МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАВЧАЛЬНО–НАУКОВИЙ КОМПЛЕКС

“ІНСТИТУТ ПРИКАДНОГО СИСТЕМНОГО АНАЛІЗУ”

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

**КУРСОВА РОБОТА**

з дисципліни

“Програмування та алгоритмічні мови”

на тему: "Ігрова програма "Шашки"

Виконав:

Студент першого курсу

гр. КА-41

Барзій Ілля Ігорович

Прийняла:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оцінка)

Назарчук Ірина Василівна

"\_\_"\_\_\_\_\_\_\_\_\_\_\_\_ 2015

КИЇВ 2015

Зміст

[1.Вступ 4](#_Toc417940243)

[2.Основна частина 5](#_Toc417940244)

[2.1 Постановка задачі 5](#_Toc417940245)

[2.2 Метод розв’язку задачі 6](#_Toc417940246)

[2.3 Алгоритм рохв’язку задачі 8](#_Toc417940247)

[2.4 Опис програмного продукту 9](#_Toc417940248)

[2.4.1 Опис модулів та їх процедур та функцій 9](#_Toc417940249)

[2.5 Опис елементів інтерфейсу 12](#_Toc417940250)

[2.6 Результат роботи программи 16](#_Toc417940251)

[3.Висновки 17](#_Toc417940252)

[4.Додаток А 18](#_Toc417940253)

[5.Список використаної літератури 32](#_Toc417940254)

# 1.Вступ

Ігрова програма, що реалізує 8-клітинні шашки для двох гравців за правилами "руських шашок" на мові програмування Delphi. Основним завданням програми є перевірка припустимості ходу. В даній версії гри користувачам, в залежності від черги та обраної шашки, пропонуються усі можливі ходи.

Дана тема була обрана через популярність гри та входження її у клас інтелектуальних ігор. Особливо цікавим завданням є розробка алгоритму до даної програми. Втілення гри за правилами "руських шашок" було обране через популярність цієї версії шашок в країнах пострадянського простору.

Гра має на меті перевірити та розвинути розумові та аналітичні здібності гравців шляхом суперництва за чітко встановленими правилами.

Бажаючи зробити програму доступною до використання широкому обсягу користувачів, було вирішено створити інтуїтивно зрозумілий інтерфейс.

# 2.Основна частина

## 2.1 Постановка задачі

Правила гри(за версією "руські шашки"), на базі яких створена дана реалізація:

1. Ігровий процес відбувається на дошці розміром 8х8, в початковій позиції кожен гравець має по 12 шашок, що займають перші три ряди з кожної сторони. Гра ведеться по темним полям. В початковій позиції усі шашки прості. При досягненні восьмого горизонтального ряду звичайна шашка перетворюється на дамку.
2. Хід; Звичайна шашка ходить по діагоналі вперед на одне поле(за умови, що на ньому не стоїть інша шашка). Дамка ходить по діагоналі на будь-яке пусте поле як вперед так і назад (за умови, що по діагоналі до цього поля усі поля пусті).
3. Взяття; Взяття обов'язкове. Побиті шашки знімаються після завершення ходу. Звичайна шашка, що знаходиться поряд з шашкою супротивника, за якою поле пусте, переноситься через цю шашку на пусте поле. Якщо є можливість продовжити взяття інших шашок супротивника, то взяття продовжується, допоки атакуюча шашка не досягне положення, у якому взяття неможливе. Дамка б'є по діагоналі як вперед, так і назад та стає на будь-яке пусте поле після збитої шашки . Дамка також повинна бити шашки супротивника доки це можливо. При бої через дамкове поле шашка стає дамкою та продовжує бити за правилами дамки. При декількох варіантах взяття, гравець обирає довільий.
4. Виграш партії; Партія вважається виграною, якщо: в одного із супротивників збиті усі шашки, або усі шашки одного з гравців заблоковані та він не може зробити хід, або якщо один з гравців закінчує партію.
5. Партія триває необмежений час, доки один із супротивників не виграє.

## 2.2 Метод розв’язку задачі

Позиції шашок на дошці задані матрицею символів, розміром 8х8. Гравець, чия черга ходити, кількість збитих шашок в обох гравців та координати обраної клітини задані п'ятьма змінними.

Після початку гри про дошці розставляються шашки та запускається таймер, що кожні 100 мілісекунд запускає сукупність процедур(далі логічне ядро). Це ядро діє наступним чином:

* отримує координати обраного поля, ім'я гравця, чия черга ходити, кількість збитих шашок в обох гравців та матрицю, що зображає дошку;
* перевіряє, чи є шашки, які треба перетворити на дамки, якщо є, то перетворює;
* перевіряє, чи є переможець партії, якщо так, то закінчує гру та виводить ім'я переможця гри;
* визначає гравця, черга якого ходити та виконує всі наступні дії для шашок цього гравця;
* якщо координати обраного поля не відрізняються від попередніх, то ядро припиняє роботу;
* перевіряє, чи є взяття в усіх шашок гравця;
* якщо взяття є, та перші координати необрані, то присвоює їм значення обраних. Якщо ці координати визначають шашку, котра може бити, то зображає цей бій на дошці, а якщо не може бити, то визначає перші координати необраними, припиняє свою роботу;
* якщо взяття є та перші координати обрані, то присвоює другим координатам значення обраних, якщо ці координати вказують на шашку, що б'ється шашкою, обраною першими координатами, то виконує цей бій, додає 1 до рахунку гравця, перевіряє, чи є в шашки, котрою щойно побили взяття, якщо так, то присвоює першим координатам значення других та переходить до ситуації, коли є взяття та обрані перші координати. Якщо повторного взяття немає, то визначає перші та другі координати необраними, перевіряє чи може супротивник зробити хід. Якщо супротивник не може зробити хід, то виводить ім'я переможця гри. Інакше, передає право ходу іншому гравцю, припиняє свою роботу;
* якщо взяття є та перша координати обрані, то присвоює другим координатам значення обраних, якщо ці координати не вказують на шашку, що б'ється шашкою, обраною першими координатами, то визначає перші та другі координати необраними, припиняє свою роботу;
* якщо взяття немає, та перші координати не обрані, то присвоює їм значення обраних, перевіряє можливі ходи для даної шашки та зображає їх на дошці, припиняє свою роботу;
* якщо взяття немає, та перші координати обрані, то присвоює другим значення обраних, якщо ці координати вказують на поле, що є ходом для шашки, обраної першими координатами, то виконує цей хід, визначає перші та другі координати необраними, перевіряє чи може супротивник зробити хід. Якщо супротивник не може зробити хід, то виводить ім'я переможця гри. Інакше, передає право ходу іншому гравцю, припиняє свою роботу;
* якщо взяття немає, та перші координати обрані, то присвоює другим значення обраних, якщо ці координати не вказують на поле, що є ходом для шашки, обраної першими координатами, то визначає перші та другі координати необраними, припиняє свою роботу;

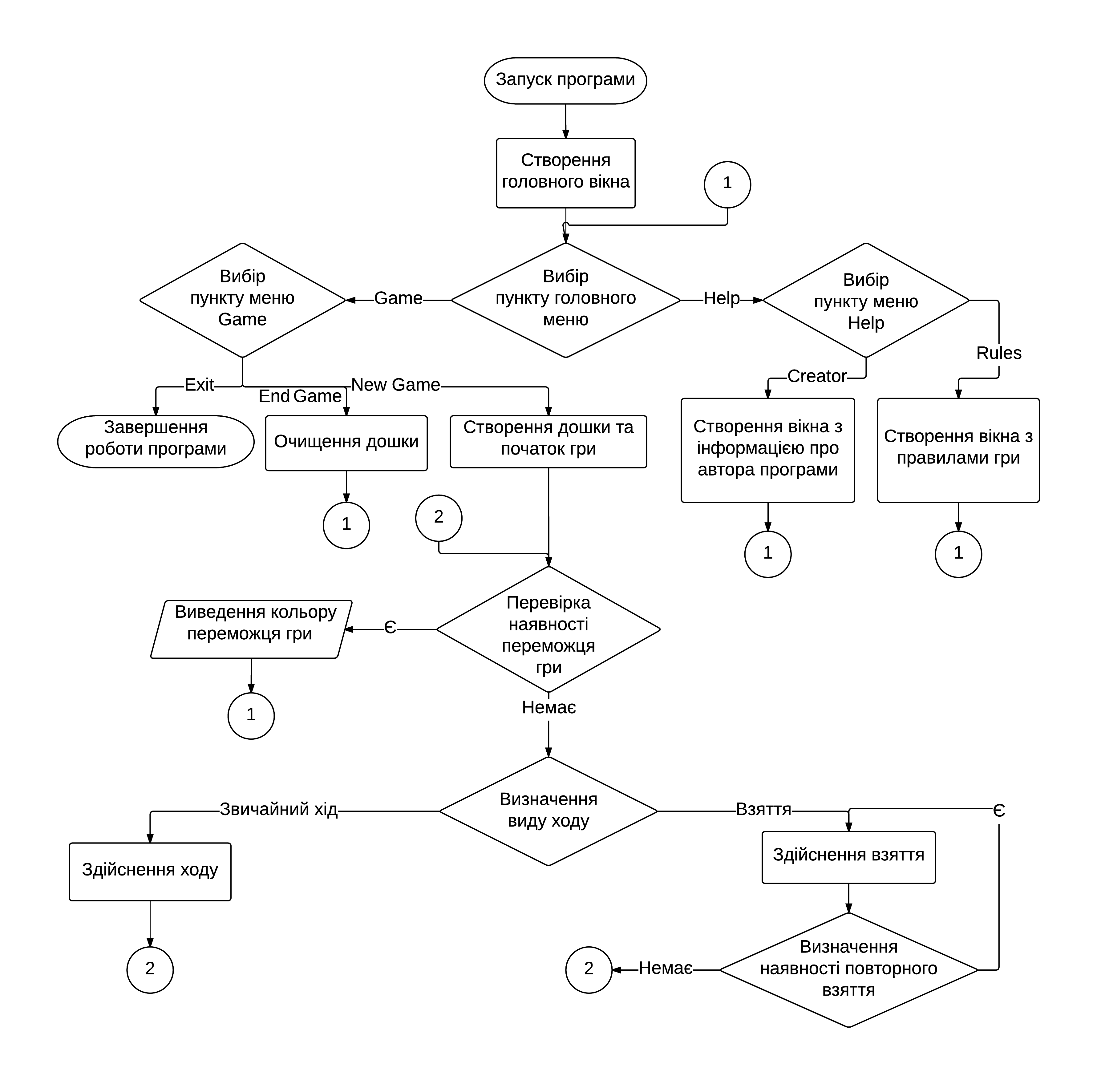
При перемозі гравця виводиться повідомлення про це.

Гравець може в будь - який момент завершити гру та вийти з програми, натиснувши відповідну кнопку в головному меню.

Також через головне вікно можна викликати вікна з правилами гри та інформацією про автора програми.

## 2.3 Алгоритм розв’язку задачі

Після запуску програми користувач потрапляє у головне вікно. Далі він, використовуючи пункти головного меню, може переходити у різні частини програми. Це показано на структурній блок-схемі,яку наведено далі.



## 

Рисунок 2.1 Структурна блок-схема алгоритму

## 2.4 Опис програмного продукту

### 2.4.1 Опис модулів та їх процедур та функцій

Модуль Unit.1

**Головні структури та змінні**

*Form1: TForm1;*

Головне вікно програми

*path:string;*

Шлях до програми

*field,b:array[1..8,1..8] of Char;*

Дошка з шашками та дошка з запропонованими ходами

*player,ka:Char;*

Гравець, чия черга ходити та наявність повторного взяття

*nowx,nowy,pointb,pointw,needx,needy:Integer;*

Обрані координати, кількість збитих гравцями шашок, координати, що

передаються(допоміжні)

*x1,x2,y1,y2,prevx,prevy:Integer;*

Перші та другі координати

*cc,work,must,l1,l2,s1,win,w1,skip:Boolean;*

Допоміжні прапорці

**Процедури та функції**

*procedure dofield*

Розставляє на дошці шашки у стартову позицію та малює їх.

*procedure checkwin;*

Перевіряє, чи є переможець партії.

*procedure move(l:Char;x,y:Integer);*

Визначає можливі ходи для заданого гравця та координат шашки.

*procedure drawmove;*

Зображає на дошці результати роботи процедури *move*.

*procedure fight( l:Char);*

Визначає можливі взяття для заданого гравця.

*procedure drawfight(x,y:integer);*

Зображає на дошці результати роботи процедури *fight* для заданих

координат.

*procedure Checker;*

Перевіряє, чи є обрані координати новими.

*procedure movedo (x1,y1,x2,y2:Integer);*

Реалізує ход з перших координат у другі.

*procedure qween;*

Перетворює необхідні шашки на дамки.

*procedure movehitdo (x1,y1,x2,y2:Integer);*

Реалізує взяття з перших координат у другі.

*procedure cani(Player:Char);*

Перевіряє заблокованість усіх шашок заданого гравця.

*procedure clean;*

Очищає дошку від запропонованих ходів.

*procedure count;*

Виводить рахунок та гравця, черга якого ходити.

*procedure Main;*

Основна процедура. Її дії описані у п.2.2

*procedure TForm1.N6Click(Sender: TObject);*

Викликає вікно з правилами.

*procedure TForm1.N2Click(Sender: TObject);*

Починає гру.

*procedure TForm1.Timer1Timer(Sender: TObject);*

Викликає процедуру *Main* кожні 100 мілісекунд.

*procedure TForm1.FormCreate(Sender: TObject);*

Створює головне вікно.

*procedure TForm1.N3Click(Sender: TObject);*

Закінчує гру.

*procedure TForm1.N7Click(Sender: TObject);*

Викликає вікно з інформацієє про автора програми.

*procedure TForm1.FormCreate(Sender: TObject);*

Створює головне вікно.

*procedure TForm1.N4Click(Sender: TObject);*

Закриває вікно.

*procedure TForm1.Image\*\*Click(Sender: TObject);*

Надає обраним координатам значення \*\* при натисканні на відповідне

поле дошки.

## 2.5 Опис елементів інтерфейсу

Після запуску користувач бачить головне вікно на рисунку 2.2. Меню відкривається при натисканні на нього. Вибрати підпункт можна за допомогою вказівника(миші)

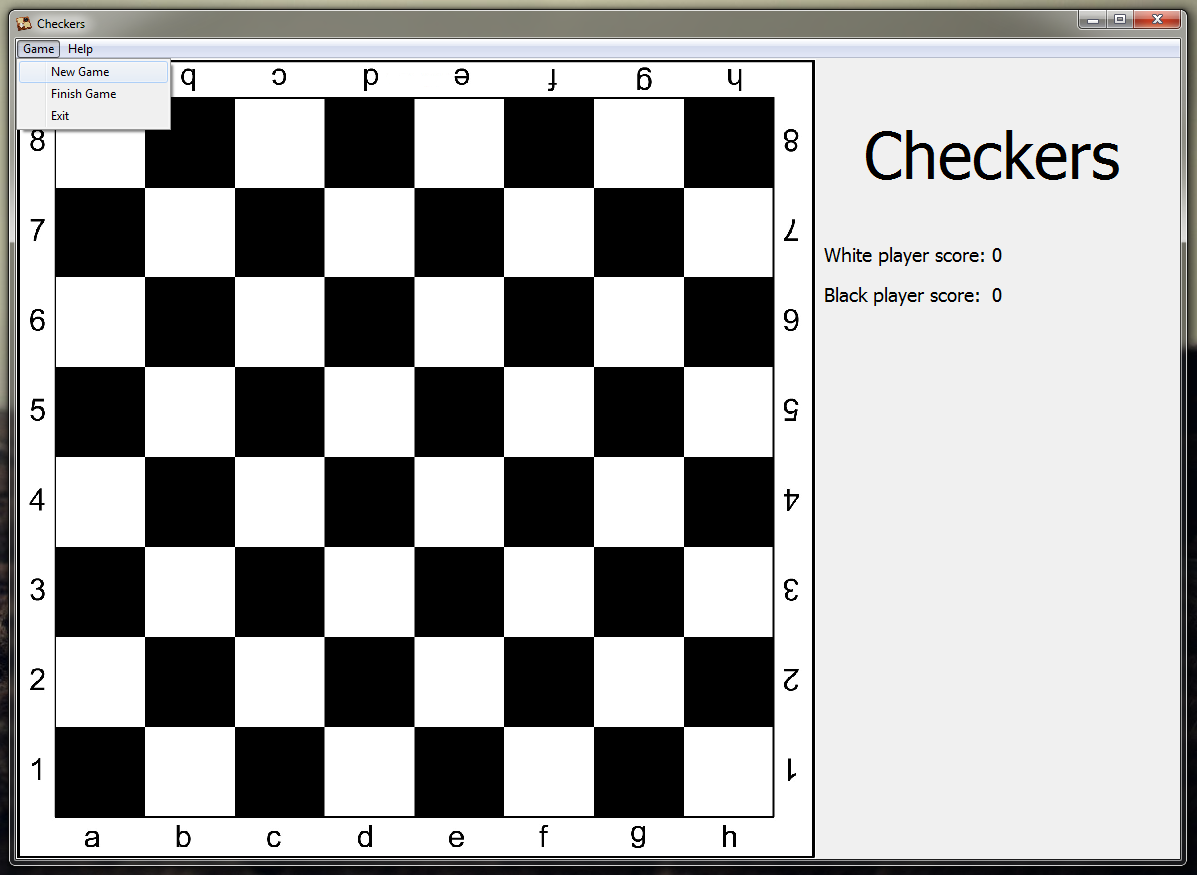


Рисунок 2.2 Головне меню

Користувач може вибрати один з двох пунктів, що містять підпункти.

Вікно з правилами гри рисунку 2.3 (Help > Rules)

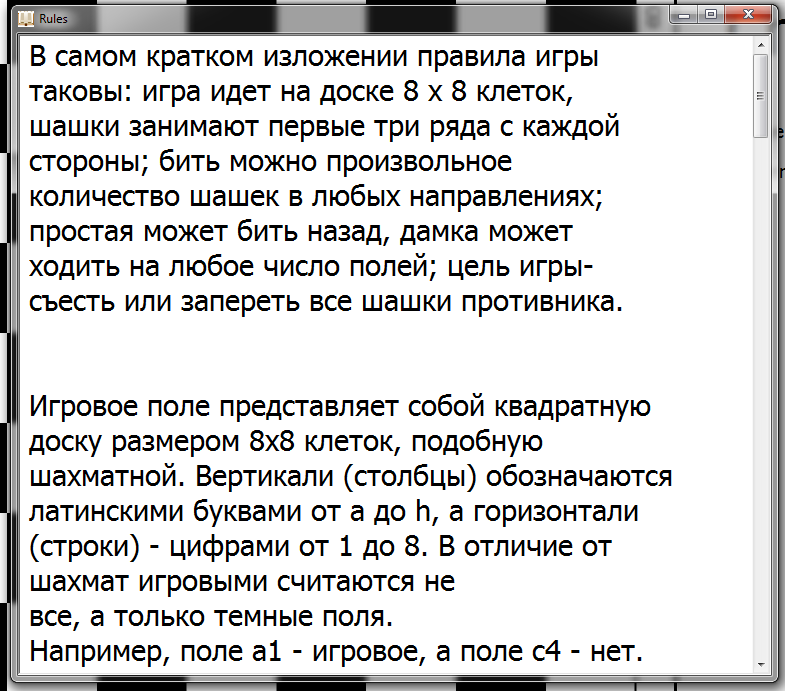


Рисунок 2.3 Вікно з правилами гри

Вікно з інформацією про розробника на рисунку 2.4 (Help > Creator)

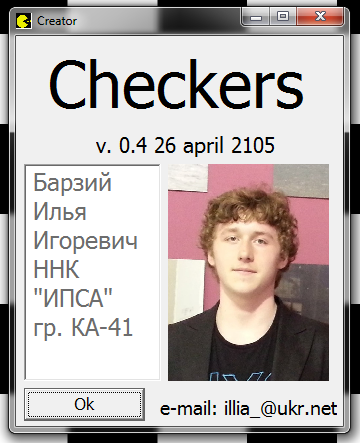


Рисунок 2.4 Вікно з інформацією про розробника

Вигляд поля після початку гри(Game > New Game) на рисунку 2.5

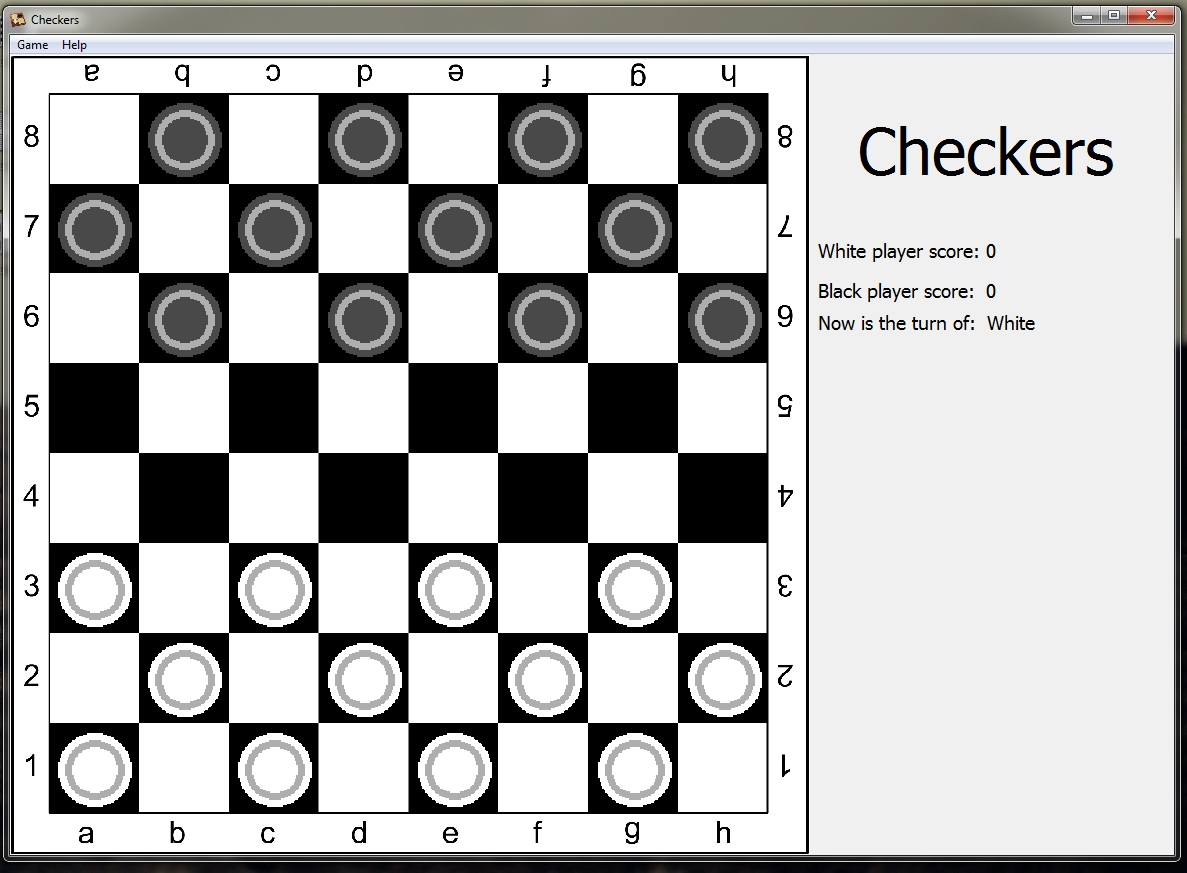


Рисунок 2.5 Ігрове поле

Вигляд запропонованого ходу для шашки на рисунок 2.6

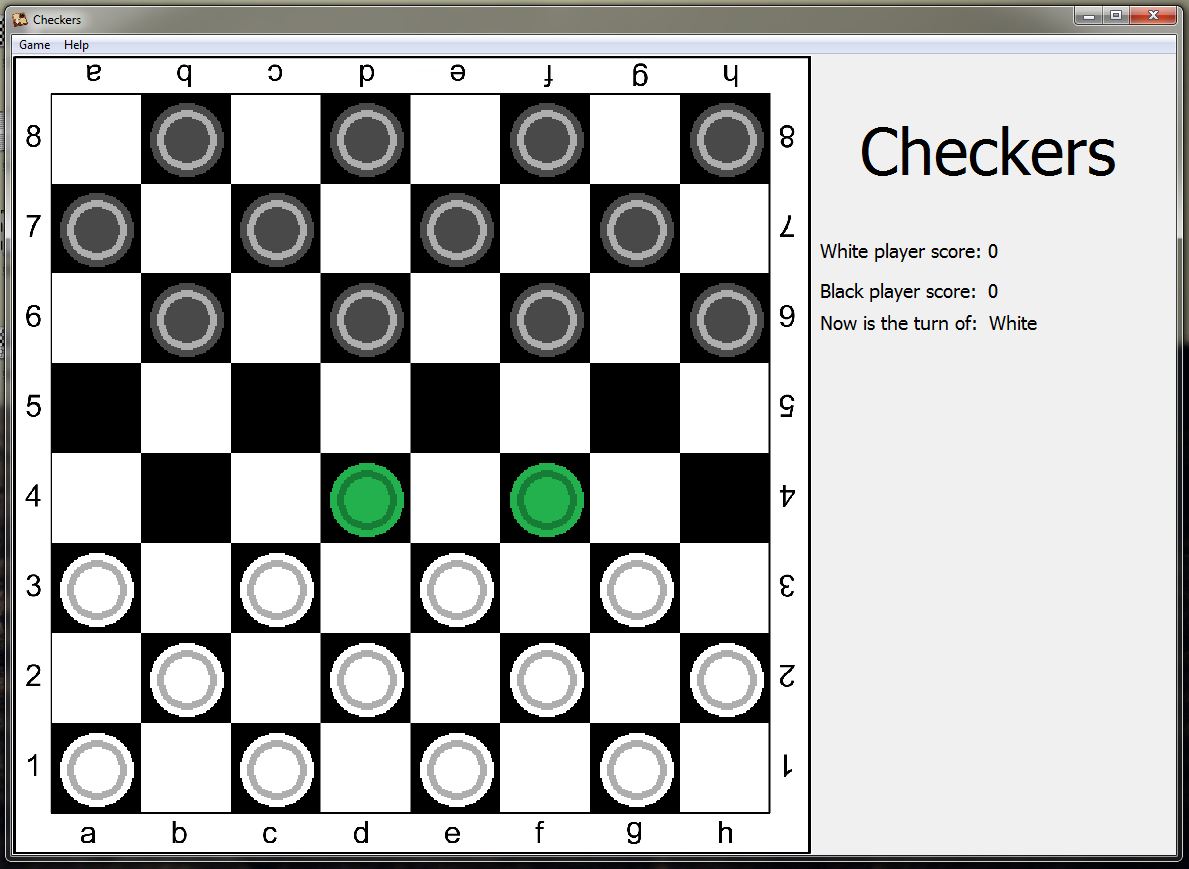


Рисунок 2.6 Пропозиція ходу

Вигляд запропонованого взяття для шашки на рисунку 2.7

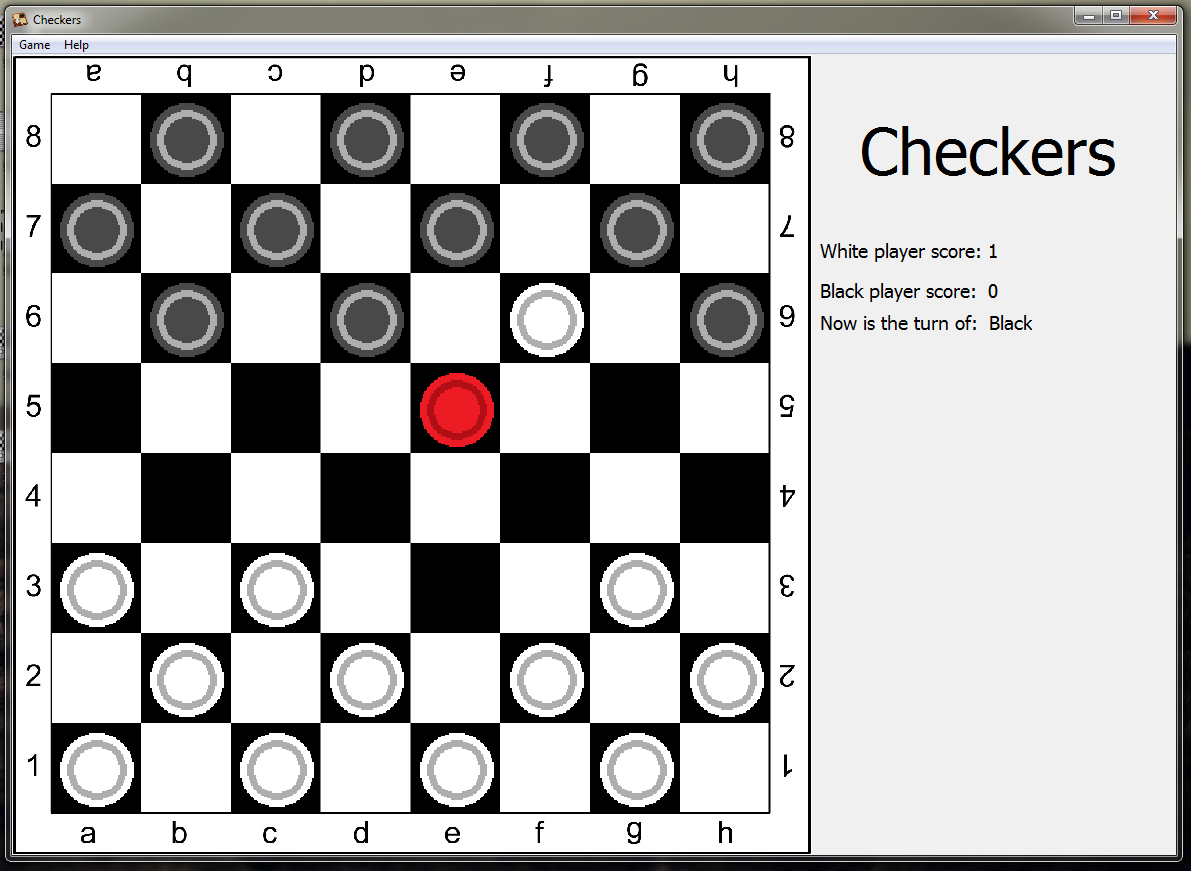


Рисунок 2.7 Пропозиція взяття

Вигляд запропонованого ходу для дамки на рисунку 2.8

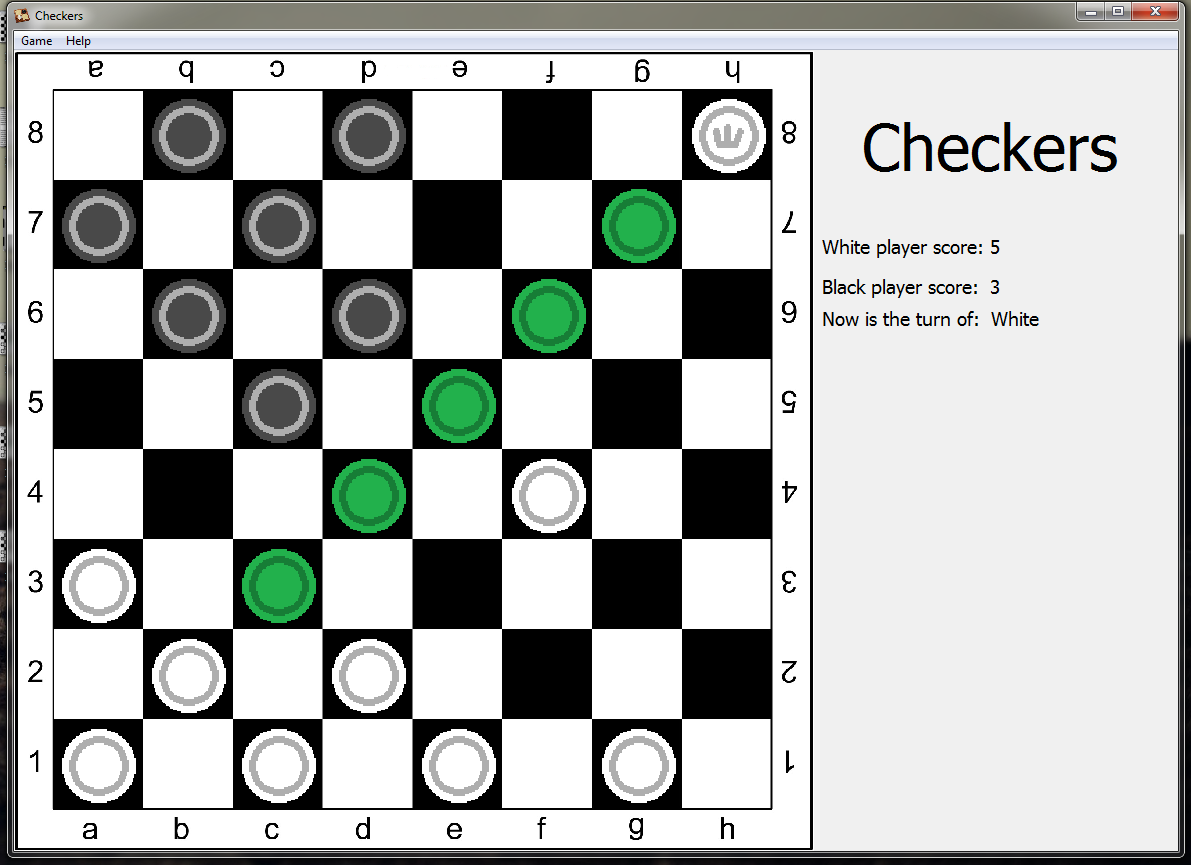


Рисунок 2.8 Пропозиція ходу дамки

Вигляд запропонованого взяття для дамки на рисунку 2.9

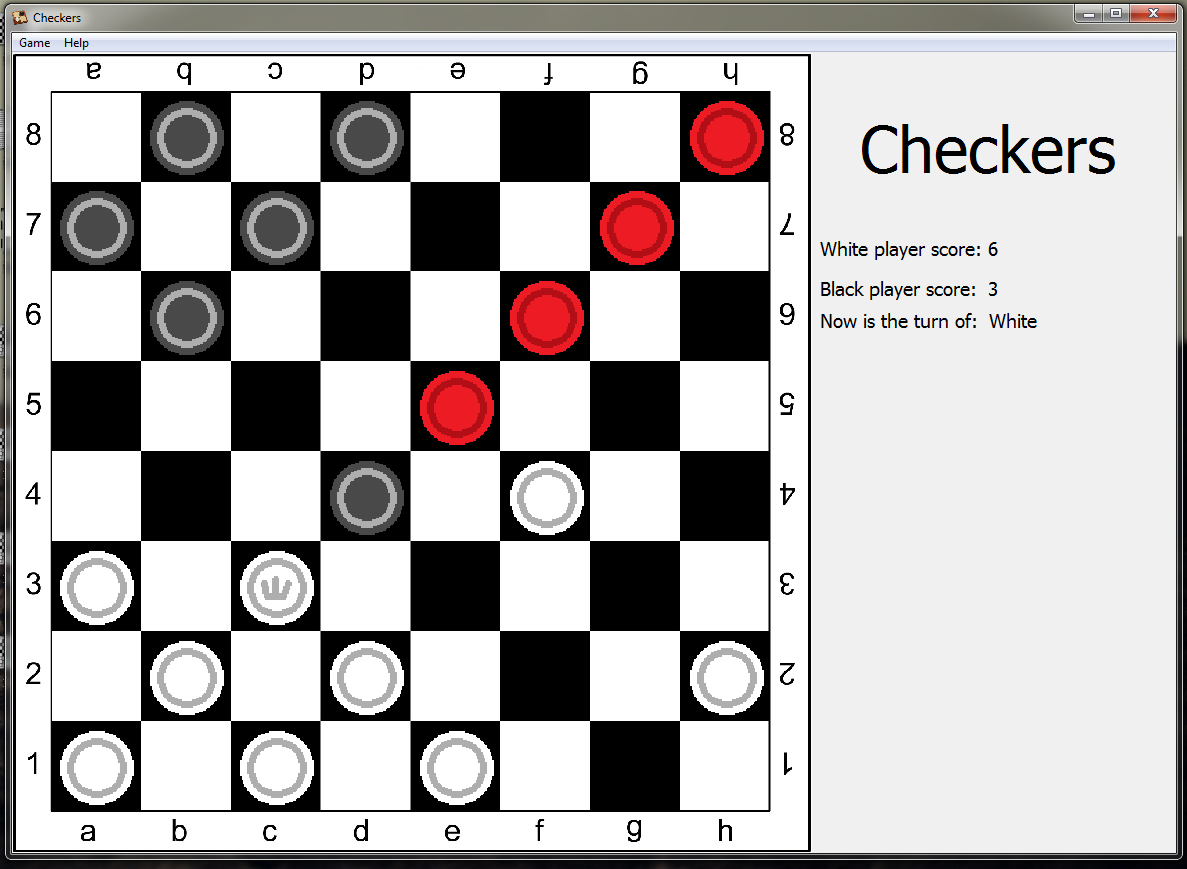


Рисунок 2.9 Пропозиція взяття для дамки

## 2.6 Результат роботи программи

Після перемоги одного з гравців буде виведено повідомлення про те, що гра завершилася перемогою певного гравця. Гру можна почати знову, обравши в головному меню "Game > New Game". Відповідне повідомлення на рисунку 2.10

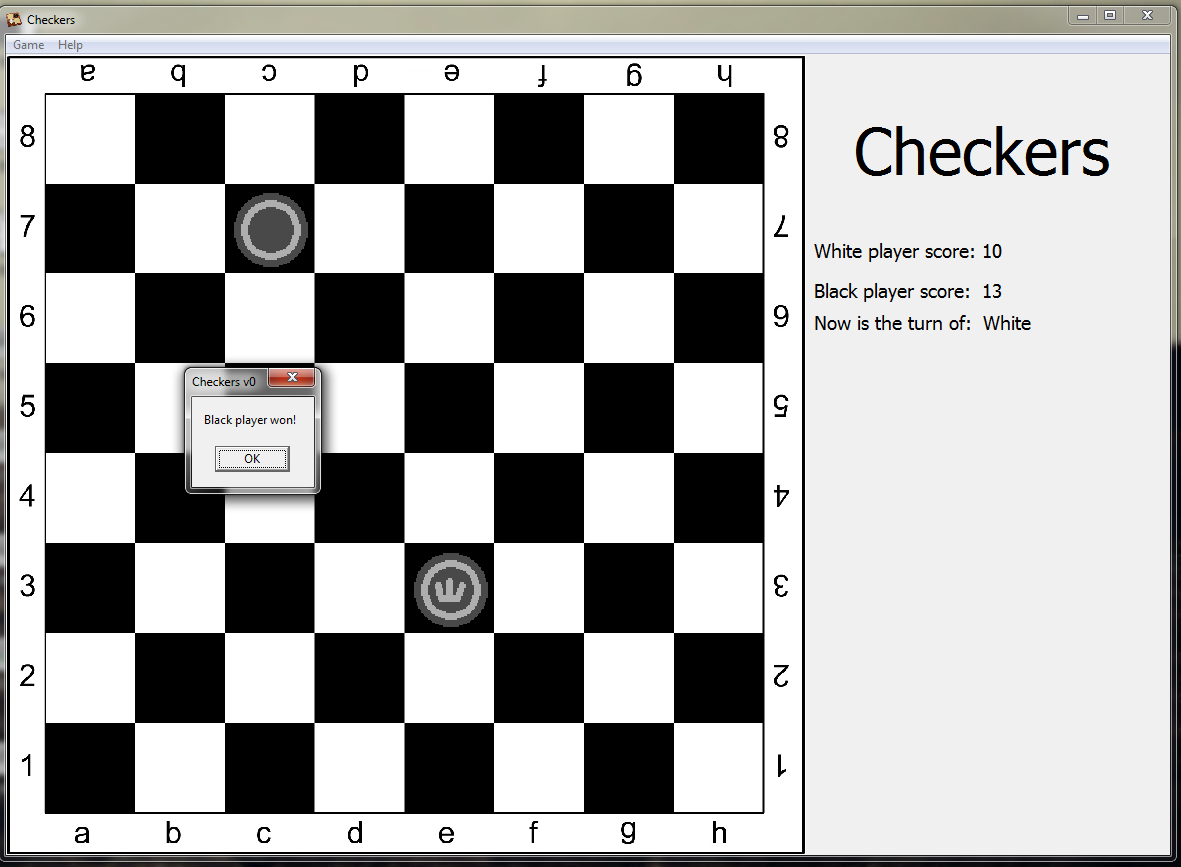


Рисунок 2.10 Завершення сеансу гри

# 3.Висновки

Кінцевий програмний продукт відповідає вимогам завдання.

До переваг даної програми належать:

* Інтуїтивно зрозумілий інтерфейс.
* Перенесення усіх правил гри за версією "руські шашки" у цю реалізацію.
* Стійкість до непердбачуваних дій користувача.
* Блокова структура програми,і , як наслідок, можливість її зміни та доповнення.

Недоліки:

* Гра реалізована тільки для операційних систем сімейства Windows
* Відстуність масштабування під різні екрани.
* Відсутьність режиму гри з обмеженим часом.
* Відсутність режиму гри проти комп'ютера.

Шляхи подальшого покращення:

* Введення режиму гри проти комп'ютера.
* Реалізація збереження гри у файл та завантаження з файлу
* Виведення нотації гри та можливість ії збереження.
* Адаптація програмного коду для версій під операційні системи на базі ядра Linux та мобільних платформ.

# 4.Додаток Лістинг програми

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Menus, ExtCtrls, StdCtrls;

type

Matrix = array[1..8] of array[1..8] of char;

TForm1 = class(TForm)

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N6: TMenuItem;

N7: TMenuItem;

Image1: TImage;

Image11: TImage;

Image31: TImage;

Image51: TImage;

Image71: TImage;

Image22: TImage;

Image42: TImage;

Image62: TImage;

Image82: TImage;

Image73: TImage;

Image53: TImage;

Image33: TImage;

Image13: TImage;

Image24: TImage;

Image44: TImage;

Image55: TImage;

Image15: TImage;

Image35: TImage;

Image75: TImage;

Image84: TImage;

Image64: TImage;

Image26: TImage;

Image46: TImage;

Image66: TImage;

Image86: TImage;

Image17: TImage;

Image37: TImage;

Image57: TImage;

Image77: TImage;

Image88: TImage;

Image68: TImage;

Image48: TImage;

Image28: TImage;

Timer1: TTimer;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

procedure N6Click(Sender: TObject);

procedure N4Click(Sender: TObject);

procedure N2Click(Sender: TObject);

procedure Image11Click(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

procedure Image22Click(Sender: TObject);

procedure Image13Click(Sender: TObject);

procedure Image15Click(Sender: TObject);

procedure Image17Click(Sender: TObject);

procedure Image24Click(Sender: TObject);

procedure Image26Click(Sender: TObject);

procedure Image28Click(Sender: TObject);

procedure Image31Click(Sender: TObject);

procedure Image33Click(Sender: TObject);

procedure Image35Click(Sender: TObject);

procedure Image37Click(Sender: TObject);

procedure Image42Click(Sender: TObject);

procedure Image44Click(Sender: TObject);

procedure Image46Click(Sender: TObject);

procedure Image48Click(Sender: TObject);

procedure Image51Click(Sender: TObject);

procedure Image53Click(Sender: TObject);

procedure Image55Click(Sender: TObject);

procedure Image57Click(Sender: TObject);

procedure Image62Click(Sender: TObject);

procedure Image64Click(Sender: TObject);

procedure Image66Click(Sender: TObject);

procedure Image68Click(Sender: TObject);

procedure Image71Click(Sender: TObject);

procedure Image73Click(Sender: TObject);

procedure Image75Click(Sender: TObject);

procedure Image77Click(Sender: TObject);

procedure Image82Click(Sender: TObject);

procedure Image84Click(Sender: TObject);

procedure Image86Click(Sender: TObject);

procedure Image88Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure N7Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

path:string;

field,b:array[1..8,1..8] of Char;

player,ka:Char;

nowx,nowy,pointb,pointw,needx,needy:Integer;

x,y,x1,x2,y1,y2,prevx,prevy:Integer;

cc,work,must,l1,l2,s1,win,w1,skip:Boolean;

implementation

uses Unit2, Unit3;

{$R \*.dfm}

procedure dofield;

var n,i,j:Integer;

begin

for j:=1 to 8 do

begin

for i:=1 to 3 do

if ((i+j) mod 2= 0) then Field[i,j]:='w';

for i:=6 to 8 do

if ((i+j) mod 2= 0) then Field[i,j]:='b';

for i:=4 to 5 do

if ((i+j) mod 2= 0) then Field[i,j]:='0';

end;

for i:=1 to 8 do

for j:=1 to 8 do

begin

n:=i\*10+j;

if field[i,j]='w' then

begin

with Form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\0.bmp');

end;

end;

if field[i,j]='b' then

begin

with Form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin Picture.LoadFromFile(path+'\pic\2.bmp');

end;

end;

end;

end;

procedure checkwin;

begin

if (pointb>12) or (pointw>12) then win:=true;

end;

procedure move(l:Char;x,y:Integer);

var i,j,i1:Integer;

way1,way2,way3,way4:boolean;

u:char;

begin

if l='w' then u:='v';

if l='b' then u:='d';

if (field[x,y]=l) or (field[x,y]=u) then

begin

way1:=true;way2:=true;way3:=true;way4:=true;

for i:=1 to 8 do

for j:=1 to 8 do

if ((i+j)mod 2 = 0) then

b[i,j]:='0';

if (field[x,y]='w') or (field[x,y]='b') then

begin

if l='w' then i1:=1;

if l='b' then i1:=-1;

i:=x; j:=y;

if ((i+i1)<9) and ((i+i1)>0) and ((j+1)<9) then

if field[i+i1,j+1]='0' then b[i+i1,j+1]:='m';

if ((i+i1)<9) and ((i+i1)>0) and ((j-1)>0) then

if field[i+i1,j-1]='0' then b[i+i1,j-1]:='m';

end;

if (field[x,y]='v') or (field[x,y]='d') then

begin

for i:=1 to 7 do begin

if ((x+i)<9) and ((y+i)<9) and (way1) then

begin

if (field[x+i,y+i]='0') then b[x+i,y+i]:='m'

else way1:=false;

end;

if ((x+i)<9) and ((y-i)>0) and (way2) then

begin

if (field[x+i,y-i]='0') then b[x+i,y-i]:='m'

else way2:=false;

end;

if ((x-i)>0) and ((y+i)<9) and (way3) then

begin

if (field[x-i,y+i]='0') then b[x-i,y+i]:='m'

else way3:=false;

end;

if ((x-i)>0) and ((y-i)>0) and (way4) then

begin

if (field[x-i,y-i]='0') then b[x-i,y-i]:='m'

else way4:=false;

end;

end;

end;

end;

end;

procedure drawmove;

var n,i,j:Integer;

begin

for i:=1 to 8 do

for j:=1 to 8 do

begin

n:=i\*10+j;

if b[i,j]='m' then

begin

with form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\5.bmp');

end;

end;

if not (b[i,j]='m') and (field[i,j]='0') and ((i+j) mod 2 = 0) then

begin

with form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture:=nil;

end;

end;

end;

end;

procedure fight( l:Char);

var i,j,i1,mx1,mx2,mx3,mx4,my1,my2,my3,my4:Integer;

op1,op2,f:Char;

way1,way2,way3,way4,k1,k2,k3,k4,g1,g2,g3,g4:boolean;

begin

if l='w' then

begin

op1:='b'; op2:='d'; f:='v';

end;

if l='b' then

begin

op1:='w'; op2:='v'; f:='d';

end;

for i:=1 to 8 do

for j:=1 to 8 do

begin

if ((i+j)mod 2 =0) then

b[i,j]:='0';

end;

for i:=1 to 8 do

for j:=1 to 8 do

begin

if (field[i,j]=l) and ((i+j)mod 2 = 0) then

begin

if (i+2<9) and (j+2<9) then

if (field[i+1,j+1]=op1) or (field[i+1,j+1]=op2) then

if (field[i+2,j+2]='0') then

begin

b[i+2,j+2]:='f';

b[i+1,j+1]:='h';

b[i,j]:='s';

must:=true;

end;

if (i+2<9) and (j-2>0) then

if (field[i+1,j-1]=op1) or (field[i+1,j-1]=op2) then

if (field[i+2,j-2]='0') then

begin

b[i+2,j-2]:='f';

b[i+1,j-1]:='h';

b[i,j]:='s';

must:=true;

end;

if (i-2>0) and (j+2<9) then

if (field[i-1,j+1]=op1) or (field[i-1,j+1]=op2) then

if (field[i-2,j+2]='0') then

begin

b[i-2,j+2]:='f';

b[i-1,j+1]:='h';

b[i,j]:='s';

must:=true;

end;

if (i-2>0) and (j-2>0) then

if (field[i-1,j-1]=op1) or (field[i-1,j-1]=op2) then

if (field[i-2,j-2]='0') then

begin

b[i-2,j-2]:='f';

b[i-1,j-1]:='h';

b[i,j]:='s';

must:=true;

end;

end;

if field[i,j]=f then

begin

way1:=false;

way2:=false;

way3:=false;

way4:=false;

k1:=true;

k2:=true;

k3:=true;

k4:=true;

g1:=true;

g2:=true;

g3:=true;

g4:=true;

for i1:=1 to 7 do

begin

if ((i+i1)<9) and ((j+i1)<9) then

begin

if (way1) and not (field[i+i1,j+i1]='0') then way1:=false;

if (g1) and (way1) and (field[i+i1,j+i1]='0') then

begin

b[i+i1,j+i1]:='f';

b[i,j]:='s';

b[mx1,my1]:='h';

must:=true;

end;

if (not (way1)) and (k1) and ((field[i+i1,j+i1]=op1) or (field[i+i1,j+i1]=op2)) then

begin

way1:=true;

k1:=false;

mx1:=i+i1;

my1:=j+i1;

end;

if not (field[i+i1,j+i1]='0') and (not (way1)) and (k1) then g1:=False;

end;

if ((i+i1)<9) and ((j-i1)>0) then

begin

if (way2) and not (field[i+i1,j-i1]='0') then way2:=false;

if (g2) and (way2) and (field[i+i1,j-i1]='0') then

begin

b[i+i1,j-i1]:='f';

b[i,j]:='s';

b[mx2,my2]:='h';

must:=true;

end;

if not (way2) and (k2) and ((field[i+i1,j-i1]=op1) or (field[i+i1,j-i1]=op2)) then

begin

way2:=true;

k2:=false;

mx2:=i+i1;

my2:=j-i1;

end;

if not (field[i+i1,j-i1]='0') and (not (way2)) and (k2) then g2:=False;

end;

if ((i-i1)>0) and ((j+i1)<9) then

begin

if (way3) and not (field[i-i1,j+i1]='0') then way3:=false;

if (g3) and (way3) and (field[i-i1,j+i1]='0') then

begin

b[i-i1,j+i1]:='f';

b[i,j]:='s';

b[mx3,my3]:='h';

must:=true;

end;

if not (way3) and (k3) and ((field[i-i1,j+i1]=op1) or (field[i-i1,j+i1]=op2)) then

begin

way3:=true;

k3:=false;

mx3:=i-i1;

my3:=j+i1;

end;

if not (field[i-i1,j+i1]='0') and (not (way3)) and (k3) then g3:=False;

end;

if ((i-i1)>0) and ((j-i1)>0) then

begin

if (way4) and not (field[i-i1,j-i1]='0') then way4:=false;

if (g4) and (way4) and (field[i-i1,j-i1]='0') then

begin

b[i-i1,j-i1]:='f';

b[i,j]:='s';

b[mx4,my4]:='h';

must:=true;

end;

if not (way4) and (k4) and ((field[i-i1,j-i1]=op1) or (field[i-i1,j-i1]=op2)) then

begin

way4:=true;

k4:=false;

mx4:=i-i1;

my4:=j-i1;

end;

if not (field[i-i1,j-i1]='0') and (not (way4)) and (k4) then g4:=False;

end;

end;

end;

end;

end;

procedure drawfight(x,y:integer);

var n,i,j,i1:Integer;

begin

for i:=1 to 8 do

for j:=1 to 8 do

if (b[i,j]='l') then b[i,j]:='f';

for i:=1 to 8 do

for j:=1 to 8 do

begin;

if (b[x,y]='s') then

begin

if (field[x,y]='b') or (field[x,y]='w') then

begin

if (b[x+2,y+2]='f') then b[x+2,y+2]:='l';

if (b[x+2,y-2]='f') then b[x+2,y-2]:='l';

if (b[x-2,y+2]='f') then b[x-2,y+2]:='l';

if (b[x-2,y-2]='f') then b[x-2,y-2]:='l';

end;

if (field[x,y]='v') or (field[x,y]='d') then

begin

for i1:=1 to 7 do

begin

if (b[x+i,j+i1]='f') then b[x+i,j+i1]:='l';

if (b[x+i,j-i1]='f') then b[x+i,j-i1]:='l';

if (b[x-i,j+i1]='f') then b[x-i,j+i1]:='l';

if (b[x-i,j-i1]='f') then b[x-i,j-i1]:='l';

end;

end;

end;

end;

for i:=1 to 8 do

for j:=1 to 8 do

begin

n:=i\*10+j;

if b[i,j]='l' then

begin

with form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\4.bmp');

end;

end

else if (field[i,j]='0') and ((i+j) mod 2 = 0) then

begin

with form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture:=nil;

end;

end;

end;

end;

procedure Checker;

var x,y:Integer;

begin

if (needx<>0) then begin

x:=needx;

y:=needy;

prevx:=x;

prevy:=y;

needx:=0;

needy:=0;

cc:=true;

end

else

if (work) then

begin

x:= nowx;

y:=nowy;

cc:=False;

if (x<>prevx) or (y<>prevy) then

begin

prevx:=x;

prevy:=y;

cc:=True;

end;

end;

work:=True;

end;

procedure movedo (x1,y1,x2,y2:Integer);

var g:Char; n,m:Integer;

begin

g:=field[x1,y1];

n:=x2\*10+y2;

m:=x1\*10+y1;

if g='b' then

begin

field[x1,y1]:='0';

field[x2,y2]:=g;

with Form1.FindComponent('Image' + IntToStr(m)) as TImage do

begin

Picture:=nil;

end;

with Form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\2.bmp');

end;

end;

if g='w' then

begin

field[x1,y1]:='0';

field[x2,y2]:=g;

with Form1.FindComponent('Image' + IntToStr(m)) as TImage do

begin

Picture:=nil;

end;

with Form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\0.bmp');

end;

end;

if g='v' then

begin

field[x1,y1]:='0';

field[x2,y2]:=g;

with Form1.FindComponent('Image' + IntToStr(m)) as TImage do

begin

Picture:=nil;

end;

with Form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\1.bmp');

end;

end;

if g='d' then

begin

field[x1,y1]:='0';

field[x2,y2]:=g;

with Form1.FindComponent('Image' + IntToStr(m)) as TImage do

begin

Picture:=nil;

end;

with Form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\3.bmp');

end;

end;

end;

procedure qween;

var i,j:Integer;

begin

for j:=1 to 8 do

begin

if (field[1,j]='b') then

begin

field[1,j]:='d';

with Form1.FindComponent('Image' + IntToStr(10+j)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\3.bmp');

end;

end;

if (field[8,j]='w') then

begin

field[8,j]:='v';

with Form1.FindComponent('Image' + IntToStr(80+j)) as TImage do

begin

Picture.LoadFromFile(path+'\pic\1.bmp');

end;

end;

end;

end;

procedure movehitdo (x1,y1,x2,y2:Integer);

var i,j,n,m:integer;

way1,way2,way3,way4:boolean;

begin

movedo(x1,y1,x2,y2);

way1:=false;

way2:=false;

way3:=false;

way4:=false;

if (x2>x1) and (y2>y1) then way1:=true;

if (x2>x1) and (y2<y1) then way2:=true;

if (x2<x1) and (y2>y1) then way3:=true;

if (x2<x1) and (y2<y1) then way4:=true;

if (way1) then

begin

for i:=x1 to x2 do

for j:=y1 to y2 do

if ((j+i) mod 2 = 0) then

if (b[i,j]='h') then

begin

field[i,j]:='0';

with Form1.FindComponent('Image' + IntToStr(10\*i+j)) as TImage do

begin

Picture:=nil;

end;

end;

end;

if (way2) then

begin

for i:=x1 to x2 do

for j:=y1 downto y2 do

if ((j+i) mod 2 = 0) then

if (b[i,j]='h') then

begin

field[i,j]:='0';

with Form1.FindComponent('Image' + IntToStr(10\*i+j)) as TImage do

begin

Picture:=nil;

end;

end;

end;

if (way3) then

begin

for i:=x1 downto x2 do

for j:=y1 to y2 do

if ((j+i) mod 2 = 0) then

if (b[i,j]='h') then

begin

field[i,j]:='0';

with Form1.FindComponent('Image' + IntToStr(10\*i+j)) as TImage do

begin

Picture:=nil;

end;

end;

end;

if (way4) then

begin

for i:=x1 downto x2 do

for j:=y1 downto y2 do

if ((j+i) mod 2 = 0) then

if (b[i,j]='h') then

begin

field[i,j]:='0';

with Form1.FindComponent('Image' + IntToStr(10\*i+j)) as TImage do

begin

Picture:=nil;

end;

end;

end;

for i:=1 to 8 do

for j:=1 to 8 do

begin

if (field[i,j]='0') and ((i+j) mod 2 = 0) then

with Form1.FindComponent('Image' + IntToStr(10\*i+j)) as TImage do

begin

Picture:=nil;

end;

end;

end;

procedure cani(Player:Char);

var i,j,k,i1,i2:Integer; l:Char;

begin

k:=0;

if Player='w' then

begin

i1:=1;

l:='v'

end

else

begin

l:='d';

i1:=-1;

end;

for i:=1 to 8 do

for j:=1 to 8 do

begin

if field[i,j]=Player then

begin

if (i+i1<9) and (i+i1>0) and (j+1<9) and (field[i+i1,j+1]='0') then k:=k+1;

if (i+i1<9) and (i+i1>0) and (j-1>0) and (field[i+i1,j-1]='0') then k:=k+1;

end;

if field[i,j]=l then

for i2:=1 to 7 do

begin

if ((x+2)<9) and ((y+2)<9) and (field[i+2,j+2]='0')then k:=k+1;

if ((x+2)<9) and ((y-2)>0) and (field[i+2,j-2]='0')then k:=k+1;

if ((x-2)>0) and ((y+2)<9) and (field[i-2,j+2]='0')then k:=k+1;

if ((x-2)>0) and ((y-2)>0) and (field[i-2,j-2]='0')then k:=k+1;

end;

end;

if (k = 0) then

begin

win:=True;

if Player='w' then pointb:=12

else pointw:=12;

end;

end;

procedure clean;

var i,j,n:Integer;

begin

for i:=1 to 8 do

for j:=1 to 8 do

begin

n:=i\*10+j;

if (field[i,j]='0') and ((i+j) mod 2 = 0) then

with Form1.FindComponent('Image' + IntToStr(n)) as TImage do

begin

Picture:=nil;

end;

end;

end;

procedure count;

begin

if player = 'w' then Form1.Label7.Caption:='White'

else Form1.Label7.Caption:='Black';

Form1.Label4.Caption:=IntToStr(pointw);

Form1.Label5.Caption:=IntToStr(pointb);

end;

procedure Main;

var olo:Char;

i,j:Integer;

m:Boolean;

begin

qween;

checkwin;

if win then

begin

Form1.Timer1.Enabled:=False;

if (pointb>=12) and (w1) then ShowMessage('Black player won!');

if (pointw>=12) and (w1) then ShowMessage('White player won!');

end

else

if not (ka='0') then

begin

Checker;

if not (skip) then

begin

skip:=True;

x1:=x2;

y1:=y2;

cc:=True;

end;

if cc then begin

x2:=nowx;

y2:=nowy;

end;

end;

case Player of

'w':

begin

if ka = '0' then Checker;

work:=True;

if (cc) then

begin

if (l1) then

fight('w');

if (must) then

begin

if (x1=0) then

begin

x1:=nowx;

y1:=nowy;

if (b[x1,y1]='s') then

begin

drawfight(x1,y1);

s1:=True;

end

else

begin

x1:=0;

y1:=0;

clean;

end;

end

else

begin

if (s1) then

begin

drawfight(x1,y1);

x2:=nowx;

y2:=nowy;

if (b[x2,y2]='l') then

begin

movehitdo(x1,y1,x2,y2);

cani('b');

pointw:=pointw+1;

m:=False;

must:=False;

fight('w');

drawfight(x2,y2);

if (b[x2,y2]='s') then m:=True;

if (m) then

begin

ka:='w';

clean;

count;

skip:=False;

m:= False;

end

else

begin

y1:=0;

x1:=0;

x2:=0;

y2:=0;

must:=false;

Player:='b';

l1:=True;

s1:=false;

clean;

count;

skip:=False;

ka:='0';

end;

end

else

begin

needy:=y2;

needx:=x2;

x1:=0;

x2:=0;

y1:=0;

y2:=0;

end;

end;

end;

end

else

begin

l1:=False;

if (x1=0) then

begin

x1:=nowx;

y1:=nowy;

move('w',x1,y1);

drawmove;

end

else

begin

x2:=nowx;

y2:=nowy;

olo:=b[x2,y2];

if (olo='m') then

begin

movedo(x1,y1,x2,y2);

cani('b');

x1:=0;

x2:=0;

y1:=0;

y2:=0;

Player:='b';

count;

l1:=True;

clean;

end;

if not (olo='m') then

begin

needy:=y2;

needx:=x2;

move('w',x1,y1);

x1:=0;

x2:=0;

y1:=0;

y2:=0;

clean;

work:=False;

l1:=true;

end;

end;

end;

end;

end;

'b':

begin

if ka = '0' then Checker;

work:=True;

if (cc) then

begin

if (l1) then

fight('b');

if (must) then

begin

if (x1=0) then

begin

x1:=nowx;

y1:=nowy;

if (b[x1,y1]='s') then

begin

drawfight(x1,y1);

s1:=True;

end

else

begin

x1:=0;

y1:=0;

clean;

end;

end

else

begin

if (s1) then

begin

drawfight(x1,y1);

x2:=nowx;

y2:=nowy;

if (b[x2,y2]='l') then

begin

movehitdo(x1,y1,x2,y2);

cani('w');

pointb:=pointb+1;

m:=False;

must:=False;

fight('b');

drawfight(x2,y2);

if (b[x2,y2]='s') then m:=True;

if m then

begin

ka:='b';

clean;

count;

skip:=False;

m:=False;

end

else

begin

x1:=0;

y1:=0;

x2:=0;

y2:=0;

must:=false;

Player:='w';

l1:=True;

s1:=False;

clean;

count;

skip:=False;

ka:='0';

end;

end

else

begin

needy:=y2;

needx:=x2;

x1:=0;

x2:=0;

y1:=0;

y2:=0;

clean;

end;

end;

end;

end

else

begin

l1:=False;

if (x1=0) then

begin

x1:=nowx;

y1:=nowy;

move('b',x1,y1);

drawmove;

end

else

begin

x2:=nowx;

y2:=nowy;

olo:=b[x2,y2];

if (olo='m') then

begin

movedo(x1,y1,x2,y2);

cani('w');

x1:=0;

x2:=0;

y1:=0;

y2:=0;

Player:='w';

count;

l1:=True;

clean;

end;

if not (olo='m') then

begin

needy:=y2;

needx:=x2;

move('w',x1,y1);

x1:=0;

x2:=0;

y1:=0;

y2:=0;

clean;

work:=False;

l1:=true;

end;

end;

end;

end;

end;

end;

end;

procedure TForm1.N6Click(Sender: TObject);

begin

if (not Assigned(Form2)) then

Form2:=TForm2.Create(Self);

Form2.Show;

end;

procedure TForm1.N4Click(Sender: TObject);

begin

Close;

end;

procedure TForm1.N2Click(Sender: TObject);

begin

dofield;

clean;

Player:='w' ;

pointw:=0;

pointb:=0;

w1:=true;

must:=False;

work:=True;

x1:=0;

y1:=0;

x2:=0;

y2:=0;

l1:=True;

l2:=True;

count;

ka:='0';

Label7.Visible:=True;

Label6.Visible:=True;

Timer1.Enabled:=True;

end;

procedure TForm1.Image11Click(Sender: TObject);

begin

nowx:=1;

nowy:=1;

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

Main;

end;

procedure TForm1.Image22Click(Sender: TObject);

begin

nowx:=2;

nowy:=2;

end;

procedure TForm1.Image13Click(Sender: TObject);

begin

nowx:=1;

nowy:=3;

end;

procedure TForm1.Image15Click(Sender: TObject);

begin

nowx:=1;

nowy:=5;

end;

procedure TForm1.Image17Click(Sender: TObject);

begin

nowx:=1;

nowy:=7;

end;

procedure TForm1.Image24Click(Sender: TObject);

begin

nowx:=2;

nowy:=4;

end;

procedure TForm1.Image26Click(Sender: TObject);

begin

nowx:=2;

nowy:=6;

end;

procedure TForm1.Image28Click(Sender: TObject);

begin

nowx:=2;

nowy:=8;

end;

procedure TForm1.Image31Click(Sender: TObject);

begin

nowx:=3;

nowy:=1;

end;

procedure TForm1.Image33Click(Sender: TObject);

begin

nowx:=3;

nowy:=3;

end;

procedure TForm1.Image35Click(Sender: TObject);

begin

nowx:=3;

nowy:=5;

end;

procedure TForm1.Image37Click(Sender: TObject);

begin

nowx:=3;

nowy:=7;

end;

procedure TForm1.Image42Click(Sender: TObject);

begin

nowx:=4;

nowy:=2;

end;

procedure TForm1.Image44Click(Sender: TObject);

begin

nowx:=4;

nowy:=4;

end;

procedure TForm1.Image46Click(Sender: TObject);

begin

nowx:=4;

nowy:=6;

end;

procedure TForm1.Image48Click(Sender: TObject);

begin

nowx:=4;

nowy:=8;

end;

procedure TForm1.Image51Click(Sender: TObject);

begin

nowx:=5;

nowy:=1;

end;

procedure TForm1.Image53Click(Sender: TObject);

begin

nowx:=5;

nowy:=3;

end;

procedure TForm1.Image55Click(Sender: TObject);

begin

nowx:=5;

nowy:=5;

end;

procedure TForm1.Image57Click(Sender: TObject);

begin

nowx:=5;

nowy:=7;

end;

procedure TForm1.Image62Click(Sender: TObject);

begin

nowx:=6;

nowy:=2;

end;

procedure TForm1.Image64Click(Sender: TObject);

begin

nowx:=6;

nowy:=4;

end;

procedure TForm1.Image66Click(Sender: TObject);

begin

nowx:=6;

nowy:=6;

end;

procedure TForm1.Image68Click(Sender: TObject);

begin

nowx:=6;

nowy:=8;

end;

procedure TForm1.Image71Click(Sender: TObject);

begin

nowx:=7;

nowy:=1;

end;

procedure TForm1.Image73Click(Sender: TObject);

begin

nowx:=7;

nowy:=3;

end;

procedure TForm1.Image75Click(Sender: TObject);

begin

nowx:=7;

nowy:=5;

end;

procedure TForm1.Image77Click(Sender: TObject);

begin

nowx:=7;

nowy:=7;

end;

procedure TForm1.Image82Click(Sender: TObject);

begin

nowx:=8;

nowy:=2;

end;

procedure TForm1.Image84Click(Sender: TObject);

begin

nowx:=8;

nowy:=4;

end;

procedure TForm1.Image86Click(Sender: TObject);

begin

nowx:=8;

nowy:=6;

end;

procedure TForm1.Image88Click(Sender: TObject);

begin

nowx:=8;

nowy:=8;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

path:=ExtractFileDir(Application.ExeName);

end;

procedure TForm1.N3Click(Sender: TObject);

var i,j:integer;

begin

for i:=1 to 8 do

for j:=1 to 8 do

if (i+j) mod 2 = 0 then

begin

field[i,j]:='0';

b[i,j]:='0';

end;

Form1.Label4.Caption:='';

Form1.Label5.Caption:='';

clean;

Player:='0' ;

pointw:=0;

pointb:=0;

w1:=true;

must:=False;

work:=True;

x1:=0;

y1:=0;

x2:=0;

y2:=0;

l1:=True;

l2:=True;

count;

ka:='0';

Label7.Visible:=False;

Label6.Visible:=False;

Timer1.Enabled:=False;

end;

procedure TForm1.N7Click(Sender: TObject);

begin

if (not Assigned(Form3)) then

Form3:=TForm3.Create(Self);

Form3.Show;

end;

end

# 

# 5.Список використаної літератури

1. Програмування та алгоритмічні мови (мова прогр. Паскаль, частина І): Метод. вказівки до комп’юерн. практикуму для студентів напрямів підготовки 6.040303 «Системний аналіз», 6.050101 «Комп’ютерні науки»/ Уклад.: Г.Г.Швачко, І.В.Назарчук. – К. ННК «ІПСА» НТУУ «КПІ», 2011. – 106с.
2. Програмування та алгоритмічні мови (мова прогр. Паскаль, частина ІІ): Метод. вказівки до комп’юерн. практикуму для студентів напрямів підготовки 6.040303 «Системний аналіз», 6.050101 «Комп’ютерні науки»/ Уклад.: Г.Г.Швачко, І.В.Назарчук. – К. ННК «ІПСА» НТУУ «КПІ», 2011. – 106с.
3. Швачко Г.Г. Конспект лекцій з програмування.
4. Правила игры "Русские шашки" [Электронный ресурс] - https://ru.wikipedia.org/wiki/Русские\_шашки