# Программа "Крестики-нолики 5 в ряд на неограниченном игровом поле"

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Курсовая работа

по дисциплине: программирование

«Крестики-нолики 5 в ряд на неограниченном игровом поле»

Выполнил студент А.С.Лебедев

Руководитель М.Н.Суслов

Санкт-Петербург 2010

СОДЕРЖАНИЕ

1. Цель работы

2. Описание игры

3. Описание входных и выходных данных

4. Описание переменных и функций программы

5. Алгоритм работы программы

6. Текст программы

7. Примеры выполнения программы

Выводы

Список литературы

программа игра данные алгоритм

1. Цель работы

Разработать программу игры в крестики-нолики пять в ряд на неограниченном поле. Программа должна быть написана на языке С++ в среде Visual Studio 2008 или Visual Studio 2010.

2. Описание игры

Игра ведется на бесконечном поле, разлинованном в клетку. Перед игрой противники решают, кто будет играть крестиками, а кто ноликами. В ходе игры противники ставят по очереди крестик или нолик (в зависимости от договоренности перед началом игры) в свободную клетку на поле.

Цель игры – построить линию из 5 стоящих рядом по вертикали, горизонтали или диагонали крестиков или ноликов. Первый игрок, построивший такую комбинацию из знаков своего типа (крестиков или ноликов) выигрывает.

Несмотря на то, что по определению игровое поле по определению бесконечно, обычно пользуются ограниченными размерами поля. Например, в игре гомоку, поле имеет размер 15x15 (ранее имело 19x19).

Пример игровой ситуации на игровом поле (показана только часть поля):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 0 |  |  | X |
| X | 0 |  |  |  |
| 0 | X | 0 |  |  |
| 0 | X |  | 0 |  |
| 0 | X | X | X |  |

Пример выигрыша крестиков:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  | 0 |  |  |  |
|  | X | 0 | X | 0 |  |  |  |
|  |  | X |  | 0 |  |  |  |
|  |  | 0 | X | X | 0 |  |  |
|  |  | 0 | X | X |  | 0 |  |
|  |  | 0 | 0 | X | X | X |  |

Пример выигрыша ноликов:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 0 |  |  | 0 | X | 0 | X | X |
| X | X |  | 0 | 0 | 0 | 0 | 0 | X |
| X | X | X | 0 |  | X |  |  |  |
| 0 | X | 0 | 0 |  | X |  |  |  |
| 0 | 0 |  | X | X | X |  |  |  |
| X | 0 | 0 | X |  | 0 | 0 |  |  |
| 0 | X |  | X |  | 0 | 0 |  |  |
| 0 | X |  | 0 | X | 0 |  | 0 |  |
| 0 | X |  |  | 0 | X | X | X | X |

3. Описание входных и выходных данных

Входные данные:

1)      Размер игрового поля (10x10, 19x19, 30x30, 50x50 или 100х100) – задается из главного меню программы.

2)      Уровень игры компьютера (новичок, любитель или профессионал) – задается из главного меню программы.

3)      Очередность первого хода (человек, компьютер).

4)      Координаты щелчка левой кнопкой мыши по игровому полю (x,y) – координаты очередного хода.

Выходные данные:

1)         Игровое поле, заполненное крестиками и ноликами, отображаемое на экран.

2)         В случае выигрыша одного из игроков, отображение выигрышной последовательности и соответствующего сообщения.

4. Описание переменных и функций программы

Функции расчета оценочной функции, состояния игрового поля, обработки нажатия клавиши мыши, вывода игрового поля на экран объявлены в классе представления клиентской области главного окна CChildView.

Объявление класса находится в модуле СhieldView.h. Определения функций и объявления глобальных переменных программы, отвечающих за алгоритм расчета расположены в модуле СhieldView.cpp.

Описание переменных:

unsigned char\*\* fields – Динамический двумерный массив, представляющий игровое поле. Каждый элемент массива представляет клетку на поле. Индексы массива соответствуют положению клеток на поле.

Элемент массива может принимать следующие значения:

0 – клетка пуста;

1 – в клетке нолик;

2 – в клетке крестик;

3 – в клетке нолик, входящий в выигрышный ряд по окончанию игры, служит для выделения другим цветом на поле выигрышного ряда;

4 – в клетке крестик, входящий в выигрышный ряд по окончанию игры, служит для выделения другим цветом на поле выигрышного ряда;

5 – в клетке нолик, поставленный последним ходом, служит для выделения другим цветом на поле последнего хода;

6 – в клетке крестик, поставленный последним ходом, служит для выделения другим цветом на поле последнего хода;

float\*\* calc\_fields – Динамический двумерный массив, представляющий значение оценочной функции для каждой клетки игрового поля.

int size\_x – Размер игрового поля по х, по умолчанию равен 19.

int size\_y – Размер игрового поля по y, по умолчанию равен 19.

int old\_size\_x – Предыдущий размер игрового поля по x, используется при изменении размеров игрового поля.

int old\_size\_y – Предыдущий размер игрового поля по y, используется при изменении размеров игрового поля.

int attack\_factor – Коэффициент агрессивности игры компьютера, используется при расчете оценочной функции. При меньшем значении данного параметра компьютер играет более атакующе. По умолчанию равен 1 для игрока уровня эксперт.

int valuation\_factor – Коэффициент, используемый при расчете оценочной функции. По умолчанию равен 3.

bool end\_game – Установка данной переменной в значение true обозначает конец игры. Дальнейшие щелчки мышью по игровому полю не воспринимаются до старта новой игры. Начальное значение – false.

int last\_x – Координата x последнего хода.

int last\_y – Координата y последнего хода.

bool player\_first\_step – Определяет приоритетность хода при старте новой игры. При значении true первым ходит человек, при false – компьютер. Значение по умолчанию – true.

int comp\_level – Уровень игры компьютера. Возможные значения:

0 – профессионал, сильный уровень, играет агрессивно;

1 – любитель, придерживается защитной стратегии;

2 – новичок, играет слабо, но достаточно агрессивно.

Описание функций программы:

void CChildView::OnPaint() – выполняет перерисовку клиентской области окна.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Производит перерисовку клеток игрового поля. В зависимости от значений массива fields выводит в клетку:

0 – ничего не выводит;

1 – нолик синим цветом;

2 – крестик зеленым цветом;

3 – нолик красным цветом (входит в выигрышный ряд);

4 – крестик красным цветом (входит в выигрышный ряд);

5 – нолик желтым цветом (последний сделанный ход);

6 – крестик желтым цветом (последний сделанный ход);

void CChildView::OnLButtonDown(UINT, CPoint xy) – Обработка нажатия левой кнопки мыши на клиентской области окна.

Входные параметры:

UINT – флаги, не используется;

CPoint xy – координаты точки нажатия.

Возвращаемое значение:

Нет.

Алгоритм работы:

По нажатию левой кнопки мыши, если игра не закончена выполняются следующие действия:

1)      Обновляется массив fields c учетом последнего поставленного нолика.

2)      Осуществляется проверка, не закончена ли игра с помощью функции end\_analyze.

3)      Вычисляется ход компьютера с помощью функции ii.

4)      Осуществляется проверка, не закончена ли игра с помощью функции end\_analyze.

5)      Производится перерисовка окна.

Алгоритм работы функции приведен на рисунке 1 в разделе 5.

int CChildView::end\_analyze() – Функция определяет не закончена ли игра, т. е. не составлен ли выигрышный ряд на поле одним из игроков, на основании значений элементов массива fields.

Входные параметры:

Нет.

Возвращаемое значение:

int – реузультат работы, = 1 – игра окончена, = 0 – игра не окончена.

Алгоритм работы:

Для каждой клетки на игровом поле просматриваются соседние клетки по горизонтали, вертикали, вниз и вправо по диагонали. Если в одном из направлений символы во всех клетках на расстоянии до 4 совпадают с символом в текущей клетке, то игра считается выигранной. К значениям найденных клеток в массиве fields прибавляется 2 для отображения выигрышной ситуации в окно.

void CChildView::ii() – Функция расчета очередного хода компьютера.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Функция рассчитывает оценочную функцию для каждой клетки игрового поля, с помощью вызова функции calculate, и заносит рассчитанные значения в массив calc\_field. Исходя из значений массива calc\_field выбирает очередной ход компьютера.

Алгоритм работы функции приведен в разделе 6.

unsigned long CChildView::calculate(int id,int x,int y) – Расчет оценочной функции для клетки игрового поля, с учетом установки в нее крестика или нолика.

Входные параметры:

int id – определяет какой символ ставится в клетку (= 1 – нолик, = 2 – крестик);

int x – координата x клетки.

int y – координата y клетки.

Возвращаемое значение:

unsigned long – значение оценочной функции.

Алгоритм работы:

Алгоритм работы функции приведен в разделе 6.

void CChildView::OnNewGame() – В главном меню нажата кнопка «Новая игра».

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Вызывается функция new\_game для начала новой игры. Перерисовывается игровое поле.

void CChildView::OnX1010() – В главном меню выбран размер поля 10x10.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяются значения size\_x, size\_y. Вызывается функция new\_game для начала новой игры. С помощью функции resize\_window устанавливаются новые размеры окна.

void CChildView::OnX1919() – В главном меню выбран размер поля 19x19.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяются значения size\_x, size\_y. Вызывается функция new\_game для начала новой игры. С помощью функции resize\_window устанавливаются новые размеры окна.

void CChildView::OnX3030() – В главном меню выбран размер поля 30x30.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяются значения size\_x, size\_y. Вызывается функция new\_game для начала новой игры. С помощью функции resize\_window устанавливаются новые размеры окна.

void CChildView::OnX5050() – В главном меню выбран размер поля 50x50.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяются значения size\_x, size\_y. Вызывается функция new\_game для начала новой игры. С помощью функции resize\_window устанавливаются новые размеры окна.

void CChildView::OnX100100() – В главном меню выбран размер поля 100x100.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяются значения size\_x, size\_y. Вызывается функция new\_game для начала новой игры. С помощью функции resize\_window устанавливаются новые размеры окна.

void CChildView::new\_game() – Функция начала новой игры.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Функция начинает новую игру, при этом:

1)      Перевыделяется память для динамических массивов fields и calc\_fields в зависимости от значений old\_size\_x, old\_size\_y и size\_x, size\_y.

2)      Сбрасывается в false флаг end\_game.

3)      Если переменная player\_first\_step равна false, то рассчитывается первый ход компьютера с помощью вызова функции ii.

void CChildView::resize\_window() – Функция установки размеров окна.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Устанавливает новые размеры окна, в зависимости от переменных size\_x, size\_y.

void CChildView::set\_chеcked\_menu(unsigned int old\_id,unsigned int new\_id) – Служит для снятия галочки и установки новой в главном меню при выборе размеров поля, уровня игры компьютера и очереднсоти хода.

Входные параметры:

unsigned int old\_id – id элемента меню, с которого необходимо снять галочку;

unsigned int new\_id – id элемента меню, на который необходимо поставить галочку;

Возвращаемое значение:

Нет.

Алгоритм работы:

Снимает галочку в главном меню с элемента, определяемого переменной old\_id и ставит галочку на элемент, определяемый переменной new\_id.

void CChildView::OnStepH() – В главном меню выбрана очередность хода – человек.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяется значение переменной player\_first\_step. Вызывается функция new\_game для начала новой игры. Выполняется перерисовка окна.

void CChildView::OnStepС() – В главном меню выбрана очередность хода – компьютер.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяется значение переменной player\_first\_step. Вызывается функция new\_game для начала новой игры. Выполняется перерисовка окна.

void CChildView::OnLevelBeg() – В главном меню выбран уровень сложности начинающий.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяется значение переменной comp\_level. Значение коэффициента агрессивности игры компьютера attack\_factor устанавливается в 1. Вызывается функция new\_game для начала новой игры. Выполняется перерисовка окна.

void CChildView::OnLevelAmat() – В главном меню выбран уровень сложности любитель.

Входные параметры:

Нет.

Возвращаемое значение:

Нет.

Алгоритм работы:

Изменяется значение переменной comp\_level. Значение коэффициента агрессивности игры компьютера attack\_factor устанавливается в 1. Вызывается функция new\_game для начала новой игры. Выполняется перерисовка окна.

5. Алгоритм работы программы

Алгоритм выполнения очередного хода:

Игрок выполняет очередной ход при нажатии левой кнопки мыши на игровом поле. При этом вызывается функция обработчика OnLButtonDown, которая содержит основной алгоритм выполнения очередного хода игроком и компьютером. Алгоритм работы функции приведен на рисунке 1.

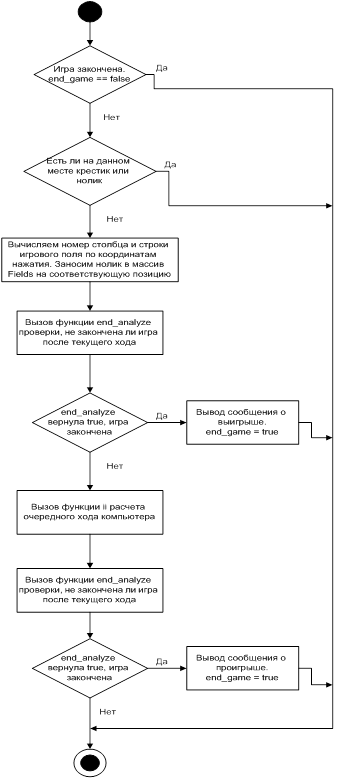


Рисунок 1 – Блок-схема работы функции OnLButtonDown.

Алгоритм расчета очередного хода компьютерного соперника:

Расчет очередного хода компьютерного соперника выполняется при помощи вызова функции ii. Расчет производится при старте игры, если приоритет хода принадлежит компьютеру или после хода игрока вызовом из функции OnLButtonDown.

Расчет производится по следующему алгоритму:

1)         Рассчитывается суммарная оценочная функция  для всех непустых клеток игрового поля. Под оценочной функцией понимается некое значение, присваиваемое клетке и обозначающее выгодность хода в данную клетку.



Расчет производится при помощи функции calculate, которая позволяет рассчитать оценочную функцию для отдельной клетки в случае хода в эту клетку игроком или компьютером.

Суммарная оценочная функция вычисляется по формуле:

, где



 – значение суммарной оценочной функции;



– значение оценочной функции при постановке в клетку крестика, рассчитанное с помощью вызова функции calculate;



 – значение оценочной функции при постановке в клетку нолика, рассчитанное с помощью вызова функции calculate;



 – коэффициент агрессивности ИИ, задается переменной attack\_factor.



Как видно из формулы, значение суммарной оценочной функции учитывает выгоду как своего хода в данную клетку, так и выгоду, которую получил бы соперник от хода в данную клетку. Причем чем больше коэффициент агрессивности, тем больше учитывается выгода соперника и соответственно компьютерный игрок следует защитной стратегии.

Значение коэффициента агрессивности на уровнях новичок и профессионал равно 1, компьютер ведет себя агрессивно. На уровне любитель значение равно 10, алгоритм находится в глубокой обороне.

Для уровней новичок и любитель также дополнительно вносится случайность значения суммарной оценочной функции: на уровне новичок значение суммарной оценочной функции умножается на случайное число, принимающее значение [0,1], на уровне любитель умножается на случайное число, принимающее значение [0.5,1].

2)         Находится клетка с максимальным значением суммарной оценочной функции. Если таких клеток несколько, то из них выбирается случайная. Ход делается в найденную клетку, массив fields обновляется путем установки соответствующего элемента в значение 2.

Алгоритм расчета оценочной функции:

Расчет оценочной функции для клетки игрвого поля производится вызовом функции calculate. Входными параметрами являются индексы поля в массиве fields (текущая клетка) и идентификатор, какой символ(крестик или нолик) будет поставлен в текущую клетку. Этот символ временно ставится в массив fields до окончания работы функции.

Расчет производится по следующей схеме. Просматриваются все ряды, продолжением которых может являться текущая клетка. Под рядом подразумевается последовательность из 5 клеток, каждая из которых может быть пустой или содержащей такой же символ, как и в текущей. Для этого проходятся все клетки на расстоянии не более 4 от данной, сначала по вертикали, затем по горизонтали, затем по 2 диагоналям. Для каждой проходимой клетки рассчитывается длина ряда, в которую она входит вместе с текущей. Под длиной ряда понимается количество одинаковых символов(крестиков или ноликов) в последовательности из 5 клеток. Если ряд прерывается противоположным символом, то длина ряда принимается равной нулю.

Значение оценочной функции рассчитывается по формуле:

, где  – общее количество ненулевых длин рядов;



– коэффициент оценочной функции;



 – длина -го ряда.



Степенная функция выбрана потому, что увеличение длины ряда даже на 1 существенно повышает его важность и не может быть выражена линейной функцией.

В случае нахождения ряда длиной 5, т. е. выигрышной ситуации, значение длины приравнивается к 10000, если расчет ведется при постановке крестика в текущую клетку и 1000 при постановке нолика в текущую клетку. Значение при постановке крестика выше, т. к. при нахождении такой ситуации компьютерный игрок должен в первую очередь стремиться к своему выигрышу, а не к предотвращению выигрыша соперника.

6. Текст программы

Файл ChildView.cpp:

//Модуль, содержащий основной алгоритм работы программы

#include "stdafx.h"

#include "XO.h"

#include "ChildView.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CChildView

CChildView::CChildView()

{

}

CChildView::~CChildView()

{

}

BEGIN\_MESSAGE\_MAP(CChildView, CWnd)

ON\_WM\_PAINT()

ON\_WM\_LBUTTONDOWN()

ON\_COMMAND(ID\_NEW\_GAME, &CChildView::OnNewGame)

ON\_COMMAND(ID\_X1010, &CChildView::OnX1010)

ON\_COMMAND(ID\_X100100, &CChildView::OnX100100)

ON\_COMMAND(ID\_X1919, &CChildView::OnX1919)

ON\_COMMAND(ID\_X3030, &CChildView::OnX3030)

ON\_COMMAND(ID\_X5050, &CChildView::OnX5050)

ON\_COMMAND(ID\_STEP\_H, &CChildView::OnStepH)

ON\_COMMAND(ID\_STEP\_C, &CChildView::OnStepC)

ON\_COMMAND(ID\_LEVEL\_BEG, &CChildView::OnLevelBeg)

ON\_COMMAND(ID\_LEVEL\_AMAT, &CChildView::OnLevelAmat)

ON\_COMMAND(ID\_LEVEL\_PROF, &CChildView::OnLevelProf)

END\_MESSAGE\_MAP()

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)

{

if (!CWnd::PreCreateWindow(cs))

return FALSE;

cs.dwExStyle |= WS\_EX\_CLIENTEDGE;

cs.style &= ~WS\_BORDER;

cs.lpszClass = AfxRegisterWndClass(CS\_HREDRAW|CS\_VREDRAW|CS\_DBLCLKS,

::LoadCursor(NULL, IDC\_ARROW), reinterpret\_cast<HBRUSH>(COLOR\_WINDOW+1), NULL);

srand((unsigned)time( NULL ));

//Начало новой игры

new\_game();

return TRUE;

}

//Глобальные переменные

unsigned char\*\* fields;//Игровое поле ( = 0 - ничего нет, = 1 - нолик,

//= 2 - крестик, = 3 - выигравший нолик, = 4 - выигравший крестик

//= 5 - последний поставленный нолик, = 6 - последний поставленный крестик)

float\*\* calc\_fields;//Рассчитанное значение оценочной функции

int size\_x = 19;//Размер поля по x (19 - по умолчанию)

int size\_y = 19;//Размер поля по y (19 - по умолчанию)

int old\_size\_x = 0;//Старый размер поля по x

int old\_size\_y = 0;//Старый размер поля по y

int attack\_factor = 1;   //Коэффициент агрессивности ИИ (1 - по умолчанию)

int valuation\_factor = 3;//Оценочный коэффициент (4 - по умолчанию)

bool end\_game = false;//Наступил конец игры?

int last\_x = 0;//Координата x последнего хода

int last\_y = 0;//Координата y последнего хода

bool player\_first\_step = true;//Приоритетность хода (true - человек)

int comp\_level = 0;//Уровень игры компьютера (0 - эксперт)

//Перерисовка окна

void CChildView::OnPaint()

{

CPaintDC dc(this);

//Выведем поле

CBrush brushBgnd(RGB(0xF4, 0xA4, 0x60));//Кисть для заднего фона

CPen penO(PS\_SOLID,2,RGB(0x00, 0x00, 0xFF));//Перо для нолика

CPen penX(PS\_SOLID,2,RGB(0x00, 0x80, 0x00));//Перо для крестика

CPen penWin(PS\_SOLID,2,RGB(0xFF, 0x00, 0x00));//Перо для выигрышных крестика или нолика

CPen penLast(PS\_SOLID,2,RGB(0xFF, 0xFF, 0x00));//Перо для вывода последнего крестика или нолика

CPen penBlack(PS\_SOLID,1,RGB(0x00, 0x00, 0x00)); //Перо для вывода сетки

dc.SelectObject(&brushBgnd); //Выбираем кисть с задним фоном

for (int y=0;y<size\_y;y++)

{

for(int x=0;x<size\_x;x++)

{

//Выводим сетку

dc.SelectObject(&penBlack);

if ((x == 0) && (y == 0))

{

dc.Rectangle(0,0,y\*15+15,x\*15+15);

}

else if (x == 0)

{

dc.Rectangle(y\*15-1,0,y\*15+15,x\*15+15);

}

else if (y == 0)

{

dc.Rectangle(0,x\*15-1,y\*15+15,x\*15+15);

}

else

{

dc.Rectangle(y\*15-1,x\*15-1,y\*15+15,x\*15+15);

}

//Устанавливаем перо для содержимого клетки

switch (fields[x][y])

{

case 0:

continue;

break;

case 1:

dc.SelectObject(&penO);

break;

case 2:

dc.SelectObject(&penX);

break;

case 3:

case 4:

dc.SelectObject(&penWin);

break;

}

//Проверяем, не текущий ли ход

if ((last\_x == x) && (last\_y == y))

{

dc.SelectObject(&penLast);

}

//Выводим содержимое клетки

switch (fields[x][y])

{

case 0:

continue;

break;

//Выводим нолик

case 1:

case 3:

case 5:

dc.Ellipse(y\*15+2,x\*15+2,y\*15+13,x\*15+13);

break;

//Выодим крестик

case 2:

case 4:

case 6:

dc.MoveTo(y\*15+2,x\*15+2);

dc.LineTo(y\*15+12,x\*15+12);

dc.MoveTo(y\*15+2,x\*15+12);

dc.LineTo(y\*15+12,x\*15+2);

break;

}

//Возвращаем значения для последних выведенных элементов

if ((fields[x][y] == 5) || (fields[x][y] == 6))

{

fields[x][y] -= 2;

}

}

}

}

//Нажатие на левую кнопку мыши, здесь производится основной расчет

void CChildView::OnLButtonDown(UINT, CPoint xy)

{

//Проверка, не закончена ли игра

if (!end\_game)

{

//Ставим в массив игрового поля нолик в зависимости от координат мыши

if (fields[xy.y/15][xy.x/15] == 0)

{

fields[xy.y/15][xy.x/15] = 1;

last\_x = xy.y;

last\_y = xy.x;

}

else

return;

//Проводим анализ, не закончена ли игра

if (!end\_analyze())

{

//Игра не закончена

//Ход компьютера

ii();

//Проверяем, не закончена ли игра

if (end\_analyze())

{

//Игра закончена, выводим сообщение и отрисовываем на экран

this->OnPaint();

this->Invalidate();

this->MessageBox(CA2T("Вы проиграли"),0,0);

end\_game = true;//Ставим признак конца игры

return;

}

}

else

{

//Игра закончена, выводим сообщение и отрисовываем на экран

this->OnPaint();

this->Invalidate();

this->MessageBox(CA2T("Вы выиграли"),0,0);

end\_game = true;//Ставим признак конца игры

return;

}

//Перерисовка окна

this->OnPaint();

this->Invalidate();

}

return;

}

//Функция вычисления, не закончена ли игра

int CChildView::end\_analyze()

{

//Проход по всему полю (расчет ведется для каждой клетки)

for (int i=0;i<size\_x;i++)

{

for (int j=0;j<size\_y;j++)

{

//Пропускаем пустую клетку

if (fields[i][j]==0) continue;

int tek = fields[i][j];//Значение текущей клетки

int end;//Текущая длина ряда

int u;//Доп. счетчик

/////////////////////////////////////////

//Смотрим вправо от текущей клетки

end = 0;

for (int k = j;k<j+5;k++)

{

if ((k == size\_y) || (fields[i][k] != tek))

{

//Нет ряда из 5

break;

}

end++;

}

if (end == 5)

{

//Есть ряд из 5 - конец игры

for (int k = j;k<j+5;k++)

{

fields[i][k]=tek+2; //Заполняем все клетки в ряду значением + 2 для отсветки красным цветом

}

return 1;

}

/////////////////////////////////////////

//Смотрим вниз и вправо от текущей клетки

end = 0;

u=i;

for (int k = j;k<j+5;k++)

{

if ((k == size\_y) || (u==size\_x) || (fields[u][k] != tek))

{

//Нет ряда из 5

break;

}

end++;

u++;

}

if (end == 5)

{

//Есть ряд из 5 - конец игры

u=i;

for (int k = j;k<j+5;k++)

{

fields[u][k]=tek+2; //Заполняем все клетки в ряду значением + 2 для отсветки красным цветом

u++;

}

return 1;

}

/////////////////////////////////////////

//Смотрим вниз и влево от текущей клетки

end = 0;

u=i;

for (int k = j;k>j-5;k--)

{

if ((k == -1) || (u==size\_x) || (fields[u][k] != tek))

{

//Нет ряда из 5

break;

}

end++;

u++;

}

if (end == 5)

{

//Есть ряд из 5 - конец игры

u=i;

for (int k = j;k>j-5;k--)

{

fields[u][k]=tek+2; //Заполняем все клетки в ряду значением + 2 для отсветки красным цветом

u++;

}

return 1;

}

/////////////////////////////////////////

//Смотрим вниз от текущей клетки

end = 0;

for (int k = i;k<i+5;k++)

{

if ((k == size\_x) || (fields[k][j] != tek))

{

//Нет ряда из 5

break;

}

end++;

}

if (end == 5)

{

//Есть ряд из 5 - конец игры

for (int k = i;k<i+5;k++)

{

fields[k][j]=tek+2; //Заполняем все клетки в ряду значением + 2 для отсветки красным цветом

}

return 1;

}

}

}

//Игра не окончена

return 0;

}

//Функция расчета действий компьютера

void CChildView::ii()

{

float max = -1;//Максимальное значение оценочной функции

int cur\_x = 0,cur\_y = 0;//Текущие x и у

int povtor\_num = 0;//Количество повторов одинаковых значений оценочной функции

int cur\_povtor = 0;//Номер текущего повтора

//Расчитываем оценочную функцию для всех клеток

for (int i=0;i<size\_x;i++)

{

for (int j=0;j<size\_y;j++)

{

if (fields[i][j] == 0)

{

//Расчет оценочной функции

calc\_fields[i][j] = calculate(2,i,j) + calculate(1,i,j)\*(float)attack\_factor;

//Берем в расчет уровень (для профессионала случайности нет)

if (comp\_level == 1)//Для любителя (небольшая случайность)

{

calc\_fields[i][j] \*= (1 + ((float)rand() / 32767)) / 2;

}

if (comp\_level == 2)//Для новичка (максимальная случайность)

{

calc\_fields[i][j] \*= ((float)rand() / 32767);

}

if (calc\_fields[i][j] == max)

{

//Еще одна клетка с максимальным значением оценочной функции

povtor\_num++;

}

if (calc\_fields[i][j] > max)

{

//Клетка с максимальным значением оценочной функции

max = calc\_fields[i][j];

povtor\_num = 0;

cur\_x = i;

cur\_y = j;

}

}

}

}

//Проверяем, есть ли вообще свободные клетки на поле

if (max == -1)

{

return;

}

//Выбираем куда сделать ход

if (povtor\_num > 0)

{

//Выбираем куда ходить случайным образом из клеток с одинаковыми значениями оценочной функции

cur\_povtor = rand() / (32767 / povtor\_num);//Номер элемента, куда надо ходить

//Ищем его по полю

int buf\_povtor = -1;

for (int i=0;i<size\_x;i++)

{

for (int j=0;j<size\_y;j++)

{

if (calc\_fields[i][j] == max)

{

buf\_povtor++;

if (buf\_povtor == cur\_povtor) //Клетка найдена

{

fields[i][j] = 2;//Ставим крестик

last\_x = i;//Запоминаем координаты последнего хода

last\_y = j;

return;

}

}

}

}

}

else

{

//Одна клетка с максимальным знаечением

fields[cur\_x][cur\_y] = 2;//Ставим крестик

last\_x = cur\_x;//Запоминаем координаты последнего хода

last\_y = cur\_y;

}

}

//Функция расчета оценочной функции

unsigned long CChildView::calculate(int id,int x,int y)

{

//Подсчет оценочной функции

//Ставим в массиве временно значение == id

fields[x][y] = id;

int series\_length = 0;//Текущая длина ряда

unsigned long sum = 0;//Общее значение оценочной функции

///////////Расчет сверху вниз/////////

//Проход по каждой клетки, которая может входить в ряд

for (int i = 0;i<5;i++)

{

//Проверка, не вышли ли за границы поля

if ((x-4+i) < 0) continue;

if ((x+i) > (size\_x - 1)) break;

//Проход по всем возможным рядам, отстоящим от клетки не более чем на 5

for (int j=0;j<5;j++)

{

if ((fields[x-4+i+j][y] != id) && (fields[x-4+i+j][y] != 0))

{

//Конец ряда

series\_length = 0;

break;

}

if (fields[x-4+i+j][y] != 0) series\_length++; //Ряд увеличивается

}

if (series\_length == 1) series\_length = 0;//Ряд из самой клетки не учитываем

if (series\_length == 5) series\_length = 100; //Выигрышная ситуация, ставим большое значение

//Плюсуем серию к общей сумме

unsigned long pow\_st = valuation\_factor;

if (series\_length == 100)

{

if (id == 2)

pow\_st = 10000;//Большое значение при своем выигрыше

else

pow\_st = 1000; //Большое значение при выигрыше соперника, но меньшее, чем при своем

}

else

{

for (int i=0;i<series\_length;i++)//Возводим оценочный коэффициент в степень длины серии

{

pow\_st\*=valuation\_factor;

}

}

sum += pow\_st;

series\_length = 0;

}

///////////Расчет слева направо/////////

//Проход по каждой клетки, которая может входить в ряд

for (int i = 0;i<5;i++)

{

//Проверка, не вышли ли за границы поля

if ((y-4+i) < 0) continue;

if ((y+i) > (size\_y - 1)) break;

//Проход по всем возможным рядам, отстоящим от клетки не более чем на 5

for (int j=0;j<5;j++)

{

if ((fields[x][y-4+i+j] != id) && (fields[x][y-4+i+j] != 0))

{

//Конец ряда

series\_length = 0;

break;

}

if (fields[x][y-4+i+j] != 0) series\_length++; //Ряд увеличивается

}

if (series\_length == 1) series\_length = 0; //Ряд из самой клетки не учитываем

if (series\_length == 5) series\_length = 100; //Выигрышная ситуация, ставим большое значение

//Плюсуем серию к общей сумме

unsigned long pow\_st = valuation\_factor;

if (series\_length == 100)

{

if (id == 2)

pow\_st = 10000;//Большое значение при своем выигрыше

else

pow\_st = 1000; //Большое значение при выигрыше соперника, но меньшее, чем при своем

}

else

{

for (int i=0;i<series\_length;i++)//Возводим оценочный коэффициент в степень длины серии

{

pow\_st\*=valuation\_factor;

}

}

sum += pow\_st;

series\_length = 0;

}

///////////Расчет по диагонали с левого верхнего/////////

//Проход по каждой клетки, которая может входить в ряд

for (int i = 0;i<5;i++)

{

//Проверка, не вышли ли за границы поля

if ((y-4+i) < 0) continue;

if ((x-4+i) < 0) continue;

if ((x+i) > (size\_x - 1)) break;

if ((y+i) > (size\_y - 1)) break;

//Проход по всем возможным рядам, отстоящим от клетки не более чем на 5

for (int j=0;j<5;j++)

{

if ((fields[x-4+i+j][y-4+i+j] != id) && (fields[x-4+i+j][y-4+i+j] != 0))

{

//Конец ряда

series\_length = 0;

break;

}

if (fields[x-4+i+j][y-4+i+j] != 0) series\_length++; //Ряд увеличивается

}

if (series\_length == 1) series\_length = 0; //Ряд из самой клетки не учитываем

if (series\_length == 5) series\_length = 100; //Выигрышная ситуация, ставим большое значение

//Плюсуем серию к общей сумме

unsigned long pow\_st = valuation\_factor;

if (series\_length == 100)

{

if (id == 2)

pow\_st = 10000;//Большое значение при своем выигрыше

else

pow\_st = 1000; //Большое значение при выигрыше соперника, но меньшее, чем при своем

}

else

{

for (int i=0;i<series\_length;i++)//Возводим оценочный коэффициент в степень длины серии

{

pow\_st\*=valuation\_factor;

}

}

sum += pow\_st;

series\_length = 0;

}

///////////Расчет по диагонали с левого нижнего/////////

//Проход по каждой клетки, которая может входить в ряд

for (int i = 0;i<5;i++)

{

//Проверка, не вышли ли за границы поля

if ((y-4+i) < 0) continue;

if ((x+4-i) > (size\_x - 1)) continue;

if ((x-i) < 0) break;

if ((y+i) > (size\_y - 1)) break;

//Проход по всем возможным рядам, отстоящим от клетки не более чем на 5

for (int j=0;j<5;j++)

{

if ((fields[x+4-i-j][y-4+i+j] != id) && (fields[x+4-i-j][y-4+i+j] != 0))

{

//Конец ряда

series\_length = 0;

break;

}

if (fields[x+4-i-j][y-4+i+j] != 0) series\_length++; //Ряд увеличивается

}

if (series\_length == 1) series\_length = 0; //Ряд из самой клетки не учитываем

if (series\_length == 5) series\_length = 100; //Выигрышная ситуация, ставим большое значение

//Плюсуем серию к общей сумме

unsigned long pow\_st = valuation\_factor;

if (series\_length == 100)

{

if (id == 2)

pow\_st = 10000;//Большое значение при своем выигрыше

else

pow\_st = 1000; //Большое значение при выигрыше соперника, но меньшее, чем при своем

}

else

{

for (int i=0;i<series\_length;i++)//Возводим оценочный коэффициент в степень длины серии

{

pow\_st\*=valuation\_factor;

}

}

sum += pow\_st;

series\_length = 0;

}

//Возвращаем исходное значение

fields[x][y] = 0;

return sum;

}

//В меню нажата кнопка Новая игра

void CChildView::OnNewGame()

{

new\_game(); //Функция начала новой игры

this->OnPaint();

this->Invalidate();

}

//Выбрано поле 10x10

void CChildView::OnX1010()

{

//Ставим галочку в меню

switch (size\_x)

{

case 19:

set\_chеcked\_menu(ID\_X1919,ID\_X1010);

break;

case 30:

set\_chеcked\_menu(ID\_X3030,ID\_X1010);

break;

case 50:

set\_chеcked\_menu(ID\_X5050,ID\_X1010);

break;

case 100:

set\_chеcked\_menu(ID\_X100100,ID\_X1010);

break;

}

size\_x = 10;

size\_y = 10;

//Стартуем новую игру

new\_game();

//Устанавливаем новые размеры окна

resize\_window();

}

//Выбрано поле 19x19

void CChildView::OnX1919()

{

//Ставим галочку в меню

switch (size\_x)

{

case 10:

set\_chеcked\_menu(ID\_X1010,ID\_X1919);

break;

case 30:

set\_chеcked\_menu(ID\_X3030,ID\_X1919);

break;

case 50:

set\_chеcked\_menu(ID\_X5050,ID\_X1919);

break;

case 100:

set\_chеcked\_menu(ID\_X100100,ID\_X1919);

break;

}

size\_x = 19;

size\_y = 19;

//Стартуем новую игру

new\_game();

//Устанавливаем новые размеры окна

resize\_window();

}

//Выбрано поле 19x19

void CChildView::OnX3030()

{

//Ставим галочку в меню

switch (size\_x)

{

case 10:

set\_chеcked\_menu(ID\_X1010,ID\_X3030);

break;

case 19:

set\_chеcked\_menu(ID\_X1919,ID\_X3030);

break;

case 50:

set\_chеcked\_menu(ID\_X5050,ID\_X3030);

break;

case 100:

set\_chеcked\_menu(ID\_X100100,ID\_X3030);

break;

}

size\_x = 30;

size\_y = 30;

//Стартуем новую игру

new\_game();

//Устанавливаем новые размеры окна

resize\_window();

}

//Выбрано поле 50x50

void CChildView::OnX5050()

{

//Ставим галочку в меню

switch (size\_x)

{

case 10:

set\_chеcked\_menu(ID\_X1010,ID\_X5050);

break;

case 19:

set\_chеcked\_menu(ID\_X1919,ID\_X5050);

break;

case 30:

set\_chеcked\_menu(ID\_X3030,ID\_X5050);

break;

case 100:

set\_chеcked\_menu(ID\_X100100,ID\_X5050);

break;

}

size\_x = 50;

size\_y = 50;

//Стартуем новую игру

new\_game();

//Устанавливаем новые размеры окна

resize\_window();

}

//Выбрано поле 100x100

void CChildView::OnX100100()

{

//Ставим галочку в меню

switch (size\_x)

{

case 10:

set\_chеcked\_menu(ID\_X1010,ID\_X100100);

break;

case 19:

set\_chеcked\_menu(ID\_X1919,ID\_X100100);

break;

case 30:

set\_chеcked\_menu(ID\_X3030,ID\_X100100);

break;

case 50:

set\_chеcked\_menu(ID\_X5050,ID\_X100100);

break;

}

size\_x = 100;

size\_y = 100;

//Стартуем новую игру

new\_game();

//Устанавливаем новые размеры окна

resize\_window();

}

//Функция начала новой игры

void CChildView::new\_game()

{

if ((old\_size\_x != 0) || (old\_size\_y != 0))

{

//Освобождаем память динамических массивов

try

{

for (int i=0;i<old\_size\_x;i++)

{

delete fields[i];

delete calc\_fields[i];

}

delete fields;

delete calc\_fields;

}

catch(...)

{

this->MessageBox(CA2T("Ошибка работы с памятью"),0,0);

this->CloseWindow();

}

}

//Присваиваем старым значениям новые

old\_size\_x = size\_x;

old\_size\_y = size\_y;

//Выделяем память под массивы

try

{

fields = new unsigned char\*[size\_x];

calc\_fields = new float\*[size\_x];

for(int i=0;i<size\_x;i++)

{

fields[i] = new unsigned char[size\_y];

calc\_fields[i] = new float[size\_y];

}

}

catch(...)

{

this->MessageBox(CA2T("Ошибка выделения памяти"),0,0);

this->CloseWindow();

}

//Обнуляем массивы

for (int i=0;i<size\_x;i++)

for (int j=0;j<size\_y;j++)

{

fields[i][j] = 0;

}

//Сбрасываем флаг начала игры

end\_game = false;

//Проверяем, не должен ли компьютер ходить первым

if (!player\_first\_step)

ii();

}

//Установка новых размеров окна

void CChildView::resize\_window()

{

//Устанавливаем размеры окна в соответствии с размером поля

RECT Rect;//Размеры внутренней области

RECT WindowRect;//Размеры окна

this->GetWindowRect(&Rect);

this->GetParent()->GetWindowRect(&WindowRect);

this->GetParent()->MoveWindow(WindowRect.left,WindowRect.top,

size\_y\*15 + 4 + (WindowRect.right - Rect.right) + (Rect.left - WindowRect.left),

size\_x\*15 + 4 + (WindowRect.bottom - Rect.bottom) + (Rect.top - WindowRect.top),1);

}

//Установка галочки в меню

void CChildView::set\_chеcked\_menu(unsigned int old\_id,unsigned int new\_id)

{

//Создаем структуру элемента меню

MENUITEMINFO menuItemInfo;

menuItemInfo.cbSize = sizeof(MENUITEMINFO);

menuItemInfo.fMask = MIIM\_STATE;//Работа с состоянием элемента

//Получаем указатель на меню

CMenu \*pMenu=this->GetParent()->GetMenu();

//Ставим галочку в новом элементе

menuItemInfo.fState = MFS\_CHECKED;

pMenu->SetMenuItemInfoW(new\_id,&menuItemInfo,0);

//Снимаем галочку в старом элементе

menuItemInfo.fState = MFS\_UNCHECKED;

pMenu->SetMenuItemInfoW(old\_id,&menuItemInfo,0);

return;

}

//Выбран первый ход человека

void CChildView::OnStepH()

{

//Ставим галочку в меню

if (player\_first\_step == false)

set\_chеcked\_menu(ID\_STEP\_C,ID\_STEP\_H);

player\_first\_step = true;

//Старт новой игры

new\_game();

//Перерисовка окна

this->OnPaint();

this->Invalidate();

}

//Выбран первый ход компьютера

void CChildView::OnStepC()

{

//Ставим галочку в меню

if (player\_first\_step == true)

set\_chеcked\_menu(ID\_STEP\_H,ID\_STEP\_C);

player\_first\_step = false;

//Старт новой игры

new\_game();

//Перерисовка окна

this->OnPaint();

this->Invalidate();

}

//Выбран уровень компьютера - новичок

void CChildView::OnLevelBeg()

{

//Ставим галочку в меню

if (comp\_level == 0)

{

set\_chеcked\_menu(ID\_LEVEL\_PROF,ID\_LEVEL\_BEG);

}

else if (comp\_level == 1)

{

set\_chеcked\_menu(ID\_LEVEL\_AMAT,ID\_LEVEL\_BEG);

}

comp\_level = 2;

attack\_factor = 1;

//Старт новой игры

new\_game();

//Перерисовка окна

this->OnPaint();

this->Invalidate();

}

//Выбран уровень компьютера - любитель

void CChildView::OnLevelAmat()

{

//Ставим галочку в меню

if (comp\_level == 2)

{

set\_chеcked\_menu(ID\_LEVEL\_BEG,ID\_LEVEL\_AMAT);

}

else if (comp\_level == 0)

{

set\_chеcked\_menu(ID\_LEVEL\_PROF,ID\_LEVEL\_AMAT);

}

comp\_level = 1;

attack\_factor = 10;

//Старт новой игры

new\_game();

//Перерисовка окна

this->OnPaint();

this->Invalidate();

}

//Выбран уровень компьютера - профессионал

void CChildView::OnLevelProf()

{

//Ставим галочку в меню

if (comp\_level == 2)

{

set\_chеcked\_menu(ID\_LEVEL\_BEG,ID\_LEVEL\_PROF);

}

else if (comp\_level == 1)

{

set\_chеcked\_menu(ID\_LEVEL\_AMAT,ID\_LEVEL\_PROF);

}

comp\_level = 0;

attack\_factor = 1;

//Старт новой игры

new\_game();

//Перерисовка окна

this->OnPaint();

this->Invalidate();

}

7. Примеры выполнения программы

Примеры выполнения программы приведены на рисунках 2, 3, 4, 5.



Рисунок 2 – Вид окна программе при запуске.

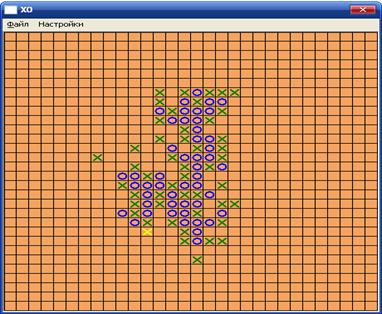


Рисунок 3 – Игровая ситуация при игре на поле 30x30 на сложности любитель.

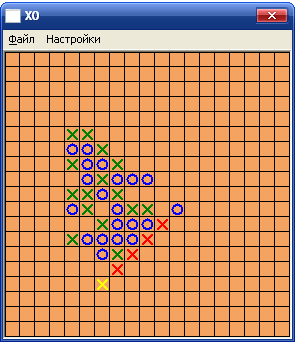


Рисунок 4 – Выигрыш компьютера при игре на сложности профессионал.

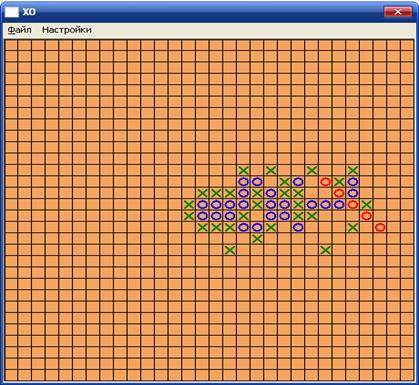


Рисунок 5 – Победа игрока при игре на сложности любитель на поле 30x30.

Выводы

В ходе выполнения курсовой работы был разработан алгоритм действий компьютерного игрока при игре в крестики-нолики пять в ряд на бесконечном поле.

В ходе выполнения работы удалось добиться достаточно сильного уровня игры на сложности профессионал.

Разработанный алгоритм был реализован на языке C++ на платформе Microsoft Visual Studio 2008.

Список литературы

1.         Арчер Т., Уайтчепел Э. Visual C++ .NET. Библия пользователя — Вильямс, 2007.

2.         Либерти Д., Джонс Б. Освой самостоятельно C++ за 21 день. — Вильямс, 2007.

3.         Материалы сайта http://www.firststeps.ru/mfc/steps.

4.         Подбельский В.В. Язык C++. — М.:"Финансы и статистика", 2003.