

Наредба у C# језику је једна инструкција програмског кода која се завршава са знаком **" ; "**.

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.

У просте наредбе спадају наредбе декларације и иницијализације променљивих, наредба доделе и наредба позива метода.

Примери простих наредби:

```
//Naredba deklaracije promenljivih x, y i s tipa int
int x, y, s;
//Naredba deklaracije promenljive ocena tipa int i inicijalizacija vrednosti
promenljive na 5
int ocena = 5;
//Naredba dodele koja promenljivoj x dodeljuje vrednost te promenljive uvecanu za 3
x = x + 3;
//Naredba poziva metoda ReadLine klase Console
string ime=Console.ReadLine();
```

У наредном примеру сложене наредбе (блока) се замењују вредности целобројних променљивих x и y.

Пример сложене наредбе:

```
//Naredbe deklaracije i inicijalizacije promenljivih x i y tipa int
int x = 3;
int y = 5;
{
    //Naredba deklaracije pomocne promenljive z tipa int
    int z;
    //Promenljivoj z dodeljujemo staru vrednost promenljive x
    z = x;
    //Promenljivoj x dodeljujemo staru vrednost promenljive y
    x = y;
    //Promenljivoj y dodeljujemo vrednost promenljive z, tj.staru vrednost promenljive
x
    y = z;
}
```

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 није затворена заграда, или ако није стављена ";" на крају наредбе, или није затворена витичаста заграда на крају блока наредби, апликација која их садржи неће моћи да се компајлира, а самим тим ни да се покрени јер ће се јавити грешке. Такође, до синтаксне грешке може доћи ако оператор %, који даје остатак при дељењу, применимо 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.

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).

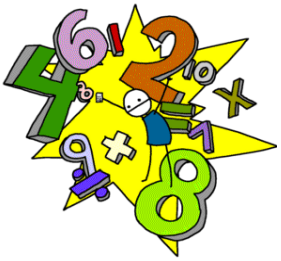
Код семантичких грешака апликација је успешно компајлирана, приликом рада не пријављује никакве грешке, нити престаје са радом. Све је у реду, али апликација "само" не ради оно што треба да ради. Враћа погрешне резултате, не исписује податке... Ово су вероватно грешке које је најтеже открити јер су последице лошег дизајна апликације. 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.

- У првом изразу x = y променљивој x се додељује вредност променљиве y.
- У другом изразу x == y проверава се да ли је вредност променљиве x једнака вредности променљиве y.

Сличне грешке се праве код оператора **&** и **&&**, као и код оператора | и ||, па је неопходно правити разлику између логичких оператора (**&&**, ||) и битовских оператора (**&**, |).

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:

- **једноставни** - састоје се само од једног операнда
- **сложени** – састоје се од два или више операнада повезаних аритметичким операторима

Операнди могу бити:

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

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

Аритметички оператори	
бинарни	
+	сабирање
-	одузимање
*	множење
/	дељење
%	остатак при дељењу првог операнда другим
унарни	
+	промена знака операнда или израза
-	

Слика 5.1. Аритметички оператори

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 и следећи пример.

Пример 1. Израчунати вредност количника:

а) 9 / 5

б) 9.0 / 5.0

в) 9.0 / 5

Решење:

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 ћемо показати на следећем примеру.

Пример 2. Израчунати вредност променљивих након извршавања следећих наредби:

а)

```
//Deklarisanje promenljivih
int n, x;
//Promenljivoj n dodeljujemo vrednost 3
n = 3;
//Promenljivoj x dodeljujemo vrednost izraza ++n-2
x = ++n - 2;
```

б)

```
//Deklarisanje promenljivih
int n, x;
//Promenljivoj n dodeljujemo vrednost 3
n = 3;
//Promenljivoj x dodeljujemo vrednost izraza n++ - 2
x = n++ - 2;
```

Решење:

а) Овде имамо префиксно коришћење 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 имамо постфиксно коришћење 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.***

Приоритет аритметичких оператора:

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 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

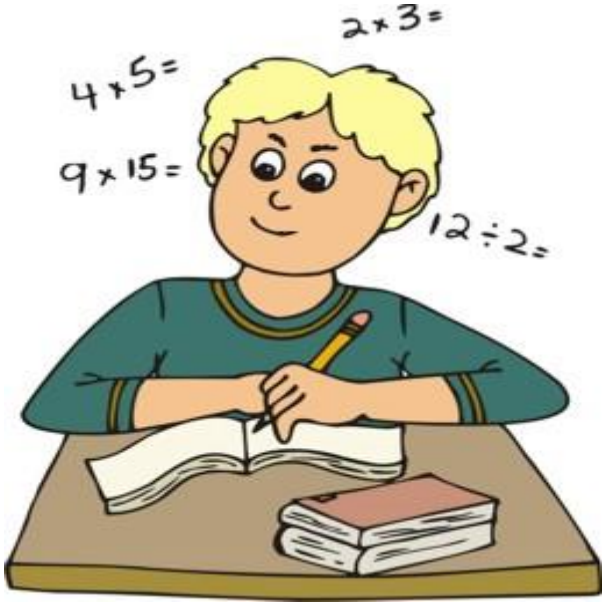
5. 4 + 24 % (2 * 3)

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 4. Израчунати вредност промењљивих x и y након извршавања следећег дела кода:

```
//Deklarisanje promenljivih
int x, y;
int a = 0;
int b = 0;
//Uvecavanje vrednosti promenljive a za 1
a++;
//Uvecavanje vrednosti promenljive b za 1
++b;
//Promenljivoj x se dodeljuje vrednost promenljive a uvecana za 1
x = ++a;
//Promenljivoj x se dodeljuje vrednost promenljive b koja se zatim uveca za 1
y = b++;
```

Решење:

Из a++; следи да се вредност промењљиве a увећа за 1, тј. a=1.

Из ++b; следи да се вредност промењљиве b увећа за 1, тј. b=1.

Из x = ++a; следи да префиксни оператор ++ увећава промењљиву a, па затим користи њену вредност, односно промењљива x добија вредност једнаку увећаној вредности промењљиве a. Дакле, a = 2, x = 2.

Из y = b++; следи да прво промењљива y добије вредност промењљиве b = 1, па је и y = 1, а затим постфиксни оператор ++ увећа вредност промењљиве b за 1. Дакле, b = 2, y = 1.

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 промењљиве (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.

x	!x
true	false
false	true

Слика 5.2. Таблица логичког оператора !

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

&&	false	true
false	false	false
true	false	true

Слика 5.3. Таблица логичког оператора &&

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

||	false	true
false	false	true
true	true	true

Слика 5.4. Таблица логичког оператора ||

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

Лењо израчунавање

У изразима у којима се јавља логички оператор &&, уколико је вредност првог операнда једнака false, онда се не израчунава вредност другог операнда. 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, 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. На пример, након израчунавања вредности израза (1 < 2) || (x++ > 10), се не мења вредност променљиве x јер подизраз (1 < 2) има вредност true, па се вредност другог операнда не рачуна, а након (2 < 1) || (x++ > 10), се мења (увећава за 1) вредност променљиве x јер је вредност подизраза (2 < 1) једнака false, па се израчунава и вредност другог операнда.

Пример 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, можемо 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), можемо закључити, без рачунања вредности другог операнда, да наш израз (y + 2 == 4) || (x / 3 != 0) има вредност true, на основу лењог израчунавања.

Пример 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 3. Записати у облику логичких израза:

1. x Є [0, 1]
2. x Є [-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)

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.

Из тог разлога 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*

Напомена: Свака наредба доделе вредности променљивој поништава њен претходни садржај јер променљива може чувати само једну вредност.

Примери наредбе доделе:

- x = 5;
- x = x + 2;
- a = b * 3 + 78;
- p = (a > 5) && (a < 20);
- s ="Ja se zovem " + ime + " " + prezime;

Приметимо да се у другом примеру 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 % , називамо **сложеном наредбом доделе.** Можемо је записати коришћењем сложених оператора доделе**<оп>=** на следећи начин:

< променљива > < оп > = < израза >

Примери сложене наредбе доделе:

- x *= 5; што је еквивалентно са x = x * 5;
- a -= 2 + y; што је еквивалентно са a = a - (2 + y);
- x += 3 - (y - 6) + 5; што је еквивалентно са x = x + 3 * (y - 6) + 5;
- a /= b + c; што је еквивалентно са a = a / (b + c);

Пример 1. Напиши наредбу која променљивој:

а) x додељује вредност -77.7

b) n додељује збир вредности променљиве n и броја 5

c) s додељује аритметичку средину реалних бројева a,b и c

d) c додељује дужину хипотенузе правоуглог троугла на основу дужина катета a и b

e) r додељује дужину растојања између тачака са координатама (x_1 ,y_1) и (x_2 ,y_2)

Решење:

a) x = -77.7;

b) n = n + 5;

c) s = (a + b + c) / 3;

d) c = sqrt(sqr(a) + sqr(b));

e) r = sqrt(sqr(x_1 - x_2) + sqr (y_1 - y_2));

Пример 2. Шта ће бити вредност променљивих након извршења следећих наредби:

```
//Deklarisanje i inicijalizacija vrednosti promenljive a
int a = 4;
//Deklarisanje i inicijalizacija vrednosti promenljive b
int b = 7;
//Promenljivoj a dodeljujemo vrednost izraza na desnoj strani
a = 2 * a - b;
//Promenljivoj b dodeljujemo vrednost izraza na desnoj strani
b = 7 * a - b;
//Promenljivoj a dodeljujemo vrednost izraza na desnoj strani
a = a - a;
```

Решење:

Из наредбе *a = 2 * a - b;* добијамо да је *a = 2 * 4 - 7 = 1*.

Из наредбе *b = 7 * a - b;* добијамо да је *b = 7 * 1 - 7 = 0*.

Из последње наредбе *a = a - a;* добијамо да је и *a = 0*.

Након извршења ових наредби, вредност обе променљиве ће бити *0*, дакле, *a = 0, b = 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.

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 смо у ситуацији да изводимо 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:

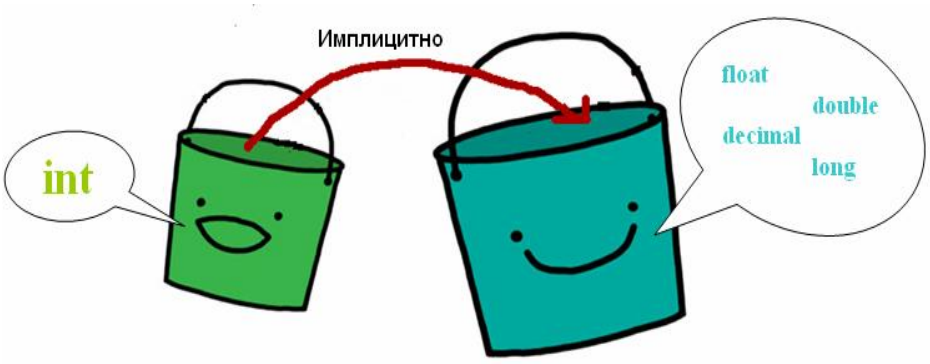
• при израчунавању вредности израза компајлер аутоматски усклађује типове података који се користе у изразу, уколико је то могуће

• 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.*



Слика 5.5. Приказ имплицитне конверзије ужег у шири тип (int у long, float, double или decimal)

На *слици 5.5* је приказана имплицитна конверзија помоћу две кофе, мање и веће, тако што све податке из мање кофе који су ужег типа (у овом случају int) можемо пребацити у већу кофу, односно у шири тип (long, float, double или decimal), без губитка података.

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
char	ushort, int, uint, long, ulong, float, double, decimal
float	double
ulong	float, double, decimal

Слика 5.6. Табела дозвољених имплицитних конверзија

Пример 1. Израчунати вредност променљиве b након извршавања следећих наредби:

```
//Deklarisanje i inicijalizacija promenljive b
double b = 12.45;
//Deklarisanje promenljive x
int x = 10;
//Promenljivoj b dodeljujemo njenu vrednost uvecanu za vrednost x
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. Дакле, након конверзије је *x = 10.0*, па је нова вредност променљиве *b = 12.45 + 10.0 = 22.45*

Пример 2. **Израчунати вредности променљивих *a, c, d* након извршавања следећих наредби.**

```
//Deklarisanje i inicijalizacija promenljivih
int  x = 15, y = 4, a;
double c, d, z = 15.0;
//Promenljivoj a dodeljujemo vrednost izraza x / y
a = x / y;
//Promenljivoj c dodeljujemo vrednost izraza x / y
c = x / y;
//Promenljivoj d dodeljujemo vrednost izraza z / y
d = z / y;
```

Решење:

Пошто су променљиве *x* и *y* истог типа (*int*), 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

Користи се 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*** . Оператор *cast* користимо тако што у заградама наведемо тип у који желимо да конвертујемо вредност израза који следи за оператором *cast*: **(тип) променљива**

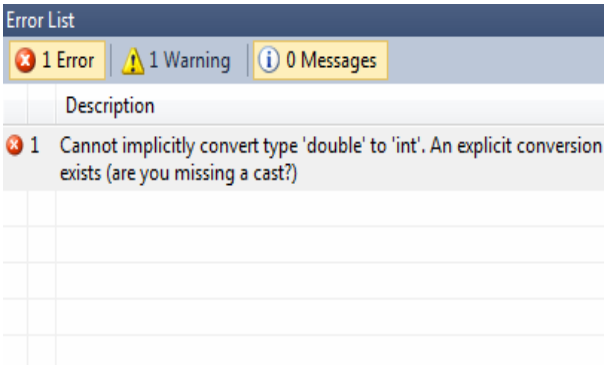
Тип	Може се експлицитно конвертовати у
sbyte	byte, ushort, uint, ulong, char
byte	sbyte, char
short	sbyte, byte, ushort, uint, ulong, char
ushort	sbyte, byte, short, char
int	sbyte, byte, short, ushort, uint, ulong, char
uint	sbyte, byte, short, ushort, int, char
long	sbyte, byte, short, ushort, int, 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

Слика 5.7. Табела дозвољених експлицитних конверзија

Посматрајмо следеће наредбе:

```
//Deklarisanje promenljive a
int a;
//Deklarisanje i inicijalizacija promenljive b
double b = 12.78;
//Promenljivoj a dodeljujemo vrednost promenljive b
a = b;
```

При покушају превођења овог кода добијамо следећу поруку да је дошло до грешке:



Слика 5.8. 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:

```
//Deklarisanje promenljive a
int a;
//Deklarisanje i inicijalizacija promenljive b
double b = 12.78;
//Promenljivoj a dodeljujemo vrednost promenljive b koja je eksplicitno konvertovana u
int
a =(int) b;
```

По извршавању ових наредби неће доћи до грешке и променљива a сада добија вредност 12. Можемо 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 3. Одредити вредности променљиве *x* након извршавања наредби:

```
//Deklarisanje i inicijalizacija promenljivih a i b
int a = 11, b = 4;
//Deklarisanje promenljivih x i y
double x, y;
//Promenljivoj x dodeljujemo vrednost izraza a / b
x = a / b;
//Promenljivoj y dodeljujemo vrednost izraza (double)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 у изразу *a / b* употребимо оператор *cast*

   x = (double)a / b;

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

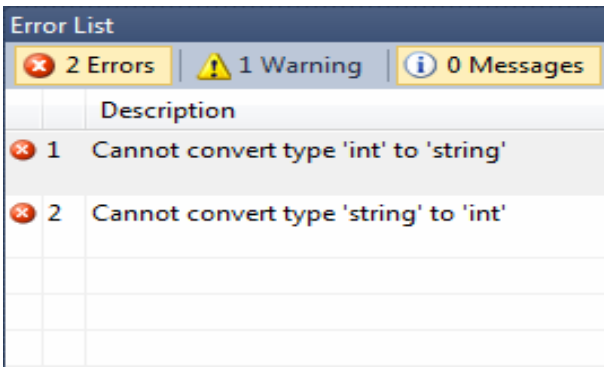
   Дакле, *x=2.75*.

   У овом примеру коришћењем експлицитне конверзије постигли смо прецизније рачунање вредности израза.

Ако бисмо покушали да преведемо наредни код:

```
//Deklarisanje i inicijalizacija promenljive x
int x = 212;
//Promenljivu x tipa int konvertujemo u tip string
//operatorom cast i dodeljujemo tu vrednost promenljivoj s tipa string
string s = (string)x;
//Deklarisanje i inicijalizacija promenljive x
string p = ”123”;
//Promenljivu p tipa string konvertujemo u tip int
//operatorom cast i dodeljujemo tu vrednost promenljivoj y tipa int
int y = (int)p;
```

добили бисмо извештај о следећим грешкама:



Слика 5.9. Листа грешака које се јављају при извршавању наредби

Што значи да наведене експлицитне конверзије није могуће извести оператором cast.

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 пример можемо записати на следећи начин:

```
//Deklarisanje i inicijalizacija promenljive x
int x = 212;
//Promenljivu x tipa int konvertujemo u tip string
//metodom Convert.ToString() i dodeljujemo tu vrednost promenljivoj s tipa string
string s = Convert.ToString(x);
//Deklarisanje i inicijalizacija promenljive p
string p = ”123”;
//Promenljivu p tipa string konvertujemo u tip int
//metodom Convert.ToInt32() i dodeljujemo tu vrednost promenljivoj y tipa int
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 промењиву x можемо превести у stringна 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. Упознаћемо се са уношењем и приказивањем података у командној линији, односно у конзоли. За почетак да објаснимо 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 користимо ***методе класе Console***.

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

System.Console.WriteLine();

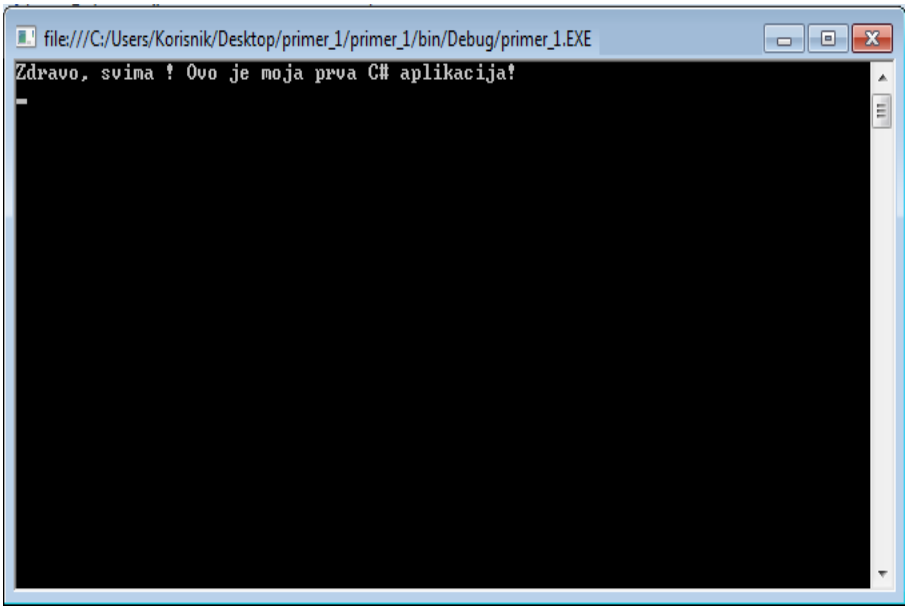
Ако се на почетку програма напише директива **using System**; наша наредба за испис текста је могла изгледати и овако:

Console.WriteLine();

ReadLine metoda је једна од улазних метода класе Console. Користи се за добијање вредности из корисниковог уноса у конзоли.

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

Пример 1. Направити конзолну апликацију која на екрану исписује: *"Zdravo, svima! Ovo je moja prva C# aplikacija!"*.



Слика 5.10. Изглед конзоле након покретања програма у примеру 1.

Решење:

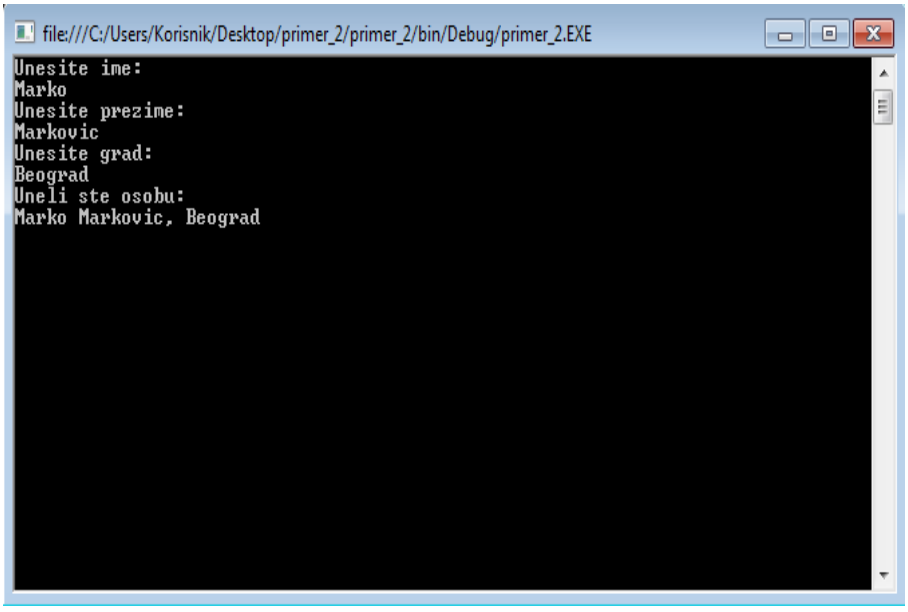
У теми *Увод у развојно окружење* упознали смо се са креирањем новог пројекта, као и конзолне апликације. Када креирамо конзолну апликацију, отвара се прозор за писање кода са већ датим

костуром апликације. Линија ***static void Main(string[ ] args)*** дефинише главну (Main) функцију која се прва покреће приликом извршавања програма. У телу ове функције треба уписати следећи код који је потребно извршити.

```
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 zelimo da ispisemo
    Console.WriteLine("Zdravo, svima! Ovo je moja prva C# 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 вршимо помоћу функцијског тастера **F5** на тастатури. Ако је код исправно написан, појављује се прозор конзоле са исписаним текстом, као на слици 5.10.

Пример 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.



Слика 5.11. Изглед конзоле након покретања програма у примеру 2.

Решење:

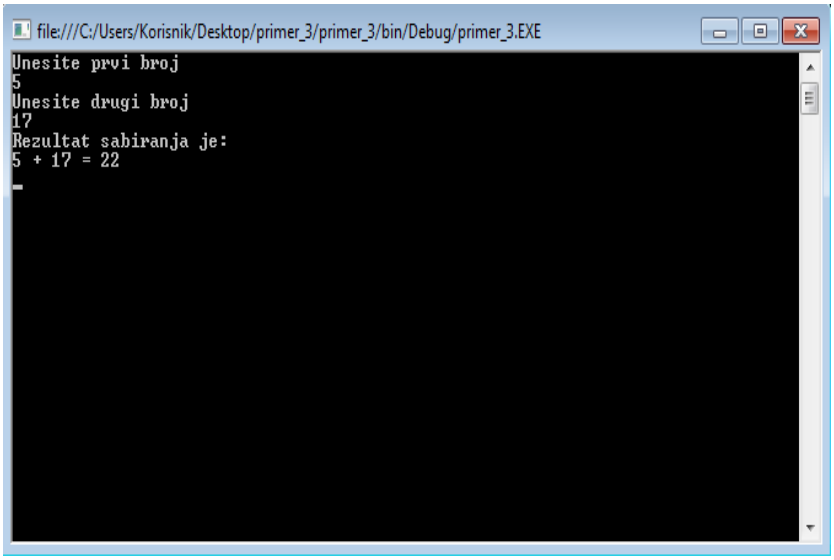
Као и у претходном примеру, направимо конзолну апликацију и у телу Main функције уписаћемо следећи код који треба да се изврши.

```
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 ucitavamo 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 ucitavamo 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 ucitavamo 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 покренемо апликацију помоћу тастера **F5** отвара нам се прозор у коме пише **Unesite ime**: . Унесемо неко име (нпр. Марко) и притиснимо **Enter** на тастатури, након тога се појављује следећа линија у којој пише **Unesite prezime**: и унесемо презиме особе (нпр. Марковић).Када притиснемо поново **Enter** појавиће се линија **Unesite grad:** (нпр. Београд) и на крају када притиснемо **Enter** исписаће се **Uneli ste osobu:** и у следећој линије појавиће нам се име и презиме унете особе и град у којем живи(у нашем примеру Марко Марковић, Београд). Изглед прозора треба да буде као на слици 5.11.

Пример 3. Направити конзолну апликацију за унос и сабирање два цела броја.



Слика 5.12. Изглед конзоле након покретања програма у примеру 3.

Решење:

Као и у претходним примерима, направимо конзолну апликацију и у телу Main функције уписаћемо следећи код који треба да се изврши.

```
static void Main(string[] args)
{
    //Deklaracija promenljivih
    int prvi, drugi, rezultat;
    //U ove promenljive smesticemo unos iz konzole
    string string1, string2;
    //Ispisuje u konzoli da korisnik unese prvi broj
    Console.WriteLine("Unesite prvi broj");
    //ReadLine metodom dobijamo vrednost koju je korisnik uneo u konzoli
    string1 = Console.ReadLine();
    //Metodom int.Parse iz string1 izdvajamo celobrojnu vrednost
    prvi = int.Parse(string1);
    //Na isti nacin kao prvi broj, dobijamo i drugi broj iz konzole
    Console.WriteLine("Unesite drugi broj");
    string2 = Console.ReadLine();
    drugi = int.Parse(string2);
    //Odredjujemo zbir ta dva broja i ispisemo ga u konzoli
    rezultat = prvi + drugi;
    Console.WriteLine("Rezultat sabiranja je: ");
    Console.WriteLine(prvi + " + " + drugi + " = " + rezultat);
    Console.ReadKey();
}
```

Када покренемо апликацију из радног окружења помоћу тастера **F5** отвара нам се прозор конзоле у коме пише **Unesite prvi broj**. Унесимо неки број (нпр.5) и притиснимо **Enter** на тастатури, након тога се појављује **Unesite drugi broj** и унесемо и други број (нпр.17). Када притиснемо поново **Enter** појавиће нам се тражени резултат сабирања (у нашем примеру то је 22) . Изглед прозора треба да буде као на слици 5.12.

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**. Са компонентама TextBox и Label већ смо се упознали у теми 3.*Увод у развојно окружење* тако да знамо на који начин се постављају на форму и њихова основна својства.

Унос података у TextBox врши се преко тастатуре,тако што се TextBox прво постави на форму, а онда у њега укуцамо тражени податак преко тастатуре. Ако у TextBox-у желимо да прикажемо податке, на пример, вредност промењљиве *x*, то ћемо урадити следећом наредбом:

textBox1.text = x.toString();

Као што смо исписивали податке у TextBox, 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.

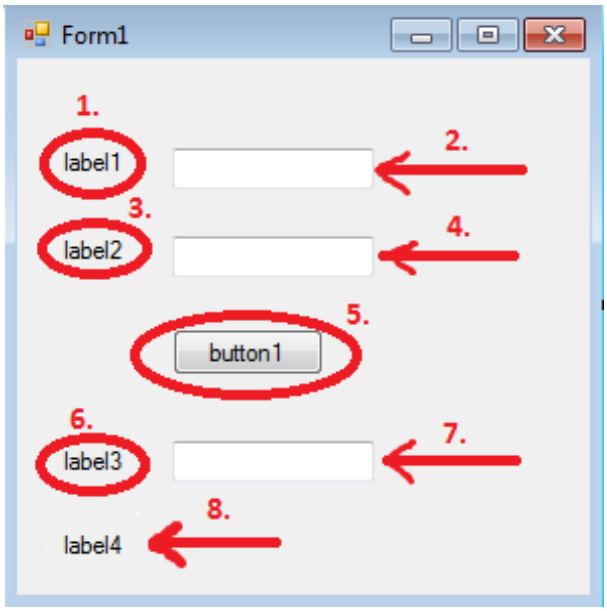
label1.text = "Vrednost promenljive x je " + x + "."; - на лабели се исписује: **Вредност променљиве x је 5**. (ако је x=5)

Следећи пример илуструје унос и приказивање података у TextBox-у i Label-у.

Пример 4. Написати програм који притиском на командно дугме SABERI узима два цела броја које је корисник унео у TextBox-ове и израчунава њихов збир и исписује резултат у TextBox и на Label-у.

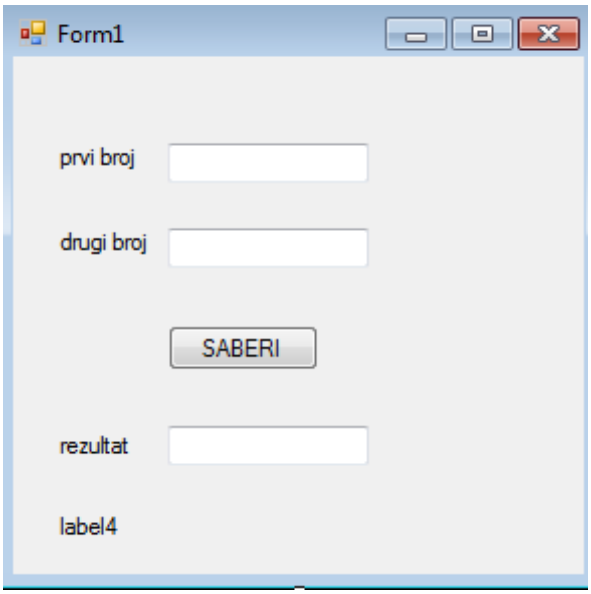
Решење:

Правимо форму облика:



1. label1 - ово ћемо преименовати у prvi broj
2. textBox1 - У ово поље корисник уноси цео број
3. label2 - ово ћемо преименовати у drugi broj
4. textBox2 - У ово поље корисник уноси цео број
5. button1 - преименујемо дугме у SABERI
6. label3 - ово ћемо преименовати у rezultat
7. textBox3 - У ово поље ћемо исписати резултат рада програма
8. label4 - овде ћемо такође исписати резултат рада програма

Слика 5.13. Почетни изглед форме

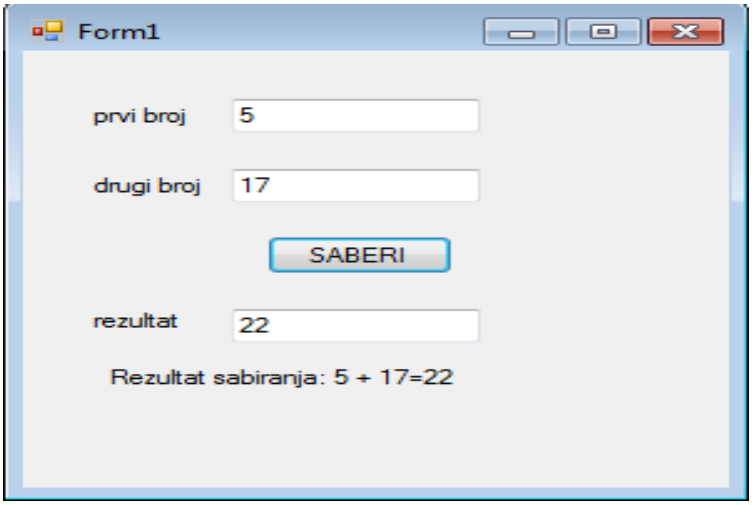


Слика 5.14. Изглед форме када смо преименовали компоненте

Када наша форма има изглед као на *слици 5.14* идемо двоструки клик на дугме ***SABERI*** и када нам се отвори нови прозор у телу функције ***button1_Click***куцамо код који треба да се извршава када кликнемо на то дугме.

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

Када покренемо апликацију из радног окружења помоћу тастера F5 отвара нам 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 5.15.



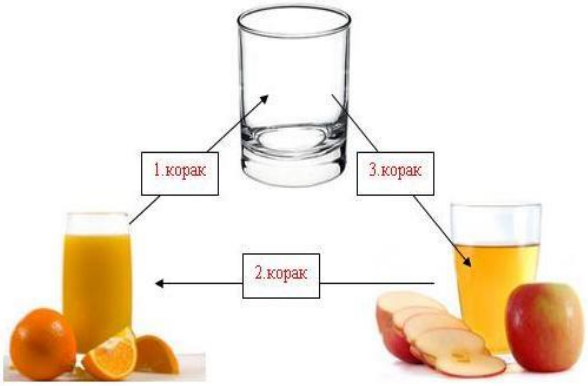
Слика 5.15. Приказ резултата рада програма

Алгоритам размене вредности две променљиве

Посматрајмо следећи пример из живота: имамо две чаше, у првој се налази сок од поморанџе, а у другој од јабуке и потребно је заменити садржај чаша, односно да сада у првој чаши буде сок од јабуке, а у другој од поморанџе (*Слика 5.16*).

Слика 5.16.

На који начин бисмо то могли урадити? Очигледно нам је потребна једна празна чаша која ће представљати помоћну чашу. Најпре ћемо сок из прве чаше пресути у празну чашу (1), а затим сок из друге чаше пресути у прву (2). Након тога, у првој чаши ћемо имати сок од јабуке и потребно је још само пресути сок из помоћне чаше у другу чашу (3) и имаћемо у другој чаши сок од поморанџе (*Слика 5.17*).



Слика 5.17. Размена садржаја две чаше

Након овог примера са пресипањем сокова, да видимо на који начин би се могла извршити размена вредности две променљиве.

Разменом вредности две променљиве, на пример *a* и *b*, вредност променљиве *a* постаје вредност променљиве *b* и вредност променљиве *b* постаје вредност променљиве *a*.

Потребно је одредити алгоритам који врши размену вредности две променљиве.

Пример 1. Шта ће се десити ако размену вредности две променљиве a и b урадимо на следећи начин:

```
//Deklaracija i inicijalizacija promenljivih a i b
int a = 1;
int b = 2;
//Promenljivoj a dodeljujemo vrednost promenljive b
a = b;
//Promenljivoj b dodeljujemo vrednost promenljive a
b = a;
```

Да ли ћемо на овај начин заиста заменити њихове вредности?

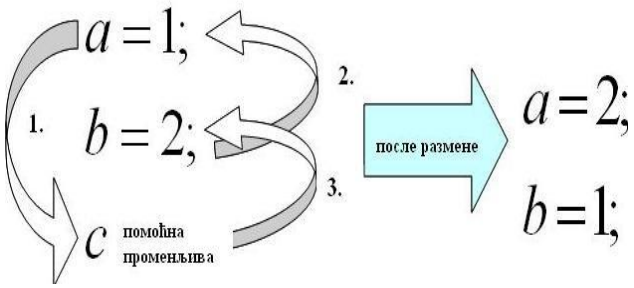
Решење:

Из прве наредбе a = b; следи да је нова вредност променљиве a једнака 2, а из друге наредбе b = a;добијамо да и b има вредност 2.

Дакле, a = 2 и b = 2, па на овај начин није извршена размена вредности две променљиве.

Слика 5.18. Неправилна размене вредности две променљиве

Из претходног можемо закључити да за размену вредности две променљиве није довољно само заменити њихове вредности, већ нам је потребна нека помоћна променљива *c*. Дакле, алгоритам за размену вредности две променљиве требало би да изгледа овако:



Слика 5.19. Размена вредности две променљиве

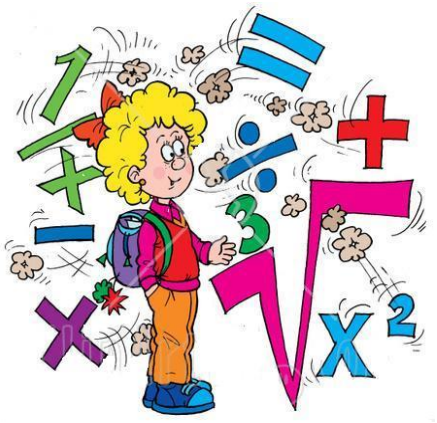
- **уводи се помоћна променљива *c***
- ***c* = *a*;** - вредност променљиве *a* смо сачували у помоћној променљивој *c*
- ***a* = *b*;** - стару вредност променљиве *a* заменимо вредношћу променљиве *b*
- ***b* = *c*;** - променљивој *b* доделимо вредност помоћне променљиве *c* у којој смо сачували стару вредност променљиве *a*

Пример 2. Део кода којим је представљен алгоритам размене вредности две целобројне променљиве:

```
//Deklaracija promenljivih a i b
int a;
int b;
//Deklaracija pomocne promenljive
int c;
//Pomocnoj promenljivoj dodeljujemo vrednost promenljive a
c = a;
//Promenljivoj a dodeljujemo vrednost promenljive b
a = b;
//Promenljivoj b dodeljujemo vrednost promenljive c
b = c;
```

Програмирање израчунавања по једноставним математичким формулама

Сви знамо да израчунамо збир, разлику, производ, количник два или више бројева у математици, знамо да одредимо обим и површину квадрата, правоугаоника, троугла, круга и других геометријских фигура, затим растојање између две тачке које су задате својим координатама и све то коришћењем познатих математичких формула.



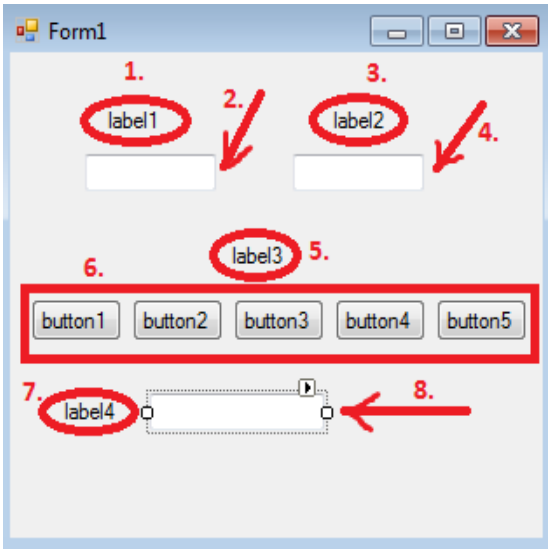
На који начин је могуће у C# програмирати по једноставним математичким формулама, видећемо на следећим примерима. За почетак ћемо узети једноставан пример са основним аритметичким операцијама.

Пример 1. Написати програм који за два цела броја која корисник уноси преко TextBox-а примењује основне аритметичке операције (+, -, *, /, %), а резултат да исписује у TextBox.

Напомена: Уколико желимо да израчунамо количник унетих бројева, као други број не смемо унети 0, да не би дошло до дељења нулом. Касније, када будемо научили ***наредбу гранања if***, моћи ћемо да проверимо да ли је унети број једнак 0 и ако јесте, исписаће се одговарајућа порука. За сада ћемо се задржати на томе да делилац (други број) буде различит од нуле, у случају дељења.

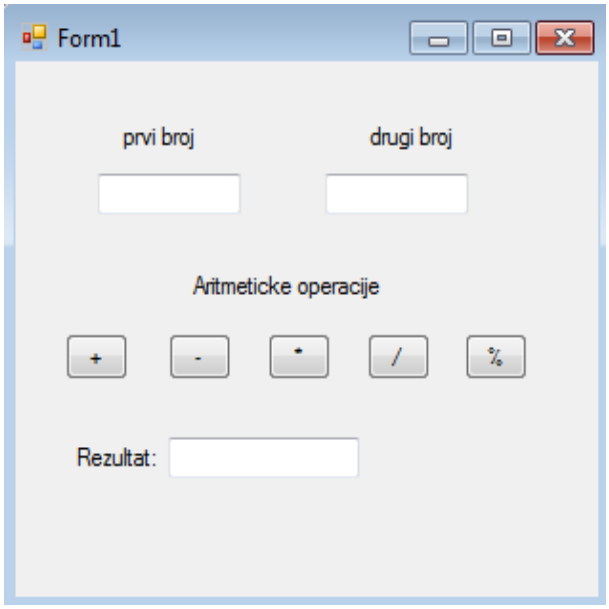
Решење:

Правимо форму облика:



Слика 5.20. Почетни изглед форме

1. label1 - у овој лабели ћемо написати **prvi broj**
2. textBox1 - У ово поље корисник уноси цео број
3. label2 - у овој лабели ћемо написати **drugi broj**
4. textBox2 - у ово поље корисник уноси цео број
5. label3 - у овој лабели ћемо написати **Aritmeticke operacije**
6. button1 - ово ћемо преименовати у **+**, button2 - ово ћемо преименовати у **-**, button3 - ово ћемо преименовати у *****, button4 - ово ћемо преименовати у **/**, button5 - ово ћемо преименовати у **%**
7. label4 - у овој лабели ћемо написати **Rezultat:**
8. textBox3 - з ово поље ћемо исписати резултат рада програма



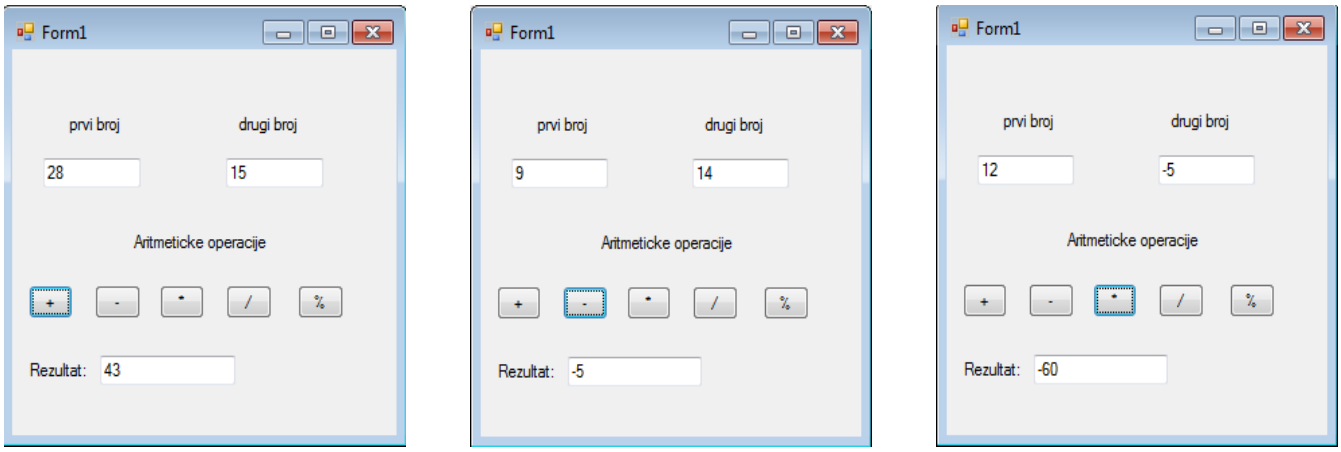
Слика 5.21. Изглед форме када смо преименовали компоненте

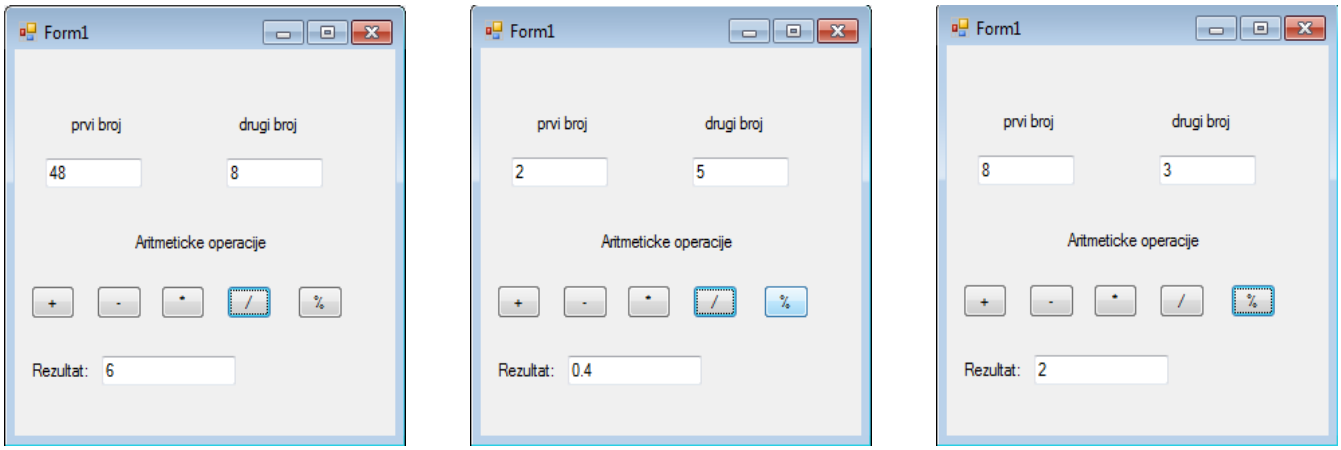
Када наша форма има изглед као на *слици 5.21* идемо двоструки клик на дугме **+** и када нам се отвори нови прозор у телу функције ***button1_Click***куцамо код који треба да се извршава када кликнемо на то дугме. Аналогно откуцамо и кодове за преосталу дугмад (**-, *, /, %**).

```
//Deklarisemo promenljive kao globalne jer cemo da ih koristimo
//u svakoj od narednih funkcija
int x, y;
//U telu ove funkcije kucamo kod koji se izvrsava kada
//kliknemo na dugme +
private void button1 Click(object sender, EventArgs e)
{
    //Promenljiva u koju cemo smestiti zbir unetih brojeva
    int zbir;
    //Uzimamo celobrojnu vrednost iz prvog textBox-a
    x = int.Parse(textBox1.Text);
    //Uzimamo celobrojnu vrednost iz prvog textBox-a
    y = int.Parse(textBox2.Text);
    //Racunamo zbir
    zbir = x + y;
    //Ispisujemo rezultat (zbir) u textBox
    textBox3.Text = zbir.ToString();
}
//U telu ove funkcije kucamo kod koji se izvrsava kada
//kliknemo na dugme -, analogno prethodnom, samo umesto zbira
//imamo razliku unetih brojeva
private void button2_Click(object sender, EventArgs e)
{
    int razlika;
    x = int.Parse(textBox1.Text);
    y = int.Parse(textBox2.Text);
    razlika = x - y;
    textBox3.Text = razlika.ToString();
}
//U telu ove funkcije kucamo kod koji se izvrsava kada
//kliknemo na dugme *, analogno prethodnim funkcijama,
//samo sto je u pitanju mnozenje
private void button3_Click(object sender, EventArgs e)
{
    int proizvod;
    x = int.Parse(textBox1.Text);
    y = int.Parse(textBox2.Text);
    proizvod = x * y;
    textBox3.Text = proizvod.ToString();
}
//U telu ove funkcije kucamo kod koji se izvrsava kada
//kliknemo na dugme /
private void button4_Click(object sender, EventArgs e)
{
```

```
    //Deklarisemo promenljivu kolicnik kao double
    //jer kolicnik dva broja ne mora biti ceo broj
    double kolicnik;
    x = int.Parse(textBox1.Text);
    y = int.Parse(textBox2.Text);
    //Stavili smo double da se ne bi primenjivalo
    //samo celobrojno deljenje, npr. 2/5 je 0.4, a ne 0
    kolicnik = (double)x / y;
    textBox3.Text = kolicnik.ToString();
}
//U telu ove funkcije kucamo kod koji se izvrsava kada
//kliknemo na dugme %
private void button5_Click(object sender, EventArgs e)
{
    int ostatak;
    x = int.Parse(textBox1.Text);
    y = int.Parse(textBox2.Text);
    //Odredjujemo ostatak pri deljenju prvog operanda drugim,
    //i to je moguce uraditi samo kada su oba celobrojnog tipa
    ostatak = x % y;
    textBox3.Text = ostatak.ToString();
}
```

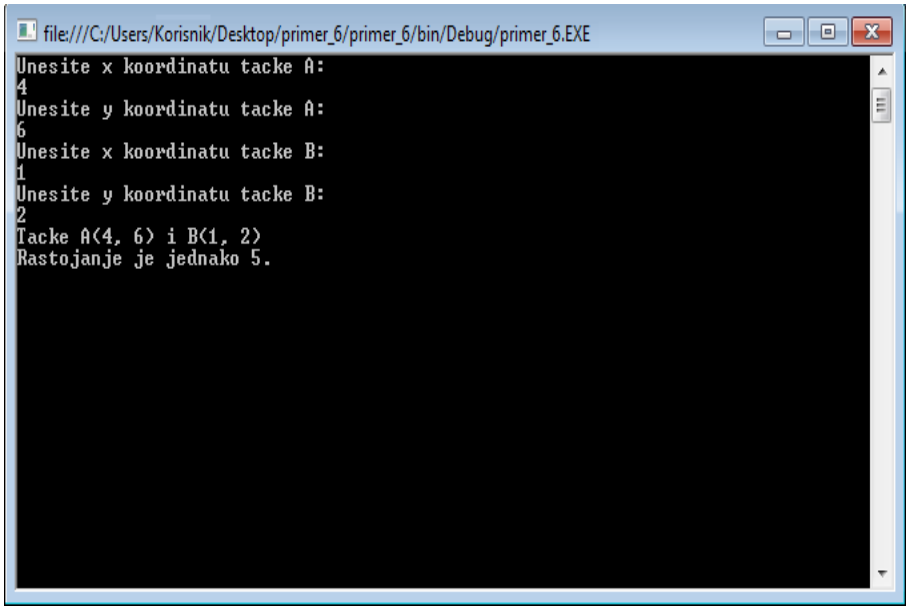
Када покренемо апликацију из радног окружења помоћу тастера F5 отвара нам се форма и потребно је унети бројеве у TextBox и затим притиснути жељено дугме. Након тога ће се исписати резултат у TextBox, као на наредној слици, за произвољне унете бројеве и изабрано дугме (Слика 5.22)





Слика 5.22. Резултат рада програма за унете вредности и изабране операције

Пример 2. Направимо конзолну апликацију која ће да рачуна растојање између две тачке и резултат да испише у конзоли. Координате обе тачке корисник уноси преко конзоле, као на слици 5.18.



Слика 5.23. Изглед конзоле након извршавања програма

Направимо конзолну апликацију и када се отвори прозор са већ датим костуром апликације и главном функцијом Main, у телу ове функције уписаћемо следећи код који треба да се изврши.

```
static void Main(string[] args)
{
    //Deklaracija koordinata tacaka A i B
    int x1, y1, x2, y2;
    double rastojanje;
    //U ove promenljive tipa string smestamo unos iz konzole
    string stringAx, stringAy, stringBx, stringBy;
    //Ispisuje u konzoli da korisnik unese x koordinatu tacke A
    Console.WriteLine("Unesite x koordinatu tacke A:");
    //ReadLine metodom dobijamo vrednost koju je korisnik uneo
    stringAx = Console.ReadLine();
    //Metodom int.Parse iz stringa izdvajamo celobrojnu vrednost
    x1 = int.Parse(stringAx);
```

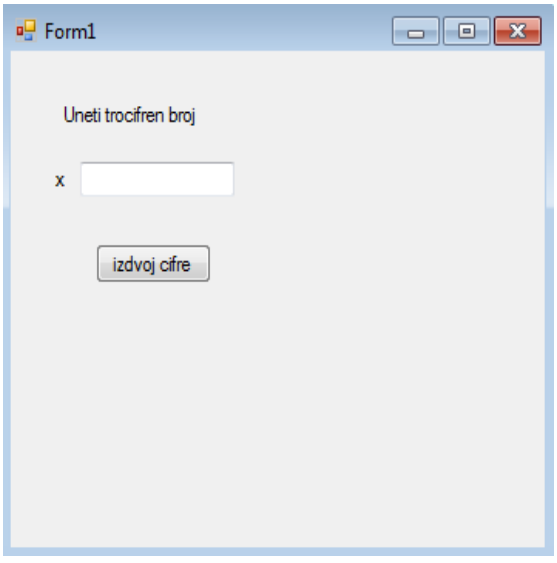
```
    //Analogno prethodnom, uradimo isto za preostale tri koordinate, y1, x2 i y2.
    Console.WriteLine("Unesite y koordinatu tacke A:");
    stringAy = Console.ReadLine();
    y1 = int.Parse(stringAy);
    Console.WriteLine("Unesite x koordinatu tacke B:");
    stringBx = Console.ReadLine();
    x2 = int.Parse(stringBx);
    Console.WriteLine("Unesite y koordinatu tacke B:");
    stringBy = Console.ReadLine();
    y2 = int.Parse(stringBy);
    //Odredjujemo rastojanje izmedju tacaka po matematickoj formuli
    rastojanje = Math.Sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    //Ispisujemo tacke sa koordinatama u konzolu
    Console.WriteLine("Tacke A(" + x1 + ", " + y1 + ") i B(" + x2 + ", " + y2 + ")");
    //Ispisujemo rastojanje izmedju tacaka u konzolu
    Console.WriteLine("Rastojanje je jednako "  + rastojanje +".");
    Console.ReadKey();
}
```

Када покренемо апликацију помоћу тастера **F5** отвара нам се прозор командне линије у коме пише **Unesite x koordinatu tacke A**:. Унесемо координату и притиснимо **Enter** на тастатури, након тога се појављује **Unesite y koordinatu tacke A:** и унесемо другу координату тачке А. Када притиснемо поново **Enter** писаће **Unesite x koordinatu tacke B**:, унесемо је и поново **Enter**, појавиће се **Unesite y koordinatu tacke B:**. Унесемо и ту координату и када следећи пут притиснемо **Enter**, исписаће нам се тачке А и В са координатама и растојање између њих.

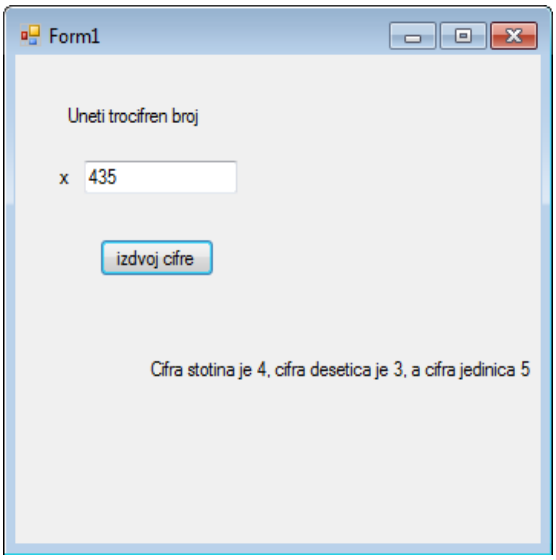
Ако је све правилно написано, конзола треба да изгледа као на *слици 5.23* за унете тачке А(4,6) и В(1,2).

Додатни примери

Пример 1. Написати програм који омогућава кориснику да унесе троцифрен број, и издваја цифре тог броја.



Слика 5.24. Изглед форме



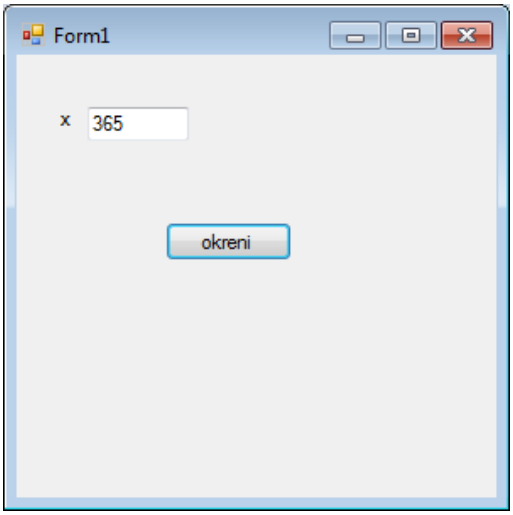
Слика 5.24б. Изглед апликације након клика на дугме

Решење:

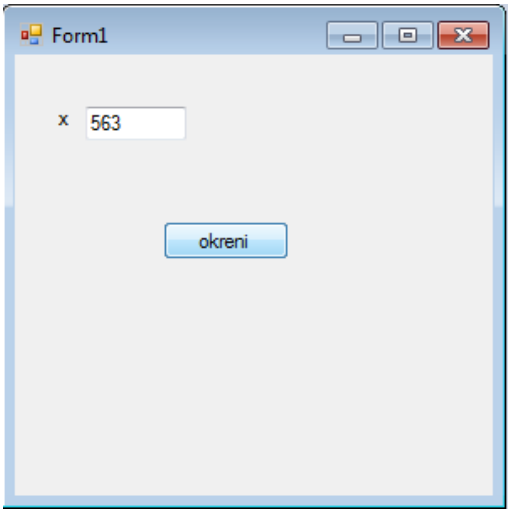
У овом примеру треба да издвојимо цифре троцифреног броја. Кренимо од последње цифре. Њу можемо да добијемо ако посматрамо остатак при дељењу датог броја бројем 10. Како можемо издвојити другу цифру (цифру десетица)? Најједноставније нам је да проблем сведемо на претходни. Ако извршимо целобројно дељење датог броја бројем 10, добићемо двоцифрен број чије су цифре исте као прве две цифре почетног броја. Сада поновимо поступак да бисмо добили прву цифру (цифру стотина).

```
private void button1_Click(object sender, EventArgs e)
{
    //Deklarisemo promenljivu i dodeljujemo joj vrednost
    //koju je korisnik uneo u prvi textBox, prevedenu u tip int
    int a = Convert.ToInt32(textBox1.Text);
    //Deklarisemo promenljive
    int cifraJedinica, cifraDesetica, cifraStotina;
    //Nalazimo ostatak pri deljenju broja a sa 10, tj. izdvajamo poslednju cifru
    cifraJedinica = a % 10;
    //Vrsimo celobrojno deljenje broja a sa 10, cime dobijamo dvocifreni broj,
    //odnosno "eliminisemo" poslednju cifru broja a
    a = a / 10;
    //Nalazimo ostatak pri deljenju "novog" broja a sa 10, cime zapravo izdvajamo
    //cifru desetica pocetnog broja a
    cifraDesetica = a % 10;
    //Vrsimo celobrojno deljenje broja a sa 10, cime dobijamo jednocifreni broj
    a = a / 10;
    //Cifra koja nam je ostala je zapravo cifra stotina iz pocetnog broja a
    cifraStotina = a;
    //Ispisujemo rezultat u labeli
    label3.Text = "Cifra stotina je " + cifraStotina + ", cifra desetica je "
    + cifraDesetica + ", a cifra jedinica " + cifraJedinica;
}
```

Пример 2. Направити апликацију којом цифре троцифреног броја исписујемо у обрнутом редоследу. На пример, ако корисник унесе број 247, након клика на дугме, у textBox-у треба да пише 742.



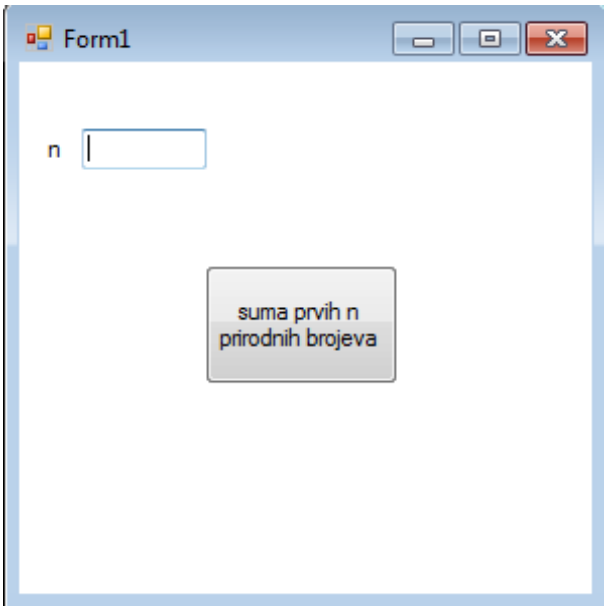
Слика 5.25. Изглед апликације након покретања програма



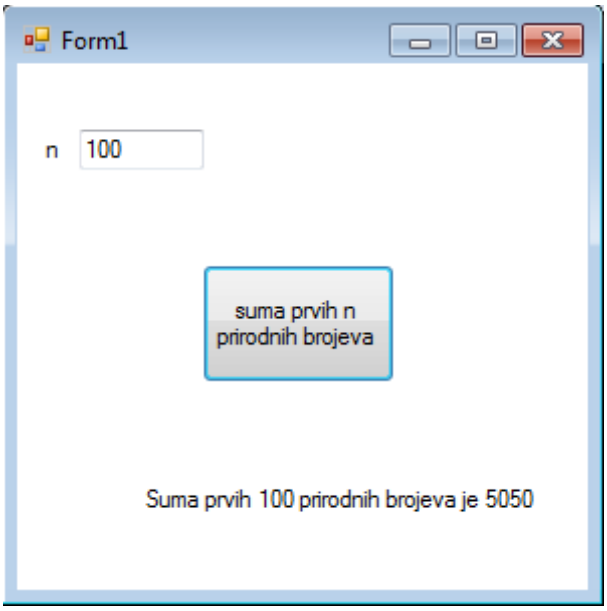
Слика 5.25б. Изглед апликације након клика на дугме

```
private void button1_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt32(textBox1.Text);
    int a, b, c;
    //Izdvajamo cifru jedinica
    a = n % 10;
    //"Odbacujemo" poslednju cifru
    n = n / 10;
    //Izdvajamo cifru desetica
    b = n % 10;
    //"Odbacujemo" poslednju cifru novog broja n
    n = n / 10;
    //Cifra stotina nam je jedina preostala cifra
    c = n;
    //Formiramo novi broj n sa rasporedom cifara koji je trazen u zadatku
    n = a * 100 + b * 10 + c;
    //Ispisujemo rezultat
    textBox1.Text = n.toString();
}
```

Пример 3. Написати програм који омогућава кориснику да унесе природан број n, и за унето n рачуна суму првих n природних бројева.



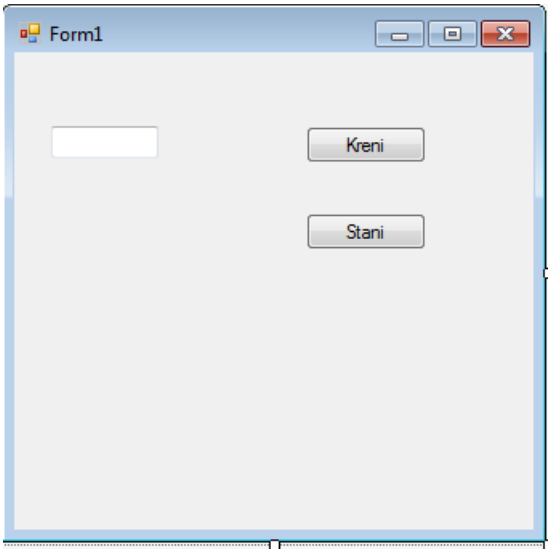
Слика 5.26. Изглед форме



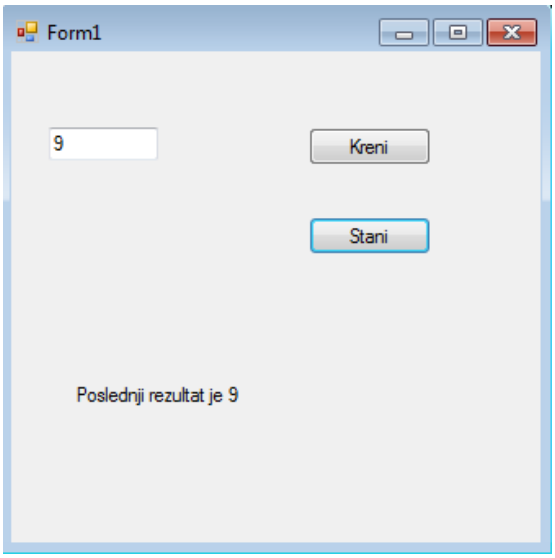
Слика 5.26б. Изглед апликације након клика на дугме

```
private void button1 Click(object sender, EventArgs e)
{
    int n = Convert.ToInt32(textBox1.Text);
    int s;
    //Koristimo matematicku formulu za racunanje sume prvih n prirodnih brojeva
    s = (n * (n + 1)) / 2;
    //Ispisujemo rezultat
    label3.Text = "Suma prvih " + n + " prirodnih brojeva je " + s;
}
```

Пример 4. Направити апликацију Штоперица која, након клика на дугме Kreni започиње одбројавање, након клика на дугме Stani зауставља одбројавање и враћа штоперицу на почетак и у лабели исписује последњи резултат.



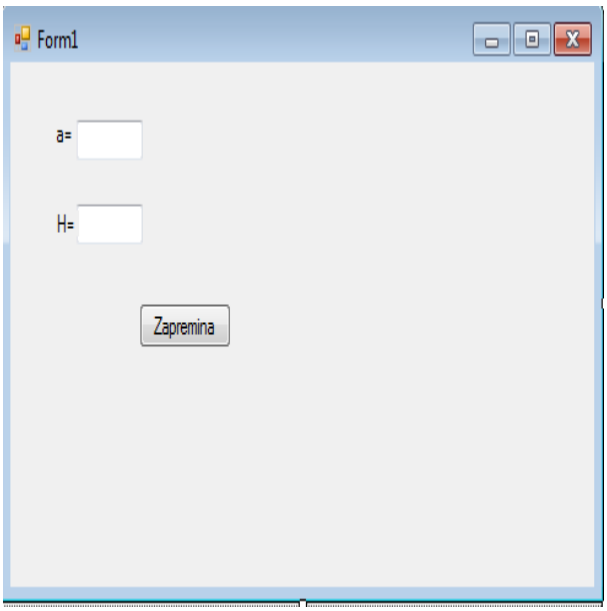
Слика 5.27. Изглед форме



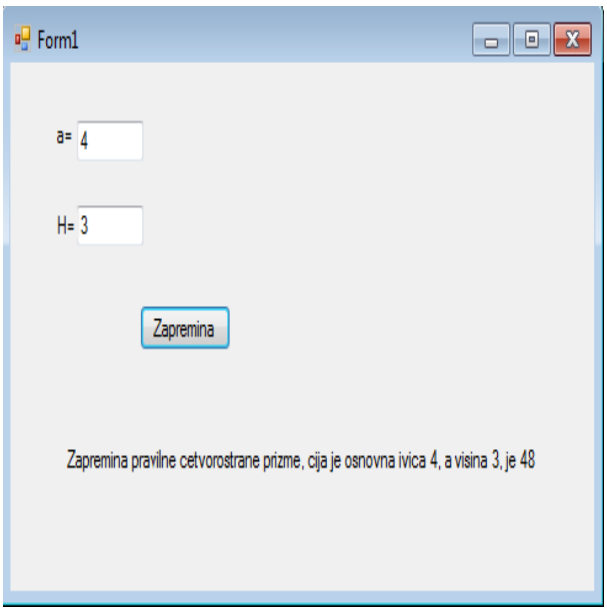
Слика 5.27б. Изглед апликације након клика на дугме Stani

```
int  broj = 0;
private void button1_Click(object sender, EventArgs e)
{
    //Ukljucujemo tajmer, tj. pocinjemo brojanje
    timer1.Start();
}
private void button2_Click(object sender, EventArgs e)
{
    //Iskljucujemo tajmer, tj. zaustavljamo brojanje
    timer1.Stop();
    //Ispisujemo poslednji rezultat
    label1.Text = Convert.ToString(broj);
    //Vracamo stopericu da ponovo broji od nule
    broj = 0;
}
private void timer1_Tick(object sender, EventArgs e)
{
    //Brojimo, odnosno uvecavamo promenjivu broj za 1 i svaki put
    //rezultat ispisujemo u textBox-u
    broj += 1;
    textBox1.Text = Convert.ToString(broj);
}
```

Пример 5. Написати програм који рачуна запремину правилне четворостране призме, ако корисник унесе дужину основне ивице и висину призме.



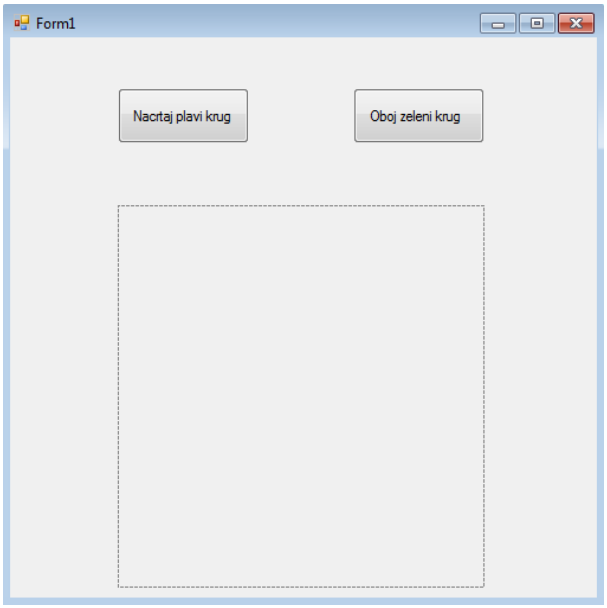
Слика 5.28. Изглед форме



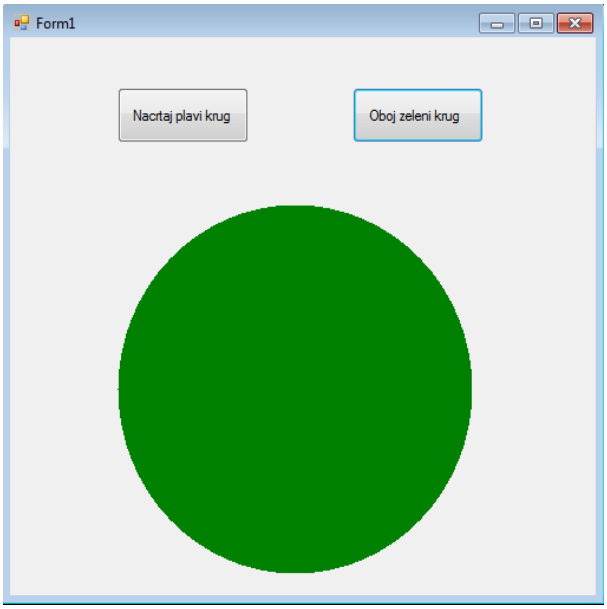
Слика 5.28б. Изглед апликације након клика на дугме

```
private void button1_Click(object sender, EventArgs e)
{
    double a, H;
    double V, B;
    a = Convert.ToDouble(textBox1.Text);
    H = Convert.ToDouble(textBox2.Text);
    //Racunamo povrsinu baze
    B = a * a;
    //Racunamo zapreminu
    V = B * H;
    //Ispisujemo rezultat
    label3.Text = "Zapremina pravilne cetvorostrane prizme, cija je osnovna ivica "
    + a + ", a visina " + H + ", je " + V ;
}
```

Пример 6. Направити апликацију којом се у PictureBox-у црта плава кружна линија или црта и боји зелени круг, у зависности од тога на које је од ова два дугмета кликнуо корисник. Претходно поставити ширину и висину PictureBox-а на 310.



Слика 5.29. Изглед форме



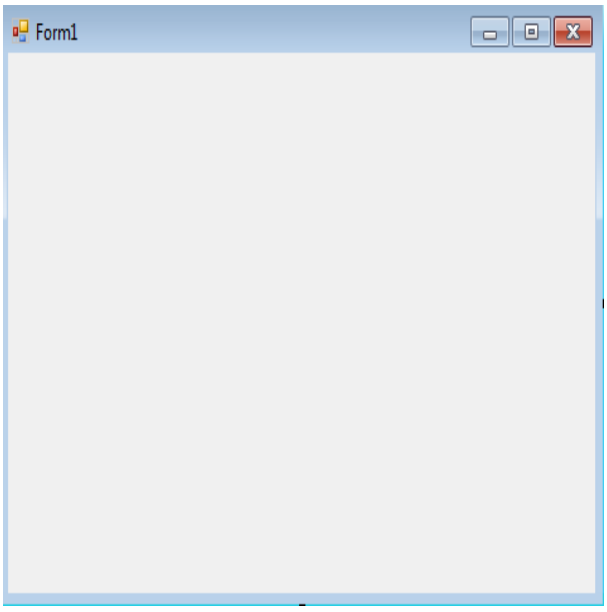
Слика 5.29б. Изглед апликације након клика на дугме

У овом примеру желимо да цртамо по pictureBox-у, а за то нам је потребан објекат класе Graphics.Како у c#-у немамо команду којом цртамо круг, приметићемо да је круг заправо једна специфична елипса. Како овде елипсе цртамо тако што дајемо податке о правоугаонику описаном око ње, уочићемо да ће нам за круг бити потребан квадрат који је око њега описан.

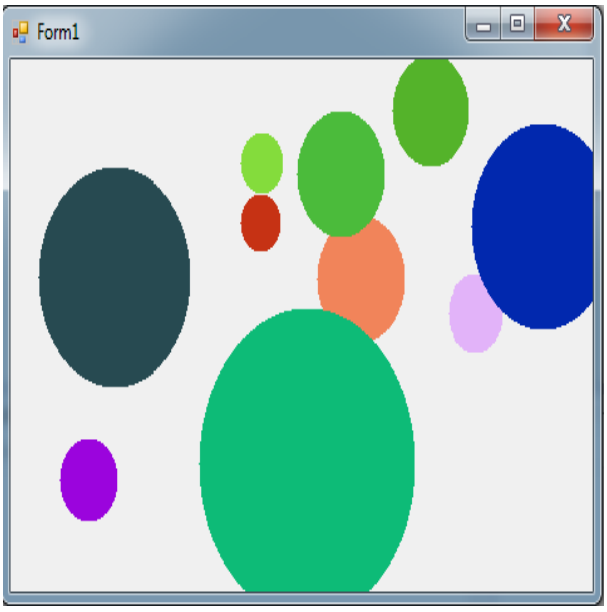
```
private void button1_Click(object sender, EventArgs e)
{
    //Brisemo sadrzaj pictureBox-a
    pictureBox1.Refresh();
    //Kreiramo objekat klase Graphics
    Graphics g = pictureBox1.CreateGraphics();
    //Kreiramo objekat klase Pen, njime cemo crtati
```

```
    //Crtacemo plavom bojom, a debljina linije bice 1
    Pen olovka = new Pen(Color.Blue, 1);
    //Crtamo elipsu, koordinate gornjeg levog temena
    //pravougaonika opisanog oko te elipse su (0,0),
    //a sirina i visina tog pravougaonika su po 300
    g.DrawEllipse(olovka, 0, 0, 300,300);
    //Oslobadjamo resurse koje su ovi objekti koristili
    olovka.Dispose();
    g.Dispose();
}
private void button2_Click(object sender, EventArgs e)
{
    //Brisemo sadrzaj pictureBox-a
    pictureBox1.Refresh();
    //Kreiramo objekat klase Graphics
    Graphics g = pictureBox1.CreateGraphics();
    //Kreiramo objekat klase SolidBrush, njime cemo crtati, odnosno bojiti krug
    //Bojicemo zelenom bojom
    SolidBrush cetka = new SolidBrush(Color.Green);
    //Crtamo elipsu, koordinate gornjeg levog temena
    //pravougaonika opisanog oko te elipse su (0,0),
    //a sirina i visina tog pravougaonika su po 300
    g.FillEllipse(cetka, 0, 0, 300,300);
    //Oslobadjamo resurse koje su ovi objekti koristili
    olovka.Dispose();
    g.Dispose();
}
```

Пример 7. Направити апликацију којом се, када корисник кликне на форму, исцртавају кругови произвољног полупречника и боје, са центром у тачки на коју је корисник кликнуо мишем.



Слика 5.30. Изглед форме



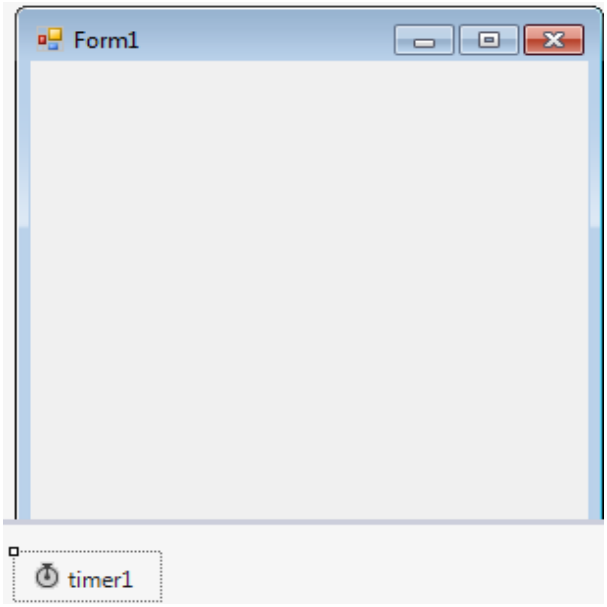
Слика 5.30б. Изглед апликације након клика на дугме

Овим примером ћемо се подсетити догађаја MouseClick.

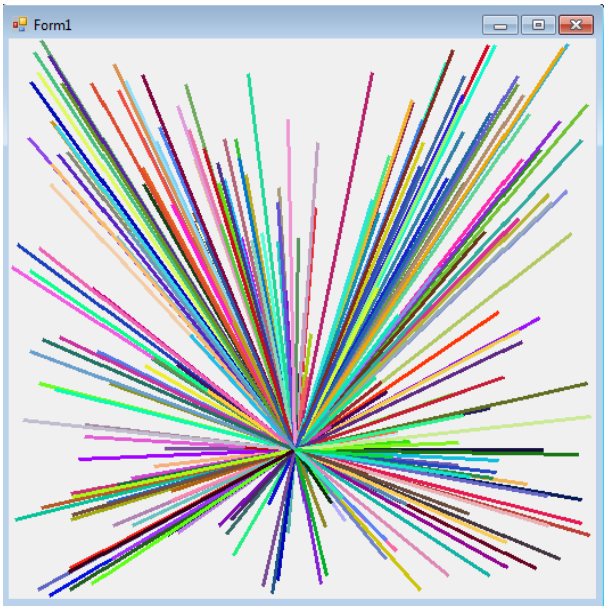
```
//Kreiramo objekat klase Random
Random R = new Random();
int x1, y1, r;

private void Form1_MouseClick(object sender, MouseEventArgs e)
{
    //Kreiramo objekat klase Graphics
    Graphics g = CreateGraphics();
    //Biramo proizvoljnu boju
    Color boja = Color.FromArgb(R.Next(256), R.Next(256), R.Next(256));
    //Kreiramo objekat klase SolidBrush, njime cemo crtati, odnosno bojiti krugove
    SolidBrush cetka = new SolidBrush(boja);
    //Pamtimo koordinate tacke na koju je korisnik kliknuo misem
    x1 = e.X;
    y1 = e.Y;
    //Biramo poluprecnik kruga
    r = R.Next(1, 150);
    //Crtamo krug
    g.FillEllipse(cetka, x1 - r, y1 - r, 2 * r, 2 * r);
    //Oslobadjamo resurse
    g.Dispose();
    cetka.Dispose();
}
```

Пример 8. Направити апликацију којом се, када корисник кликне на форму, исцртавају дужи са заједничким почетком у тачки на коју је корисник кликнуо мишем, као на слици.



Слика 5.31. Изглед форме



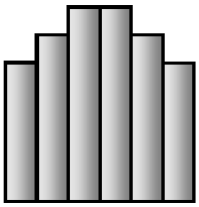
Слика 5.31б. Изглед апликације након клика на дугме

```
//Kreiramo objekat klase Random
Random R = new Random();
int x1, y1, x2, y2;

private void Form1_MouseClick(object sender, MouseEventArgs e)
{
    //Pamtimo koordinate tacke na koju je korisnik kliknuo misem
    x1 = e.X;
    y1 = e.Y;
    //Brisemo sve sa forme
    Refresh();
    //Ukljucujemo tajmer
    timer1.Start();
}
private void timer1_Tick(object sender, EventArgs e)
{
    Graphics g = CreateGraphics();
    Color boja = Color.FromArgb(R.Next(256), R.Next(256), R.Next(256));
    Pen olovka = new Pen(boja, 3);
    //Odredjujemo koordinate drugog kraja duzi
    x2 = R.Next(0, ClientRectangle.Width);
    y2 = R.Next(0, ClientRectangle.Height);
    //Crtamo duz
    g.DrawLine(olovka, x1, y1, x2, y2);
    //Oslobadjamo resurse
    g.Dispose();
    olovka.Dispose();
}
```

Наредна три задатка су задаци са основношколских такмичења из информатике из претходних година. Предлажемо ученицима да их ураде сами за домаћи.

Пример 9. За припрему декора за школску представу, деца су подељена у три групе и свака група је добила задатак да направи по један део декора. Прва група је добила задатак да направи декор који изгледа као замак у даљини. За основу су узели 6 дасака поређаних једну до друге, које ће обојити на одговарајући начин. Прва даска је дужине K, друга за 20 центиметара дужа, трећа за 20 центиметара дужа од друге, четврта исте дужине као трећа, пета исте дужине као друга и шеста исте дужине као прва. Написати програм ZAMAK у коме се за унету дужину прве даске K израчунава укупна дужина дасака.

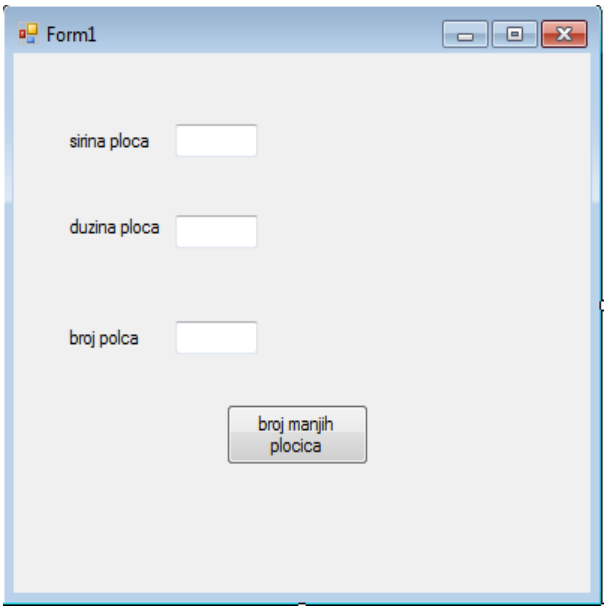


Слика 5.32. Изглед форме

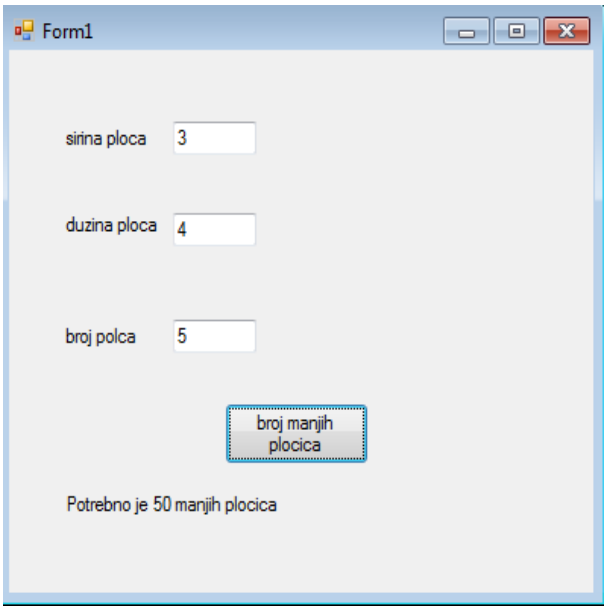
Слика 5.32б. Изглед апликације након клика на дугме

```
private void btDuzina_Click(object sender, EventArgs e)
{
    //Ucitavamo duzinu prve daske
    int k = Convert.ToInt32(textBox1.Text);
    int duzina;
    //Racunamo ukupnu duzinu dasaka
    duzina = 2 * k + 2 * (k + 20) + 2 * (k + 40);
    //Ispisujemo rezultat
    label2.Text = "Ukupna duzina dasaka je " + duzina;
}
```

Пример 10. Кнез Љубинко је решио да у свом родном месту направи мост преко реке. Сакупио је најбоље мајсторе који су почели изградњу каменог моста. У каменоломима су секли огромне камене плоче једнаких димензија које су постављали на стубове и то им је представљало основу моста. Ширина плоче је увек одређивала ширину моста. Да се ивице ових плоча не би временом оштетиле, након што поређају плоче у низ они су око ових великих плоча постављали мање плоче квадратне основе димензије 1 метар. Написати програм MOST у коме се уносе ширина плоче S и дужина плоче D, задате у метрима и број плоча N који је потребан да се направи основа једног моста, а затим се израчунава број мањих квадратних плоча које су потребне да се поставе по ивицама моста. Вредности променљивих S, D и N су позитивни цели бројеви, 0 < S, D, N < 100.



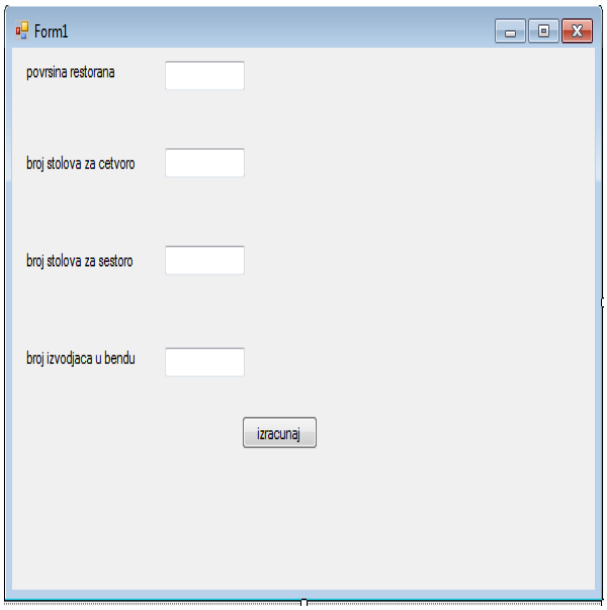
Слика 5.33. Изглед форме



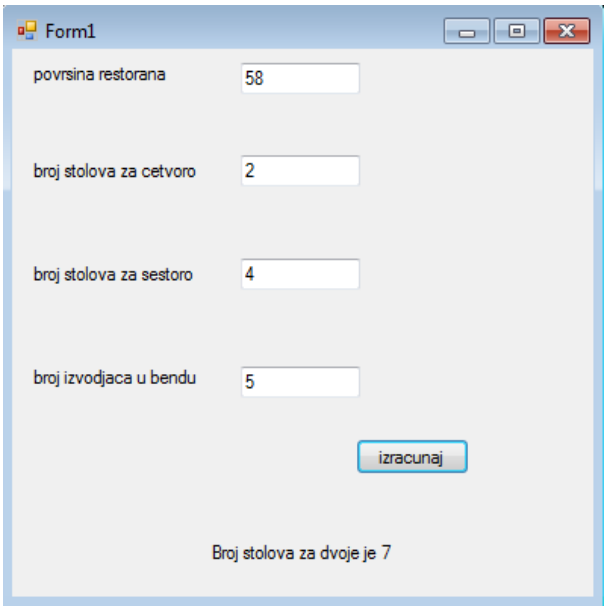
Слика 5.33б. Изглед апликације након клика на дугме

```
private void button1_Click(object sender, EventArgs e)
{
    //Ucitavamo promenljive iz textBox-ova
    int s = Convert.ToInt32(textBox1.Text);
    int d = Convert.ToInt32(textBox2.Text);
    int n = Convert.ToInt32(textBox3.Text);
    //Deklarisemo promenljivu
    int brPlocica;
    //Racunamo broj manjih plocica
    brPlocica = 2 * d * n + 2 * s + 4;
    //Ispisujemo rezultat
    label4.Text = "Potrebno je " + brPlocica + " manjih plocica";
}
```

Пример 11. Ресторан "36" је због добре хране и одличне забаве увек пун гостију, због чега се запослени у њему труде да направе што више места за седење. Ресторан располаже столовима за шест, четири и две особе, од којих прва врста заузима површину од 7 квадратних метара, друга 5 квадратних метара, а најмања 2 квадратна метара. Поред столова у ресторану се мора наћи место и за бенд који забавља госте и који сваке вечери наступа са различитим бројем извођача, при чему је по извођачу потребно обезбедити 1 квадрани метар. С обзиром на то да број извођача варира, столови се сваке вечери изнова распоређују. Зна се да столова за 4 и 6 особа има толико да се сигурно сви могу увек распоредити и они се први постављају. Простор који бенд заузима никада није већи од простора који остаје слободан након што се поставе столови за 4 и 6 особа. Након распоређивања столова за 4 и 6 особа и бенда, преостали простор се попуњава столовима за две особе. Написати програм STOLOVI који након уноса површине ресторана, броја столова за 4 и 6 особа и броја извођача у бенду одређује колико столова за две особе може да се постави.



Слика 5.34. Изглед форме



Слика 5.34б. Изглед апликације након клика на дугме

```
private void button1_Click(object sender, EventArgs e)
{
    //Deklarisemo promenljive
    int x;
    int r = Convert.ToInt32(textBox1.Text);
    int s1 = Convert.ToInt32(textBox2.Text);
    int s2 = Convert.ToInt32(textBox3.Text);
    int b = Convert.ToInt32(textBox4.Text);
    //Racunamo broj stolova za dvoje
    x = (r - (s1 * 5 + s2 * 7 + b)) / 2;
    //Ispisujemo rezultat
    label5.Text = "Broj stolova za dvoje je " + x;
}
```