

Osmosmjerka (C#)

Luka Jukić

```
mirror_mod = modifier_ob.  
set mirror object to mirror  
mirror_mod.mirror_object =  
operation = "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation = "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation = "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly one  
mirror object")
```

--- OPERATOR CLASSES ---

```
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"
```

```
context):  
context.active_object is not None
```

Unos podataka

► PRIMJER ULAZA:

5 6 (dimenzije osmosmjerke u formatu {razmak}{broj redaka}{razmak}{broj stupaca})

O L O V K A (osmosmjerka u formatu {slovo}{razmak}{slovo}{razmak}...Enter-kraj unosa retka)
N B O N J I B
O C I T I R
T G K L S O
E T P O L T

6 (unos broja riječi koje treba naći u osmosmjerci)
L O P T E (unos riječi u obliku {1.riječ}{Enter}{2.riječ}{Enter}...)
T O R B A
O L O V K A
K N J I G E
N O T E
S T O L

► PRIMJER IZLAZA (RIJEŠENJE):

► B I C I K L (slova koja ostanu nezaokružena)

Podaci

- ▶ Riječi koje se traže u osmosmjerci unesene su u polje stringova
- ▶ Osmosmjerka je unesena u 2D polje stringova
- ▶ Pojedina slova svake riječi koja se traži su tipa char
- ▶ Pojedino slovo, tj. element osmosmjerkke je string zbog dvoslova u hrvatskom jeziku, koji ne mogu biti spremljeni u jednom char-u (osim ako im se dodijeli poseban kod)
- ▶ Kreirana je i kopija osmosmjerkke, matrica identična onoj u kojoj je osmosmjerka u kojoj će se znakovi „zaokruženi” u osmosmjerci pretvarati u „0” te će se iz te matrice na kraju dobiti traženi izlaz, tj. svi znakovi koji su različiti od „0” biti će ispisani

Kada ne bi bilo kopije osmosmjerkke, jedna od ovih riječi ne bi bila pronađena jer bi se „K” pretvorilo u „0”

L	I	K
x	x	A
x	x	R

RAK
LIK

Ideja algoritma prolaska kroz osmosmjerku

Za i=1 do M činiti {

 Za j=1 do N činiti {

 Za z=1 do K činiti

 {

 Funkcija_smjer_gore (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 Funkcija_smjer_goreDesno (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 Funkcija_smjer_goreLijevo (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 Funkcija_smjer_desno (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 Funkcija_smjer_lijevo (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 Funkcija_smjer_doljeLijevo (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 Funkcija_smjer_doljeDesno (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 Funkcija_smjer_dolje (osmosmjerka, kopija_osmosmjerke, i, j, riječ[K], M, N)

 }

 }

}

M - broj redaka osmosmjerke

N - broj stupaca osmosmjerke

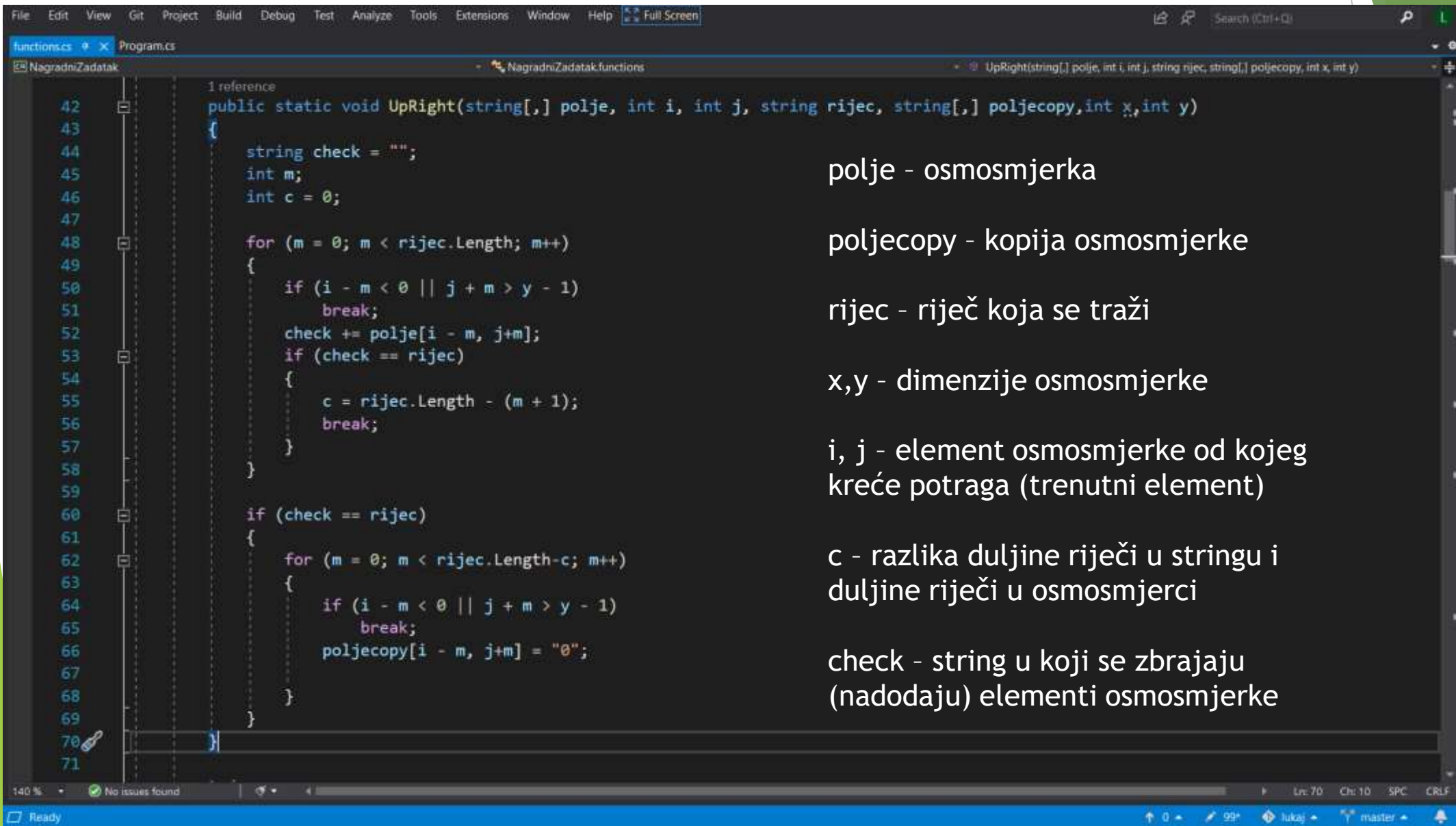
K - broj riječi koji se traži

Potrebno samo u nekim funkcijama(u onima koje mogu izaći iz jedne ili objiju dimenzija polja u smjeru dolje ili desno)

Optimizacija

- ▶ U pogledu da se na temelju početnog slova u osmosmjerci odbaci cijela riječ jer ne počinje s tim slovom, optimizacija je moguća ako bi se slovo riječi koja se traži (tip `char`) pretvorilo u tip `string` funkcijama u C#. Tako bi se moglo uspoređivati s ostalim stringovima u osmosmjerci. Kada bi naišli na dvoslov u osmosmjerci, umjesto prvog slova u riječi, pretvarali bi prva 2 slova riječi u `string` i uspoređivali radi li se o istom slovu.
- ▶ Primjer:
NJIHALJKA - u osmosmjerci početno slovo (`string`) ove riječi je „NJ”, dok je to ustvari samo slovo N ako pokušamo pristupiti 1. slovu riječi pomoću `NJIHALJKA[0]`, zato treba uzeti prva dva slova (NJ), pretvoriti ih u `string` i usporediti s elementom osmosmjerke (u slučaju da smo naišli na dvoslov u osmosmjerci).
- ▶ Optimizacija je moguća i u slučaju da se svaka riječ pojavljuje samo 1 u osmosmjerci, tako bismo mogli svaki put kada pronađemo riječ u svakoj sljedećoj iteraciji preskakati tu riječ jer je pronađena, npr. ako je riječ nađena kod obilaska smjerova oko mjesta `[i,j]`, tada bi ju mogli ignorirati kod prolaska smjerova oko mjesta `[i,j+1]` itd.

Primjer funkcije za smjer (C#)



```
1 reference
public static void UpRight(string[,] polje, int i, int j, string rijec, string[,] poljecopy, int x, int y)
{
    string check = "";
    int m;
    int c = 0;

    for (m = 0; m < rijec.Length; m++)
    {
        if (i - m < 0 || j + m > y - 1)
            break;
        check += polje[i - m, j+m];
        if (check == rijec)
        {
            c = rijec.Length - (m + 1);
            break;
        }
    }

    if (check == rijec)
    {
        for (m = 0; m < rijec.Length - c; m++)
        {
            if (i - m < 0 || j + m > y - 1)
                break;
            poljecopy[i - m, j+m] = "0";
        }
    }
}
```

polje - osmosmjerka

poljecopy - kopija osmosmjerke

rijec - riječ koja se traži

x,y - dimenzije osmosmjerke

i, j - element osmosmjerke od kojeg kreće potraga (trenutni element)

c - razlika duljine riječi u stringu i duljine riječi u osmosmjerki

check - string u koji se zbrajaju (nadodaju) elementi osmosmjerke

Primjer : LOPTA

K	R	C	D	F	L
M	L	O	P	T	A
S	O	G	G	NJ	Đ

Tražena riječ : LOPTA
Trenutna pozicija [2,2]
Funkcija: smjer_desno (j++)

Osmosmjerka (1. petlja)

Iteracija(broj slova riječi)	Tražena riječ	Check (osmosmjerka)	Pozicija
1.	„LOPTA”	„L”	[2,2]
2.	„LOPTA”	„LO”	[2,3]
3.	„LOPTA”	„LOP”	[2,4]
4.	„LOPTA”	„LOPT”	[2,5]
5.	„LOPTA”	„LOPTA”	[2,6]

Kopija osmosmjerke (2. petlja)

Iteracija(broj slova) - 0(c)	Pozicija	Slovo na poziciji	Slovo nakon prolaska
1.	[2,2]	„L”	„0”
2.	[2,3]	„O”	„0”
3.	[2,4]	„P”	„0”
4.	[2,5]	„T”	„0”
5.	[2,6]	„A”	„0”

Check==riječ

(c=0)

Primjer : KONJ

K	R	C	D	F	L
M	K	O	NJ	I	C
S	O	G	G	NJ	Đ

Tražena riječ : KONJ

Trenutna pozicija [2,2]

Funkcija: smjer_desno (j++)

Osmosmjerka (1. petlja)

Iteracija(broj slova riječi)	Tražena riječ	Check (osmosmjerka)	Pozicija
1.	„KONJ”	„K”	[2,2]
2.	„KONJ”	„KO”	[2,3]
3.	„KONJ”	„KONJ”	[2,4]
4.	„KONJ”	„KONJI”	[2,5]

Check==riječ

Check!=riječ

(c=4-3=1) - da ga nema, bio bi prekrižen i znak „I” na poziciji [2,5]

Kopija osmosmjerke (2. petlja)

Iteracija(broj slova) - 1(c)	Pozicija	Slovo na poziciji	Slovo nakon prolaska
1.	[2,2]	„K”	„0”
2.	[2,3]	„O”	„0”
3.	[2,4]	„NJ”	„0”

Ispis i složenost

- Ispisuju se svi elementi kopije osmosmjerke koji su različiti od „0”, tj. nisu zaokruženi, a ako takvi ne postoje, ispisuje se -

Flag= FALSE

Za svaki i=1 do M činiti{

 Za svaki j=1 do N činiti{

 Ako je poljekopija[i,j] različit od „0” onda {

 Ispiši poljekopija[i,j]

 Flag=TRUE

 }

 }

}

Ako nije Flag onda ispiši „-” i Završi algoritam.

- Složenost: $O(n*m*k*L)$:
n, m - dimenzije osmosmjerke, k - broj traženih riječi, L - (prosječan) broj slova riječi