

Прва београдска гимназија

Цара Душана 61, Београд

Матурски рад из програмирања

Mastermind

Ментор:

Милош Пушић, професор

Ученик:

Душан Стојанов, IV-9

Прва београдска гимназика, јун 2025. год.

Садржај

1. Увод.....	3
Основни елементи игре:.....	3
Ток игре:.....	4
Историја игре Mastermind.....	5
2. Почетак програма.....	6
Две верзије игре:.....	6
3. Компјутер замишља, играч погађа.....	9
Битније променљиве су:.....	9
4. Играч замисли, компјутер погађа.....	14
Доналд Кнут:.....	16
5. Закључак.....	19

Mastermind

Овај матурски рад описује процес дизајнирања и имплементације компјутерске верзије популарне игре *Mastermind*. Игра је прављена у програму *Visual Studio*, а пројекат је типа *Windows Forms*.

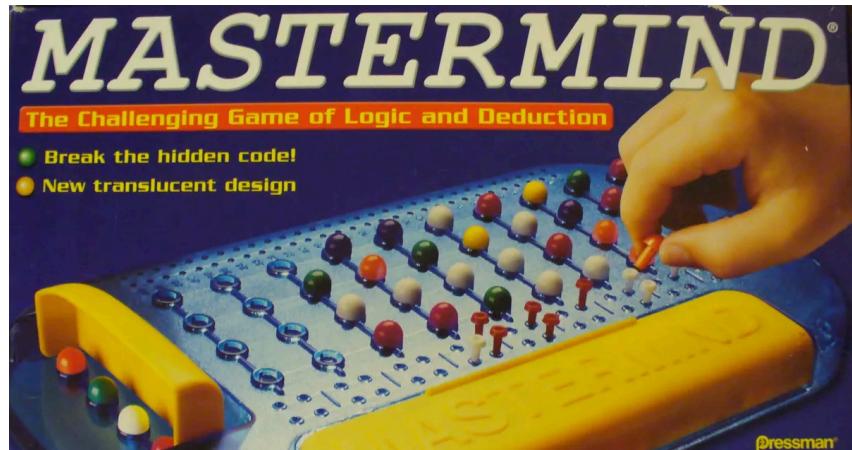
Mastermind је логичка игра за два играча, у којој један играч замишља тајну комбинацију боја, док други играч треба да је погоди.

1. Увод

Основни елементи игре:

- **Табла за играње:** Најчешће се састоји од 7 редова са по 4 рупице, од којих је 1 ред сакривен и користи га играч који смишља тајну комбинацију, а 6 за играча који погађа.
- **Чунићи (клинови) у боји:** Са њима постављач боја замишља комбинацију, док их други играч користи да погоди тајну комбинацију. Црне и беле чуниће постављач боје користи да другом играчу каже колико је погодио са својом комбинацијом
- **Тајна комбинација:** Састоји се од 4 боје. Постављач кода ставља чуниће у замишљеном редоследу у свој сакривени ред.
- **Поља за погађање:** Ова поља играч који погадђа користи да унесе чуниће за које он мисли да представљају замишљену комбинацију. Након уношења комбинације, играч који замишља тајни код му даје одговор са црним и белим чунићима које ставља у поља са стране. Црни представљају бој погођених боја, а бели представљају боје које има у тајној комбинацији, али нису на доброј позицији.





Ток игре:

- **Замишљање кода:** Постављач кода замишља комбинацију и ставља чуниће у свој ред.
- **Погађање кода:** Погађач кода уноси комбинацију своју комбинацију
- **Добијање повратних информација:** Постављач кода другом играчу даје одговор у виду црних и белих чунића.
- **Следећи корак:** Погађач кода, имајући у виду одговор на прошлу комбинацију, смишља нову комбинацију.
- **Остатак игре:** Погађач кода има најчешће 6 покушаја да погоди комбинацију.
- **Крај игре:** Погађач кода побеђује ако успе да погоди тајну комбинацију, а постављач кода побеђује ако постављач не успе да је погоди.

Игра *Mastermind* је занимљива зато што захтева логичко закључивање, стратешко размишљање, дедукцију, препознавање образца и много стрпљења.

Игра може имати варијације у боји, у дужини и броју редова, да ли је дозвољено понављање боја, све што утиче на њену комплексност. Поставка је једноставна, а има пуно простора за стратешко размишљање.

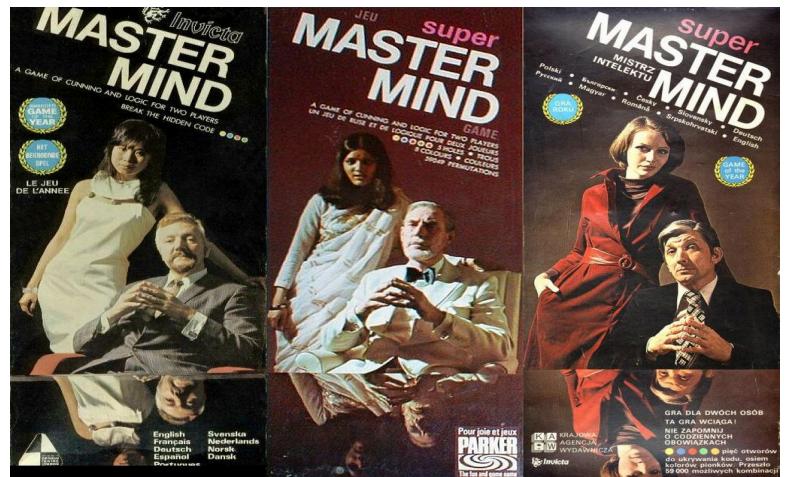
Историја игре *Mastermind*

Игра *Mastermind* је настала у Израелу 1970. године. Направио ју је Мордекај Мејровиц (Mordecai Meirowitz), израелски управник поште и експерт у телекомуникацији. Слична је игри Бикови и краве, која је настала пре једног века или више. Та игра се играла са цифрама, са папиром и оловком, и за сваку погођену цифру на добром месту би играч добијао “бика,” а за сваку погођену цифру на лошем месту “краву.” Мејровиц је ову игру пренео са бројева на боје.

Мејровиц је своју игру представио великим компанијама, али је у почетку био одбијан. Игра је представљена на међународном сајму играчака у Нирнбергу 1970. или 1971. године. Ту је игру запазила британска компанија Инвикта Пластикс (Invicta Plastics).

Компанија је купила игру и почела да је производи под именом “Master Mind.” Дизајн игре, са карактеристичном таблом и шареним чунићима, постао је иконичан. Кутија је често приактивала слику елегантно обученог мушкарца (понекад са женом поред себе), што је допринело у стварању слике игре.

Игра је постала позната 1970-их година и продала око 50 милиона примерака. У Великој Британији је 1973. године освојила награду за најбољу игру године, што је додатно учврстило њену популарност.



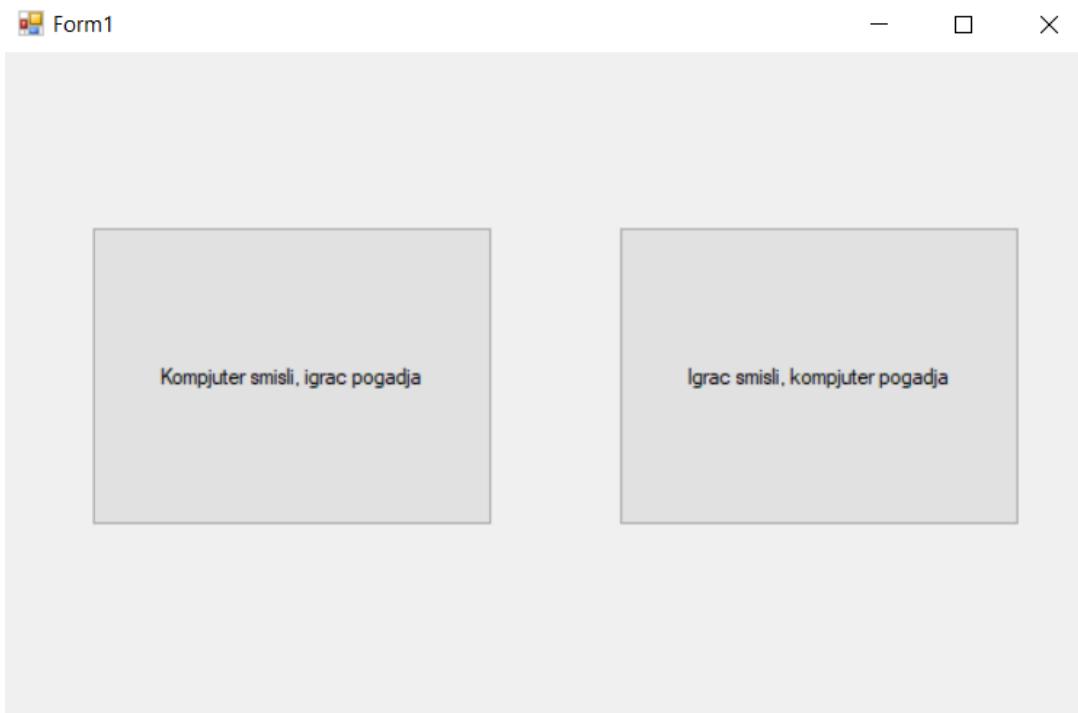
2. Почетак програма

Од овог поглавља креће објашњење компјутерске верзије игре *Mastermind*. Почетак програма садржи почетни прозор, у ком корисник бира коју верзију игре ће да игра.

Две верзије игре:

- Дугме које покреће игру у којој компјутер замисли комбинацију, а играч мора да је погоди
- Дугме које покреће игру у којој играч замисли комбинацију, а компјутер мора да је ппогоди. За ову верзију се користи Доналд Кнутов (Donald Knuth) алгоритам за погађање тајног кода.

Када корисник покрене апликацију, прво се отвори следећи прозор:



Овде корисник може да бира између два типа игра. Кликом на лево дугме отвара се игра у којој компјутер смисли комбинацију и играч треба да је погоди.

```
1 reference
private void btnIgracPogadja_Click(object sender, EventArgs e)
{
    IgracPogadja igracPogadja = new IgracPogadja("covek");
    igracPogadja.ShowDialog();
}
```

Кликом на десно дугме се отвори игра у којој играч за себе смисли комбинацију, а компјтер треба да је погоди (користећи Кнутов алгоритам).

```
1 reference
private void btnKompjuterPogadja_Click(object sender, EventArgs e)
{
    IgracPogadja igracPogadja = new IgracPogadja("kompjuter");
    igracPogadja.ShowDialog();
}
```

Оба дугмета отварају исту форму, али је разлика у **конструктору**. У првој слици се прослеђује стринг “covek,” а у другој стринг “kompjuter.” Тај стринг говори форми ко ће да погадђа комбинацију и које контроле треба да упали, угаси или промени.

У обе верзије игре се користи **Custom Control**-а **Ред**. **Custom Control**-а је контрола коју програмер може да направи сам. Користе се када постојеће контроле не могу ураде оно што нам је потребно, или када нам се иста група контрола више пута јавља у програму.



Контрола **Ред** у себи садржи **4 PictureBox-а** и **1 TextBox**. Користи се зато што, без ње, програм садржи превише слика и дугмића. PictureBox-еви се користе да би у себи садржали боје које играч унесе, а TextBox да би у њега написали колико има тачних боја на добром месту и колико има тачних боја на погрешном месту.

Ред од променљивих у себи садржи низ од 4 интиџера назван **indexBoja** и низ од 4 PictureBox-ева назван **слике**. Свака боја је означена индексом од 0 до 5, и indexBoja у себи чува индекс боја у свакој од 4 слици. Низ од 4 PictureBox-ева у себи чува 4 PictureBox-а који су стављени у Design-у Реда. У конструктору се изврши поставка ових променљивих.

```
public int[] indexBoja = new int[4];
public PictureBox[] slike = new PictureBox[4];
```

```
7 references
public Red()
{
    InitializeComponent();

    slike[0] = boja1;
    slike[1] = boja2;
    slike[2] = boja3;
    slike[3] = boja4;

    for (int i = 0; i < 4; i++)
    {
        indexBoja[i] = 7;
    }
}
```

Чак иако су индекси боја од 0 до 5, поставе се на 7 да би се низови попунили. Кликом на неки од PictureBox-ева се слика која је у њима избрисана, ако ред у ком се слике налазе одговара тренутном потезу играча.

```
1 reference
private void boja1_Click(object sender, EventArgs e)
{
    IgracPogadja.kolona = 0;
    indexBoja[0] = 7;
    boja1.BackColor = DefaultBackColor;
}
```

3. Компјутер замишља, играч погађа

Кликом на лево дугме у почетном прозору се отвара прозор у ком играч треба да погоди тајну комбинацију коју је компјутер замислио. Састоји се из:

- Дугмића који представљају 6 **боја** које играч може да унесе.
- Дугмета “**Потврди**” којим компјутеру прослеђујемо унету комбинацију да је провери и да нам да одговор колико има тачних боја на добром месту и тачних боја

на покрешном месту. Након што пртиснемо ово дугме, више не можемо да мењамо унету комбинацију.

- **7 редова** од којих 6 служе да у њих унесемо боје, а седми је ту да открије тајну комбинацију на крају игре.

Битније променљиве су:

- **Редови:** Представља низ од 7 редова (*Custom Control-a*). Првих 6 редова се користи за уношење боја, а седми се користи за показивање тајног решења.
- **Ред:** Нема везе са *Custom Control-om*. Променљива је интицер који показује до ког реда су низу **Редови** смо стигли. Повећава се кликом на дугме за потврђивање унете комбинације.
- **Колона:** Интицер који означава до које слике у једном реду смо стигли. Повећава се кликом на дугмиће за стављање боја, а када обришемо неку боју, постаје индекс првог слободног поља у реду.

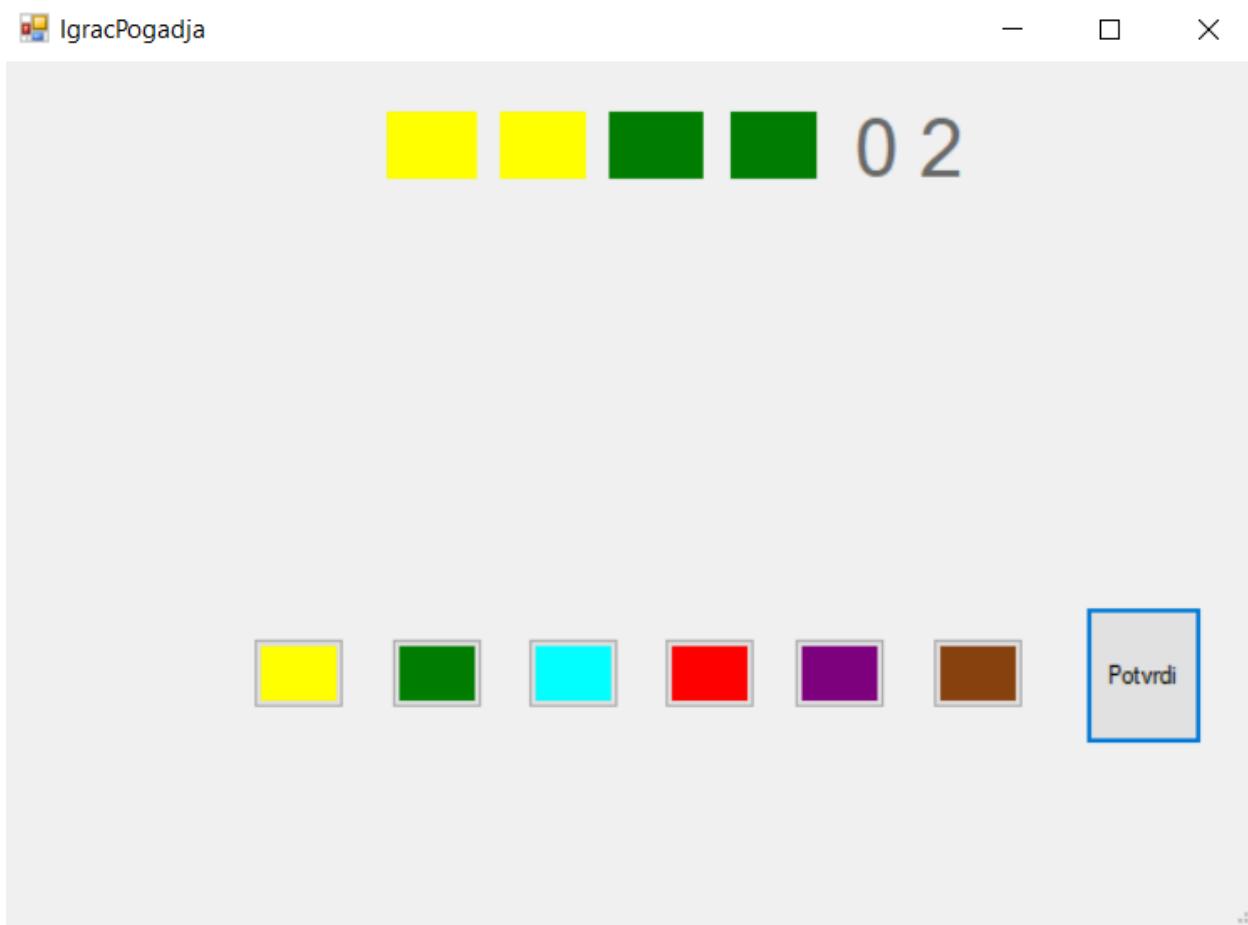
```
10 references
public partial class IgracPogadja : Form
{
    int IgracevoResenje = 0;
    public bool nastavi;
    public int[,] RezultatKompjutera = new int[6, 4];
    public int[] drzac = Data.PocniKompjuterSredjuje();

    public Red[] redovi = new Red[7];
    Red resenje;
    int red = 0;
    public static int kolona = 0;
    string rezultat;

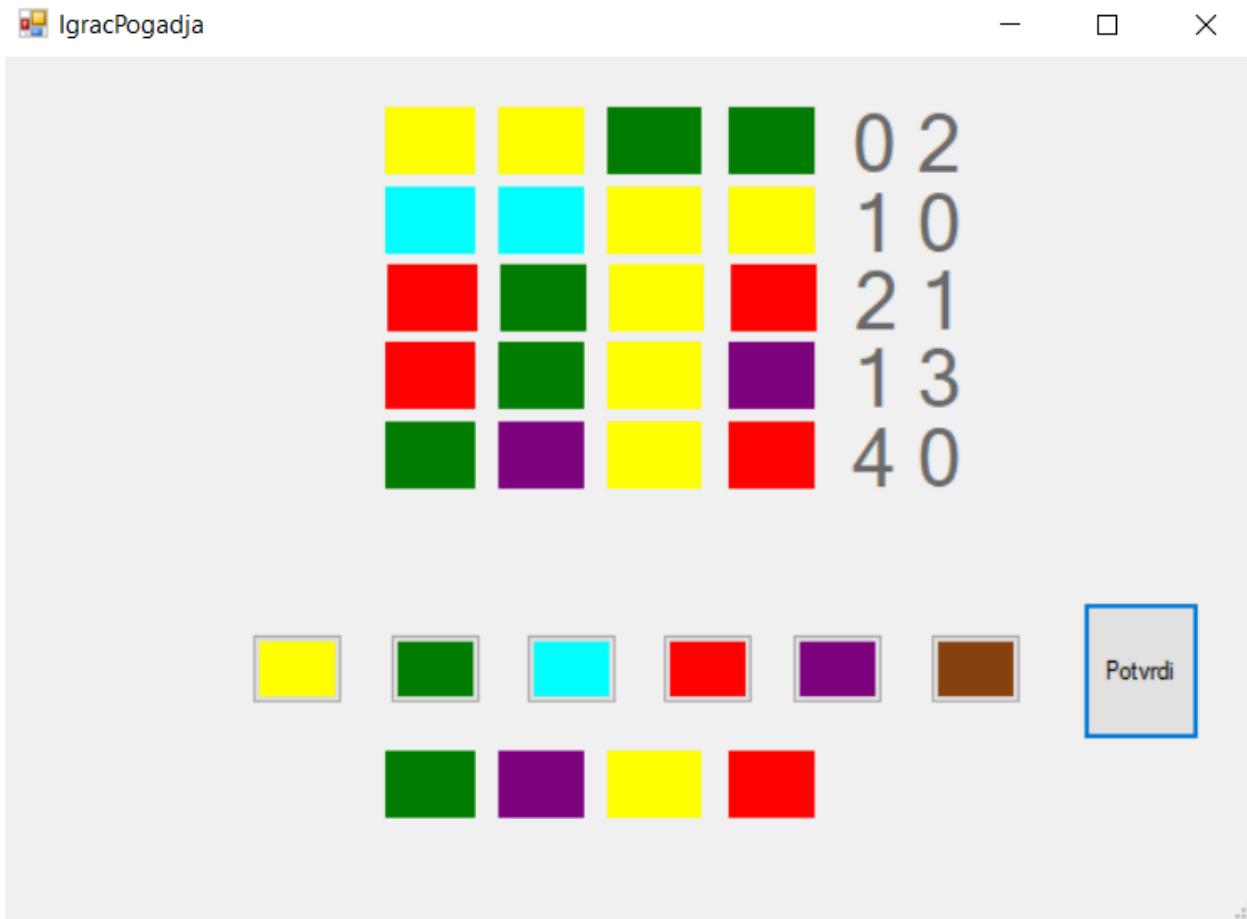
    Label[] odgovori = new Label[6];

    public string player;
```

На почетку игре се виде само дугмићи за стављање боја и дугме за потврђивање унете комбинације. Након уношења жељене комбинације и притискања дугмета за потврђивање, компјутер нам врати бројеве који означавају колико боја смо погодили. Леви број означава колико има тачних боја на тачном месту, а десно колико има тачних боја на погрешном месту.



Притискање дугмета за потврђивање повећава интиџер **ред** за 1 и тако куцамо нашу следећу комбинацију. Игра се наставља све док играч не погоди тајну комбинацију, или, у случају да играч не успе да погоди, док не прође 6 покушаја.



Кликом на дугмиће за постављање боја се извршава следећи код:

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    namestiKolonu();
    redovi[red].StaviBoju(0, kolona);
}
```

Процедура namestiKolonu() проверава колика треба да буде вредност иницијера колона тако што гледа где се налази прва слободна слика у реду.

```
6 references
public void namestiKolonu()
{
    for (int i = 0; i < 4; i++)
    {
        if (redovi[red].slike[i].BackColor == DefaultBackColor)
        {
            kolona = i;
            return;
        }
    }
}
```

Процедура StaviBoju(int boja, int kolona) се налази у класи Ред. Она проверава који смо иницијер проследили за боју и ставља одговарајућу боју у поље које смо му рекли да стави иницијером колона.

```
9 references
public void StaviBoju(int boja, int kolona)
{
    if (boja == 0) slike[kolona].BackColor = Color.Yellow;
    if (boja == 1) slike[kolona].BackColor = Color.Green;
    if (boja == 2) slike[kolona].BackColor = Color.Cyan;
    if (boja == 3) slike[kolona].BackColor = Color.Red;
    if (boja == 4) slike[kolona].BackColor = Color.Purple;
    if (boja == 5) slike[kolona].BackColor = Color.SaddleBrown;

    indexBoja[kolona] = boja;
    /* zuta,
    zelena,
    plava,
    crvena,
    ljubicasta,
    braon,*/
}
```

Боје се бришу тако што се притисне на њих. То позове процедуру у класи **Ред**.

```
1 reference
private void boja1_Click(object sender, EventArgs e)
{
    IgracPogadja.kolona = 0;
    indexBoja[0] = 7;
    boja1.BackColor = DefaultBackColor;
}
```

За проверавање колико је тачна унета комбинација боја, користи се функција **ProveriResenje(int[] niz, int[] rez)** која се налази у класи **Data**. Она враћа структуру **resenje**, која у себи садржи 2 интицера: тачно и скоро. Тачно представља број тачних боја на тачном месту, а скоро број тачних на погрешном месту.

Функцији **ProveriResenje(int[] niz, int[] rez)** прослеђујемо унету комбинацију (низ) и тајну комбинацију (рез). Она ради тако што:

1. Упореди боје из низа и из решења са истих позиција. Ако су боје исте, означи те две позиције да су искоришћене и повећа интицер који означава број тачних на тачним позицијама за 1.

```
for (int i = 0; i < 4; i++)
{
    if (niz[i] == rez[i])
    {
        rezultat.tacno += 1;
        NizUsed[i] = true;
        RezUsed[i] = true;
    }
}
```

2. Након што добијемо број тачних на доброј позицији, упоредимо сваку боју из низа са сваком бојом из решења. Ако су боје исте, и обе боје нису означене да су искоришћене, означимо да су искоришћене и повећамо интиџер који означава број тачних боја на погрешним местима за 1.

```
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 4; j++)
    {
        if ((niz[i] == rez[j]) && NizUsed[i] == false && RezUsed[j] == false)
        {
            rezultat.skoro += 1;
            NizUsed[i] = true;
            RezUsed[j] = true;
        }
    }
}
```

4. Играч замисли, компјутер погађа

Кликом на десно дугме у почетном прозору се отвара прозор у ком компјутер треба да погоди тајну комбинацију коју је играч замислио. Чак иако отвара исту форму, због другацијег стригна који је прослеђен конструктору неке контроле ће бити другачије.

У овој верзији игре, играч сам замисли комбинацију и никде је не да компјутеру. Компјутер може да погоди шта је тачна комбинације само користећи одговоре које му играч даје у виду броја тачних боја. Форма се састоји из:

- Дугмића за повезавање и смањивање броја који представља број тачних боја на тачним и на погрешним позицијама. Компјутеру прослеђујемо интиџер чије десетице представљају број тачних на добрим позицијама и јединице представљају број тачних на погрешним позицијама
- Дугмета за потврђивања броја који се даје компјутеру. Када притиснемо ово дугме, компјутер нам врати своју следећу комбинацију.

Компјутер на почетку сваке партије прво почне са комбинацијом где су прве две и друге две боје исте.

IgracPogadja

- □ ×

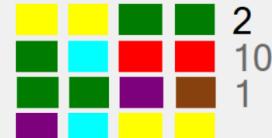


Potvrdi
kombinaciju



Рецимо да смо ми замислили комбинацију
ЛУБИЧАСТО ПЛАВО ЗУТО ЗУТО.
Та партија би изгледала овако

IgracPogadja

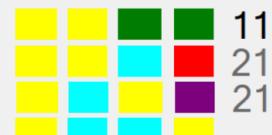


Potvrdi
kombinaciju



Сада замислимо комбинацију ЗУТО ПЛАВО
ПЛАВО ЗУТО.

За погађање тајне комбинације користи се
Доналд Кнutoов алгоритам



Potvrdi
kombinaciju



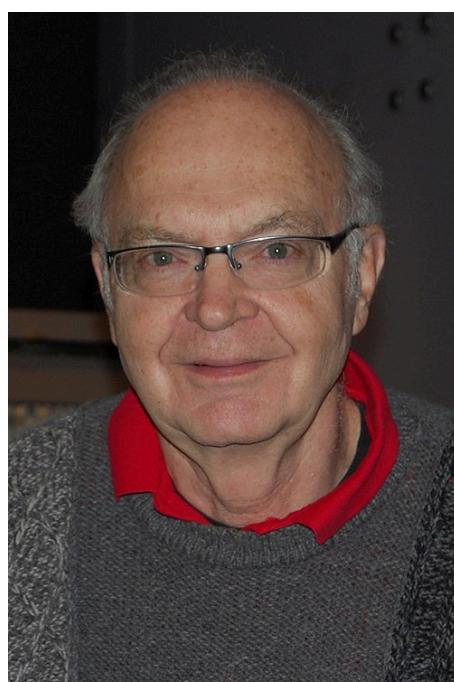
Доналд Кнут:

Доналд Ервин Кнут (Donald Ervin Knuth) је амерички математичар, информатичар и професор. Рођен је 1938. године у Милвокију (Milwaukee). Сматра се једним од најзначајнијих личности у историји рачунарства и често га називају “оцем анализе алгоритама” због његовог доприноса развоју и систематизацији математичких техника за анализу сложености рачунарских алгоритама.

Његова највећа дела су:

- Књиге “*The Art of Computer Programming*” (Уметност рачунарског програмирања) је серија књига која се сматра најдетаљнијом анализом алгоритама и структура података. Иако још увек није завршена, сматра се јако битном у области рачунарске науке.
- *TeX* и *METAFONT*: *TeX* је систем за припремање текста, посебно погодан за слагање сложених математичких и научних текстова. *METAFONT* је систем за дизајнирање фонтова који ради са *TeX*-ом.
- Смислио је разне алгоритме који се користе широм света, нпр. заједно са Џејмсом Х. Морисом и Воном Пратом је смислио Кнут-Морис-Пратов алгоритам за проналажење једног мањен стринга у неком другом, већем стрингу.

Доналд Кнут је такође смислио алгоритам за добијање тајног кода у игри *Mastermind*. Он је на класичан алгоритам додао један део који нам омогућава да увек добијемо тачно решење у 5 или мање покушаја.



Кнутов алгоритам је у коду откуцан у класи **Data**. Ради на следећи начин:

1. Прво генерише низ са свим могућим комбинацијама од 4 боје. Укупно их има 1296. Поред главног низа, у мом коду генерише и остале потребне листе.

```
for (int i = 0; i < 6; i++)
{
    for (int j = 0; j < 6; j++)
    {
        for (int r = 0; r < 6; r++)
        {
            for (int t = 0; t < 6; t++)
            {
                skup.Add(new int[] { i, j, r, t });
                skupSvihResenja.Add(new int[] { i, j, r, t });
                brojKodova.Add(new int[14]);
                najveciBrojKodova.Add(0);
            }
        }
    }
}
```

2. Након генерисања потребних листи, компјутер унесе своју прву комбинацију. Једна од најбољих почетних комбинација је комбинација где су прве две и друге две боје исте. Процедура *sredi()* избацује унету комбинацију из низа.
3. Након тога, компјутер узме комбинацију коју је унео и упореди је са скупом свих могућих потенцијалних комбинација. Ако нека комбинација када се упореди са унетом даје исто решење које је унета добила када је била унесена, комбинација остаје у скупу, иначе је избачена.

```
for (int i = 0; i < skup.Count; i++)
{
    privkod = ProveriResenje(trenutni, skup[i]);

    if ((privkod.tacno != kod.tacno) || (privkod.skoro != kod.skoro))
    {
        skup.RemoveAt(i);
        i--;
    }
}
```

4. Када се прође кроз скуп могућих комбинација, онда се упореди свака могућа комбинација (свих 1296) са сваком из скупа могућих кодова. За свих 1296 кодова се забележи које од 14 могућих решења (00,01,02,03,04,10,11,12,13,20,21,22,30,40) је добио и колико пута.

```

for (int i = 0; i < skupSvihResenja.Count; i++)
{
    for (int j = 0; j < skup.Count; j++)
    {
        resenje priv = ProveriResenje(skupSvihResenja[i], skup[j]);

        if (priv.tacno == 0 && priv.skoro == 0) brojKodova[i][0]++;
        if (priv.tacno == 0 && priv.skoro == 1) brojKodova[i][1]++;
        if (priv.tacno == 0 && priv.skoro == 2) brojKodova[i][2]++;
        if (priv.tacno == 0 && priv.skoro == 3) brojKodova[i][3]++;
        if (priv.tacno == 0 && priv.skoro == 4) brojKodova[i][4]++;
        if (priv.tacno == 1 && priv.skoro == 0) brojKodova[i][5]++;
        if (priv.tacno == 1 && priv.skoro == 1) brojKodova[i][6]++;
        if (priv.tacno == 1 && priv.skoro == 2) brojKodova[i][7]++;
        if (priv.tacno == 1 && priv.skoro == 3) brojKodova[i][8]++;
        if (priv.tacno == 2 && priv.skoro == 0) brojKodova[i][9]++;
        if (priv.tacno == 2 && priv.skoro == 1) brojKodova[i][10]++;
        if (priv.tacno == 2 && priv.skoro == 2) brojKodova[i][11]++;
        if (priv.tacno == 3 && priv.skoro == 0) brojKodova[i][12]++;
        if (priv.tacno == 4 && priv.skoro == 0) brojKodova[i][13]++;
    }
}

```

5. За сваки од 1296 кодова се изабере оно решење које се појавило највише пута. Овај код чије је решење које се појавило **највише** пута **најмање** се узима за следећи код који компјутер уноси. Након овога се понављају кораци 3, 4 и 5 све док се не дође до тачне комбинације.

```

for (int i = 0; i < skupSvihResenja.Count; i++)
{
    int max = 0;
    for (int j = 0; j < 14; j++)
    {
        if (brojKodova[i][j] > max)
        {
            max = brojKodova[i][j];
        }
    }
    najveciBrojKodova[i] = max;
}

```

```

int min = najveciBrojKodova.Min();
for (int i = 0; i < najveciBrojKodova.Count; i++)
{
    if (najveciBrojKodova[i] == min)
    {
        indexi.Add(i);
    }
}

```

```
bool flag = false;
for (int i = 0; i < indexi.Count; i++)
{
    for (int j = 0; j < skup.Count; j++)
    {
        if (skupSvihResenja[indexi[i]].SequenceEqual(skup[j]))
        {
            trenutni = skup[j];
            flag = true;
            break;
        }

        if (flag == true)
        {
            break;
        }
    }
}
```

Обичани алгоритам би, након што је проверио код који је унео са скупом могућих кодова, узео насумично нови код и наставио да ради. Кнутов алгоритам је посебан зато што, уместо да узме код насумично, он узме код који највише смањује број потребних покушаја и због тога може да погоди сваки пут у 5 или мање покушаја.

5. Закључак

Кроз овај матурски рад детаљно је анализирана игра *Mastermind*. У почетку смо видели нешто о историји и правилима игре, а централни део је посвећен компјутерској верзији игре

Основни циљ апликације је успешан. Корисник и компјутер могу да заједно играју друштвену игру *Mastermind*. Током писања кода, највише је пажње посвећено Кнутовом алгоритму за тражење тачног кода.

Процес израде програма је био интересантно и драгоцено искуство. Изучавање како Кнутов алогирам ради је био незгодан, али врло занимљиво искуство. Научио сам и да користим *Custom Control-e*, које су значајно унапредила функционалност целе апликације и значајно олакшали њену израду.

Иако је програм функционалан, и даље има простора за унапређивање. Больни *UI*, простије и боље објашњене контроле, опција да се мења број редова, колона и боја су само од неких могућих унапређења за ову апликацију. Али поред свега тога, пројекат је заокружен и током израдње је стечено значајно искуство, и у писању кода и у шисању овог документа, које ћи сигурно у будућности значити.

