**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 6 з дисципліни

«Проектування алгоритмів»

„**Пошук в умовах протидії, ігри з елементом випадковості, ігри з неповною інформацією**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*Спаських Микола Дмитрович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.Н.*

Київ 2021

Зміст

[1 Мета лабораторної роботи 3](#_Toc86770239)

[2 Завдання 4](#_Toc86770240)

[3 Виконання 6](#_Toc86770241)

[3.1 Програмна реалізація алгоритму 6](#_Toc86770242)

[3.1.1 Вихідний код 6](#_Toc86770243)

[3.1.2 Приклади роботи 6](#_Toc86770244)

[Висновок 7](#_Toc86770245)

[Критерії оцінювання 8](#_Toc86770246)

# Мета лабораторної роботи

Мета роботи - вивчити основні підходи до формалізації алгоритмів знаходження рішень задач в умовах протидії. Ознайомитися з підходами до програмування алгоритмів штучного інтелекту в іграх з елементами випадковості та в іграх з неповною інформацією.

# Завдання

Для ігор з елементами випадковості, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм мінімакс.

Для карткових ігор, згідно варіанту (таблиця 2.1), реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Потрібно реалізувати стратегію комп'ютерного опонента, і звести гру до гри з повною інформацією (див. Лекцію), далі реалізувати стратегію гри комп'ютерного опонента за допомогою алгоритму мінімаксу або альфа-бета-відсікань.

Реалізувати анімацію процесу жеребкування (+1 бал) або реалізувати анімацію процесу роздачі карт і анімацію ходів (+1 бал).

Реалізувати варто тільки одне з бонусних завдань.

Зробити узагальнений висновок лабораторної роботи.

Таблиця 2.1 – Варіанти

|  |  |
| --- | --- |
| **№** | **Варіант** |
| 1 | Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html |
| 2 | Лудо http://www.iggamecenter.com/info/ru/ludo.html |
| 3 | Генерал http://www.rules.net.ru/kost.php?id=7 |
| 4 | Свиня http://www.rules.net.ru/kost.php?id=3 |
| 5 | Тринадцять http://www.rules.net.ru/kost.php?id=16 |
| 6 | Индійські кості http://www.rules.net.ru/kost.php?id=9 |
| 7 | Закрити сектор http://gamerules.ru/igry-v-kosti-part1#zakroi-sektor |
| 8 | Двадцять одне http://gamerules.ru/igry-v-kosti-part8#dvadtsat-odno |
| 9 | Бакси http://gamerules.ru/igry-v-kosti-part10#baksy |
| 10 | Подвійний генерал http://gamerules.ru/igry-v-kosti-part6#dvoinoi-general |
| 11 | 101 https://www.durbetsel.ru/2\_101.htm |
| 12 | Ночка http://life-games.net/index/nochka/0-192 |
| 13 | Табу https://www.durbetsel.ru/2\_taboo.htm |
| 14 | Чотири рядки https://www.durbetsel.ru/2\_chetyre\_stroki.htm |
| 15 | Свої козирі https://www.durbetsel.ru/2\_svoi-koziri.htm |
| 16 | Війна з ботами https://www.durbetsel.ru/2\_voina\_s\_botami.htm |
| 17 | Вибух https://www.durbetsel.ru/2\_vzryv.htm |
| 18 | Останній гравець https://www.durbetsel.ru/2\_posledny\_igrok.htm |
| 19 | Скарбнички https://www.durbetsel.ru/2\_sunduchki.htm |
| 20 | Богач https://www.durbetsel.ru/2\_bogach.htm |
| 21 | Редуду https://www.durbetsel.ru/2\_redudu.htm |
| 22 | Эльферн https://www.durbetsel.ru/2\_elfern.htm |
| 23 | Ремінь https://www.durbetsel.ru/2\_remen.htm |
| 24 | Курка https://www.durbetsel.ru/2\_kurisa.htm |
| 25 | Навалка https://www.durbetsel.ru/2\_navalka.htm |
| 26 | Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html |
| 27 | Лудо http://www.iggamecenter.com/info/ru/ludo.html |
| 28 | Генерал http://www.rules.net.ru/kost.php?id=7 |
| 29 | Свиня http://www.rules.net.ru/kost.php?id=3 |
| 30 | Тринадцять http://www.rules.net.ru/kost.php?id=16 |

В мене 18 варіант

# Последний игрок

Количество колод: 1  
Количество карт в колоде: 52 и 2 джокера  
Количество игроков: 2 - 6  
Старшинство карт: 2, 3, 4, 5, 6, 7, 8, 9, 10, В, Д, К, Т.  
Цель игры: набрать наименьшее количество штрафных очков.  
Правила игры. Считается, что данная игра появилась в тюрьме Уолла-Уолла, в штате Вашингтон, в США в 1980 году, хотя первые упоминания об этой игре звучали в 1970-х годах в штате Мэн, в США. Первый сдатчик в игре определяется по жребию, в следующих играх игроки сдают карты по очереди по часовой стрелке. Сдатчик тщательно тасует карты и сдает каждому игроку от четырех до восьми карт, количество сдаваемых карт игроки устанавливают до игры, оставшиеся карты кладутся по центру стола в закрытом виде в стопку и служат картами запаса. Первый ход в игре принадлежит игроку слева от сдатчика, игрок может положить свою любую карту в открытом виде по центру стола. Эта карта служит началом сброса карт. Далее ход переходит к следующему игроку по часовой стрелке, который должен положить карту такого же значения или такой же масти, если игроки играют с картами запаса, то игрок если у него нет карты для хода, берет одну карту из запаса и ход переходит к следующему игроку по часовой стрелке. В том случае, если у игрока нет карты для хода, то ход все равно переходит к следующему игроку по часовой стрелке. Некоторые карты предполагают наличие определенных действий от игроков.

|  |  |
| --- | --- |
| 2 | Следующий игрок, должен взять карту из карт запаса и пропустить ход. |
| 3 | Если игрок играет тройкой, то он может положить на нее еще одну свою любую карту. |
| 4 | Каждый следующий игрок по часовой стрелке должен положить карту такой же масти, но на одно значение старше. Например, следующий игрок должен положить 5 такой же масти, а следующий игрок должен положить 6 такой же масти и т.д. Если у какого-либо игрока нет такой карты, то он должен взять из карт запаса столько карт, сколько значение верхней карты сброса. |
| 5, 6, 7 | Разрешается класть четверку. |
| 8 | Разрешается класть на любую карту и игрок выложивший восьмерку может объявить новую масть. |
| Валет | Следующий игрок по часовой стрелке пропускает свой ход. |
| Туз | Направление игры меняется на противоположное. |

Джокер может заменять любую карту на выбор игрока. Игрок, у которого осталась одна карта на руках, должен сказать "последняя карта". Если игрок этого не говорит, то он должен взять одну карту из запаса. Таким образом, игра идет до тех пор, пока не останется одного игрока без карт. Оставшиеся игроки с картами на руках, подсчитывают штрафные очки за них. Стоимость карт в штрафных очках: король, дама, валет и 10 - стоят 10 очков; туз - 15 очков; 8 - 25 очков; джокер - 40 очков. Любой из игроков, который достигнет определенного количества баллов или превысит это количество баллов, выбывает из игры. Последний игрок, который останется, становится победителем.

# Виконання

## Програмна реалізація алгоритму

### Вихідний код

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6

{

class karta

{

public int nominal;

public int mast;

public string nazva;

public int tsinnist;

public bool joker;

public karta(int mast\_, int nominal\_)

{

mast = mast\_;

nominal = nominal\_;

gen\_nazvi();

if (nominal == 15) joker = true;

else joker = false;

}

public void gen\_nazvi()

{

if ((nominal > 1) && (nominal < 11))

{

nazva += nominal;

}

else

switch (nominal)

{

case 11:

nazva = "В";

break;

case 12:

nazva = "Д";

break;

case 13:

nazva = "К";

break;

case 14:

nazva = "Т";

break;

case 15:

nazva = "Дж";

break;

}

switch (mast)

{

case 1:

nazva += "♥"; // '\u2665'

break;

case 2:

nazva += "♦";

break;

case 3:

nazva += "♣";

break;

case 4:

nazva += "♠";

break;

}

}

public void vivid()

{

Console.Write(nazva + " ");

}

public void rozstanovka\_tsinnostey(int kilkist\_kart\_gravtsa)

{

switch (nominal)

{

case 2: tsinnist = 1;

break;

case 3:

tsinnist = 2;

break;

case 4:

tsinnist = 4;

break;

case 5:

tsinnist = 3;

break;

case 6:

tsinnist = 5;

break;

case 7:

tsinnist = 5;

break;

case 8:

tsinnist = 6;

break;

case 9:

tsinnist = 5;

break;

case 10:

tsinnist = 7;

break;

case 11:

tsinnist = 8;

break;

case 12:

tsinnist = 7;

break;

case 13:

tsinnist = 7;

break;

case 14:

tsinnist = 8;

break;

case 15:

tsinnist = 4;

break;

}

if (kilkist\_kart\_gravtsa <= 2)

{

switch (nominal)

{

case 15:

tsinnist = 10;

break;

case 8:

tsinnist = 9;

break;

}

}

}

}

class gra

{

public int skladnist;

public List<karta> koloda;

public List<karta> ruka\_kompa;

public List<karta> ruka\_gravtsa;

public List<int> spetsefekti = new List<int> {2,3,4,8,11,15};

public karta karta\_na\_stoli;

public bool osoblivi\_pravila=false;

public bool end\_game = false;

public int ochki\_gravtsa = 0;

public int ochki\_compa = 0;

public gra()

{

ochki\_compa = 0;

ochki\_gravtsa = 0;

vibir\_skladnosti();

while (ochki\_gravtsa<30 && ochki\_compa<30)

{

osoblivi\_pravila = false;

generuem\_kolodu();

pochatkova\_rozdacha();

karta\_na\_stoli = null;

end\_game = false;

while (!end\_game)

{

hid\_gravtsa();

hid\_kompa();

}

pidrahunok\_ochkiv();

Console.WriteLine($"результат гри: очки гравця {ochki\_gravtsa}, очки компа {ochki\_compa}");

}

Console.WriteLine("кінець гри");

}

public void vibir\_skladnosti()

{

Console.Write("Вибери складність (введи число, скільки карт компа закрито від тебе) 0-10:");

skladnist = int.Parse(Console.ReadLine());

}

public void pidrahunok\_ochkiv()

{

for (int i = 0; i < ruka\_gravtsa.Count; i++)

{

if (ruka\_gravtsa[i].nominal >= 10 && ruka\_gravtsa[i].nominal <= 13) ochki\_gravtsa += 10;

if (ruka\_gravtsa[i].nominal == 14 ) ochki\_gravtsa += 15;

if (ruka\_gravtsa[i].nominal == 8 ) ochki\_gravtsa += 25;

if (ruka\_gravtsa[i].nominal == 15 ) ochki\_gravtsa += 40;

}

for (int i = 0; i < ruka\_kompa.Count; i++)

{

if (ruka\_kompa[i].nominal >= 10 && ruka\_kompa[i].nominal <= 13) ochki\_compa += 10;

if (ruka\_kompa[i].nominal == 14) ochki\_compa += 15;

if (ruka\_kompa[i].nominal == 8) ochki\_compa += 25;

if (ruka\_kompa[i].nominal == 15) ochki\_compa += 40;

}

}

public void generuem\_kolodu()

{

koloda = new List<karta>();

for (int i = 1; i <= 4; i++)

{

for (int j = 2; j <= 14; j++)

{

karta k1 = new karta(i, j);

koloda.Add(k1);

}

}

karta k2 = new karta(5, 15);

koloda.Add(k2);

k2 = new karta(5, 15);

koloda.Add(k2);

}

public void vivid\_spisku(List<karta> spisok, int k=0)

{

for (int i = 0; i < spisok.Count; i++)

{

if (i >= k) spisok[i].vivid();

else Console.Write("?? ");

}

}

public void pochatkova\_rozdacha()

{

ruka\_gravtsa = new List<karta>();

ruka\_kompa = new List<karta>();

for (int i = 0; i < 4; i++)

{

igrok\_bere\_kartu();

komp\_beret\_kartu();

}

}

public void vivid\_stanu()

{

Console.Write("\n\nрука компа: ");

vivid\_spisku(ruka\_kompa, skladnist);

Console.Write("\nкарта на столі: ");

if (karta\_na\_stoli != null) karta\_na\_stoli.vivid();

Console.Write("\nрука гравця: ");

vivid\_spisku(ruka\_gravtsa);

//Console.Write("\nколода: ");

//vivid\_spisku(koloda);

}

public void igrok\_bere\_kartu()

{

Random ran = new Random();

int n = ran.Next(koloda.Count);

karta kar1 = koloda[n];

ruka\_gravtsa.Add(kar1);

koloda.RemoveAt(n);

}

public void komp\_beret\_kartu()

{

Random ran = new Random();

int n = ran.Next(koloda.Count);

karta kar1 = koloda[n];

ruka\_kompa.Add(kar1);

koloda.RemoveAt(n);

}

public void hid\_gravtsa\_kartoi(int i)

{

if (karta\_na\_stoli.joker) karta\_na\_stoli = new karta(5, 15);

koloda.Add(karta\_na\_stoli);

karta\_na\_stoli = ruka\_gravtsa[i];

ruka\_gravtsa.RemoveAt(i);

}

public void hid\_gravtsa()

{

if (ruka\_gravtsa.Count == 0)

{

end\_game = true;

Console.WriteLine("Переміг гравець");

return;

}

if (ruka\_kompa.Count == 0)

{

end\_game = true;

Console.WriteLine("Переміг комп");

return;

}

vivid\_stanu();

List<int> mojlivi\_hodi\_gravtsa = new List<int>();

int hid;

if (karta\_na\_stoli == null)

{

Console.Write($"\nвибери номер карти від 1 до {ruka\_gravtsa.Count}: ");

hid = int.Parse(Console.ReadLine()) - 1;

karta\_na\_stoli = ruka\_gravtsa[hid];

ruka\_gravtsa.RemoveAt(hid);

if (karta\_na\_stoli.nominal == 15)

{

Console.Write("\nвведи масть джокера 1-♥ 2-♦ 3-♣ 4-♠ : ");

karta\_na\_stoli.mast = int.Parse(Console.ReadLine());

Console.Write("\nвведи номінал джокера 2 3 4 5 6 7 8 9 10 11-В 12-Д 13-К 14-Т: ");

karta\_na\_stoli.nominal = int.Parse(Console.ReadLine());

string s =""+karta\_na\_stoli.nominal;

switch (karta\_na\_stoli.mast)

{

case 1:

s += "♥"; // '\u2665'

break;

case 2:

s += "♦";

break;

case 3:

s += "♣";

break;

case 4:

s += "♠";

break;

}

karta\_na\_stoli.nazva += s;

}

if (spetsefekti.Contains(karta\_na\_stoli.nominal))

{

osoblivi\_pravila = true;

}

if (karta\_na\_stoli.nominal == 3)

{

hid\_gravtsa();

}

if (karta\_na\_stoli.nominal == 8)

{

Console.Write("\nвибери масть: 1-♥ 2-♦ 3-♣ 4-♠ :");

karta\_na\_stoli.mast = int.Parse(Console.ReadLine());

osoblivi\_pravila = false;

}

}

else

{

if (osoblivi\_pravila==false)

{

for (int i = 0; i < ruka\_gravtsa.Count; i++)

{

if (

ruka\_gravtsa[i].nominal == 15 || ruka\_gravtsa[i].nominal == 8 ||

ruka\_gravtsa[i].nominal!=4 && (ruka\_gravtsa[i].nominal == karta\_na\_stoli.nominal || ruka\_gravtsa[i].mast == karta\_na\_stoli.mast) ||

ruka\_gravtsa[i].nominal == 4 && ruka\_gravtsa[i].mast == karta\_na\_stoli.mast && (karta\_na\_stoli.nominal==5 || karta\_na\_stoli.nominal == 6 || karta\_na\_stoli.nominal == 7)

)

{

mojlivi\_hodi\_gravtsa.Add(i);

}

}

}

else

{

if (karta\_na\_stoli.nominal == 4)

{

for (int i = 0; i < ruka\_gravtsa.Count; i++)

{

if (ruka\_gravtsa[i].nominal==15 || ruka\_gravtsa[i].nominal==5 && ruka\_gravtsa[i].mast == karta\_na\_stoli.mast)

{

mojlivi\_hodi\_gravtsa.Add(i);

}

}

}

if (karta\_na\_stoli.nominal==3)

{

for (int i = 0; i < ruka\_gravtsa.Count; i++) mojlivi\_hodi\_gravtsa.Add(i);

osoblivi\_pravila = false;

}

}

if (mojlivi\_hodi\_gravtsa.Count == 0)

if (karta\_na\_stoli.nominal==4 && osoblivi\_pravila == true)

{

igrok\_bere\_kartu();

igrok\_bere\_kartu();

igrok\_bere\_kartu();

igrok\_bere\_kartu();

osoblivi\_pravila = false;

return;

}

else

{

Console.WriteLine("Гравцю нічим ходити, він бере карту і пропускає хід");

igrok\_bere\_kartu();

return;

}

hid = -1;

while (!mojlivi\_hodi\_gravtsa.Contains(hid))

{

Console.Write($"\nвибери номер карти зі списку ");

for (int i = 0; i < mojlivi\_hodi\_gravtsa.Count; i++) Console.Write((mojlivi\_hodi\_gravtsa[i]+1) + " ");

Console.Write(": ");

hid = int.Parse(Console.ReadLine()) - 1;

}

if (new List<int> { 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14 }.Contains(ruka\_gravtsa[hid].nominal))

{

hid\_gravtsa\_kartoi(hid);

if (new List<int> { 2, 4, 11 }.Contains(karta\_na\_stoli.nominal)) osoblivi\_pravila = true;

return;

}

if (ruka\_gravtsa[hid].nominal == 3)

{

hid\_gravtsa\_kartoi(hid);

osoblivi\_pravila = true;

hid\_gravtsa();

// dopisat

return;

}

if (ruka\_gravtsa[hid].nominal == 8)

{

hid\_gravtsa\_kartoi(hid);

Console.Write("\nвибери масть: 1-♥ 2-♦ 3-♣ 4-♠ :");

karta\_na\_stoli.mast = int.Parse(Console.ReadLine());

return;

}

if (ruka\_gravtsa[hid].nominal == 15)

{

if (karta\_na\_stoli.nominal == 4)

{

int mast = karta\_na\_stoli.mast;

hid\_gravtsa\_kartoi(hid);

karta\_na\_stoli.nominal = 5;

karta\_na\_stoli.mast = mast;

string s = "-5";

switch (mast)

{

case 1:

s += "♥"; // '\u2665'

break;

case 2:

s += "♦";

break;

case 3:

s += "♣";

break;

case 4:

s += "♠";

break;

}

karta\_na\_stoli.nazva += s;

}

else

{

Console.Write("\nвведи масть джокера 1-♥ 2-♦ 3-♣ 4-♠ : ");

ruka\_gravtsa[hid].mast = int.Parse(Console.ReadLine());

Console.Write("\nвведи номінал джокера 2 3 4 5 6 7 8 9 10 11-В 12-Д 13-К 14-Т: ");

ruka\_gravtsa[hid].nominal = int.Parse(Console.ReadLine());

string s = "-"+ ruka\_gravtsa[hid].nominal;

switch (ruka\_gravtsa[hid].mast)

{

case 1:

s += "♥"; // '\u2665'

break;

case 2:

s += "♦";

break;

case 3:

s += "♣";

break;

case 4:

s += "♠";

break;

}

ruka\_gravtsa[hid].nazva += s;

hid\_gravtsa();

}

}

}

}

public void hid\_kompa\_kartoi(int i)

{

if (karta\_na\_stoli.joker) karta\_na\_stoli = new karta(5, 15);

koloda.Add(karta\_na\_stoli);

karta\_na\_stoli = ruka\_kompa[i];

ruka\_kompa.RemoveAt(i);

}

public int vibor\_masti()

{

List<int> mast = new List<int> { 0, 0, 0, 0 };

for (int i=0; i < ruka\_kompa.Count; i++)

{

if (ruka\_kompa[i].mast<5)

mast[ruka\_kompa[i].mast-1]++;

}

int max\_i\_mast = 0;

for (int i = 0; i < 4; i++) if (mast[i] > mast[max\_i\_mast]) max\_i\_mast = i;

Console.Write("\nКомп вибрав масть ");

switch (max\_i\_mast + 1)

{

case 1:

Console.WriteLine("♥"); // '\u2665'

break;

case 2:

Console.WriteLine("♦");

break;

case 3:

Console.WriteLine("♣");

break;

case 4:

Console.WriteLine("♠");

break;

}

return max\_i\_mast+1;

}

public void hid\_kompa\_jokerom(int i)

{

if(karta\_na\_stoli.nominal==5 || karta\_na\_stoli.nominal == 6 || karta\_na\_stoli.nominal == 7)

{

int mast = karta\_na\_stoli.mast;

hid\_kompa\_kartoi(i);

karta\_na\_stoli.nominal = 4;

karta\_na\_stoli.mast = mast;

osoblivi\_pravila = true;

string s="-4";

switch (mast)

{

case 1:

s+="♥"; // '\u2665'

break;

case 2:

s+="♦";

break;

case 3:

s += "♣";

break;

case 4:

s+="♠";

break;

}

karta\_na\_stoli.nazva += s;

}

else

{

int mast = karta\_na\_stoli.mast;

hid\_kompa\_kartoi(i);

karta\_na\_stoli.nominal = 3;

karta\_na\_stoli.mast = mast;

osoblivi\_pravila = true;

string s = "-3";

switch (mast)

{

case 1:

s += "♥"; // '\u2665'

break;

case 2:

s += "♦";

break;

case 3:

s += "♣";

break;

case 4:

s += "♠";

break;

}

karta\_na\_stoli.nazva += s;

hid\_kompa();

}

}

public void hid\_kompa()

{

vivid\_stanu();

if (ruka\_gravtsa.Count == 0)

{

end\_game = true;

Console.WriteLine("Переміг гравець");

return;

}

if (ruka\_kompa.Count == 0)

{

end\_game = true;

Console.WriteLine("Переміг комп");

return;

}

for (int i = 0; i < ruka\_kompa.Count; i++)

{

ruka\_kompa[i].rozstanovka\_tsinnostey(ruka\_gravtsa.Count);

}

ruka\_kompa.Sort((x, y) => y.tsinnist - x.tsinnist);

//vivid\_stanu();

if (osoblivi\_pravila == false)

{

int i = 0;

for (; i < ruka\_kompa.Count; i++)

{

if ((ruka\_kompa[i].mast == karta\_na\_stoli.mast || ruka\_kompa[i].nominal == karta\_na\_stoli.nominal)&&(!spetsefekti.Contains(ruka\_kompa[i].nominal)))

{

hid\_kompa\_kartoi(i);

return;

}

if(ruka\_kompa[i].nominal==2 && (ruka\_kompa[i].mast == karta\_na\_stoli.mast || ruka\_kompa[i].nominal == karta\_na\_stoli.nominal))

{

hid\_kompa\_kartoi(i);

igrok\_bere\_kartu();

hid\_kompa();

return;

}

if (ruka\_kompa[i].nominal == 3 && (ruka\_kompa[i].mast == karta\_na\_stoli.mast || ruka\_kompa[i].nominal == karta\_na\_stoli.nominal))

{

hid\_kompa\_kartoi(i);

osoblivi\_pravila = true;

hid\_kompa();

return;

}

if (ruka\_kompa[i].nominal == 4 && (karta\_na\_stoli.nominal==5 || karta\_na\_stoli.nominal == 6 || karta\_na\_stoli.nominal == 7) && ruka\_kompa[i].mast == karta\_na\_stoli.mast)

{

hid\_kompa\_kartoi(i);

osoblivi\_pravila = true;

return;

}

if (ruka\_kompa[i].nominal == 8)

{

hid\_kompa\_kartoi(i);

karta\_na\_stoli.mast=vibor\_masti();

//karta\_na\_stoli.nazva +="-"+karta\_na\_stoli.mast;

return;

}

if (ruka\_kompa[i].nominal == 11 && (ruka\_kompa[i].mast == karta\_na\_stoli.mast || ruka\_kompa[i].nominal == karta\_na\_stoli.nominal))

{

hid\_kompa\_kartoi(i);

hid\_kompa();

return;

}

if (ruka\_kompa[i].nominal == 15)

{

hid\_kompa\_jokerom(i);

return;

}

}

if (i>=ruka\_kompa.Count)

{

komp\_beret\_kartu();

}

}

else

{

osoblivi\_pravila = false;

if (karta\_na\_stoli.nominal == 2)

{

komp\_beret\_kartu();

return;

}

if (karta\_na\_stoli.nominal == 3)

{

if (!spetsefekti.Contains(ruka\_kompa[0].nominal))

{

hid\_kompa\_kartoi(0);

return;

}

else

{

if(ruka\_kompa[0].nominal==2)

{

hid\_kompa\_kartoi(0);

igrok\_bere\_kartu();

hid\_kompa();

return;

}

if (ruka\_kompa[0].nominal == 3)

{

hid\_kompa\_kartoi(0);

osoblivi\_pravila = true;

hid\_kompa();

return;

}

if (ruka\_kompa[0].nominal == 4)

{

hid\_kompa\_kartoi(0);

osoblivi\_pravila = true;

return;

}

if (ruka\_kompa[0].nominal == 8)

{

hid\_kompa\_kartoi(0);

karta\_na\_stoli.mast = vibor\_masti();

return;

}

if (ruka\_kompa[0].nominal == 11)

{

hid\_kompa\_kartoi(0);

hid\_kompa();

return;

}

if (ruka\_kompa[0].nominal == 15)

{

hid\_kompa\_jokerom(0);

return;

}

// treba dopisat

}

}

if (karta\_na\_stoli.nominal == 4)

{

for (int i = 0; i < ruka\_kompa.Count; i++)

{

if (ruka\_kompa[i].nominal==5&&ruka\_kompa[i].mast==karta\_na\_stoli.mast)

{

hid\_kompa\_kartoi(i);

return;

}

else if (ruka\_kompa[i].nominal == 15)

{

int mast = karta\_na\_stoli.mast;

hid\_kompa\_kartoi(i);

karta\_na\_stoli.mast = mast;

karta\_na\_stoli.nominal = 5;

string s = "-5";

switch (mast)

{

case 1:

s += "♥"; // '\u2665'

break;

case 2:

s += "♦";

break;

case 3:

s += "♣";

break;

case 4:

s += "♠";

break;

}

karta\_na\_stoli.nazva += s;

return;

}

}

for (int j = 0; j < 4; j++) komp\_beret\_kartu();

osoblivi\_pravila = false;

return;

}

if (karta\_na\_stoli.nominal == 11)

{

return;

}

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

Console.OutputEncoding = Encoding.UTF8;

gra g = new gra();

}

}

}

### Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

## 

Висновок

В рамках даної лабораторної роботи я розробив логіку гри останній гравець, в ній використовувалась оцінка карт в залежності від кількості карт в руці противника і номінала карти, комп’ютер викидував найменш цінні карти. Також реалізована складність гри - можна підглядати карти противника.