# Nepobjedivi algoritam za igru kružić i križić ("iks-oks") u programskom jeziku C#

**Matijević, Barbara**

**Undergraduate thesis / Završni rad**

**2018**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:200:804614

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-27**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni preddiplomski studij računarstva**

# Nepobjedivi algoritam za igru kružić i križić ("iks-oks") u programskom jeziku C#

**Završni rad**

**Barbara Matijević**

**Osijek, 2018.**

# Sadržaj

1. UVOD ..... 1

1.1. Zadatak završnog rada ..... 1

2. PROGRAMSKA PODRŠKA ..... 2

2.1. Microsoft Visual Studio ..... 2

2.2. C# programski jezik ..... 3

3. IZRADA IKS-OKS IGRE ..... 6

3.1. Dijagram tijeka programa ..... 6

3.2. Form0 ..... 8

3.3. Form1 ..... 9

ZAKLJUČAK ..... 34

LITERATURA ..... 35

SAŽETAK ..... 36

ABSTRACT ..... 36

ŽIVOTOPIS ..... 37

PRILOZI ..... 38

# 1. UVOD

Igra iks-oks datira iz davne 1300. godine prije nove ere, iz vremena drevnog Egipta. Igra iks-oks predviđena je za igru na praznom polju 3x3, u većini slučajeva na papiru. Simboli korišteni prilikom igre su X i O. Igrači naizmjenično odabiru prazna polja i unutar njih crtaju svoj znak. Igrač nosi pobjedu kada ostvari tri uzastopna znaka u nekom retku, stupcu ili dijagonali. Također, igra može završiti neriješenim rezultatom ako niti jedan igrač ne ostvari niz od tri simbola.

## 1.1. Zadatak završnog rada

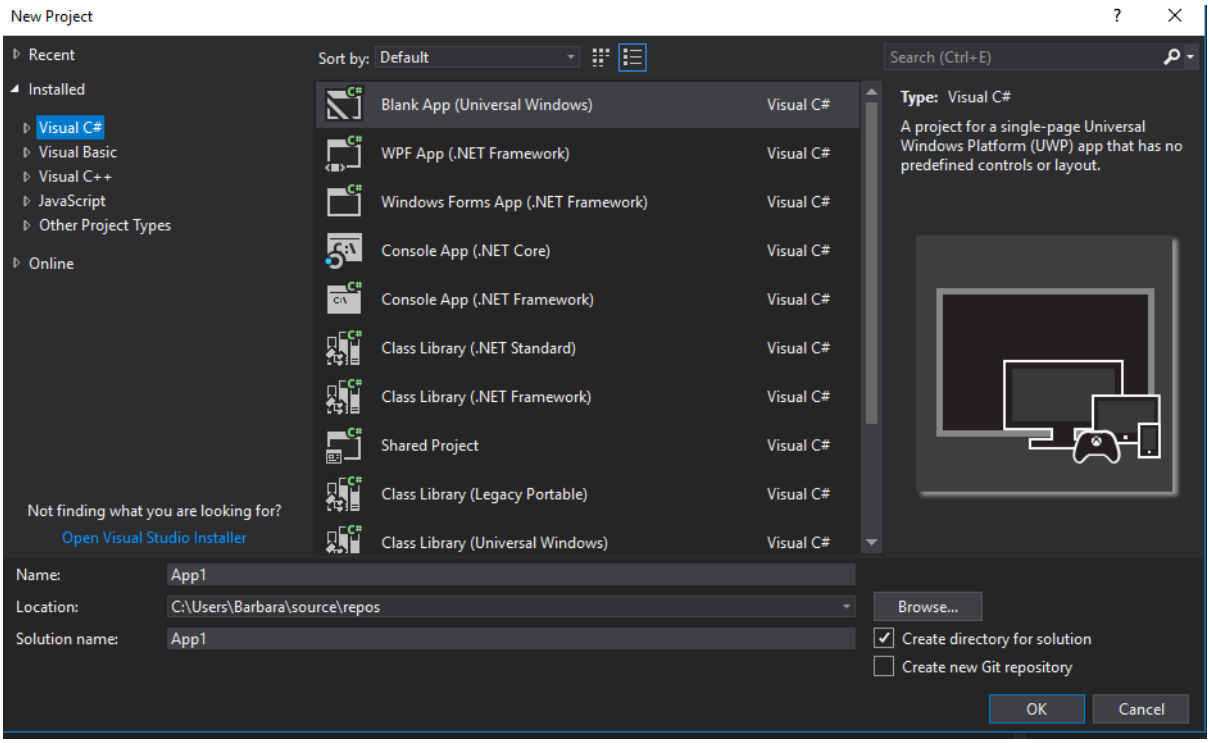
Zadatak završnog rada je konstruirati nepobjedivi algoritam za društvenu igru iks-oks koja će biti izrađena u digitalnom obliku. Iks-oks igra bit će koncipirana pomoću programskog jezika C#.

# 2. PROGRAMSKA PODRŠKA

Cilj završnog rada je nadograditi znanje programiranja i steći nove spoznaje programskim alatima: integrirano razvojno okruženje Microsoft Visual Studio 2017 te široko rasprostranjeni, objektno orijentirani C# programski jezik. Korištena programska podrška detaljnije je opisana u tekstu koji slijedi pod naslovima 2.1. i 2.2.

## 2.1. Microsoft Visual Studio

Microsoft Visual Studio je integrirano razvojno okruženje koje svojim korisnicima nudi programiranje u različitim jezicima sa različitom svrhom. [1] Koristi se za razvoj računalnih programa za Windows, Web stranice, Web aplikacije i Web usluge. Jezici koji se nalaze u osnovnom paketu prilikom instalacije Microsoft Visual Studia su: C, C#, C++, C++/CLI, VB.NET, F#. Microsoft Visual Studio, osim osnovnog paketa, također svojim korisnicima pruža podršku za druge jezike koji se tretiraju kao dodatak na osnovni paket (npr. interpretacijski programski jezik više razine – Python ) .




Slika 2.1 Radna površina Visual Studia

Slika 2.1. prikazuje kako izgleda prvi korak u izradi novog projekta putem Visual Studia 2017. Na lijevoj strani nalazi se izbornik koji nam nudi mogućnost odabira programskog jezika kojim se želimo služiti prilikom izrade projekta. Desno od njega nalazi se izbornik pomoću kojeg detaljnije odrađujemo vrstu projekta kojeg želimo izraditi. U ranije spomenutom izborniku nalaze se predlošci koji u sebi sadržavaju unaprijed implementirane biblioteke te dijelove koda ovisno o vrsti projekta kojeg izabiremo. Također, sadrži implementirane gotove strukture za određene funkcije, petlje, naredbe, itd. Visual studio ovakvim pristupom svojim korisnicima uvelike olakšava, pojednostavljuje i ubrzava sam proces pisanja programskog koda.

## 2.2. C# programski jezik

C# je elegantan objektno-orijentirani programski jezik koji omogućava programeru da izradi različite vrste sigurnih aplikacija koje rade na bazi .NET Framework-a. C# se može koristiti za izradu Windows aplikacija, XML Web usluga, klijent-server aplikacija, itd. Sintaksa C# pruža mogućnost rješavanja problema na više različitih načina, vrlo je jednostavna za naučiti i već prepoznatljiva poznavateljima C, C++ ili Jave. [2]
Kao objektno – orijentirani jezik, C# podržava i koncepte:

1. **ENKAPSULACIJE** – mogućnost objekta da sakrije podatke i „ponašanje“ koje nije nužno. Ona omogućava grupi svojstava, metoda i ostalih članova da budu prikazani kao jedinica ili objekt.

2. **NASLJEĐIVANJA** – osnovna ideja nasljeđivanja je da se prilikom razvoja identificiraju razredi koji imaju sličnu funkcionalnost te da se u izvedenim razredima samo definiraju specifična svojstva, dok se preostala svojstva nasljeđuju u nepromijenjenom obliku. Klasa čiji su članovi naslijeđeni naziva se **osnovna klasa**, a klasa koja nasljeđuje te članove naziva se **izvedena klasa**.

3. **POLIMORFIZMA** – Svaki objekt izvedenog razreda (klase) može se promatrati i kao objekt bilo kojeg javnog osnovnog razreda. Time se ne narušava integritet objekta jer su svi članovi osnovnog razreda prisutni i u objektima izvedenog razreda pa im se bez problema može i pristupiti. Tako je moguće kreirati pokazivač tipa osnovne klase i usmjeriti ga na objekt izvedene. [3]

Što je to metoda?

Metoda klase je blok naredbi koji se sastoji od zaglavlja i tijela metode. U zaglavlju se nalazi povratni tip i ime metode. Poslije zaglavlja u vitičastim zagradama se piše tijelo metode koje se sastoji od odgovarajućih naredbi programskog jezika C#.

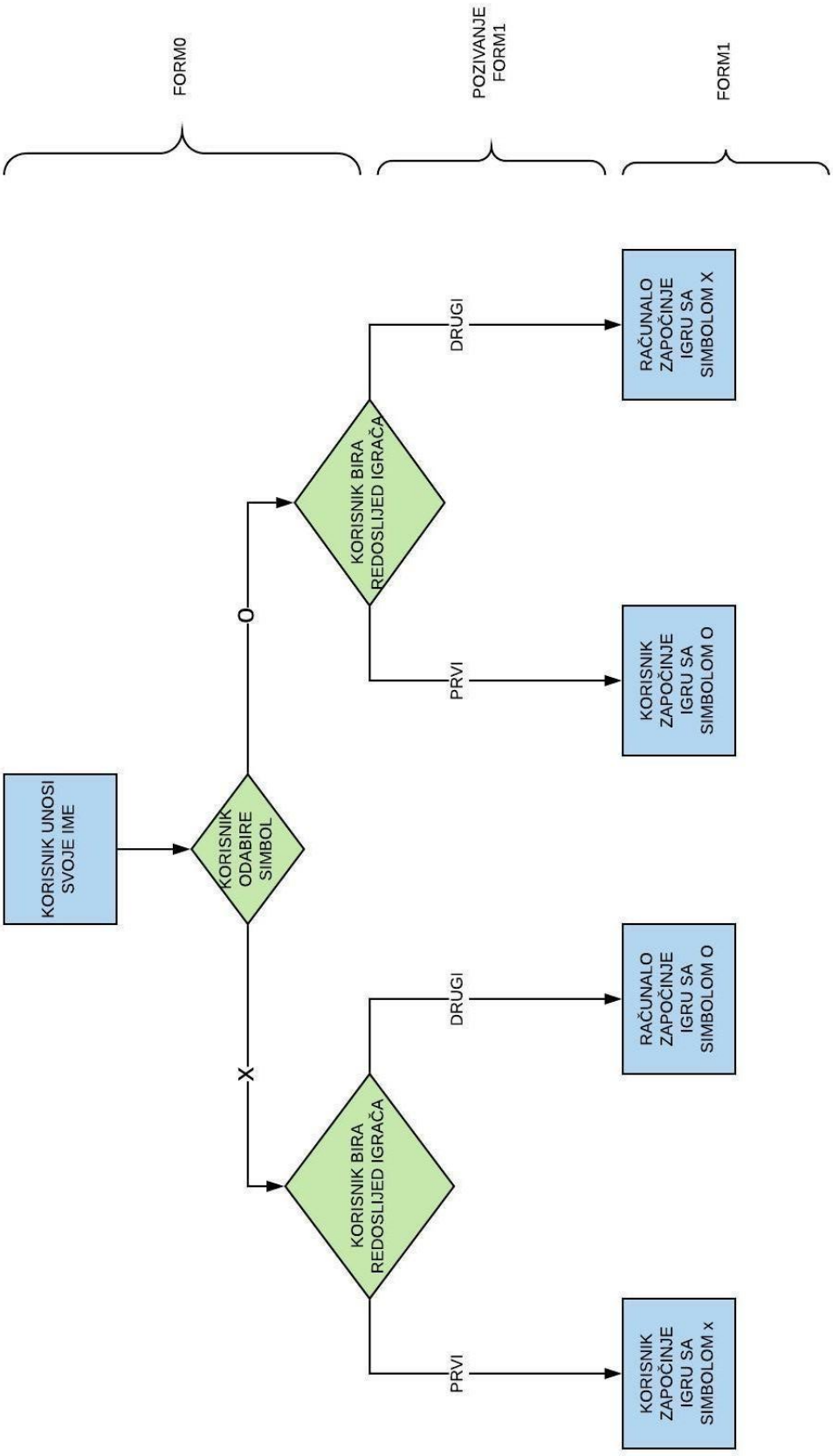
Metodama se opisuje ponašanje objekta u određenoj situaciji i pod određenim uvjetima ali i određujete nove vrijednosti na temelju osobina koje objekt posjeduje (Na taj način se opisuje funkcionalnost objekta) [4]

Sve varijable i metode, uključujući „Main()“ metodu (početna točka programa), su enkapsulirane unutar definicija KLASA. Klasa može naslijediti direktno iz jedne roditeljske klase, ali može implementirati bilo koji broj sučelja

# 3. IZRADA IKS-OKS IGRE

## 3.1. Dijagram tijeka programa

Iks-oks igra izvedena je pomoću dvije forme - „*Form0*“ i „*Form1*“. Način na koji iks-oks igra funkcionira ukratko je prikazan i objašnjen pomoću dijagrama tijeka koji se nalazi na Slici 3.1. Funkcioniranje koda i samog programa bit će detaljnije objašnjen u tekstu koji slijedi pod naslovima 3.2. i 3.3.



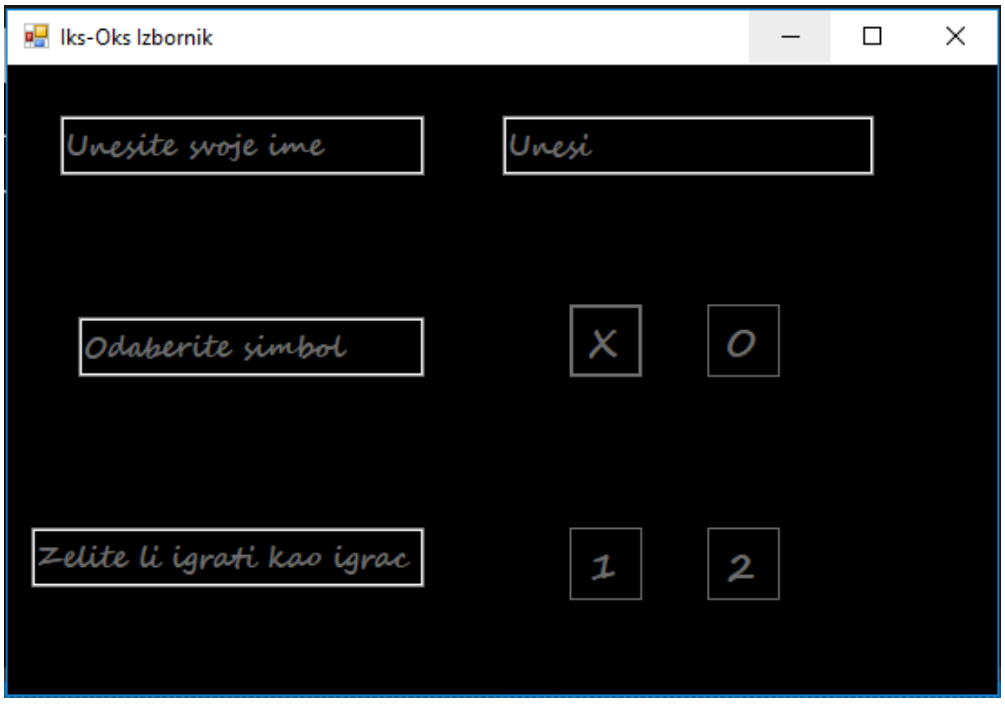

Slika 3.1. Dijagram tijeka programa [5]

## 3.2. Form0

Prilikom pokretanja aplikacije otvara se prvo programsko sučelje pod nazivom Iks-Oks Izbornik prikazano na slici 3.2.

Izbornik omogućava korisniku 3 opcije:

1. Unos korisničkog imena pomoću TextBox-a.
2. Pritiskom na gumb korisnik bira simbol kojim želi odigrati igru, može izabrati simbol X(iks) ili se pak može opredijeliti za O (oks).
3. Također, pomoću gumba se pruža mogućnost da korisnik odabere želi li odigrati prvi potez igre ili će to prepustiti računalu te odigrati korak nakon njega.




Slika 3.2. Prikaz izgleda forme „*Form0*“

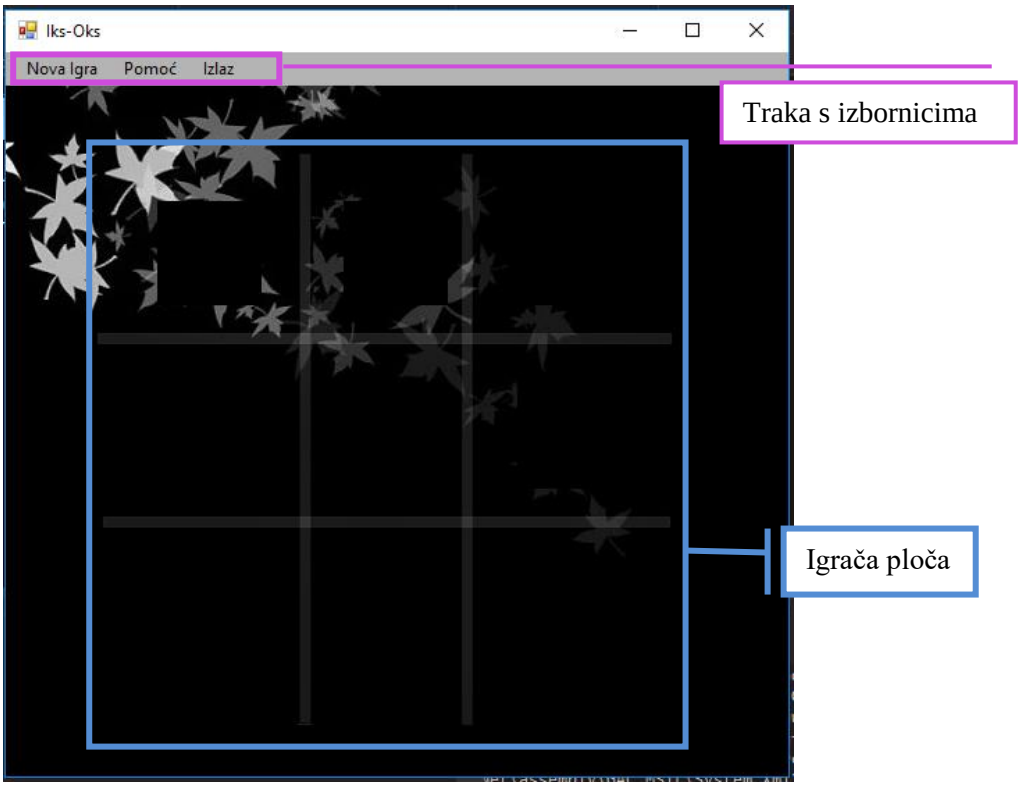
Pritiskom na bilo koji gumb (1 ili 2) koji određuje prioritete igrača poziva se forma „*Form1*“ koja će poprimiti sve parametre koje je korisnik unio popunjavanjem na formi „*Form0*“. Način na koji se poziva forma „*Form1*“ prikazan je na Slici 3.3.

```
private void prvi_btn_Click(object sender, EventArgs e)
{
    red1 = true;
    Form1 frm = new Form1(igrac1, red1, ime);
    frm.Show();
    this.Hide();
}
```

Slika 3.3. Pozivanje Fome 1

## 3.3. Form1

Prilikom otvaranja forme „*Form1*“ prethodna forma („*Form0*“), tj. prozor se zatvara. Izgled forme „*Form0*“, koji se dobiva prilikom pokretanja igre, detaljnije je prikazan na slici 3.4.




Slika 3.4 Prikaz izgleda Forme1

U formi „*Form1*“ implementirana je traka s 3 padajuća izbornika. (Vidi sliku 3.4).

3.3.1. **Izbornik „Nova igra“** - kao što sam naziv izbornika kaže, on omogućava korisniku da pokrene novu igru, odnosno da s igrom krene od početka. Metoda NovaIgra() pomoću ugniježđene „*for*“ petlje svaki element matrice izjednači s vrijednošću -1 (u ovom kodu, vrijednost -1 označava da je mjesto u matrici prazno).

```
//izbornik
private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    NovaIgra();
}  // Nova Igra
void NovaIgra()

{

    btn00.Text = btn01.Text = btn02.Text = btn10.Text = btn11.Text = btn12.Text = btn20.Text = btn21.Text = btn22.Tex
    for (int i= 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            matrica[i, j] = -1;
        }
    }
}
```

Slika 3.5 Programski kod Izbornika „Nova igra“

ŠTO JE UGNIJEŽĐENA PETLJA?

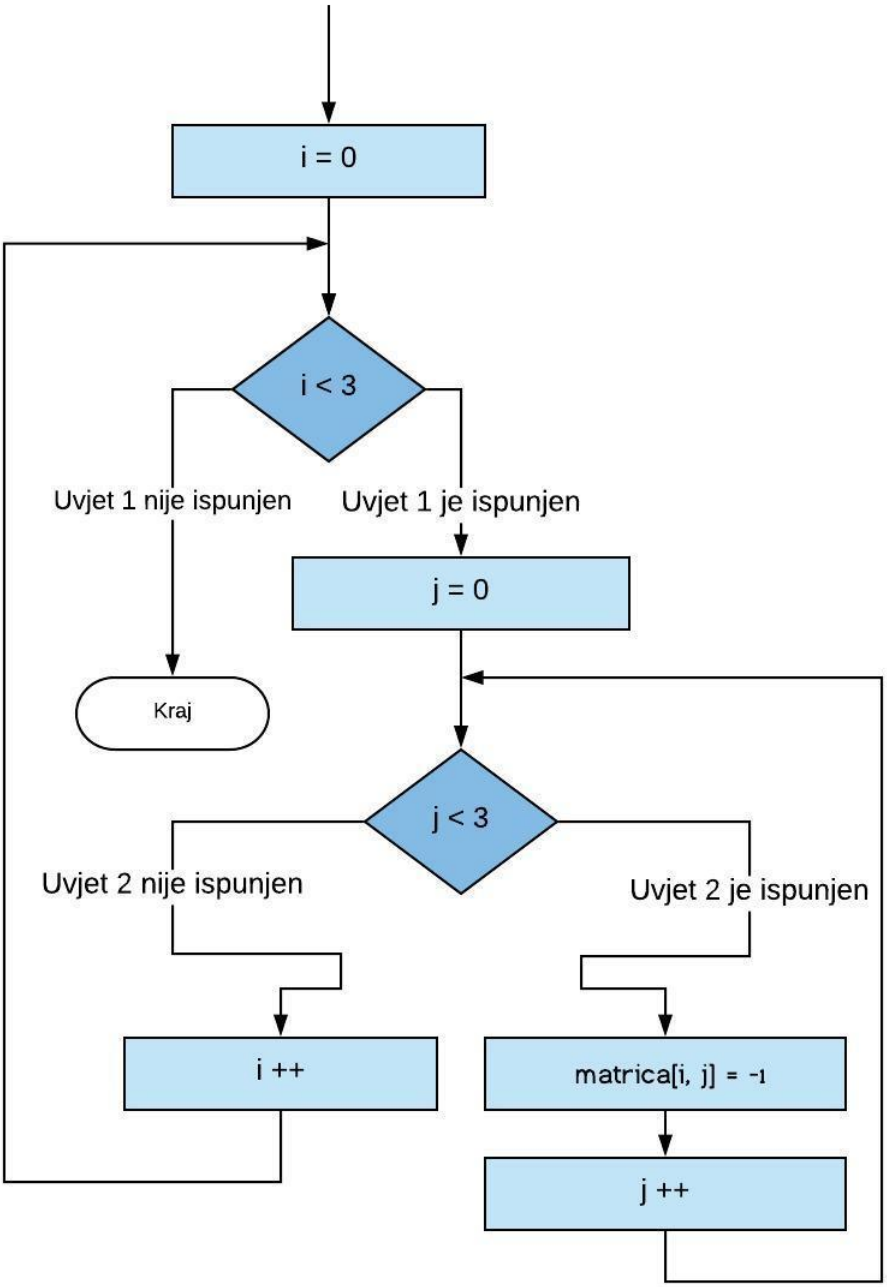
Ugniježđena „for“ petlja naziv je za „*for*“ petlju koja je napisana unutar druge „*for*“ petlje. Prva petlja naziva se **vanjska** petlja, a „*for“* petlja koja je ugniježđena unutar vanjske petlje naziva se **unutarnja** „*for*“ petlja. „*For*“ petlja radi na principu kontrolnih varijabli čije se vrijednosti mijenjaju svakim prolaskom kroz petlju ovisno o operandu koji je napisan u petlji. Za vanjsku petlju kontrolna varijabla je **i,** dok je za unutarnju to pak **j.** [6]

KAKO ONA FUNKCIONIRA?

- Ulaskom u vanjsku petlju, kontrolna varijabla **„i**“ poprima početnu vrijednost. Provjerava se uvjet vanjske petlje koji, ako je ispunjen, omogućava ulaz u unutarnju petlju.
- Pri ulazu u unutarnju petlju, **kontrolna varijabla j** poprima početnu vrijednost , zatim se provjerava **uvjet unutarnje petlje** te se izvršava **blok naredbi**

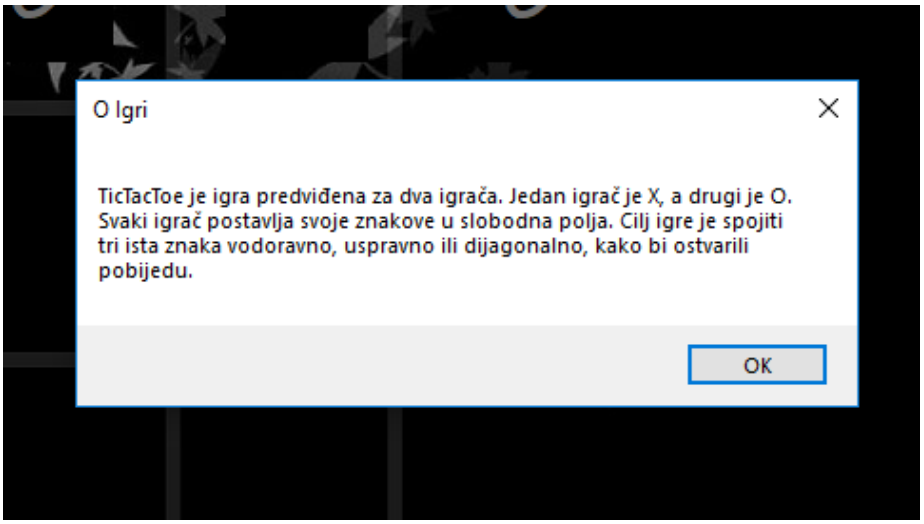
unutarnje petlje ako je ispunjen uvjet.

- Po završetku bloka naredbi unutarnje petlje mijenja se vrijednost kontrolne varijable unutarnje petlje za **vrijednost prirasta unutarnje petlje**. Sve dok je zadovoljen uvjet unutarnje petlje njen blok naredbi se ponavlja uz stalnu vrijednost kontrolne varijable vanjske petlje.
- Kada uvjet unutarnje petlje nije ispunjen, izlazi se iz unutarnje petlje i ponovo se počinje izvršavati vanjska petlja. Vrijednost kontrolne varijable vanjske petlje mijenja se za vrijednost prirasta, provjerava se uvjet vanjske petlje pa ako je ispunjen, ulazi se u unutarnju petlju. Sada se ponovo ponavlja opisan postupak izvršavanja unutarnje petlje.
- Sve se ponavlja do trenutka kada uvjet vanjske petlje više nije ispunjen. Valja uočiti da se za svaku vrijednost kontrolne varijable vanjske petlje izvodi cjelokupna unutarnja petlja.




Slika 3.6 Blok dijagram ugniježđene petlje. [5]

3.3.2. **Izbornik „Pomoć“** – Prilikom pokretanja izbornika „Pomoć“, korisniku se na ekranu ispisuje poruka prikazana na slici 3.7, sadržaj iste omogućava korisniku da se podsjeti pravila igre. Ranije spomenuta poruka je kreirana pomoću naredbe MessageBox.Show(„Sadržaj poruke“, „Naslov poruke“).




Slika 3.7 Sadržaj poruke koja se ispisuje prilikom pokretanja izbornika „Pomoć“

3.3.3. **Izbornik „Izlaz**“ – omogućava korisniku prekid ili zaustavljanje igre.

```
private void izlazToolStripMenuItem_Click(object sender, EventArgs e)   // Izlaz iz igre
{
    this.Close();
}
```

Slika 3.8 Naredba za izlazak iz igre.

VARIJABLE, METODE I AKCIJE „On Click“

U tekstu koji slijedi detaljnije će se obraditi varijable i metode. U tekstu koji slijedi nalaze se tri tablice:

a. Tablica 3.1. detaljno opisuje **varijable** koje su korištene pri izradi algoritma za „nepobjedivi iks-oks.“ U tablici se nalaze imena varijabli, njihovi tipovi kao i sam razlog zbog kojeg su korištene baš te varijable.

b. U Tablici 3.2. nalazi se lista sa svim **metodama** koje su korištene u programskom kodu, tipovi metoda te funkcionalnosti koje te metode obavljaju nad pozivajućim programom.

c. U Tablici 3.3. nalazi se lista komponenata koje su korištene kako bi se konstruirala forma „*Form1*“. Navedene komponente na određeni događaj izvršavaju određenu radnju tj. akciju. Sve korištene komponente isprogramirane su da na **klik** pozivaju određene metode što je detaljnije opisano unutar Tablice 3.3.

Tablica 3.1. Varijable

| TIP | NAZIV VARIJABLE | ZA ŠTO SE KORISTI? |
|---|---|---|
| **int** | i,j | Varijable i,j su kontrolne varijable. Što su to kontrolne varijable? Kontrolna varijabla se nalazi u „*for*“petlji, te se svakim prolaskom kroz petlju njena vrijednost mijenja ovisno o zadanoj operaciji. |
| **int** | brojac = 0 | Vrijednost varijable se povećava za jedan, prilikom svakog odigranog pokreta. Brojac se povećava svaki puta kada se promjeni vrijednost varijable iz 1 u 0 i obrnuto, pa možemo zaključiti, da ona zapravo broji broj odigranih pokreta. |
| **int** | suma = 0 | Varijabla suma se koristi se spremanje rezultata sume vrijednosti elemenata u matrici, i na temelju izračunate sume, određuje se koji je slijedeći potez računala. Ovisno o dijelu programa u kojem se suma koristi, računa se suma redaka, stupaca, dijagonale (sporedne ili glavne) ili pak cijele matrice. |
| **bool** | sljedeci | Varijabla **sljedeci**, korištena je kako bi naznačila računalu tko je sljedeći igrač.<br>Kada je njena vrijednost jednaka **false** – oks je sljedeći na potezu.<br>Kada je njena vrijednost pak jednaka **true –** iks je sljedeći na potezu. |
| **bool** | pv | Pv je privremena varijabla, čija vrijednost prikazuje koji igrač je trenutno na potezu. Kada je pv jednako **true**, na potezu je iks. No, kada je pak pv jednako **false**, na potezu je oks. |
| **bool** | red | Varijable red, koristi se prilikom odabira prioriteta igrača. Igrač postavlja varijablu red na vrijednost **false**, - prepušta računalo da odigra prvi potez. Ako se pak igrač odluči da želi on odigrati prvi potez, varijabla red postavlja se u **true** te on dobiva prioritet u igri**.** |
| **private int** | private int[,] matrica = new int[3, 3]; | U ovom kodu korištena je kvadratna matrica dimenzije 3x3, a sastoji se od 9 elemenata. Matrica je osmišljena na sljedeći način:<br>MATRICA = $\begin{bmatrix} a(0,0) & a(0,1) & a(0,2) \\ a(1,0) & a(1,1) & a(1,2) \\ a(2,0) & a(2,1) & a(2,2) \end{bmatrix}$<br>Svaki element matrice označen je dvama brojevima – indeksima. Prvi broj kojim je označena matrica predstavlja broj retka u kojem se nalazi element, a drugi broj predstavlja broj stupca u kojem se nalazi element. Indeksiranje nam omogućava brže i lakše pretraživanje, popunjavanje ili brisanje elemenata matrice. |

Tablica 3.2. Metode

| TIP | NAZIV METODE | ZA ŠTO SE KORISTI? |
|---|---|---|
| **void** | Pobjeda() | Klikom na gumb koji se nalazi na igraćoj ploči poziva se metoda Pobjeda() koja provjerava postoji li tri ista simbola u retku, stupcu ili jednoj od dijagonala. |
| **void** | ProvjeriBrojac() | Metoda ProvjeriBrojac također se poziva u program klikom na gumb, tj. nakon svakog odigranog pokreta. Ako je vrjednost brojača jednaka 9, a nije ostvaren niti jedna pobjednička kombinacija, na ekranu se ispisuje poruka da je igra završila neriješenim rezultatom. |
| **void** | NovaIgra() | Metoda NovaIgra() omogućava korisniku da pokrene igru od početka. |
| **bool** | RacunaloIgra(bool pv) | RacunaloIgra() je metoda pomoću koje se računalu ugrađuje umjetna inteligencija kako bi računalo bilo dostojan protivnik korisniku protiv kojeg igra. Ona predstavlja algoritam pomoću kojeg računalo „odlučuje“ o svom sljedećem pokretu. |
| **void** | clck(bool pv, int i, int j) | Clck() je metoda pomoću koje se na zaslonu prikazuje odigrani pokret, a ovisno o igraču koji je na potezu ispisuje se iks ili oks. |

Tablica 3.3. On Click

| TIP | NAZIV METODE | On_Click ? |
|---|---|---|
| **private void** | izlazToolStripMenuItem_Click(object sender, EventArgs e) | Prilikom klika na izbornik „Izlaz“ program se prekida i aplikacija se gasi. |
| **private void** | oIgriToolStripMenuItem_Click(object sender, EventArgs e) | Klikom na izbornik „Pomoć“ na ekranu se otvara Message Box, koji prikazuje poruku sa pravilima igre |
| **private void** | aboutToolStripMenuItem_Click(object sender, EventArgs e) | Odabirom izbornika „Nova Igra“ poziva se metoda NovaIgra(), te se igra započinje ponovno. |
| **private void** | btn22_Click(object sender, EventArgs e) | Klikom na određeni gumb na igraćoj ploči pozivaju se sve metode koje su navedene u gornjoj tablici. |

## TIJEK PROGRAMA PRILIKOM POKRETANA FORM1

U sljedećem tekstu bit će objašnjeno na koji način funkcionira program od samog pokretanja forme „*Form1*“ pa sve do krajnjeg rezultata koji se dobije po završetku igre.

### POZIVANJE FORME „*Form1*“

Pokretanjem igre korisniku se na ekranu ispisuje „*Form0*“. On popunjava potrebne podatke kako bi mogao nastaviti s igrom (upisuje svoje ime, odabire simbol kojim želi nastaviti igru, kao i prioritet igrača). Nakon popunjavanja upita koji su definirani u formi „*Form0*“, korisniku se na ekranu otvara forma „*Form1*“.

Neovisno o prethodnoj formi te odabiru simbola i prioriteta, prilikom svakog pokretanja igre brojač odigranih poteza se uvijek postavlja na nula, svim elementima matrice se dodjeljuje početna vrijednost -1 (tj. matrica se „prazni“), briše se i tekst (ako postoji) koji je ispisan na gumbovima.
Forma „*Form1*“, prilikom svog pokretanja, od pozivajućeg programa poprima određene parametre koji će kasnije biti korišteni u samom programu. Dakle, pravac u kojem će se razvijati program ovisi o primljenim parametrima, a to su odabrani simbol te prioritet igrača.

```
public Form1(String igr1, bool red1, String ime)    // Iz Form0 primamo simbol kojim igrač želi igrati
{
    InitializeComponent();
    brojac = 0;
    lbl_1.Text = ime;
    if (igr1 == "X" && red1 == true ) { pv = true; red = true;  }
    if (igr1 == "X" && red1 == false) { pv = false; red = false;  pv=RacunaloIgra(pv); brojac = brojac - 1; }
    if (igr1 == "O" && red1 == true)  { pv = false; red = true;  }
    if (igr1 == "O" && red1 == false) { pv = true; red = false; pv=RacunaloIgra(pv); brojac = brojac - 1; }

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            matrica[i, j] = -1;
        }
    }
}
```

Slika 3.9 Koraci koji se obavljaju prilikom pokretanja „Form1“

TIJEK PROGRAMA OVISNO O PRIORITETU

i. **Prvi na potezu je korisnik**

- Kada je korisnik prvi na potezu varijabla **red** uvijek poprima vrijednost „*true*“
- Korisnik igra simbolom X - pv poprima vrijednost „*true*“,
  korisnik se pak odluči za simbol O, pv poprima vrijednost „*false*“
- Korisnik sada odabire mjesto na igračoj ploči gdje želi staviti simbol, prilikom pritiska na određeno mjesto, korisnik zapravo klikne na gumb koji predstavlja element matrice te se poziva metoda *„clck()“*. Nakon klika na gumb, na njemu se ispisuje tekst X ili O, ovisno o tome kojim simbolom igrač igra rundu.

**Akcija „On Click“**

Igraču ploču čini 9 gumbova. Jedan gumb predstavlja jedan element matrice.
Kodovi gumbova su vrlo slični, jedina razlika u kodovima su brojevi kojima su numerirani stupci i redci.
Kao primjer uzet ćemo da je korisnik stisnuo gumb „btn00“, tj. element matrice $a_{00.}$ Na slici ispod prikazan je kod kojim je opisan „btn00“, a u tekstu ispod slike slijede objašnjenje koda i način na koji funkcionira program prilikom pritiska na gumb.

```
//na klik gumba, gumbu se dodijeljuje tekst x ili o, ovisno o odabranom simbolu
private void btn00_Click(object sender, EventArgs e)
{
    if (btn00.Text == "")
    {
        if (pv == true) {  clck(true, 0, 0); pv = false; pv=RacunaloIgra(pv); }
        else { clck(false, 0, 0); pv = true; pv=RacunaloIgra(pv); }
        Pobjeda(); ProvjeriBrojac();


    }
}
```

Slika 3.10. Kod gumba „btn00“

Klikom na određeno polje u matrici, prvo dolazi do provjere jeli to polje uopće dostupno za odabrati. Ranije navedena provjera, odrađuje se pomoću naredbe „ *if(btn.00== „“)* “. Dakle, ako gumbu „btn00“ nije dodijeljen nikakav tekst, odnosno ako je on prazan, korisnik ga može

odabrati. Ova provjera sprječava korisnika ili računalo da ponovno kliknu na gumb kojem je već ranije dodijeljen tekst X ili O.

Recimo da je korisnik odabrao simbol X, program provjera uvjet „*if(pv == true)*“, uvjet je istinit te se obavlja blok naredbi koje su zadane pod tim uvjetom. Ulaskom u blok naredbi poziva se metoda „Clck(true,0,0)“. Parametri koje ona poprima prilikom pozivanja naznačuju da se na igraču ploču ispiše simbol X, na elementu matrice a[0,0]. Pogledajmo kod na slici 3.11., iz slike možemo zaključiti da je samo prvi uvjet zadovoljen, a svi drugi se preskaču. Nakon što se na ekran ispiše simbol X, vrijednost brojača odigranih pokreta se poveća za 1.

```
//Funkcija koja se poziva klikom na gumb  ili kada računalo igra slijedeći pokret
void clck(bool pv, int i, int j)
{
    if (pv == true)
    {
        if (i == 0 && j == 0) { matrica[0, 0] = 1; Button b = btn00; b.Text = "X"; }
        if (i == 0 && j == 1) { matrica[0, 1] = 1; Button b = btn01; b.Text = "X"; }
        if (i == 0 && j == 2) { matrica[0, 2] = 1; Button b = btn02; b.Text = "X"; }
        if (i == 1 && j == 0) { matrica[1, 0] = 1; Button b = btn10; b.Text = "X"; }
        if (i == 1 && j == 1) { matrica[1, 1] = 1; Button b = btn11; b.Text = "X"; }
        if (i == 1 && j == 2) { matrica[1, 2] = 1; Button b = btn12; b.Text = "X"; }
        if (i == 2 && j == 0) { matrica[2, 0] = 1; Button b = btn20; b.Text = "X"; }
        if (i == 2 && j == 1) { matrica[2, 1] = 1; Button b = btn21; b.Text = "X"; }
        if (i == 2 && j == 2) { matrica[2, 2] = 1; Button b = btn22; b.Text = "X"; }
        brojac++;

    }
```

3.11. Metoda „clck()“

Nakon ispisa, vrijednost varijable pv se mijenja iz „true“ u „false“ kako bi slijedeći na redu bio igrač  sa simbolom O, čime se sprječava da igrač X odigra dva uzastopna pokreta. Potom se poziva metoda „RačunaloIgra()“  koja će biti obrađena pod podnaslovom „Računalo igra prvi potez“.

Zatim se uvjet „*else*“  preskače jer je prethodni uvjet zadovoljen. Potom, nakon svakog klika na gumb, se pozivaju dvije metode: „Pobijeda()“ i „ProvjeriBrojac()“.

Metoda „Pobjeda()“ provjerava postoji li pobjednička kombinacija simbola.

**Pobjeda()**

U ranije napisanom tekstu spominje se da je početna vrijednost svakog elementa matrice jednaka -1. Kada kliknemo na neki gumb igrače ploče dolazi do promjene vrijednost elementa matrice koji je odabran. Ako igrač igra kao iks, vrijednost se mijena iz -1 u 1, a ako pak igrač igra kao oks vrijednost se mijenja iz -1 u 2. To je način na koji računalo prepoznaje je li odigran iks ili oks simbol.

a. HORIZONTALNE PROVJERE

Horizontalna provjera pomoću uvjeta „*if*“ provjerava sve retke matrice, ako je neki od redaka u potpunosti ispunjen sa vrijednostima 1 ili 2, pomoću „MessageBox.Show()“ naredbe, na ekranu korisnika ispisuje se pobjednik. Nakon što je ispisana pobjednička poruka, pokreće se nova igra.

```
//metoda za provjeru pobijeda
void Pobjeda()
{
    //horizontalna provjera

    if ((matrica[0, 0] == 1 && matrica[0, 1] == 1 && matrica[0, 2] == 1)
        || (matrica[1, 0] == 1 && matrica[1, 1] == 1 && matrica[1, 2] == 1)
        || (matrica[2, 0] == 1 && matrica[2, 1] == 1 && matrica[2, 2] == 1))
    {
        MessageBox.Show("Pobijedio je X", "Pobjeda");
        NovaIgra();
    }

    if   ((matrica[0, 0] == 2 && matrica[0, 1] == 2 && matrica[0, 2] == 2)
       || (matrica[1, 0] == 2 && matrica[1, 1] == 2 && matrica[1, 2] == 2)
       || (matrica[2, 0] == 2 && matrica[2, 1] == 2 && matrica[2, 2] == 2))
    {
        MessageBox.Show("Pobijedio je 0", "Pobjeda");
        NovaIgra();
    }
```

3.12. Kod horizontalne provjere

b. VERTIKALNE PROVJERE

U ovom slučaju, pomoću uvjeta „if“, prolazi se kroz sve stupce matrice. Neka je svim elementima iz stupca dodijeljena vrijednost jedan, tada se na ekran korisnika ispisuje poruka koja obavještava da je pobijedio simbol iks. No, ako su pak elementi stupca popunjeni sa dvojkama, tada se ispisuje poruka o pobijedi simbola oks.

```
//vertikalna provjera

if ((matrica[0, 0] == 1 && matrica[1, 0] == 1 && matrica[2, 0] == 1)
    || (matrica[0, 1] == 1 && matrica[1, 1] == 1 && matrica[2, 1] == 1)
    || (matrica[0, 2] == 1 && matrica[1, 2] == 1 && matrica[2, 2] == 1))
{
    MessageBox.Show("Pobijedio je X", "Pobjeda");
    NovaIgra();

}

if ((   matrica[0, 0] == 2 && matrica[1, 0] == 2 && matrica[2, 0] == 2)
    || (matrica[0, 1] == 2 && matrica[1, 1] == 2 && matrica[2, 1] == 2)
    || (matrica[0, 2] == 2 && matrica[1, 2] == 2 && matrica[2, 2] == 2))
{
    MessageBox.Show("Pobijedio je O", "Pobjeda");
    NovaIgra();

}
```

3.13. Kod za vertikalnu provjeru

c. PROVJERA DIJAGONALA

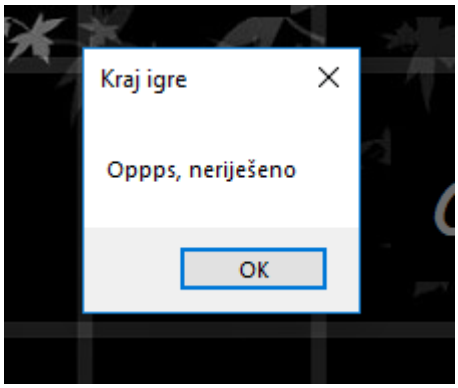
Kao i u dvije prethodne provjere, tako se i ovdje provjerava pomoću uvjeta „*if*“. Ovaj dio programa zadužen je da provjerava glavnu i sporednu dijagonalu.

```
//dijagonale
if  ((matrica[0, 0] == 1 && matrica[1, 1] == 1 && matrica[2, 2] == 1)
    || (matrica[2, 0] == 1 && matrica[1, 1] == 1 && matrica[0, 2] == 1))
{
    MessageBox.Show("Pobijedio je X", "Pobjeda");
    NovaIgra();

}

if   ((matrica[0, 0] == 2 && matrica[1, 1] == 2 && matrica[2, 2] == 2)
   || (matrica[2, 0] == 2 && matrica[1, 1] == 2 && matrica[0, 2] == 2))
{
    MessageBox.Show("Pobijedio je O", "Pobjeda");
    NovaIgra();

}
```

3.14. Kod za provjeru dijagonala

**„ProvjeriBrojac()“**

Metoda „ProvjeriBrojac()“ se poziva nakon metode „Pobijeda()“. Recimo da nije ostvarena niti jedna pobjednička kombinacija, a vrijednost brojača iznosi 9, to znači da je odigran maksimalan broj poteza te da je rezultat neriješen. Korisniku se izbacuje poruka koja je prikazana na slici 3.15.




3.15. Poruka koja se izbacuje prilikom neriješenog rezultata

**ii. Računalo igra prvi potez**

- Svaki puta računalo dobije prioritet za prvi potez, vrijednost varijable red postavlja se u „*false*“.
- Varijabla pv se postavlja u „*true*“ ili „*false*“, ovisno o odabranom simbolu.
- U ovom slučaju uvijek se poziva metoda „*RacunaloIgra()*“, u kojoj se nalazi algoritam, na temelju kojeg računalo odlučuje o pokretima.

**„RacuanaloIgra(bool pv)“**

Dio programa koji poziva metodu „RacunaloIgra()“ samoj metodi predaje parametar *pv* koji naznačuje simbol trenutnog igrača. Ovisno o tome parametru program izvršava blok naredbi ispod odgovarajućeg uvjeta. Prije svakog odigranog pokreta vrijednost varijable *sljedeći* se mijenja iz „*true*“ u „*false*“, ako je sada na redu igrač sa simbolom Iks sljedeći će biti Oks. To je povratni parametar ove metode koji ona predaje pozivajućem dijelu programa nakon svog izvođenja.
Ovo je metoda čiji se rad temelji na sumi, praznim mjestima u matrici te brojaču.

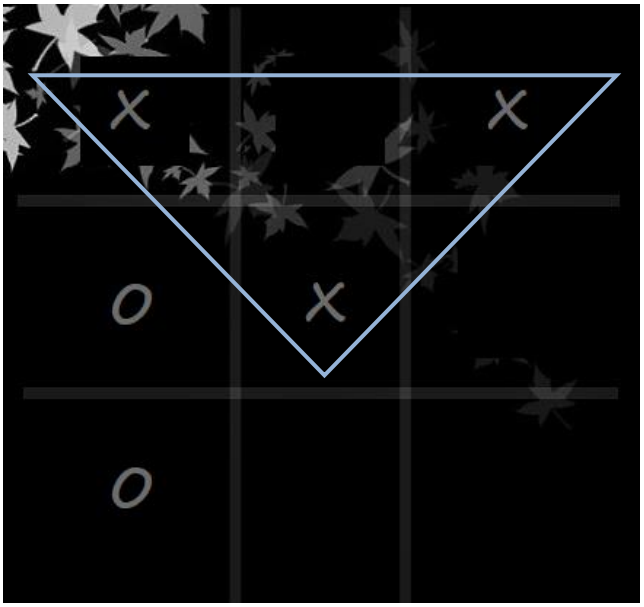
Na početku igre brojač je inicijaliziran na nula što naznačuje da nije odigran niti jedan potez. Dakle, ako je brojač jednak nula te vrijednost varijable red jednaka „*false*“, prvi potez igra računalo. Računalo će uvijek postaviti prvi simbol na mjesto matrice [1,1] kako bi imao otvorene mogućnosti za ostvariti pobjedničku kombinaciju u drugom stupcu, drugom retku, glavnoj i sporednoj dijagonali. Korisniku je blokirana većina pobjedničkih kombinacija te mu sada preostaje mogućnost za pobjedu u prvom i trećem retku i stupcu. Nakon odigranog poteza brojač se poveća za 1 te je sljedeći na redu korisnik. Daljnji tijek programa ovisi o korisnikovom odigranom potezu:

i. Računalo sigurno pobjeđuje

Nepobjedivi algoritam temelji svoj rad na principu da računalo svoje simbole slaže u formaciju „jednakokračnog trokuta“. Radi lakšeg razumijevanja slijedi par primjera.

a) Formacija „ jednakokračni gornji trokut“

Neka se korisnik odlučio za element matrice[1,0]. Računalo će uvijek postaviti svoj simbol na element matrice[0,2] (ukoliko on nije popunjen), korisnik mora igrati na element matrice[2,0] kako bi spriječio računalo da ostvari pobjedničku kombinaciju na sporednoj dijagonali. Računalo sada stavlja svoj simbol na element matrice[0,0] kako bi spriječio korisnika da pobjedi u prvom retku te je ostvarilo formaciju „gornjeg trokuta“ (pogledaj sliku 3.16.). Korisnik se sada nalazi u bezizlaznoj situaciji, a računalo u svakom slučaju odnosi pobjedu bilo to u prvom retku ili pak na glavnoj dijagonali.



Slika 3.16. Formacija „gornjeg trokuta“

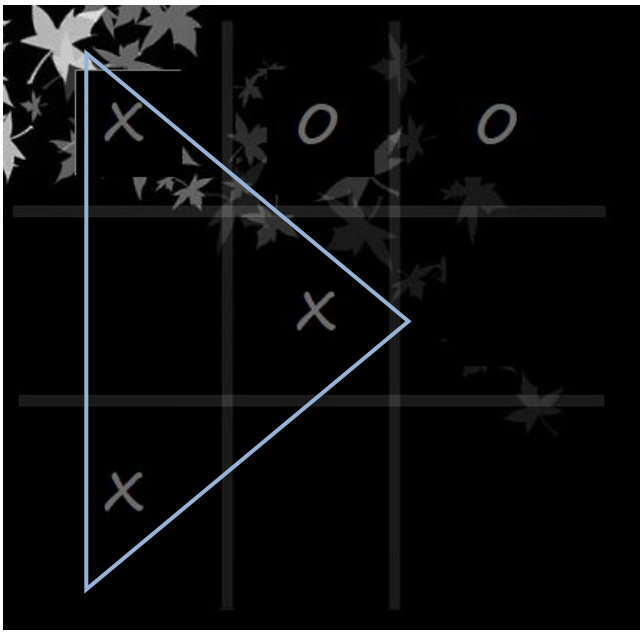
b) Formacija „jednakokračni donji trokut“

Pretpostavimo da je korisnik stavio simbol na gornji desni rub. Računalo sada stavlja svoj simbol na donji lijevi rub (suprotni rub). Korisnik će sada staviti svoj simbol na element matrice[1,2], računalo sprječava pobjedničku kombinaciju u trećem retku te stavlja simbol na donji desni rub. Ostvarena je formacija „donjeg trokuta“ te računalo zasigurno pobjeđuje.

Slika 3.17. Formacija „donjeg trokuta“

c) Formacija „jednakokračni lijevi trokut“

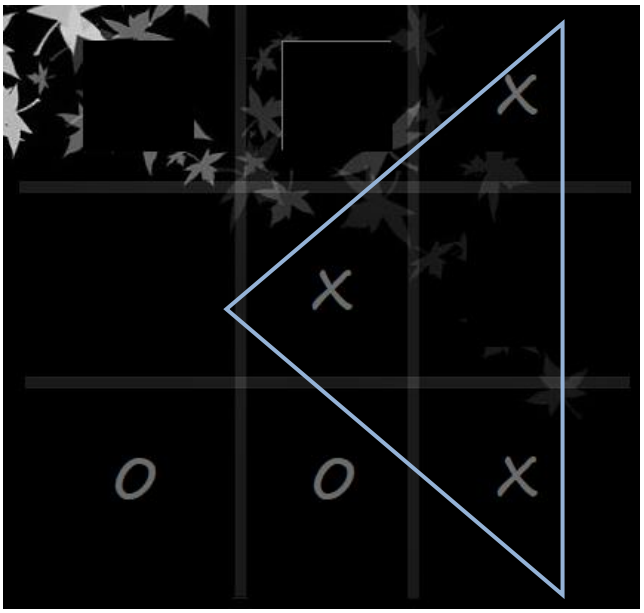
Pretpostavimo da korisnik igra u gornji desni rub, računalo igra rub nasuprotan korisnikovom rubu (donji lijevi rub). Recimo da korisnik pokušava ostvariti pobjedu u prvom retku te stavlja svoj simbol na element matrice[0,1]. Stanje na igraćoj ploči izgleda kao na slici 3.18. Ostvarena je formacija „lijevog trokuta“, računalo je sada u prednosti i neovisno o sljedećem korisnikovu potezu ono pobjeđuje.



Slika 3.18. Formacija „lijevi trokut“

d) Formacija „jednakokračni desni trokut“

Pretpostavimo da je korisnik stavio simbol na donji lijevi rub te računalo odigra nasuprotan rub(gornji desni). Korisnik se odlučio da želi ostvariti pobjedničku kombinaciju u trećem retku te stavlja simbol na polje matrice[2,1]. Računalo stavlja simbol na polje[2,2] te ostvaruje sigurnu pobjedu.



Slika 3.19. Formacija „desni trokut“

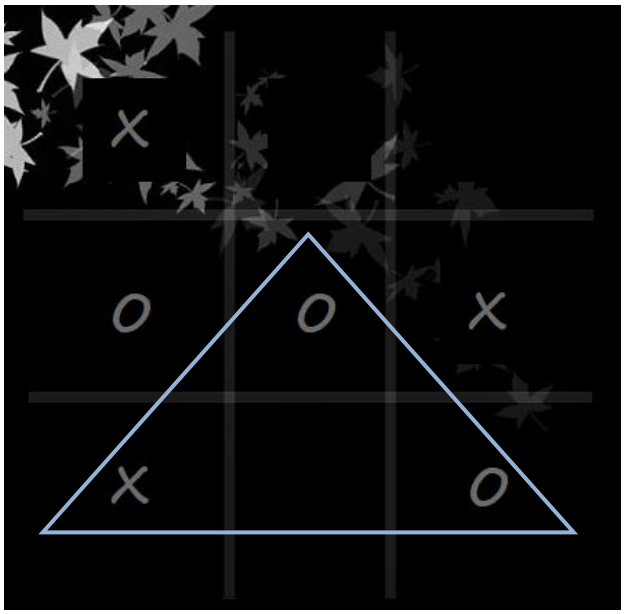
Gore navedeni primjeri nisu jedine moguće formacije trokuta, postoje neke druge kombinacije koje također kao rezultat imaju jednu od formacija „trokuta“.

ii. Računalo ima mogućnost odigrati neriješeno

Neka, na primjer, sada računalo igra s malo iskusnijim i pažljivijim protivnikom koji je „otkrio“ shemu za formaciju raznih vrsta trokuta te metodu „nasuprotnih kutova“. U takvim slučajevima računalo ne ostvaruje pobjedu, već nastoji blokirati korisnika kako on ne bi ostvario pobjedu.

1 º Korisnik pokušava ostvariti formaciju „jednakokračnog trokuta.“

Pretpostavimo da korisnik igra prvi i da se odlučio za simbol oks. Prvi potez će odigrati u sredinu igraće ploče kako bi mu bile otvorene sve mogućnosti za formaciju „jednakokračnog trokuta“. Računalo će postaviti svoj simbol na polje matrice[0,0] i time spriječiti mogućnost formiranja „lijevog“ te „gornjeg jednakokračnog trokuta“. Korisnik sada stavlja svoj simbol u donji desni rub te ostvaruje mogućnost za formiranje „desnog“ ili „donjeg trokuta“. Računalo bira donji lijevi kut i time sprječava korisnika za formaciju „donjeg trokuta“ te ga prisiljava da odigra sljedeći potez na polje matrice[1,0]. (Vidi sliku 3.20.)

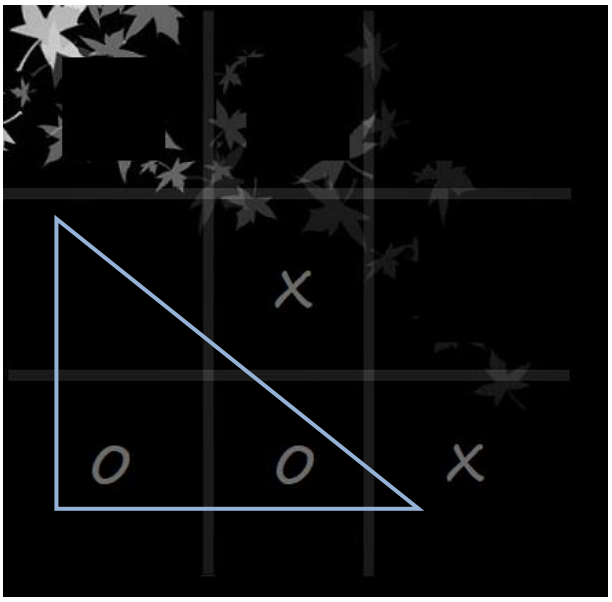



Slika 3.20. Prikaz stanja igrače ploče

Računalo sada opet blokira korisnikovu mogućnost za pobjedu u drugom retku. Neovisno o korisnikovom sljedećem koraku rezultat igre će u svakom slučaju biti neriješen jer računalo zasigurno ima mogućnost blokiranja pobijede.

2 º Korisnik pokušava ostvariti formaciju „pravokutnog trokuta“ s mogućnosti pobjede u prvom stupcu i zadnjem retku

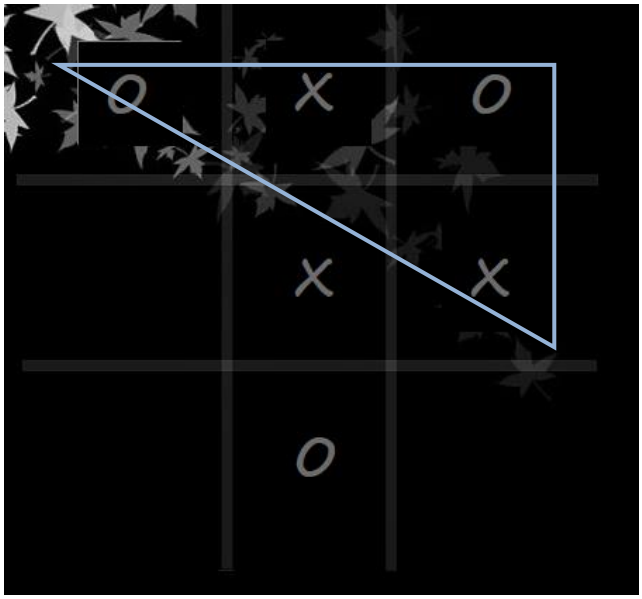
Korisnik započinje igru stavljajući svoj simbol u donji lijevi kut. Računalo stavlja svoj simbol na polje matrice[1,1] te blokira korisnikovu mogućnost za pobjedom na sporednoj dijagonali. Korisnik se odlučuje za formaciju „pravokutnog trokuta“ koja mu donosi mogućnost pobijede u prvom stupcu i zadnjem retku (Pogledaj sliku 3.21.) te bira polje matrice[2,1]. Računalo stavlja svoj simbol u donji desni kut i sprječava korisnika da odnese pobjedu u zadnjem retku, ne uspijeva ostvariti formaciju „pravokutnog trokuta“ te ga prisiljava da sljedeći potez odigra u gornjem lijevom kutu. Računalo sada stavlja svoj simbol na mjesto matrice[1,0] te ga korisnik sprječava da odnese pobjedu u srednjem retku. Rezultat igre opet je neriješen.




Slika 3.21. Trenutno stanje igrače ploče

3 º Korisnik pokušava ostvariti formaciju „pravokutnog trokuta“ s mogućnosti pobjede u prvom retku i trećem stupcu

Korisnik započinje igru te odabire gornji desni kut kao početni korak. Računalo će staviti svoj simbol na polje matrice[1,1] te sprječava korisnika da ostvari pobjedu u drugom retku, drugom stupcu te sporednoj dijagonali. Korisnik se odlučio za formaciju „pravokutnog trokuta“ koja mu daje mogućnost pobjede u prvom retku ili pak u trećem stupcu (Pogledaju Sliku 3.22.) te bira gornji lijevi kut. Računalo je predvidjelo pokušaj korisnika te ga sprječava stavljajući simbol na polje matrice [0,1]. Korisnik je prisiljen odabrati polje matrice[2,1] kako bi spriječio računalo da odnese pobjedu u drugom stupcu. Računalo sada odabire mjesto matrice[1,2] te opet prisiljava korisnika da stavi svoj simbol na polje matrice[1,0] kako bi mu spriječio pobjedu. Neovisno o sljedećem pokretu računala rezultat igre je neriješen.

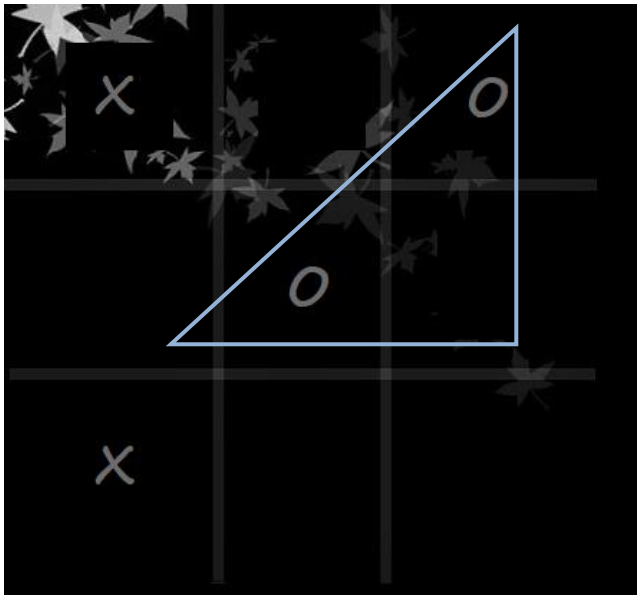



Slika 3.22. Trenutno stanje igrače ploče

4 º Korisnik pokušava ostvariti formaciju „pravokutnog trokuta“ s mogućnosti pobjede u prvom retku, trećem stupcu te na sporednoj dijagonali.

Tijekom ove igre korisnik se odlučio prevariti računalo tako što će pokušati izvesti formaciju „pravokutnog trokuta“ koja će mu omogućiti pobjedu u prvom retku, trećem stupcu ili na sporednoj dijagonali (Vidi sliku 3.23.). Dakle, korisnik se odlučuje da će prvi potez staviti na

polje matrice[1,1]. Računalo svoj simbol stavlja u gornji lijevi rub te blokira korisnika da pobijedi na glavnoj dijagonali, prvom stupcu i prvom retku. Korisnik će sada odabrati gornji desni kut jer mu još uvijek ostaje mogućnost za pobjedu na sporednoj dijagonali i u trećem stupcu. Računalo je predvidjelo korisnikov potez, odabire donji desni kut i korisniku zatvara mogućnost pobijede na sporednoj dijagonali te prisiljava korisnika da sljedeći potez odigra na polju matrice[1,0]. Korisnik sada ima mogućnost pobijede u drugom retku, no računalo i ovaj puta predviđa njegov potez te ga blokira. Neovisno o korisnikovom sljedećem pokretu igra će opet završiti neriješeno.



Slika 3.23. Korisnikov pokušaj ostvarivanja formacije „pravokutnoga trokuta“

*KAKO RAČUNALO IGRA IKS - OKS AKO NEMA FORMACIJE TROKUTA??*

**a) Dva ista simbola u nizu**

Program pomoću sume prvo provjerava postoji li negdje kombinacija od dva ista uzastopna simbola. Ako ona postoji, računalo popunjava element u retku, stupcu ili dijagonali gdje se nalazi odgovarajuća kombinacija u matrici. Računalo odigra potez koji donosi pobjedu ili sprječava korisnika da ostvari pobjedničku

kombinaciju.
Zbog lakšeg razumijevanja, u tekstu koji slijedi navest će se primjer jedne runde iks-oks igre i njegovo objašnjenje.

Uzmimo za primjer da nam trenutno stanje na igraćoj ploči izgleda kao na slici 3.24.

3.24. Trenutno stanje na igraćoj ploči koje se prikazuje korisniku

Važno za napomenuti je da samo korisnik vidi igraću ploču na način na koji je prikazan na slici 3.24. Korisniku je na zaslonu računala ispisan ovakav prikaz igraće ploče. Računalo igraću ploču tretira na drugačiji način. Igraća ploča računalu predstavlja matricu s ispunjenim elementima kao što je to prikazano u formuli (3-1).

$$\text{MATRICA} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 1 & -1 \\ 2 & -1 & -1 \end{bmatrix} \qquad (3\text{-}1).$$

Kao što je već ranije spomenuto, računalo broj -1 tretira kao prazno mjesto unutar matrice, broj 2 kao simbol oks, dok broj 1 tretira kao simbol iks.
Recimo da je računalo sljedeće na potezu i da igra simbolom iks.
Prije samog odigranog poteza računalo prvo prolazi kroz sve provjere dok ne dođe do „*for*“ petlje koja je zadužena da računa sumu stupca. Kod koji služi za „provjeru“ stupaca nalazi se na slici 3.25.

```
suma = 0;
for (i = 0; i < 3; i++)
{
    suma = 0;
    for (j = 0; j < 3; j++)
    {
        suma = suma + matrica[j, i];
    }

    if (suma == 1 || suma == 3)
    {
        if (matrica[0, i] == -1) { clck(true, 0, i);}
        if (matrica[1, i] == -1) { clck(true, 1, i);}
        if (matrica[2, i] == -1) { clck(true, 2, i); }
        return sljedeci;
    }


}
```

3.25. Kod koji provjerava elemente unutar stupaca

Program ulazi u „*for*“ petlju, koja je prikazana na slici 3.25., te računa sumu prvog stupca. Suma stupca sada iznosi 3. Dakle, računalo je zbrojilo samo elemente prvog stupca koji iznose 2 + 2 - 1 (prikazano na formuli 3.1.) što nam u konačnici daje rezultat 3. Program sada provjerava istinitost „*if*“ uvjeta koji nam kaže da, ako je suma jednaka jedan ili tri, izvršava se blok naredbi koji se nalazi ispod ovoga uvjeta. To znači da će računalo na prazno polje unutar prvog stupca matrice staviti simbol iks kako bi spriječio pobjedu korisnika. Ista ova metoda primjenjuje se u slučaju kada računalo pobjeđuje korisnika, no tada suma neće biti jednaka trojki već jedinici.

**b) Jedan element u nizu**

Recimo da ne postoji kombinacija od dva uzastopna simbola, računalo tada odigra potez u retku, stupcu ili dijagonali gdje već postoji barem jedan njegov simbol unutar matrice ili odigra suprotni rub od korisnika. Također se i u ovom slučaju sve temelji na sumi osim što su sada drugačiji uvjeti, kao što je prikazano na slici 3.26.

```
if (suma == -1)
{
    if (i == 0)
    {

        if (matrica[0, i] == 1) { clck(true, 1, 0); return sljedeci; }
        if (matrica[1, i] == 1) { clck(true, 2, 0); return sljedeci; }
        if (matrica[2, i] == 1) { clck(true, 0, 0); return sljedeci; }
        return sljedeci;
    }
    if (i == 1)
    {
        if (matrica[0, i] == 1) { clck(true, 1, 1); return sljedeci; }
        if (matrica[1, i] == 1) { clck(true, 2, 1); return sljedeci; }
        if (matrica[2, i] == 1) { clck(true, 0, 1); return sljedeci; }
        return sljedeci;
    }
    if (i == 2)
    {
        if (matrica[0, i] == 1) { clck(true, 1, 2); return sljedeci; }
        if (matrica[1, i] == 1) { clck(true, 2, 2); return sljedeci; }
        if (matrica[2, i] == 1) { clck(true, 0, 2); return sljedeci; }
        return sljedeci;
    }
```

3.26. Kod za popunjavanje stupca u kojem je samo jedan simbol

Neka računalo opet igra kao simbol iks i neka se u prvom stupcu nalazi samo jedan iks i dva slobodna mjesta. Tada je stupac ispunjen na sljedeći način,
-1 – 1 + 1. Suma sporedne dijagonale sada iznosi -1. Uvjet ispod „*for*“ petlje na slici 3.20. je ispunjen te se popunjava jedno od praznih mjesta - prvo koje je slobodno.

Brojači, sume, ugniježđene „*for*“ petlje, prazna mjesta unutar matrice, formacije „trokuta“ i metoda „nasuprotnog ruba“ – sve to je ono što „nepobjedivi iks-oks “ čini baš takvim.

# ZAKLJUČAK

Zadatak rada bio je osmisliti 2D računalnu igru „iks-oks“. Igra je realizirana tako da korisnik uvijek igra protiv računala, a sam bira prioriteta igre – želi li igrati prvi ili će pak prvi potez prepustiti računalu. Svaki puta kada je računalo na potezu poziva se metoda „RacunaloIgra()“, ona predstavlja baš taj algoritam koji računalo čini nepobjedivim i nepogrješivim. U njoj se nalaze „upute“ prema kojima računalo bira optimalan potez. Na temelju trenutnog stanja matrice računalo „nastoji“ odigrati potez koji mu donosi višestruku mogućnost za pobjedu, a ako nije u mogućnosti pobijediti onda sprječava korisnika da ne ostvari pobjedničku kombinaciju simbola. Metoda „RacunaloIgra()“ funkcionira na temelju sume, praznih i ispunjenih mjesta u matrici te brojanju odigranih pokreta.

# LITERATURA


[1] Quora, What is Microsoft's Visual Studio?, [online], Quora Inc., dostupno na: https://www.quora.com/What-is-Microsofts-Visual-Studio, [30.06.2018]

[2] Microsoft, Getting Started; Introduction to the C# Language and the .NET Framework,[online], Microsoft Corporation, dostupno na: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharplanguage-andthe-net-framework , [25.07.2018]

[3] Techopedia, Polymorphism; Techopedia explains *Polymorphism*,[online], Technopedia Inc., dostupno na: https://www.techopedia.com/definition/3240/polymorphism-c, [27.07.2018]

[4] M. Petričević, S. Davila, C# I .NET FRAMEWORK 4.0, Algebra d.o.o., Zagreb, 2010

[5] Lucidchart, Chart Maker,[online], Lucid Software, dostupno na: https://www.lucidchart.com/documents/edit/185c4a2d-36db-4c64-aea6-bbae3b3f9c68/0, [10.09.2018]

[6] Osnove programiranja u jeziku C++, Programska petlja; Ugnježđena petlja for, [online], Google Sites, dostupno na: https://sites.google.com/site/sandasutalo/osnove-programiranja/programska-petlja/ugnjezdena-petlja-for, [28.07.2018]

## SAŽETAK

Tema završnog rada je osmisliti algoritam za nepobjedivi iks-oks u objektno orijentiranom programskom jeziku C#. Programsko okruženje koje je korišteno prilikom izrade aplikacije je Microsoft Visual Studio. Aplikacija je realizirana kao funkcijska cjelina pomoću dvije forme, koje su međusobno povezne. Igra je osmišljena na način da u njoj sudjeluju samo dva igrača. Jedan igrač uvijek je računalo, a drugi igrač uvijek je korisnik. Neovisno o tome, tko započinje igru, rezultat igre uvijek mora biti u korist računala ili barem neriješeno. Računalu je implementirana umjetna inteligencija. Ona mu omogućava da „odlučuje“ koje poteze igra prilikom igre, pomoću implementiranih uvjeta i unaprijed predviđenim potezima.



Ključne riječi: križić-kružić, teorija igara, algoritmi



## ABSTRACT

The topic of my final assignment was to come up with an algorithm for unbeatable game of tic tac toe and to write it as an object-oriented program in C#. Programming environment that is used in the making of application is Microsoft Visual Studio. The application is made into functional ensemble by using two mutually connected forms. The game is designed for two players, one of which is always the computer and another one is the user. Despite of who plays first, the ending result must always be in favour of the computer, or the game can be draw, however the user can never win. The computer is implemented with an artificial intelligence that enables it to, with a help of implemented conditions and predetermined anticipated moves, "decide" it's next move during the game.



Key words: Tic-Tac-Toe, game theory, algorithms

# ŽIVOTOPIS

Barbara Matijević, rođena sam 07. rujna 1996. u Osijeku. Započinjem obrazovanje u „OŠ Gorjani, područna škola Tomašanci“. Nakon osnovnoškolskog obrazovanja upisujem Prirodoslovno-matematičku gimnaziju Osijek i uspješno ju završavam 2015 . Iste godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, smjer računarstvo. Također, imam kratko radno iskustvo kao tehnička podrška u Hrvatskom Telekomu.

Interesiraju me znanstvena područja poput logike, kombinatorike, digitalna elektronike i programiranja. Voljela bih u životu raditi posao koji je kombinacija mojih interesa. Osim poslovnih i fakultetskih obveza, volim volontirati.

# PRILOZI