МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра “Комп’ютерної математики

та математичного моделювання”

Курсова робота

Дисципліна

«Об’єктно-орієнтоване програмування»

Тема

«Хрестики-нулики на полі гри 10х10»

Виконавець:
студентка групи ІФ-57а Самилова А.О.

Харків

Содержание

Введение

Крестики-нолики — логическая игра между двумя противниками на прямоугольном поле 3 на 3 клетки или большего размера (вплоть до «бесконечного поля»). Один из игроков играет «крестиками», второй — «ноликами». Разновидностями этой игры есть рэндзю и го-моку (5 в ряд).

Крестики-нолики - весьма известная игра, как у нас, так и в англоязычном мире, где она называется tic-tac-toe или Nine Man Morris. Поразительно то, что в эту игру, в ее различные варианты играют дети и взрослые на протяжении более 3000 лет. Известно, что в tic-tac-toe играли в Египте около 1300 г. до н. э. История происхождения крестиков-ноликов неоднозначна. Одни источники указывают на то, что прототипом крестиков-ноликов была древнеримская игра Terni Lapilli, другие – на ее восточное происхождение.

Terni Lapilli - древнеримская игра для двух игроков. Смысл игры в том, чтобы пытаться выстроить ряд из 5 своих фигурок, по диагонали, горизонтали или вертикали. Игроки ходят поочередно. Важно следить за соперником и мешать ему сделать то же самое.

Игра на поле 15х15 имеет много общего с игрой рэндзю, которая возникла во втором тысячелетии до н.э. на территории Китая и вскоре распространилась по всей территории востока. Игра претерпевала некоторые изменения в правилах и меняла свои названия. Есть все основания полагать, что современная игра крестики-нолики берет свои начала именно от игры рэндзю. Однако, рэндзю, в отличие от игры крестики-нолики, имеет более сложные правила, ограничивающие ходы первого игрока с целью уравнивания шансов выигрыша, а игра крестики-нолики подобных ограничений не имеет, что делает ее более популярной. А ее упрощенная версия на поле размером 3Х3 делает эту игру доступной даже детям, именно поэтому игра крестики-нолики пользуется огромной популярностью у детей младшего школьного возраста.

Классические «крестики-нолики» на поле 3x3 не представляют никакого практического интереса — общеизвестен алгоритм, который при правильной игре гарантирует ничью любой стороне, а при ошибке противника позволяет выиграть. Таким образом, игра находится в состоянии «ничейной смерти».

Ничейная смерть — этап развития логической игры, когда разработанность теории достигает уровня, позволяющего любому владеющему теорией игроку, независимо от квалификации противника, свести партию вничью.

Однако при n = 5 игра становится намного содержательнее. Такой вариант имеет специальное название — го-моку.

Основной победной тактикой при игре на бесконечном поле считается построение пересечений («вилок»), которые не дают противнику возможности блокировать все возможные пути построения пятёрки. Чтобы не проиграть, необходимо своевременно прерывать линии противника длиной в три фигуры.

История компьютерных крестиков-ноликов восходит к самому началу современных электронных компьютеров и продолжается в некоторых самых последних исследованиях в области искусственного интеллекта.

OXO (также известна как **Noughts And Crosses**) - первая программа предназначена для воспроизведения тактики игры tic-tac-toe для компьютера EDSAC. Была написана в Кембриджском университете в 1952 году А. С. Дугласом как иллюстрация к кандидатской диссертации на тему взаимодействия человека и компьютера.

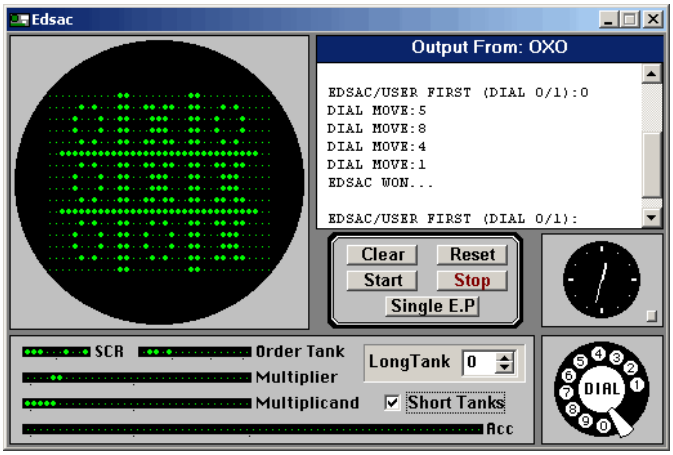


Рис. 1. OXO.

Заданием моего курсового проекта является реализация игры крестики нолики – популярной игры многих поколений. Играя в эту игру, можно не только интересно коротать время, но и развивать свои способности.

Постановка задачи

Назначение программы: программная реализация популярной игры крестики нолики на поле 10*10 и система управления «искусственным интеллектом».

Правила игры крестики нолики:

Игроки по очереди ставят на свободные клетки поля 10*10 знаки (один всегда крестики, другой нолики). Первый, выстроивший в ряд 5 своих фишек по вертикали, горизонтали или диагонали, побеждает. Первый ход делает игрок, ставящий крестики. По завершении партии выигравшая сторона зачёркивает чертой свои 5 знаков (нолики или крестики), которые составляют сплошной ряд.

Программа должна реализовать игру между пользователем и компьютером («искусственным интеллектом»).

Для игры нужно, чтобы программа предоставляла удобный визуальный интерфейс, т. е. содержала игровое поле, осуществляла наглядное отображение ходов игроков, сообщала о конечном результате игры.

В функциональность программы также должна входить система для подключения сторонних библиотек с «искусственным интеллектом» для игры. Необходимым является интерфейс взаимодействия игры и библиотеки ИИ.

Архитектура программы

Программу «Крестики-нолики (5 в ряд)» можно условно разделить на три части:

- первая – хранит информацию о расположении крестиков и ноликов, выполняет расчет очередного хода;
- вторая – рисует игровое поле, выполняет обработку команд пользователя;
- третья – отвечает за игру компьютера.

Реализация проекта осуществлена с помощью следующих классов (см. Unit1.h, Unit1.cpp):

- **TCell** –класс ячейки, содержит поля x, y и value;
- **TMatrixGame** – класс игрового поля. Содержит такие поля (параметры) как матрицу состояния игры с возможными значениями 0, 1, 2. Методы: анализ возможных ходов игроков, организация последовательности ходов, изменение матрицы состояния, оценка выигрыша (завершение игры);
- **TUserForm** – класс, который осуществляет отрисовку игрового поля, выполняет обработку команд пользователя и т.п.

В программе используется динамически подключаемая библиотека DLL (см. .../DLL/MyDLL.cpp), в которой находятся оценочные функции для игры компьютера в крестики-нолики.

Рассмотрим каждый из классов более подробно.

Класс TCell служит для хранения параметров расположения одной ячейки: номеров строки и столбца, значения веса. Структура класса показана на рис. 2.

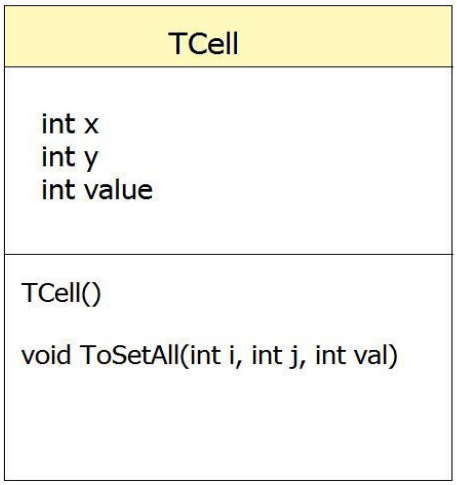


Рис. 2. Класс TCell.

Метод ToSetAll устанавливает значения ячейки.

Всю работу, связанную с хранением информации об игре и выбором очередных ходов, выполняет класс TGameMatrix. Это класс, отвечающий за организацию правил игры. Содержит такие поля:

- cross, zero, vacant – постоянные для обозначения состояний значений ячеек в массиве поля. В игре соответственно инициализированы как 1, 2, 0.

is_next_cross – переменная, указывающая на очередь последующего символа. По умолчанию равна true. Изменяет свое значение в ходе игры.

**GMatrix – квадратная матрица игры, двухмерный динамический массив целого типа.

N – размер матрицы GMatrix.

win_number – постоянное число равное количеству символов в ряде, которые необходимо для выигрыша. В классе win_number=5.

count_of_free – количество свободных ячеек в GMatrix.

BegPoint, EndPoint – выигрышные точки.

Ниже (рис. 3) указана полная структура класса TGameMatrix.

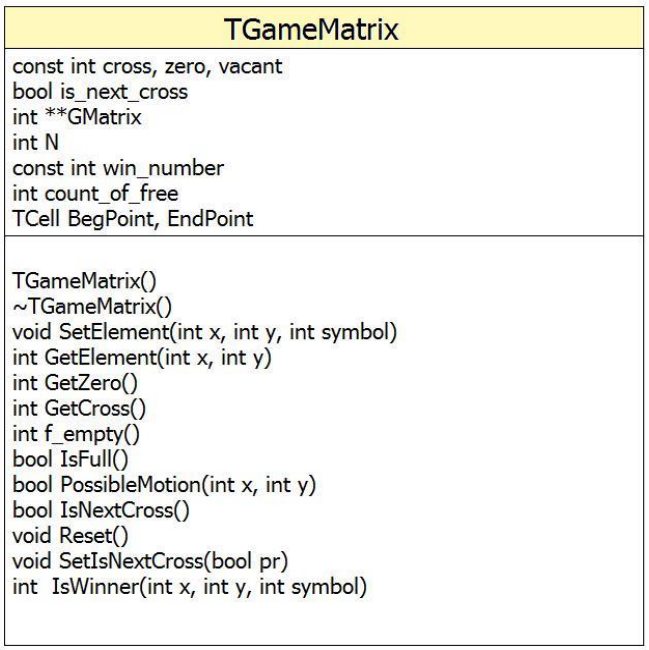


Рис. 3. Класс TGameMatrix.

С помощью метода SetElement можно добавить крестик или нолик в заданную ячейку. Метод GetElement возвращает элемент, расположенный в заданной ячейке массива. GetZero, GetCross, f_empty возвращают число целого типа, что служит за состояние ячейки: нолик, крестик, пусто.

IsFull – функция отвечающая за проверку полноты заполнения матрицы поля: если все ячейки в массиве заняты, то возвращает true; если есть свободные ячейки, то возвращает false.

Методы PossibleMove и IsNextCross служат для регулировки и проверки ходов игроков. Если ход игрока возможен и не нарушает правил игры, то PossibleMove возвращает true. IsNextCross проверяет чей сейчас ход: крестика или нолика. Если ход крестика, то возвращает значение true; если нолика, то false.

Метод Reset обнуляет все значения ячеек массива GMatrix, сбрасывает параметры следующего хода и устанавливает их в начальное положение.

SetIsNextCross служит для установки хода крестика или нолика. Используется только при загрузке сохраненного сеанса игры.

Метод IsWinner позволяет определить кто выиграл игру (крестик или нолик). Считается, что выигрывает тот, кто составит пять своих символов подряд по горизонтали, вертикали или по диагоналям.

Деструктор ~TGameMatrix очищает массив GMatrix и освобождает память.

Вторая часть программы реализована классом TUserGame, который является наследником стандартного класса TForm. Он отвечает за прорисовку всех элементов игры (создание окна, меню, крестики, нолики и т.д.), взаимодействие с пользователем (обработка сообщений мыши).

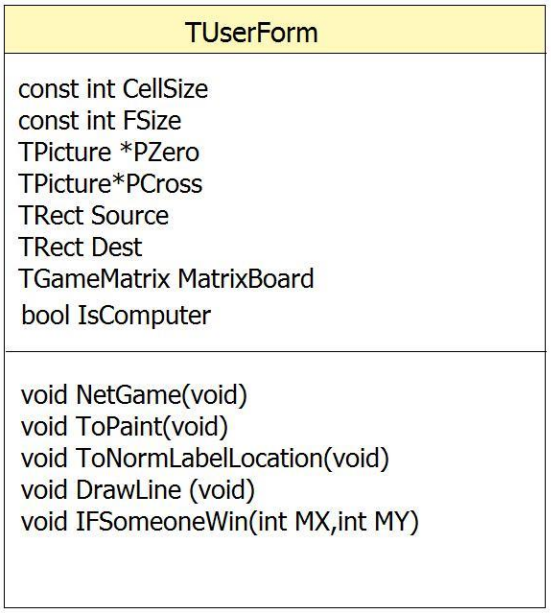


Рис. 4. Класс TUserForm.

Класс содержит следующие поля:

FSize – целочисленная постоянная, которая задает размер формы. В программе FSize=500.

CellSize – целочисленная постоянная, отвечающая за размер ячейки на форме. В программе CellSize=50.

PZero, PCross – указатели на объекты класса TPicture. Служат для хранения изображений соответственно нолика и крестика.

MatrixBoard – объект класса TGameMatrix.

IsComputer – переменная, отображающая режим игры: с компьютером или с другим пользователем. По умолчанию равна false.

Методы класса TUserForm:

NetGame осуществляет рисование сетки игрового поля. В нем также реализована загрузка изображений крестика и нолика в соответствии с выбранной темой игры.

ToPaint отвечает за рисование игрового поля по значениям массива MatrixBoard.GMatrix.

DrawLine осуществляет рисование линии на игровом поле, которая зачеркивает ряд символов победителя.

IFSomeoneWin – метод, который осуществляет проверку победы одного из игроков. В случае победы одного из игроков или ничьи выдает соответствующие сообщения.

Третья часть программы, отвечающая за игру компьютера, реализована с помощью оценочной функции, которая содержится в динамически подключаемой библиотеке MyDLL.

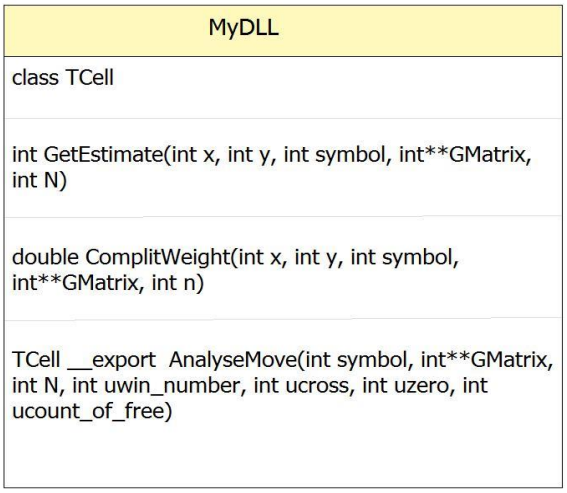


Рис. 5. Содержимое MyDLL

Динамически присоединяемая библиотека DLL — это одна из возможностей повторного использования разработанных кодов. DLL — это специального вида исполняемый файл с расширением .dll, используемый для хранения функций и ресурсов отдельно от исполняемого файла.

Содержимое MyDLL указано на рис. 5. Функции GetEstimate и CompliteWeight формируют веса значений ячеек, а функция AnalyseMove их анализирует и выбирает ячейку с максимальным весом (подробно оценочная функция будет рассмотрена в разделе Алгоритм). AnalyseMove в качестве результата экспортирует объект класса TCell.

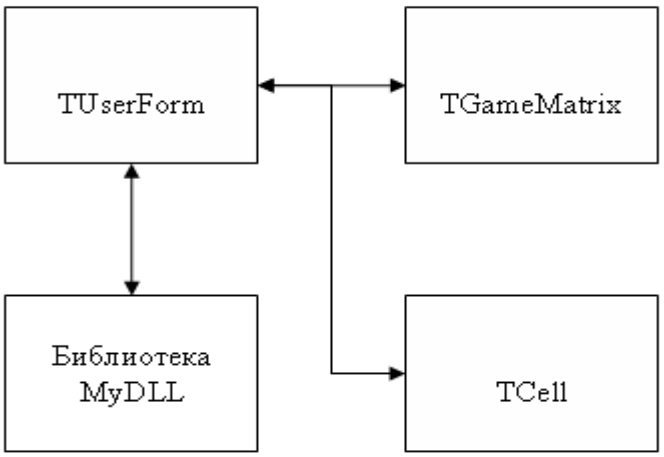


Рис. 6. Взаимодействие частей программы.

Алгоритм

Для осуществления игры компьютера используется оценочная функция.

Суть оценочной функции - оценить насколько выгодно нам поставить в данную точку свою фишку. Очевидно, нам бывает выгодно это сделать либо для создания своего длинного ряда, либо для блокирования длинного ряда противника. Также следует учесть, что бывает выгоднее заблокировать большое количество не очень длинных рядов, вместо одного длинного.

Фишка, поставленная в данную пустую клетку может одновременно участвовать в продолжении до 8 рядов (2 горизонтальных, 2 вертикальных и 4 диагональных). Считаем, что мы поставили фишку в данное место. Тогда можно сосчитать длинны каждого из наших рядов, включающих эту фишку. Введем коэффициент M = sum(Ki). Где Ki - коэффициент важности i-го ряда.

Т.к. направление ряда нам безразлично, то Ki зависит только от длинны ряда. Для простоты можно взять Ki=5*длинна ряда. Полученный коэффициент M - оценка той выгоды, которую мы получим, поставив в данную клетку свою фишку. Далее предположим, что мы не поставили в данную клетку фишку, и соответственно это сделал противник. Аналогично считаем коэффициент N - оценка выгоды, получаемой противником. Сложив M и N с неким оценочным коэффициентом, получим окончательную оценку: F = M + Q*N. Коэффициент Q - показатель агрессивности алгоритма, если он больше 1 - алгоритм сидит в глухой обороне; меньше 1 - алгоритм пытается захватить инициативу.

Описание программы

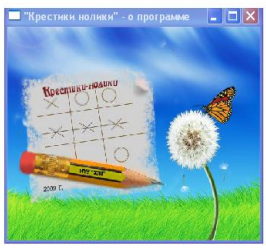
Приложение состоит из трех форм:

- UserForm;
- FormHelp;
- FAboutProg.

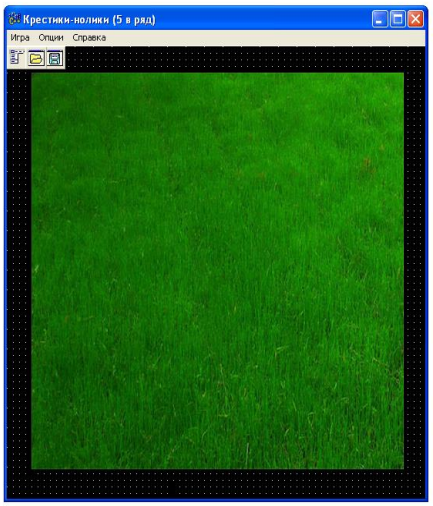
В курсовом проекте по реализации игры «Крестики-нолики» были использованы следующие компоненты:

Tform

- MainManu: MainManu1 (UserForm).
- Image: ImField (UserForm); Image1, Image2 (FormHelp); Image1 (FAboutProg).
- OpenDialog: OpenDialog1 (UserForm).
- SaveDialog: SaveDialog1 (UserForm).
- RichEdit: RichEdit1 (FormHelp).
- StaticText: StaticText1 (FormHelp).



FAboutProg



UserForm

FormHelp

Рис. 7. Формы проекта.

Программа реализована на языке C++ Builder 6.0 разработчика Borland.

Проект представлен в виде совокупности файлов:

Файл	Размер
Project1.bpr	4 320
Project1.cpp	1 302
Project1.obj	18 275
Project1.exe	1 072 128
Project1.res	876
Project1.tds	3 342 336
Unit1.cpp	11 992
Unit1.ddp	51
Unit1.dfm	1 692 244
Unit1.h	7 907
Unit1.obj	326 244
Unit2.cpp	977
Unit2.ddp	51
Unit2.dfm	336 300
Unit2.h	993

Файл	Размер
Unit2.obj	45 113
Unit3.cpp	751
Unit3.ddp	51
Unit3.dfm	41 453
Unit3.h	1 030
Unit3.obj	75 895
MyDLL.tds	786 432
MyDLL.obj	19 432
MyDLL.dll	13 824
MyDLL.lib	11 188
MyDLL.cpp	5 028
MyDLL.bpr	3 417
MyDLL.res	876
MyDLL.bpf	101

Руководство пользователя

Использование компонентов описанных в пункте выше, создает удобный и простой в использовании интерфейс программы, что является очень важной частью в создании игр. Ведь, прежде всего, интерфейс должен быть не нагроможденным, но в то же время содержать компоненты, которые бы реализовывали все задачи необходимые пользователю в игре.

При запуске программы перед пользователем появляется игровое поле, по умолчанию имеющее тему «футбольное поле», в случае необходимости пользователь может с легкостью поменять тематику, об этом подробнее будет описано позже. Вверху появившегося приложения расположено меню, которое содержит следующие разделы:

-Игра;

-Опции;

-Справка.

При выборе раздела «Игра» появляются следующие пункты:

-Новая игра;

-Сохранить;

-Загрузить;

-Выход.

Если выбрать пункт меню «Новая игра», то начинается собственно игра. По умолчанию игра настроена на двух игроков, если игрок хочет сыграть против компьютера, то ему необходимо только лишь поменять соответствующий параметр «с другим пользователем» в этом пункте на «с компьютером».

При необходимости пользователь имеет возможность сохранить текущую игру, для этого ему необходимо выбрать пункт меню «Сохранить» или нажать комбинацию клавиш Ctrl+S.Чтобы продолжить какую-то одну из сохраненных ранее пользователем игр, необходимо выбрать пункт «Загрузить».

Для того чтобы выйти из игры существует несколько способов.

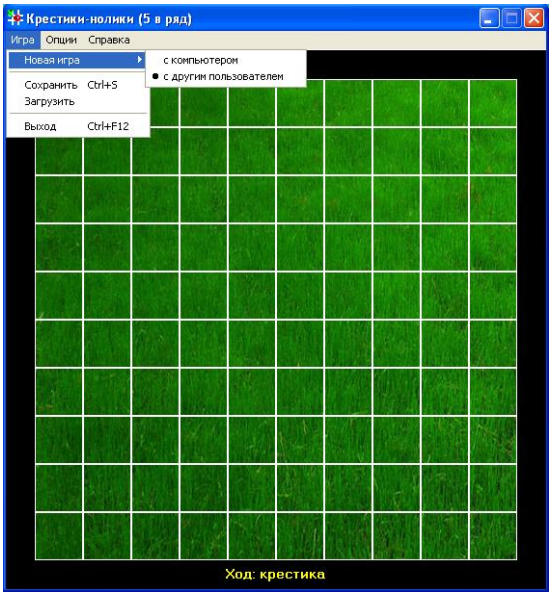


Рис. 8. Интерфейс игры.

Первый выбрать соответствующий пункт меню, второй – нажав комбинацию клавиш Ctrl+F12, или просто закрыв окно приложения.

При выборе раздела «Опции» появляется пункт «Тема игры», в которой содержатся два подпункта, соответствующие двум тематикам «морское дно» и «футбольное поле». Тематики игры разработаны для того, чтобы сделать игру более яркой, интересной и увлекательной.

Рис. 9. Темы игры

Для того чтобы ознакомиться с правилами игры, необходимо выбрать раздел меню «Справка» и пункт «О правилах игры», или нажать клавишу F1. При этом перед пользователем появляется еще одно окно, в котором детально описаны правила и руководство в использовании игры.

Если пользователя интересует информация о разработчике программы, он может её посмотреть выбрав раздел меню «Справка» и пункт «О программе…».

Итак, игрок начал игру с компьютером. Играет пользователь за «крестики», а компьютер за «нолики».Игра продолжается до тех пор, пока один из игроков не составит линию из пяти своих символов. В этом случае появляется сообщение о том, кто выиграл «крестики» или «нолики». И для наглядности линия из пяти символов перечеркивается, как это делается в классической игре «Крестики-нолики».

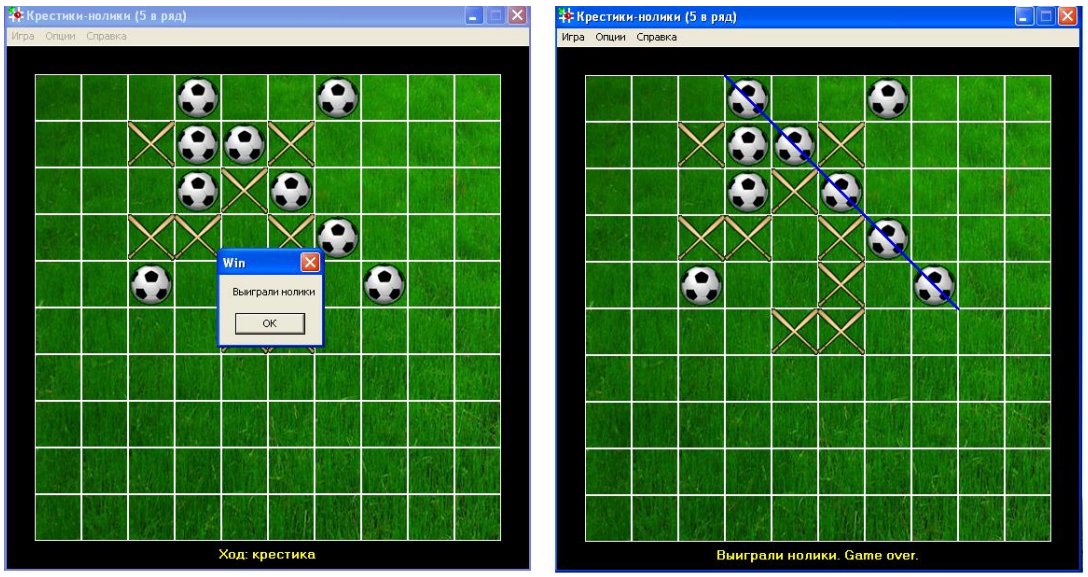


Рис. 10. Конец сеанса игры.

Заключение

Данный курсовой проект посвящен игре «Крестики – нолики на поле 10*10». Играя в нее не только можно приятно коротать время, но и развивать собственные способности: внимание, мышление, реакцию быстрого принятия решения.

В ходе выполнения курсовой работы мной было изучено создание динамически подключаемых библиотек, разработано стратегию игры компьютера, реализовано программу «Крестики-нолики (5 в ряд)» в соответствии с поставленной задачей.

В функциональность программы входит интерфейс взаимодействия игры и библиотеки с «искусственным интеллектом» (компьютером). Программа имеет красочный и приятный пользовательский интерфейс удобный в использовании. Эта версия программы не является окончательной. Возможно ее дальнейшее усовершенствование.

Список использованных источников

1. Павловская Т. А. C/C++. Программирование на языке высокого уровня. – СПб Питер, 2007. – 461 с.

2. Архангельский А. Я. Программирование в C++ Builder 6. Москва: Бином, 2003. – 1152 с.

3. Р. Хаггарити Дискретная математика для программистов. Москва: Техносфера,2004. – 320 с.

4. С. В. Мациевский. Математическая культура: Учебное пособие / Калининград: Изд-во КГУ, 2001. – 72 с.

5. http://ru.wikipedia.org/wiki/OXO.

Приложение А

Текст программы

```
//-------------------------Unit.h-----------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
….
//---------------------------------------------------------------------------
class TUserForm : public TForm
{
__published:    // IDE-managed Components
        TMainMenu *MainMenu1;
        TMenuItem *MGame;
        TMenuItem *MOptions;
        TMenuItem *MHelp;
        TMenuItem *MNewGame;
        TMenuItem *MExit;
        TMenuItem *MAboutRules;
        TMenuItem *MAboutProgram;
        TMenuItem *MTopicGame;
        TMenuItem *MSave;
        TMenuItem *MLoad;
        TMenuItem *N1;
        TMenuItem *N2;
        TMenuItem *N3;
        TImage *ImField;
        TImage *IBoder;
        TMenuItem *MtgSea;
        TMenuItem *MtgFootball;
  TLabel *SocerBoard;
  TMenuItem *NGKomp;
  TMenuItem *NGUser;
  TSaveDialog *SaveDialog1;
  TOpenDialog *OpenDialog1;
        void __fastcall MExitClick(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall MAboutProgramClick(TObject *Sender);
        void __fastcall ImFieldMouseDown(TObject *Sender,
          TMouseButton Button, TShiftState Shift, int X, int Y);
        void __fastcall FormDestroy(TObject *Sender);
        void __fastcall MAboutRulesClick(TObject *Sender);
        void __fastcall MtgSeaClick(TObject *Sender);
        void __fastcall MtgFootballClick(TObject *Sender);
  void __fastcall NGUserClick(TObject *Sender);
  void __fastcall NGKompClick(TObject *Sender);
  void __fastcall MSaveClick(TObject *Sender);
  void __fastcall MLoadClick(TObject *Sender);
```

```
private:          // User declarations
public:
          void NetGame(void);
  void ToPaint(void);
  void ToNormLabelLocation(void);
  void DrawLine (void);
  bool IsComputer;
  void IFSomeoneWin(int MX,int MY);// функция проверки и приветствия победителя
       __fastcall TUserForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TUserForm *UserForm;
//---------------------------------------------------------------------------
class TCell
{
public:
   int x,y,value;
   TCell()
   {
     randomize();
     x=random(10);
     y=random(10);
     value=0;
   }
   void ToSetAll(int i,int j,int val)
   {
    x=i;
    y=j;
    value=val;
   }

};
//---------------------------------------------------------------------------
class TGameMatrix
{
private:

    const int cross,zero,vacant; //vacant -пустой
    bool is_next_cross;
public:
    int  **GMatrix;
    const int N;   // размер поля
    const int win_number;
    int count_of_free;
    TCell BegPoint,EndPoint; //выигрышные точки
    TGameMatrix():N(10),cross(1),zero(2),vacant(0),win_number(5)
    {
      GMatrix=new int*[N];
      for (int i=0;i<N;i++)
      {
        GMatrix[i]=new int[N];
        for(int j=0;j<N;j++)
          GMatrix[i][j]=vacant;
      }
```

```
  is_next_cross=true;
  count_of_free=N*N;
  BegPoint.ToSetAll(-1,-1,-1);
  EndPoint.ToSetAll(-1,-1,-1);// пока нет выигрышной линии
}
~TGameMatrix()
{
 for(int i=0;i<N;i++)
   delete[]GMatrix[i];
 delete[]GMatrix;
}
void SetElement(int x,int y,const int symbol)
{
  if((x<N)&&(y<N))
  {
    switch(symbol)
    {
    case 1 : GMatrix[x][y]=cross; is_next_cross=false;count_of_free--; break;
    case 2:  GMatrix[x][y]=zero;is_next_cross=true;count_of_free--; break;
    default: GMatrix[x][y]=vacant;break;
    }
  }
}
int GetElement( int x,int y)
{
   if((x<N)&&(y<N))
     return GMatrix[x][y];
   return -1;
}
int GetZero()
{
 return zero;
}
int GetCross()
{
 return cross;
}
int f_empty ()
{
  return vacant;
}
bool IsFull()
{
  if(count_of_free!=0)
     return false;
  return true;
}
bool PossibleMotion(int x,int y)
{
  int El=GetElement(x,y);
  if(El==vacant)
    return true;
  return false;
}
```

```
void Reset ()
{
  for (int i=0;i<N;i++)
      for(int j=0;j<N;j++)
         GMatrix[i][j]=vacant;
  is_next_cross=true;
  count_of_free=N*N;
  BegPoint.ToSetAll(-1,-1,-1);
  EndPoint.ToSetAll(-1,-1,-1);
}
bool IsNextCross()
{
    return is_next_cross;
}
void SetIsNextCross(bool pr)
{
    is_next_cross=pr;
}
int IsWinner(int x,int y,int symbol)
{
   int ccount=0;// количество  идущих подряд символов данного типа
   int dlcount=0;// по левой диагонали
   int drcount=0;// по правой диагонали
   for(int i=x-win_number;i<x+win_number;i++)
   {
      if((i>=0)&&(i<N))
      {
        if(GMatrix[i][y]==symbol)
        {
          if(ccount==0)
            BegPoint.ToSetAll(i,y,1);
          ccount++;
          if(ccount==win_number)
          {
             EndPoint.ToSetAll(i,y,1);
             return symbol;
          }
        }
        else
        {
          ccount=0;
          BegPoint.ToSetAll(-1,-1,0);
        }
      }
   }
   ccount=0;
   for(int j=y-win_number;j<y+win_number;j++)
   {
     if((j>=0)&&(j<N))
     {
        if(GMatrix[x][j]==symbol)
        {
           if(ccount==0)
            BegPoint.ToSetAll(x,j,2);
```

```
        ccount++;
        if(ccount==win_number)
        {
          EndPoint.ToSetAll(x,j,2);
          return symbol;
        }
      }
      else
      {
        ccount=0;
        BegPoint.ToSetAll(-1,-1,0);
      }
    }
}

int i,j,r;
i=x-win_number;
j=y-win_number;
r=y+win_number;
TCell BegDiag1;
BegDiag1.ToSetAll(-1,-1,0);
int k=0;
int control=2*win_number;
while(k<control)
{
  if((i+k>=0)&&(i+k<N)&&(j+k>=0)&&(j+k<N))
  {
     if(GMatrix[i+k][j+k]==symbol)
     {
          if(dlcount==0)
            BegDiag1.ToSetAll(i+k,j+k,3);
          dlcount++;
          if(dlcount==win_number)
          {
            BegPoint=BegDiag1;
            EndPoint.ToSetAll(i+k,j+k,3);
            return symbol;
          }
     }
     else
     {
        dlcount=0;
        BegDiag1.ToSetAll(-1,-1,0);
     }
  }
  if((i+k>=0)&&(i+k<N)&&(r-k>=0)&&(r-k<N))
  {
     if(GMatrix[i+k][r-k]==symbol)
     {
        if(drcount==0)
          BegPoint.ToSetAll(i+k,r-k,4);
        drcount++;
        if(drcount==win_number)
        {
```

```
                EndPoint.ToSetAll(i+k,r-k,4);
                return symbol;
              }
            }
            else
            {
              drcount=0;
              BegPoint.ToSetAll(-1,-1,0);
            }
          }
          k++;
        }
        return -1;

    }

};
#endif
```

```
//---------------------------Unit1.cpp-----------------------------------------------

#include <vcl.h>
#include <fstream.h>
#include "iostream.h"
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TUserForm *UserForm;
typedef TCell (*TPtrFuncFromDLL)(int,int**,int,int,int,int,int);  // создание своего типа
HINSTANCE  PtrToDLL;
TPtrFuncFromDLL AIFunction;//переменная создаданного типа, загружаемый ИИ
const int CellSize=50;
const int FSize=500;
TPicture *PZero,*PCross;
TRect Source,Dest;
TGameMatrix MatrixBoard;

//---------------------------------------------------------------------------
__fastcall TUserForm::TUserForm(TComponent* Owner)
        : TForm(Owner)
{
  IsComputer=false;
}
//---------------------------------------------------------------------------
void TUserForm::ToNormLabelLocation(void)
{
  SocerBoard->Left=ImField->Left+(ImField->Width - SocerBoard->Width)/2;
}
//---------------------------------------------------------------------------
void TUserForm::NetGame(void)
{

  if(MtgSea->Checked==true)
  {
    ImField->Picture->LoadFromFile("Темы игры/Океан4.bmp");
    PZero->LoadFromFile("Темы игры/Морской еж1.bmp");
    PZero->Graphic->Transparent=true;
    PCross->LoadFromFile("Темы игры/морская звезда1.bmp");
    PCross->Graphic->Transparent=true;
    SocerBoard->Font->Color=clWhite;
  }
  else if(MtgFootball->Checked==true)
  {
    ImField->Picture->LoadFromFile("Темы игры/газон1.bmp");
    PZero->LoadFromFile("Темы игры/Мяч.bmp");
```

```
    PZero->Graphic->Transparent=true;
    PCross->LoadFromFile("Темы игры/Биты2.bmp");
    PCross->Graphic->Transparent=true;
    SocerBoard->Font->Color=clYellow;

  }
  ImField->Canvas->Pen->Color=clWhite;
  ImField->Canvas->Pen->Width=2;
  for(int i=0;i<=FSize;i+=CellSize)
  {
    ImField->Canvas->MoveTo(i,0);
    ImField->Canvas->LineTo(i,FSize);
  }
  for(int i=0;i<=FSize;i+=CellSize)
  {
    ImField->Canvas->MoveTo(0,i);
    ImField->Canvas->LineTo(FSize,i);
  }

}
//---------------------------------------------------------------------------
void TUserForm::ToPaint(void)
{

  int n=MatrixBoard.N;
  for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
    {
      if(MatrixBoard.GetElement(i,j)==MatrixBoard.GetZero())
        ImField->Canvas->Draw(i*CellSize,j*CellSize,PZero->Graphic);
      else if (MatrixBoard.GetElement(i,j)==MatrixBoard.GetCross())
        ImField->Canvas->Draw(i*CellSize,j*CellSize,PCross->Graphic);

    }
    DrawLine();
}
//---------------------------------------------------------------------------
void __fastcall TUserForm::MExitClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------

void __fastcall TUserForm::FormCreate(TObject *Sender)
{
  DoubleBuffered=true;
  Left=(Screen->Width-Width)/2;// центральное положение формы
  Top=(Screen->Height-Height)/2;
  ClientHeight=FSize+60;
  ClientWidth=FSize+60;
  ImField->Height=FSize;
  ImField->Width=FSize;
  ImField->Top=30;
  ImField->Left=30;
```

```
  PZero=new TPicture();
  PCross=new TPicture();
  IBoder->Canvas->Brush->Color=clBlack;
  IBoder->Canvas->FloodFill(0,0,clWhite,fsSurface);

  NetGame();
}
//---------------------------------------------------------------------------

void __fastcall TUserForm::MAboutProgramClick(TObject *Sender)
{
  FAboutProg->Show();
  UserForm->Enabled=false;
}
//---------------------------------------------------------------------------


void __fastcall TUserForm::ImFieldMouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{

  int MX,MY;
  MX=X/CellSize;
  MY=Y/CellSize;
  if(MatrixBoard.PossibleMotion(MX,MY))
  {
   if((Shift.Contains(ssLeft)==true)&&(Shift.Contains(ssRight)==false))
   {
     if(!MatrixBoard.IsNextCross())
     {
       if(!IsComputer)
       {
        MatrixBoard.SetElement(MX,MY,MatrixBoard.GetZero());
        ImField->Canvas->Draw(MX*CellSize,MY*CellSize,PZero->Graphic);
        SocerBoard->Caption="Ход: крестика";
        ToNormLabelLocation();
       }
       else goto m;
     }

     else
     {
     m: MatrixBoard.SetElement(MX,MY,MatrixBoard.GetCross());
       ImField->Canvas->Draw(MX*CellSize,MY*CellSize,PCross->Graphic);
       SocerBoard->Caption="Ход: нолика";
       ToNormLabelLocation();
       IFSomeoneWin(MX,MY);

       if((IsComputer)&&(ImField->Enabled==true))
       {
        TCell
Choise=AIFunction(MatrixBoard.GetZero(),MatrixBoard.GMatrix,MatrixBoard.N,MatrixBoard.
win_number,MatrixBoard.GetCross(),MatrixBoard.GetZero(),MatrixBoard.count_of_free);
         MX=Choise.x;
```

```
            MY=Choise.y;
            MatrixBoard.SetElement(MX,MY,MatrixBoard.GetZero());
            ImField->Canvas->Draw(MX*CellSize,MY*CellSize,PZero->Graphic);
            SocerBoard->Caption="Ход: крестика";
            ToNormLabelLocation();
          }
        }
        IFSomeoneWin(MX,MY);
    }
  }
}
//---------------------------------------------------------------------------
void TUserForm::DrawLine(void)
{
      if(MatrixBoard.BegPoint.value==0)
        return;
      ImField->Canvas->Pen->Color=clBlue;
      ImField->Canvas->Pen->Width=3;
      TCell MBeg,MEnd;
      MBeg.ToSetAll(MatrixBoard.BegPoint.x*CellSize,MatrixBoard.BegPoint.y*CellSize,0);
      MEnd.ToSetAll(MatrixBoard.EndPoint.x*CellSize,MatrixBoard.EndPoint.y*CellSize,0);
      int direct=MatrixBoard.BegPoint.value;
      switch(direct)
      {
       case 1:
       {
         MBeg.y=MBeg.y+CellSize/2;
         MEnd.x=MEnd.x+CellSize;
         MEnd.y=MEnd.y+CellSize/2;
         break;
       }
       case 2:
       {
         MBeg.x=MBeg.x+CellSize/2;
         MEnd.x= MEnd.x+CellSize/2;
         MEnd.y=MEnd.y+CellSize;
         break;
       }
       case 3:
       {
         MEnd.x= MEnd.x+CellSize;
         MEnd.y=MEnd.y+CellSize;
         break;
       }
       case 4:
       {
         MBeg.y=MBeg.y+CellSize;
         MEnd.x=MEnd.x+CellSize;


         break;
       }
      }
      ImField->Canvas->MoveTo(MBeg.x,MBeg.y);
```

```
      ImField->Canvas->LineTo(MEnd.x,MEnd.y);

}
//---------------------------------------------------------------------------
void TUserForm::IFSomeoneWin(int MX,int MY)
{

if(MatrixBoard.GetElement(MX,MY)==MatrixBoard.IsWinner(MX,MY,MatrixBoard.GetZero(
))\

||MatrixBoard.GetElement(MX,MY)==MatrixBoard.IsWinner(MX,MY,MatrixBoard.GetCross()
)\
    &&(ImField->Enabled==true))
   {
      AnsiString Con="Выиграли";
      if(MatrixBoard.GetElement(MX,MY)==MatrixBoard.GetCross())
         Con+=" крестики";
      else Con+=" нолики";
      Application->MessageBoxA(Con.c_str(),"Win",MB_OK);
      ImField->Enabled=false;
      Con+=". Game over.";
      SocerBoard->Caption=Con;
      ToNormLabelLocation();
      DrawLine();
   }
   else if(MatrixBoard.IsFull())
     Application->MessageBoxA("Мир"," Ничья ",MB_OK);
}

//---------------------------------------------------------------------------
void __fastcall TUserForm::FormDestroy(TObject *Sender)
{
  PZero->Free();
  PCross->Free();
}
//---------------------------------------------------------------------------


void __fastcall TUserForm::MAboutRulesClick(TObject *Sender)
{
    FormHelp->Show();
    UserForm->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TUserForm::MtgSeaClick(TObject *Sender)
{

   MtgSea->Checked=true;
   NetGame();
   ToPaint();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TUserForm::MtgFootballClick(TObject *Sender)
{
  MtgFootball->Checked=true;
  NetGame();
  ToPaint();
}
//---------------------------------------------------------------------------

void __fastcall TUserForm::NGUserClick(TObject *Sender)
{
   NGUser->Checked=true;
   MatrixBoard.Reset();
   ImField->Enabled=true;
   if(PtrToDLL!=NULL)
     FreeLibrary(PtrToDLL);
   NetGame();
   ToPaint();
   SocerBoard->Caption="Ход: крестика";
   ToNormLabelLocation();
   IsComputer=false;

}
//---------------------------------------------------------------------------

void __fastcall TUserForm::NGKompClick(TObject *Sender)
{

  NGKomp->Checked=true;
  MatrixBoard.Reset();
  ImField->Enabled=true;
  if(PtrToDLL==NULL)
  {  AnsiString WayToDLL=GetCurrentDir()+"\\DLL\\MyDLL.dll";
     PtrToDLL=LoadLibrary(WayToDLL.c_str()); //загружаем DLL
     if(PtrToDLL==NULL)
       Application->MessageBoxA("Невозможно загрузить
DLL","Ошибка",MB_OK+MB_ICONERROR);
     AIFunction =(TPtrFuncFromDLL)GetProcAddress(PtrToDLL,"_AnalyseMove");
     if(AIFunction==NULL)
      Application->MessageBoxA("Функция не
существует","Ошибка",MB_OK+MB_ICONERROR);
  }

  NetGame();
  ToPaint();
  SocerBoard->Caption="Ход: крестика";
  ToNormLabelLocation();
  IsComputer=true;
}
//---------------------------------------------------------------------------

void __fastcall TUserForm::MSaveClick(TObject *Sender)
{
 if(MatrixBoard.BegPoint.value!=0)
 {
```

```
    Application->MessageBoxA("Игра окончена","Нет смысла
сохранять",MB_OK+MB_ICONWARNING);
    return;
  }
  SaveDialog1->InitialDir=GetCurrentDir()+"\\Сохраненные игры";
  SaveDialog1->Title="Сохранение сеанса игры";
  SaveDialog1->DefaultExt="*.crz";
  AnsiString MyFileName="Игра 0";
  SaveDialog1->FileName=MyFileName;

  if(SaveDialog1->Execute())
  {
    MyFileName=SaveDialog1->FileName;
    ofstream SWrite (MyFileName.c_str());
    SWrite<<MatrixBoard.IsNextCross()<<endl;
    for(int i=0;i<MatrixBoard.N;i++)
    {
      for(int j=0;j<MatrixBoard.N;j++)
        SWrite<<MatrixBoard.GetElement(i,j)<<" ";   // транспонируема
      SWrite<<endl;
    }
    SWrite.close();

  }


}
//---------------------------------------------------------------------------

void __fastcall TUserForm::MLoadClick(TObject *Sender)
{
   OpenDialog1->InitialDir=GetCurrentDir()+"\\Сохраненные игры";
   OpenDialog1->Title="Загрузка сеанса игры";
   OpenDialog1->DefaultExt="*.crz";
   AnsiString MyFileName="Игра 0";
   OpenDialog1->FileName=MyFileName;
   if(OpenDialog1->Execute())
   {
     MyFileName=OpenDialog1->FileName;
     ifstream FRead(MyFileName.c_str());

     bool pr;
     MatrixBoard.Reset();
     int z=1;
     int i=0;
     char str[255];
     AnsiString Stroka,Dop;
     while(FRead.getline(str,255)!=0)
     {
       Stroka=str;
       Stroka=Stroka.Trim();
       if(z==1)
       {
         pr=StrToInt(Stroka);
```

```
        MatrixBoard.SetIsNextCross(pr);
        if(pr==true)
           SocerBoard->Caption="Ход: крестика";
        else SocerBoard->Caption="Ход: нолика";

       }
       else
       {
        int j=0,vh;
        Stroka=Stroka+" ";
        vh=Stroka.Pos(" ");
        while(vh!=0)
        {
         Dop=Stroka.SubString(1,vh-1);
         int el=Dop.ToInt();
         MatrixBoard.SetElement(j,i,el);
         Stroka.Delete(1,vh);
         vh=Stroka.Pos(" ");
         j++;
        }
        i++;

       }
       z++;
      }
      FRead.close();
   }
   ImField->Enabled=true;
   NetGame();
   ToPaint();
   ToNormLabelLocation();



}
//---------------------------------------------------------------------------
```

```
//---------------------------------MyDLL.cpp-----------------------------------------

#include <vcl.h>
#include <windows.h>
#include <math.h>
#pragma hdrstop
//---------------------------------------------------------------------------
//-----------------------Искусственный ителлект----------------------------
//---------------------------------------------------------------------------

#pragma argsused
int win_number,cross,zero,count_of_free;
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fwdreason, LPVOID lpvReserved)
{
  return 1;
}
//---------------------------------------------------------------------------
class TCell
{
public:
   int x,y,value;
   TCell()
   {
      randomize();
      x=random(10);
      y=random(10);
      value=0;
   }
   void ToSetAll(int i,int j,int val)
   {
     x=i;
     y=j;
     value=val;
   }
};
//---------------------------------------------------------------------------
int GetEstimate (int x,int y,int symbol,int**GMatrix,int N)
{

      int k[8];
      for(int i=0;i<8;i++)
         k[i]=0;
      for(int i=x-1;i>=x-win_number;i--)
      {
         if(i>=0&&i<N)
         {
           if(GMatrix[i][y]==symbol)
             k[0]++;
           else break;
         }
         else break;
      }
      for(int i=x+1;i<x+win_number;i++)
```

```
{
  if(i>=0&&i<N)
  {
    if(GMatrix[i][y]==symbol)
      k[1]++;
    else break;
  }
  else break;
}
for(int i=y-1;i>=y-win_number;i--)
{
  if(i>=0&&i<N)
  {
    if(GMatrix[x][i]==symbol)
      k[2]++;
    else break;
  }
  else break;
}
for(int i=y+1;i<y+win_number;i++)
{
  if(i>=0&&i<N)
  {
    if(GMatrix[x][i]==symbol)
      k[3]++;
    else break;
  }
  else break;
}
int i=x;
int j=y;
int l=1;
while(l<win_number)          //left up
{
 if((i-l>=0)&&(i-l<N)&&(j-l>=0)&&(j-l<N))
 {
   if(GMatrix[i-l][j-l]==symbol)
     k[4]++;
   else break;
 }
 else break;

 l++;
}
l=1;
i=x;
j=y;
while(l<win_number)      // right down
{
 if((i+l>=0)&&(i+l<N)&&(j+l>=0)&&(j+l<N))
 {
   if(GMatrix[i+l][j+l]==symbol)
     k[5]++;
   else break;
```

```
        }
        else break;
        l++;

      }
      l=1;
      i=x;
      j=y;
      while(l<win_number)       // right up
      {
       if((i-l>=0)&&(i-l<N)&&(j+l>=0)&&(j+l<N))
       {
         if(GMatrix[i-l][j+l]==symbol)
           k[6]++;
         else break;
       }
       else break;
       l++;
      }
      l=1;
      i=x;
      j=y;
      while(l<win_number)       // left down
      {
       if((i+l>=0)&&(i+l<N)&&(j-l>=0)&&(j-l<N))
       {
         if(GMatrix[i+l][j-l]==symbol)
           k[7]++;
         else break;
       }
       else break;
       l++;
      }

      int M=0;
      for(i=0;i<8;i++)
        M=M+pow(15,k[i]);

      return M;
}
//---------------------------------------------------------------------------

double ComplitWeight(int x, int y,int symbol,int **GMatrix,int n)
{
       int M=GetEstimate(x,y,symbol,GMatrix,n);
       int other_symbol=cross;
       if(other_symbol==symbol)
         other_symbol=zero;
       int N=GetEstimate(x,y,other_symbol,GMatrix,n);
       double Q=0.1;
       double F=M+Q*N;
       return F;
}
//----------------------Основная функция-------------------------------------
```

```
extern "C" TCell __export  AnalyseMove(int symbol,int**GMatrix,int N,int uwin_number,int
ucross, int uzero, int ucount_of_free)
{
      win_number=uwin_number;
      cross=ucross;
      zero=uzero;
      count_of_free=ucount_of_free;
      if(count_of_free==0)
      {
       TCell Step;
       Step.ToSetAll(-1,-1,-4);
       return Step;

      }
      TCell *ArrayPossibles=new TCell[count_of_free];
      int k=0;
      for(int i=0;i<N;i++)
       for(int j=0;j<N;j++)
       {
          if(GMatrix[i][j]==0)
          {
            ArrayPossibles[k].ToSetAll(i,j,ComplitWeight(i,j,symbol,GMatrix,N));
            k++;
            if(k==count_of_free) break;
          }
       }
       int r=random(count_of_free);
       TCell Max=ArrayPossibles[r];
       for(int i=0;i<count_of_free;i++)
          if(Max.value<ArrayPossibles[i].value)
            Max=ArrayPossibles[i];
       delete[]ArrayPossibles;
      // SetElement(Max.x,Max.y,symbol);   нужно записать из вне
       return Max;


}



//---------------------------------------------------------------------------
```