Міністерство освіти і науки України

Національний технічний університет «ХПІ»

Кафедра «Інформатика та інтелектуальна власність»

**КУРСОВА РОБОТА**

З дисципліни «Основи програмування»

**Тема**

«Комп’ютерна гра «Шашки»»

Керівник курсової роботи

Максим Соболь

Виконавець

Анатолій Климчук

Харків 2020

ЗАВДАННЯ КУРСОВОЇ РОБОТИ

Гра являє собою ігрове поле, кнопки Старт, Здатися, а також показник кількості шашок, що залишилися, для кожної зі сторін. Також при перемозі чи поразці виводиться відповідне повідомлення. Гра відбувається за правилами Російських шашок.

РЕФЕРАТ

Пояснювальна записка містить: 32 стор., 2 рис, 5 джерел.

Предмет дослідження ­ – сучасні методи розробки програм таких, як об’єктно-орієнтоване програмування та візуальне проектування, а також структурне і модульне програмування.

Мета курсової роботи ­­– систематизація, поглиблення і активне застосування знань з програмування, закріплення знань, отриманих в лекційному курсі, а також на практичних і лабораторних заняттях.

Метод дослідження – вивчення літератури, складання і налагодження програм на комп’ютері.

Програма «Checkers», що реалізована в цій роботі, є реалізацією комп’ютерної гри «Шашки». Розроблено проект «Checkers» повністю відповідний умові завдання і має зручний інтерфейс.

ключові слова: шашки, checkers, visual c++, проект, програма, повідомлення, Метод, windows forms.

ЗМІСТ

Вступ....………………………………………………………………………………….5

1. Аналіз предметної області..………………………..……………………………….6
   1. Основні особливості середовища програмування VS.......................................6
   2. Стандартні компоненти………………………………………………………...6
2. Розробка програми…………………………………………………………………..7
   1. Опис алгоритму створення форми.…………………………………………….7
   2. Опис алгоритму створення елементів форми…………………………………8
   3. Опис використовуваних методів і подій……………………………………...10
   4. Текст програми з коментарями………………………………………………..28
3. Опис програмної реалізації………………………………………………………..33
   1. Інтерфейс користувача………………………………………………………...33
   2. Інструкція користувача………………………………………………………..33

Висновок……………………………………………………………………………….34

Список джерел інформації……………………………………………………………35

ВСТУП

У сучасному світі існує багато варіантів реалізації комп’ютерної гри «Шашки», що є дуже популярною. Грати у шашки на комп’ютері зручніше, ніж на справжній дошці, тому що програма автоматично розставляє фігури, слідкує за дотриманням правил. Тому серед користувачів є попит на якомога функціональну та зручну реалізацію цієї гри, а спроби зробити таку програму є актуальними.

Програма може бути застосована як розвага, або тренажер уваги та логічного мислення.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

# Основні особливості середовища програмування VS

Visual Studio ­– це середовище для програмування від Microsoft, яке має зручний інтерфейс, що легко налаштовується під будь-якого користувача, а також велику кількість інструментів для програмування.

# Основні компоненти.

У середовищі програмування Microsoft Visual Studio є інструменти для програмування на багатьох мовах. Це різноманітні інструменти для редагування тексту (коду), перевірки правильності написання коду, налагодження програми (відладчик, точки останову та ін.), редагування складу рішення (Solution Explorer), інструменти для зміни налаштувань проекту, властивостей елементів проекту, та ін.

Також Microsoft Visual Studio містить компілятор і будівник програм, які перетворюють написаний код у машинний та таким чином будують програму.

1. РОЗРОБКА ПРОГРАМИ

# Опис алгоритму створення форми

Для створення програми використовується форма Windows Forms. [1] Для її створення у проекті С++ треба перейти у меню Проект – Додати – Новий елемент, та у меню вибрати Visual С++ – UI – Windows Form, як показано на рис. 2.1.

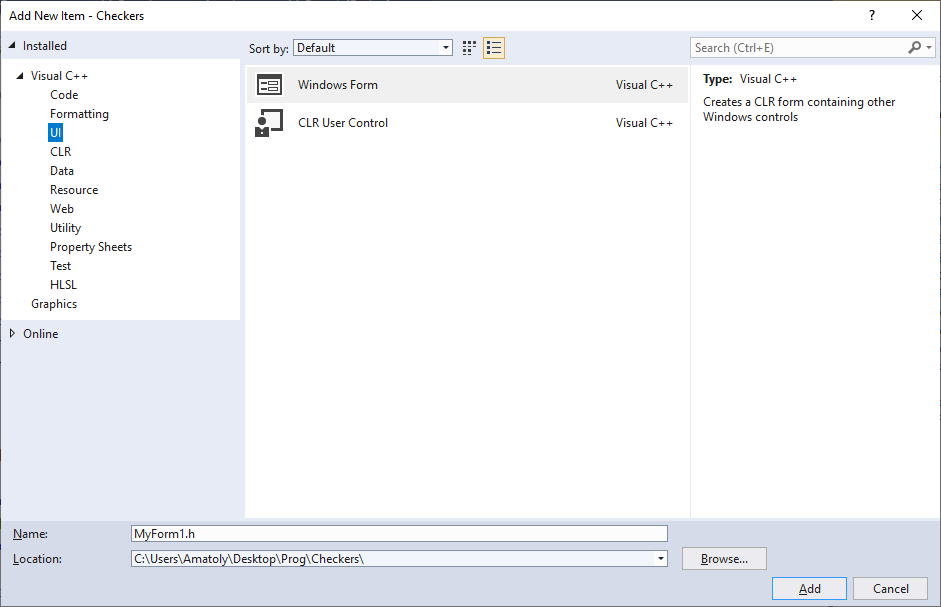


Рисунок 2.1– Вибір пункту Windows Form

Таким чином з’явиться 2 файли форми \*Назва\*.h та \*Назва\*.cpp, а також вікно редагування форми. У першому файлі містяться дані про форму та її елементи, у другому файлі пишуться усі методи та програмний код форми.

# Опис алгоритму створення елементів форми.

Для того, щоб розташувати елемент форми, треба у вікні редактора форми обрати вікно «Панель інструментів» («Toolbox») та у ньому обираємо елемент, який бажаємо розташувати на формі, як наведено на Рис. 2.2.

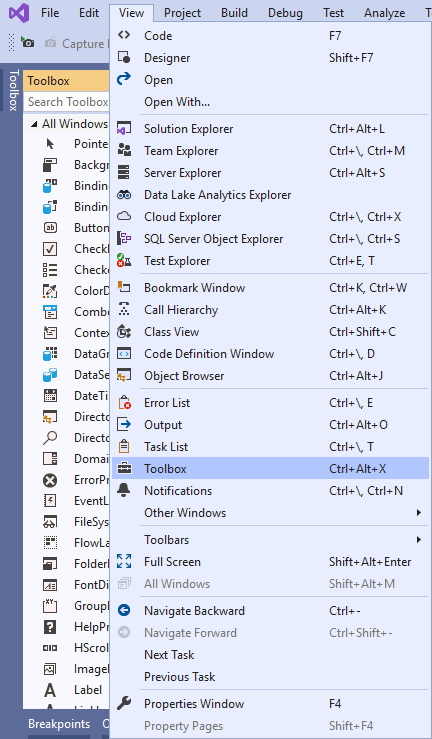


Рисунок 2.2 – Вибір елементу Toolbox

Після вибору елементу треба натиснути на те місце на формі, куди треба розташувати елемент. Після цього він з’явиться на екрані. Щоб редагувати властивості елемента, треба виділити елемент 1 натисканням або серед списку у вікні «Властивості» («Properties»), яке можна викликати через пункти «Вид – Вікно властивостей (Properties Window)», або за допомогою кнопки F4. У вікні властивостей містяться властивості елемента або, якщо обрано більше 1 елемента, властивості, що можна застосувати для усіх вибраних елементів.

Можна змінити властивості самого вікна. Вікно програми «Checkers» неможливо розвернути на весь екран.

Таким чином, у програмі є елемент Label, що виводить текстуру дошки, елемент GroupBox, який об'єднує 4 кнопки – елементи Label: «Выход», «Сдаться», «Ничья» та «Правила». У межах дошки на місцях чорних клітин розташовані 32 кнопки. Їх назви – це перетин відповідних координат на дошці за прикладом «А1». Вони виводять текстуру фігури, розташованої у цій клітині а також виконують дії з фігурами. Як само – описано нижче.

Кінцевий вид програми приведений на Рис. 2.3

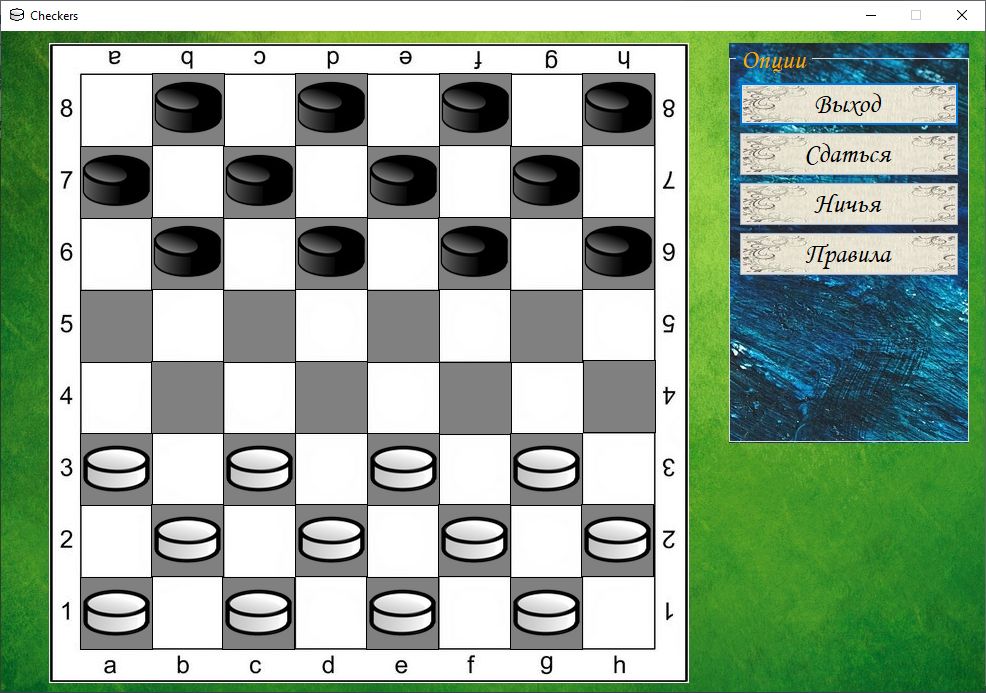


Рис. 2.3 – Кінцевий вигляд програми

# Опис використовуваних методів і подій

У програмі «Checkers» використовуються такі події та методи:

Подія Click для кожної кнопки форми.

Методи Windows Forms:

Обробники натискання:

//Обработчики нажатий

//Кнопка Выход - завершение программы.

private: System::Void Exit\_Click(System::Object^ sender, System::EventArgs^ e);

//Кнопка Сдаться - вывод сообщения "победа ... " и перезапуск программы.

private: System::Void GiveUp\_Click(System::Object^ sender, System::EventArgs^ e);

//Кнопка Сдаться - вывод сообщения "... предложили ничью." и перезапуск программы.

private: System::Void Standoff\_Click(System::Object^ sender, System::EventArgs^ e);

//Кнопка "Правила" - вывод сообщения с правилами игры.

private: System::Void Rules\_Click(System::Object^ sender, System::EventArgs^ e);

//Обработчики нажатий на кнопки на доске

private: System::Void B8\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void D8\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void F8\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void H8\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void A7\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void C7\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void E7\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void G7\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void B6\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void D6\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void F6\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void H6\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void A5\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void C5\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void E5\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void G5\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void B4\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void D4\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void F4\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void H4\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void A3\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void C3\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void E3\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void G3\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void B2\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void D2\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void F2\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void H2\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void A1\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void C1\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void E1\_Click(System::Object^ sender, System::EventArgs^ e);

private: System::Void G1\_Click(System::Object^ sender, System::EventArgs^ e);

Методи класу «Checkers»:

public:

ch();//Конструктор

~ch();//Деструктор

//Основная функция, с которой начинаются проверки.

short go();

//Ввод данных

short SetXY(unsigned short x, unsigned short y,short st);

// Общие функции

void refresh();//Заново

bool anycan();//проверка на возможность побить/походить любой фигурой

void convert();//Функция перевода шашек в дамки по достижению ими соотв. полей

//Геттеры и Сеттеры.

short Getdesk(short x, short y) { return desk[y][x]; }//содержимое 1 ячейки на доске.

short Getcol() { return col; }

unsigned short GetPlayer() { return Plnow; }// Узнать какой игрок сейчас ходит.

//Узнать первые и вторые X и Y

unsigned short GetFX() { return firstX; }

unsigned short GetFY() { return firstY; }

unsigned short GetSX() { return secondX; }

unsigned short GetSY() { return secondY; }

//Узнать координаты побитой фигуры.

int GetBX() { return beatX; }

int GetBY() { return beatY; }

private:

void move();//Ход фигурой

//Проверки для пешки

short checker();

void checkerhit();//Удар пешкой

//проверки для дамки

short king();

void kinghit(short i, short j);//Удар дамкой

//Задать первые и вторые X и Y

void SetFX(unsigned short x) { firstX = x; }

void SetFY(unsigned short y) { firstY = y; }

void SetSX(unsigned short x) { secondX = x; }

void SetSY(unsigned short y) { secondY = y; }

//Задать координаты побитой фигуры.

void SetBX(unsigned short x) { beatX = x; }

void SetBY(unsigned short y) { beatY = y; }

//Проверки на возможность удара

bool canhit();//Для всей доски

bool CanHitChecker(short i, short j);//Для одной шашки

bool CanHitKing(short i, short j);//Для одной дамки

Реалізація конструктора класу

ch::ch() {

Змінні гравців ставляться у положення білого гравця, а координати у положення 0

Plnow = WHITE;

col = W;

FX = 0;

FY = 0;

SX = 0;

SY = 0;

Об’являється двовимірний масив desk та заповнюється цифрами, які потрібні у началі гри.

desk = new short\* [K];

for (short i = 0; i < K; i++) {

desk[i] = new short[K];

}

for (short y = 0; y < K; y++) {

for (short x = 0; x < K; x++)

{

desk[y][x] = 0;

}

}

for (short y = 0; y < 3; y++) {

for (short x = 0; x < K; x++)

{

if ((x + y) % 2 == 0) {

desk[y][x] = CHECKER + WHITE;

}

}

}

for (short i = 5; i < K; i++) {

for (short j = 0; j < K; j++)

{

if ((i + j) % 2 == 0) {

desk[i][j] = CHECKER + BLACK;

}

}

}

Можливість для розробника на етапі написання алгоритмів у вигляді консольної програми перевірити різні ситуації на полі за допомогою ручного заповнювання масиву, тобто дошки.

//Ручной ввод массива для проверки различных ситуаций.

//desk[7] = new short[8]{ 0,3,0,3,0,3,0,3 };

//desk[6] = new short[8]{ 3,0,3,0,3,0,3,0 };

//desk[5] = new short[8]{ 0,3,0,3,0,3,0,3 };

//desk[4] = new short[8]{ 0,0,0,0,0,0,0,0 };

//desk[3] = new short[8]{ 0,0,0,0,0,0,0,0 };

//desk[2] = new short[8]{ 1,0,1,0,1,0,1,0 };

//desk[1] = new short[8]{ 0,1,0,1,0,1,0,1 };

//desk[0] = new short[8]{ 1,0,1,0,1,0,1,0 };

}

Реалізація деструктора класу

ch::~ch() {

delete (desk);

}

Основний Метод, який викликається після натискання другої клітинки. Її задача – перевірити основні умови для можливості ходу, що є однаковими для обох типів фігур: клітинка, куди здійснюється хід потрібна бути пустою, розташовуватися по діагоналі до першої клітинки. Якщо ці вимоги виконуються, йде перевірка, чи є своя фігура на стартовій клітинці, та в залежності від того, що це – пішак або дама, запускає перевірку умов для пішака або дамки.

short ch::go() {

//Клетка "куда идти" пуста?

if (!SP) {

//Ход по диагонали?

if ((FX - SX) == (FY - SY) || (FX - SX) == -(FY - SY)) {

// это пешка нынешнего игрока?

if (FP == CHECKER + Plnow) {

// проверка условий хода для пешки и дейстиве если условия выполнятся.

return checker();

}

// это дамка нынешнего игрока?

else if (FP == KING + Plnow) {

// проверка условий хода для дамки

return king();

}

else {

MessageBox::Show("Нельзя ходить не своими фигурами!", "Ошибка!");

return -1;

}

}

else {

MessageBox::Show("В шашках ходят только по диагонали!", "Ошибка!");

return -1;

}

}

else {

MessageBox::Show("Нельзя ходить на непустую клетку!", "Ошибка!");

return -1;

}

}

short ch::SetXY(unsigned short x, unsigned short y, short st)

{

if (st == 0) {

FX = x;

FY = y;

return -2;

}

else {

SX = x;

SY = y;

if (SY != FY && SY != -FY) return go();

else {

return -1;

MessageBox::Show("Нельзя ходить на ту же клетку, с которой пытаетесь походить!", "Ошибка");

}

}

}

Метод перезапуску гри (є потреба тільки у вигляді консольної програми). Дії аналогічні конструктору класу.

void ch::refresh() {

Plnow = WHITE;

col = W;

FX = 0;

FY = 0;

SX = 0;

SY = 0;

for (short y = 0; y < K; y++) {

for (short x = 0; x < K; x++)

{

desk[y][x] = 0;

}

}

for (short y = 0; y < 3; y++) {

for (short x = 0; x < K; x++)

{

if ((x + y) % 2 == 0) {

desk[y][x] = CHECKER + WHITE;

}

}

}

for (short i = 5; i < K; i++) {

for (short j = 0; j < K; j++)

{

if ((i + j) % 2 == 0) {

desk[i][j] = CHECKER + BLACK;

}

}

}

}

Метод звичайного ходу. Переміщує фігуру з 1 місця у друге, передає право ходу іншому гравцю та перевіряє ситуацію на перемогу (якщо гравець не може нічого зробити , перемога присуджується іншому та програма перезапускається).

void ch::move()

{

SP = FP; // перенос шашки\дамки на новую клетку

FP = NONE;// опустошение старой клетки

Plnow = 2 - Plnow;// переход права хода к другому игроку

col = col \* (-1);

convert();

if (!anycan()) {

if (Plnow == WHITE) {

MessageBox::Show("Победа ЧЕРНОГО игрока!", "Победа!");

Application::Restart();

}

else {

MessageBox::Show("Победа БЕЛОГО игрока!", "Победа!");

Application::Restart();

}

}

}

Перевірка на можливість гравця зробити хоч якусь дію. Якщо хоч 1 вимога виконується, гравець може щось робити.

bool ch::anycan() {//проверка на возможность ходить или бить для всей доски

//вызов проверки на возможность бить

if (canhit()) {

return 1;

}

//если бить нельзя проверяем возможность хода, проверяя, свободны ли соседние клетки.

else {

for (short i = 0; i < K; i += 2) {//Проход по массиву

for (short j = 0; j < K; j += 2) {

if (desk[i][j] == CHECKER + Plnow) {//Это наша шашка?

//Пешка может ходить?

if (Plnow == WHITE) {

if (i < 7) {//фишка не на верхнем краю?

if (j < 7) {//фишка не на правом краю?

if (!desk[i + 1][j + 1]) {

return 1;

}

}

if (j > 0) { //фишка не на левом краю?

if (!desk[i + 1][j - 1]) {

return 1;

}

}

}

}

else

{

if (i > 0) { //фишка не на нижнем краю?

if (j < 7) { //фишка не на правом краю?

if (!desk[i - 1][j + 1]) {

return 1;

}

}

if (j > 0) {//фишка не на левом краю?

if (!desk[i - 1][j - 1]) {

return 1;

}

}

}

}

}

else if (desk[i][j] == KING + Plnow) {//Это дамка?

//Дамка может ходить?

if (i < 7) { //фишка не на верхнем краю?

if (j < 7) { //фишка не на правом краю?

if (!desk[i + 1][j + 1]) {

return 1;

}

}

if (j > 0) {//фишка не на левом краю?

if (!desk[i + 1][j - 1]) {

return 1;

}

}

}

if (i > 0) { //фишка не на нижнем краю?

if (j < 7) { //фишка не на правом краю?

if (!desk[i - 1][j + 1]) {

return 1;

}

}

if (j > 0) { //фишка не на левом краю?

if (!desk[i - 1][j - 1]) {

return 1;

}

}

}

}

}

}

for (short i = 1; i < K; i += 2) {//Проход по массиву

for (short j = 1; j < K; j += 2) {

if (desk[i][j] == CHECKER + Plnow) {//Это наша шашка?

//Пешка может ходить?

if (Plnow == WHITE) {

if (i < 7) {

if (j < 7) {

if (!desk[i + 1][j + 1]) {

return 1;

}

}

if (j > 0) {

if (!desk[i + 1][j - 1]) {

return 1;

}

}

}

}

else//Black

{

if (i > 0) {

if (j < 7) {

if (!desk[i - 1][j + 1]) {

return 1;

}

}

if (j > 0) {

if (!desk[i - 1][j - 1]) {

return 1;

}

}

}

}

}

else if (desk[i][j] == KING + Plnow) {//Это дамка?

//Дамка может ходить?

if (i < 7) {

if (j < 7) {

if (!desk[i + 1][j + 1]) {

return 1;

}

}

if (j > 0) {

if (!desk[i + 1][j - 1]) {

return 1;

}

}

}

if (i > 0) {

if (j < 7) {

if (!desk[i - 1][j + 1]) {

return 1;

}

}

if (j > 0) {

if (!desk[i - 1][j - 1]) {

return 1;

}

}

}

}

}

}

}

return 0;

}

Метод перетворення пішаків на дами якщо вони дійшли до краю карти.

void ch::convert() {

//Если белая шашка на 7 строке(для игрока на 8) - превращаем в дамку.

for (short i = 1; i < K; i += 2) {

if (desk[7][i] == CHECKER + WHITE) {

desk[7][i]++;

}

}

//Если черная шашка на 0 строке(для игрока на 1) - превращаем в дамку.

for (short i = 0; i < K; i += 2) {

if (desk[0][i] == CHECKER + BLACK) {

desk[0][i]++;

}

}

}

Розгалуження перевірок для пішаків.

short ch::checker()// для Пешек

{

if (SY - FY == 1 \* col) {

//возможно ли бить?

if (!canhit())//если нет

{

move();// Перемещение фишки

return 1;

}

else {

MessageBox::Show("Ход невозможен, вы можете бить!", "Ошибка!");

return -1;

}

}

//Удар

else if (abs(SY - FY) == 2) {

//проверка на враждебную фишку между точками назначения.

if ((desk[SY - ((SY - FY) / 2)][SX - ((SX - FX) / 2)]) == CHECKER + (2 - Plnow)

|| (desk[SY - ((SY - FY) / 2)][SX - ((SX - FX) / 2)]) == KING + (2 - Plnow)) {

//Удар пешкой

checkerhit();

return 2;

}

else {

MessageBox::Show("Вы пытаетесь побить чвою фигуру или пустую клетку!", "Ошибка!");

return -1;

}

}

else {

MessageBox::Show("Пешка может ходить на 1 клетку вперед или бить через фигуру на 2 клетки!", "Ошибка!");

return -1;

}

}

Метод удару пішаком. Після удару перевіряється можливість бити ще раз та перевірка на перемогу 1 з сторін.

void ch::checkerhit() {

SP = FP; // перенос шашки на новую клетку

desk[SY - ((SY - FY) / 2)][SX - ((SX - FX) / 2)] = NONE;//Опустошение клетки, где стояла побитая фигура.

this->beatX = (SX - ((SX - FX) / 2));

this->beatY = (SY - ((SY - FY) / 2));

// опустошение старой клетки

FP = NONE;

//Конвертирование пешки в дамку если она достигла дамочного поля

convert();

// Переход права хода к другому игроку, если он не может бить снова.

if (SP == CHECKER + Plnow) {

if (!CanHitChecker(SY, SX)) {

Plnow = 2 - Plnow;

col = col \* (-1);

if (!anycan()) {

if (Plnow == WHITE) {

MessageBox::Show("Победа ЧЕРНОГО игрока!", "Победа!");

Application::Restart();

}

else {

MessageBox::Show("Победа БЕЛОГО игрока!", "Победа!");

Application::Restart();

}

}

}

else {

MessageBox::Show("Вы можете бить дальше!", "Ходите снова");

SX = FX;

SY = FY;

}

}

//Случай, когда шашка стала дамкой после побития фишки врага.

else {

if (!CanHitKing(SY, SX)) {

Plnow = 2 - Plnow;

col = col \* (-1);

convert();

if (!anycan()) {

if (Plnow == WHITE) {

MessageBox::Show("Победа ЧЕРНОГО игрока!", "Победа!");

Application::Restart();

}

else {

MessageBox::Show("Победа БЕЛОГО игрока!", "Победа!");

Application::Restart();

}

}

}

else {

MessageBox::Show("Вы можете бить дальше!", "Ходите снова");

SX = FX;

SY = FY;

}

}

}

Перевірки для дами.

short ch::king()

{

short p;

short d;

//Если идет на 1 клетку то это не удар. проверка только на ход.

if (abs(FY - SY) == 1) {

//возможно ли бить?

if (!canhit())

{

move();

return 1;

}

else {

MessageBox::Show("Ход невозможен, вы можете бить!", "Ошибка!");

return -1;

}

}

else if (abs(FX - SX) > 1) {

short rez = 0, a = 0, b = 0;

d = (SY - FY) / (abs(SY - FY));

p = (SX - FX) / (abs(SX - FX));

//Считаем сколько чужих фигур на пути у дамки.

for (short i = FY + 1 \* d, j = FX + 1 \* p, c = 0; i != SY, j != SX, c < abs(SX - FX); i += d, j += p, c++)

{

if (desk[i][j] == CHECKER + (2 - Plnow) || desk[i][j] == KING + (2 - Plnow)) {

rez++;

a = i;

b = j;

}

else if (desk[i][j] == CHECKER + Plnow || desk[i][j] == KING + Plnow) {

MessageBox::Show("На пути не должно быть собственных фигур!", "Ошибка!");

return -1;

}

}

//Если на пути нету ничьих фишек - просто ход.

if (rez == 0) {

if (!canhit())

{

move();

return 1;

}

else {

MessageBox::Show("Ход невозможен, вы можете бить!", "Ошибка!");

return -1;

}

}

//На пути у дамки 1 фишка противника, производится удар.

if (rez == 1) {

kinghit(a, b);

return 2;

}

// на пути больше 2 фишек противника.

else {

MessageBox::Show("На пути дамки должна быть только 1 фигура противника!", "Ошибка!");

return -1;

}

}

else return -1;

}

Метод удару дамою. Записуються координати побитої фігури, перевіряється можливість бити ще раз та умови перемоги.

void ch::kinghit(short a, short b) {

// перенос шашки на новую клетку

SP = FP;

//Опустошение клетки, где была взята шашка.

desk[a][b] = NONE;

//Запись её координат

beatX = b;

beatY = a;

// опустошение старой клетки

FP = NONE;

convert();

// переход права хода к другому игроку, если он не может бить снова.

if (!CanHitKing(SY, SX)) {

Plnow = 2 - Plnow;

col = col \* (-1);

if (!anycan()) {

if (Plnow == WHITE) {

MessageBox::Show("Победа ЧЕРНОГО игрока!", "Победа!");

Application::Restart();

}

else {

MessageBox::Show("Победа БЕЛОГО игрока!", "Победа!");

Application::Restart();

}

}

}

else {

MessageBox::Show("Вы можете бить дальше!", "Ходите снова");

SX = FX;

SY = FY;

}

}

Метод перевірки на можливість бити хоч 1 фігурою

bool ch::canhit() {// Проверки для всей доски на возможность бить. Если ни разу не дойдет до return 1 - ходить можно

for (short i = 0; i < K; i += 2) {//Проход по массиву

for (short j = 0; j < K; j += 2) {

if (desk[i][j] == CHECKER + Plnow) {//Это наша шашка?

if (CanHitChecker(i, j))return 1;//Проверка условий для ШАШКИ

}

else if (desk[i][j] == KING + Plnow) {//Это дамка?

if (CanHitKing(i, j))return 1;//Проверка условий для ДАМКИ

}

}

}

for (short i = 1; i < K; i += 2) {//Проход по массиву

for (short j = 1; j < K; j += 2) {

if (desk[i][j] == CHECKER + Plnow) {//Это наша шашка?

if (CanHitChecker(i, j))return 1;//Проверка условий для ШАШКИ

}

else if (desk[i][j] == KING + Plnow) {//Это дамка?

if (CanHitKing(i, j))return 1;//Проверка условий для ДАМКИ

}

}

}

return 0;

}

Метод для перевірки можливості бити пішаком, що знаходиться за зазначеними координатами.

bool ch::CanHitChecker(short i, short j) {

if (i < 6) {//Не выйдем ли за верхний край доски?

if (j < 6) {//Не выйдем ли за правый край доски?

if (!desk[i + 2][j + 2]) {//через клетку справа свободно?

if (desk[i + 1][j + 1] == CHECKER + (2 - Plnow)

|| desk[i + 1][j + 1] == KING + (2 - Plnow)) {//Есть ли враг?

return 1;

}

}

}

if (j > 1) {//Не выйдем ли за левый край доски?

if (!desk[i + 2][j - 2]) {//через клетку слева свободно?

if (desk[i + 1][j - 1] == CHECKER + (2 - Plnow)

|| desk[i + 1][j - 1] == KING + (2 - Plnow)) {//Есть ли враг?

return 1;

}

}

}

}

if (i > 1) {//Не выйдем ли за нижний край доски?

if (j < 6) {//Не выйдем ли за правый край доски?

if (!desk[i - 2][j + 2]) {//через клетку справа свободно?

if (desk[i - 1][j + 1] == CHECKER + (2 - Plnow)

|| desk[i - 1][j + 1] == KING + (2 - Plnow)) {//Есть ли враг?

return 1;//Бить возможно, ход недоступен

}

}

}

if (j > 1) {//Не выйдем ли за левый край доски?

if (!desk[i - 2][j - 2]) {//через клетку слева свободно?

if (desk[i - 1][j - 1] == CHECKER + (2 - Plnow)

|| desk[i - 1][j - 1] == KING + (2 - Plnow)) {//Есть ли враг?

return 1;//Бить возможно, ход недоступен

}

}

}

}

return 0;

}

Перевірка на можливість бити дамою за зазначеними координатами.

bool ch::CanHitKing(short i, short j) {

if (i < 6) {//Не выйдем ли за верхний край доски?

if (j < 6) {//Не выйдем ли за правый край доски?

for (int k = 0; i + k < 7, j + k < 7; k++) {

if ((desk[i + 1][j + 1] == CHECKER + (2 - Plnow)

|| desk[i + 1][j + 1] == KING + (2 - Plnow))

&& !desk[i + k + 1][j + k + 1])

{//Если есть враг и пустая клетка за ним, возврат 1, т. к. можно бить.

return 1;

}

else if ((desk[i + 1][j + 1] == CHECKER + (2 - Plnow)

|| desk[i + 1][j + 1] == KING + (2 - Plnow))

&& desk[i + k + 1][j + k + 1])

{

break;

}

}

if (j > 1) {//Не выйдем ли за левый край доски?

for (int k = 0; i + k < 7, j - k >0; k++) {

if ((desk[i + 1][j - 1] == CHECKER + (2 - Plnow)

|| desk[i + 1][j - 1] == KING + (2 - Plnow))

&& !desk[i + k + 1][j - k - 1])

{//Если есть враг и пустая клетка за ним, возврат 1, т. к. можно бить.

return 1;

}

else if ((desk[i + 1][j - 1] == CHECKER + (2 - Plnow)

|| desk[i + 1][j - 1] == KING + (2 - Plnow))

&& desk[i + k + 1][j - k - 1])

{

break;

}

}

}

}

}

if (i > 1) {//Не выйдем ли за нижний край доски?

if (j < 6) {//Не выйдем ли за правый край доски?

for (int k = 0; i - k > 0, j + k < 7; k++) {

if ((desk[i - 1][j + 1] == CHECKER + (2 - Plnow)

|| desk[i - 1][j + 1] == KING + (2 - Plnow))

&& !desk[i - k - 1][j + k + 1])

{//Если есть враг и пустая клетка за ним, возврат 1, т. к. можно бить.

return 1;

}

else if ((desk[i - 1][j + 1] == CHECKER + (2 - Plnow)

|| desk[i - 1][j + 1] == KING + (2 - Plnow))

&& desk[i - k - 1][j + k + 1])

{

break;

}

}

}

if (j > 1) {//Не выйдем ли за левый край доски?

for (int k = 0; i - k > 0, j - k > 0; k++) {

if ((desk[i - 1][j - 1] == CHECKER + (2 - Plnow)

|| desk[i - 1][j - 1] == KING + (2 - Plnow))

&& !desk[i - k - 1][j - k - 1])

{//Если есть враг и пустая клетка за ним, возврат 1, т. к. можно бить.

return 1;

}

else if ((desk[i - 1][j - 1] == CHECKER + (2 - Plnow)

|| desk[i - 1][j - 1] == KING + (2 - Plnow))

&& desk[i - k - 1][j - k - 1])

{

break;

}

}

}

}

return 0;

}

У формі програми використовується тільки обробники натискань на кнопки.

Обробник натискання кнопки «Выход» викликає функцію Application::Exit(); що завершує роботу програми.

//ВЫХОД

System::Void Checkers::MyForm::Exit\_Click(System::Object^ sender, System::EventArgs^ e)

{

Application::Exit();

return System::Void();

}

Обробник натискання кнопки «Сдаться» виводить повідомлення про те, що інший гравець переміг та перезапускає програму.

//СДАТЬСЯ

System::Void Checkers::MyForm::GiveUp\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (game.GetPlayer() == BLACK) MessageBox::Show("ПОБЕДА БЕЛЫХ!", "Черные сдались!");

else MessageBox::Show("ПОБЕДА ЧЕРНЫХ!", "Белые сдались!");

Application::Restart();

return System::Void();

}

Обробник натискання кнопки «Сдаться» виводить повідомлення про те, що гравець, який ходив у даний час запропонував нічию та перезапускає програму.

//НИЧЬЯ

System::Void Checkers::MyForm::Standoff\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (game.GetPlayer() == BLACK) MessageBox::Show("Черные предложили ничью.", "Ничья");

else MessageBox::Show("Белые предложили ничью.", "Ничья");

Application::Restart();

return System::Void();

}

Обробник тискання кнопки «Правила» виводить повідомлення з правилами гри (правила Російських Шашок).

//ПРАВИЛА

System::Void Checkers::MyForm::Rules\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Правила русских шашек:\n"

"Шашки ходят только по клеткам тёмного цвета.\n"

"Простая шашка ходит по диагонали вперёд на одну клетку.\n"

"Простая шашка бьёт вперёд и назад, дамка ходит и бьёт на любое поле диагонали.\n"

"Дамка бьёт по диагонали, как вперёд, так и назад, и становится на любое свободное поле после побитой шашки.\n"

"Дамка может бить несколько фигур соперника и должна бить до тех пор, пока это возможно.\n"

"Взятие обязательно. Если есть возможность продолжить взятие других шашек соперника, то это взятие продолжается,\n"

"пока бьющая шашка не достигнет положения, из которого бой невозможен.\n"

"Во время боя простая шашка может превратиться в дамку и сразу продолжить бой по правилам дамки.\n"

"При наличии нескольких вариантов боя можно выбрать любой из них.\n", "Правила");

return System::Void();

}

Обробник натискання кнопки на дошці однаковий для кожної клітинки, окрім різних даних у тих самих діях, тому буде опишемо алгоритм тільки 1 кнопки на прикладі кнопки B8. Також верхні та нижні клітинки перевіряють, чи треба змінювати текстуру пішака на текстуру дами.

Опис обробника натискання кнопки B8:

System::Void Checkers::MyForm::B8\_Click(System::Object^ sender, System::EventArgs^ e) {

Спочатку викликається метод ініціалізації координат. Туди передається статус. Спочатку він дорівнює 0, що означає – вказати координати першої клітинки, тобто тієї, з якої гравець ходить. Якщо у метод SetXY() передається 0, то передані координати (різні для кожної клітинки) записуються у змінні перших координат, а назад повертається значення статусу -2. Якщо ж у метод SetXY() передається статус -2, то передані координати записуються як координати другої клітки (тобто тої, на яку гравець хоче піти своєю фігурою) та маючи координати обидвох клітинок викликається метод go(). Значення статусу повертається такий, який отримується у результаті методу go().

status = game.SetXY(1, 7, status);

Після того, як ми отримали новий статус, перевіряється, що це за статус.

У результаті методу go() може вийти значення -1 – виникла помилка, метод go() викликав MessageBox з поясненням помилки, нічого не відбулось, тому статус просто становиться у 0 і цикл з 2 натисканнями закінчився.

if (status == -1) {

status = 0;

}

Якщо ми натиснули на кнопку перший раз, отримаємо статус -2. Програмі потрібно запам’ятати, яка кнопка була натиснута першою, щоб мати змогу змінити у ній текстуру після 2 натискання. Для цього існує невидима кнопка buf, якій програма прирівнює першу кнопку. Ця ділянка коду є однаковою для усіх кнопок за виключенням назви кнопки у коді. Воно співпадає з назвою натиснутої кнопки.

else if (status == -2) {

buf = B8;

}

Якщо ж у результаті методу go() був получений статус 1, відбувся звичайний хід, а тому текстура з першої кнопки, прирівняної до buf, переміщується у кнопку, на яку натиснули останньою (та відбувається перевірка на перетворення у даму), а першій кнопці приписується однотонна текстура пустоти «Void.png». статус становиться у положення 0 і цикл з 2 натиснень завершиться. Ця ділянка коду є однаковою для усіх кнопок за виключенням назви кнопки у коді. Воно співпадає з назвою натиснутої кнопки.

else if (status == 1) {

if ( game.Getdesk(1, 7) == KING + WHITE) {

B8->BackgroundImage = B8->BackgroundImage->FromFile(WKINGTEX);

buf->BackgroundImage = buf->BackgroundImage->FromFile(VOIDTEX);

}

else {

B8->BackgroundImage = buf->BackgroundImage;

buf->BackgroundImage = buf->BackgroundImage->FromFile(VOIDTEX);

}

status = 0;

}

Якщо у результаті методу go() отримуємо 2, то відбувся удар. Усе як і з звичайним ходом, але клітці, у якій була побита фігура приписується текстура пустоти. Цю кнопку запам’ятати неможливо, тому треба знайти її. Для цього при ударі в елементі класу ch записуються координати побитої фігури. Через перевірку, які координати у побитої фигури, дізнаємось, текстуру якої кнопки требя міняти на текстуру пустоти. Ця ділянка коду є абсолютно однаковою для усіх кнопок.

//ПРОИСХОДИТ УДАР, ОЧЕНЬ МНОГО УСЛОВИЙ ДЛЯ ЗАМЕНЫ ТЕКСТУРЫ ПОБИТОЙ ШАШКИ

else {

status = 0;

if (game.Getdesk(1, 7) == CHECKER + WHITE || game.Getdesk(1, 7) == KING + WHITE) {

B8->BackgroundImage = B8->BackgroundImage->FromFile(WKINGTEX);

buf->BackgroundImage = buf->BackgroundImage->FromFile(VOIDTEX);

}

else {

B8->BackgroundImage = buf->BackgroundImage;

buf->BackgroundImage = buf->BackgroundImage->FromFile(VOIDTEX);

}

if (game.GetBY() == 0) {

if (game.GetBX() == 0)A1->BackgroundImage = A1->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 2)C1->BackgroundImage = C1->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 4)E1->BackgroundImage = E1->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 6)G1->BackgroundImage = G1->BackgroundImage->FromFile(VOIDTEX);

}

else if (game.GetBY() == 1) {

if (game.GetBX() == 1)B2->BackgroundImage = B2->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 3)D2->BackgroundImage = D2->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 5)F2->BackgroundImage = F2->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 7)H2->BackgroundImage = H2->BackgroundImage->FromFile(VOIDTEX);

}

else if (game.GetBY() == 2) {

if (game.GetBX() == 0)A3->BackgroundImage = A3->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 2)C3->BackgroundImage = C3->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 4)E3->BackgroundImage = E3->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 6)G3->BackgroundImage = G3->BackgroundImage->FromFile(VOIDTEX);

}

else if (game.GetBY() == 3) {

if (game.GetBX() == 1)B4->BackgroundImage = B4->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 3)D4->BackgroundImage = D4->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 5)F4->BackgroundImage = F4->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 7)H4->BackgroundImage = H4->BackgroundImage->FromFile(VOIDTEX);

}

else if (game.GetBY() == 4) {

if (game.GetBX() == 0)A5->BackgroundImage = A5->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 2)C5->BackgroundImage = C5->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 4)E5->BackgroundImage = E5->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 6)G5->BackgroundImage = G5->BackgroundImage->FromFile(VOIDTEX);

}

else if (game.GetBY() == 5) {

if (game.GetBX() == 1)B6->BackgroundImage = B6->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 3)D6->BackgroundImage = D6->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 5)F6->BackgroundImage = F6->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 7)H6->BackgroundImage = H6->BackgroundImage->FromFile(VOIDTEX);

}

else if (game.GetBY() == 6) {

if (game.GetBX() == 0)A7->BackgroundImage = A7->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 2)C7->BackgroundImage = C7->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 4)E7->BackgroundImage = E7->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 6)G7->BackgroundImage = G7->BackgroundImage->FromFile(VOIDTEX);

}

else {

if (game.GetBX() == 1)G1->BackgroundImage = B8->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 3)G1->BackgroundImage = D8->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 5)G1->BackgroundImage = F8->BackgroundImage->FromFile(VOIDTEX);

if (game.GetBX() == 7)G1->BackgroundImage = H8->BackgroundImage->FromFile(VOIDTEX);

}

}

Після цього – завершення методу.

return System::Void();

}

# Текст програми з коментарями

У склад проекту входять файли «MyForm.h», «MyForm.cpp», «Ch.h», «Ch.cpp», «func.h».

Файл «func.h» містить потрібні бібліотеки та директиви препроцесора для роботи класу Сh.h та спрощення написання та розуміння коду.

Файл «func.h»:

#pragma once

#include <corecrt\_math.h> //Для работы функции abs()(модуль).

#define K 8 //Размер доски

#define NONE 0 //Индекс пустой клетки

#define WHITE 0// Индекс белого игрока

#define W 1 // Индекс белого цвета

#define BLACK 2// Индекс черного игрока

#define B -1// Индекс черного игрока

#define CHECKER 1 //Индекс шашки

#define KING 2 //Индекс дамки

#define FP desk[FY][FX] //Содержимое начальной клетки.

#define SP desk[SY] [SX] //Содержимое конечной клетки.

#define FX firstX //Координаты по X начальной клетки.

#define FY firstY //Координаты по Y начальной клетки.

#define SX secondX //Координаты по X конечной клетки.

#define SY secondY //Координаты по Y конечной клетки.

Файл «Ch.h» містить усі дані та прототипи методів класу «ch», який є реалізацією усіх внутрішніх алгоритмів.

Файл «Ch.h»:

#pragma once

#include "func.h"

class ch {

Змінні класу:

private:

Масив, де зберігаються положення клітинок на дошці (0 – пуста клітинка, 1 або 3 – білий або чорний пішак, 2 або 4 – біла або чорна дама).

short\*\* desk;

Змінні гравців (білий – col = 1, Plnow = 0; чорний – col = -1, Plnow = 2;)

short col;

unsigned short Plnow;

Координати клітинки, з якої гравець хоче зробити хід

unsigned short firstX;

unsigned short firstY;

Координати клітинки, на яку гравець хоче зробити хід

unsigned short secondX;

unsigned short secondY;

Координати клітинки, що була побита останнім ударом.

int beatX;

int beatY;

public:

ch();

~ch();

//Основная функция, с которой начинаются проверки.

short go();

//Ввод данных

short SetXY(unsigned short x, unsigned short y,short st);

// Общие функции

void refresh();//Заново

bool anycan();//проверка на возможность побить/походить любой фигурой

void convert();//Функция перевода шашек в дамки по достижению ими соотв. полей

//Геттеры и Сеттеры.

short Getdesk(short x, short y) { return desk[y][x]; }//содержимое 1 ячейки на доске.

short Getcol() { return col; }

unsigned short GetPlayer() { return Plnow; }// Узнать какой игрок сейчас ходит.

//Узнать первые и вторые X и Y

unsigned short GetFX() { return firstX; }

unsigned short GetFY() { return firstY; }

unsigned short GetSX() { return secondX; }

unsigned short GetSY() { return secondY; }

//Узнать координаты побитой фигуры.

int GetBX() { return beatX; }

int GetBY() { return beatY; }

private:

void move();//Ход фигурой

//Проверки для пешки

short checker();

void checkerhit();//Удар пешкой

//проверки для дамки

short king();

void kinghit(short i, short j);//Удар дамкой

//Задать первые и вторые X и Y

void SetFX(unsigned short x) { firstX = x; }

void SetFY(unsigned short y) { firstY = y; }

void SetSX(unsigned short x) { secondX = x; }

void SetSY(unsigned short y) { secondY = y; }

//Задать координаты побитой фигуры.

void SetBX(unsigned short x) { beatX = x; }

void SetBY(unsigned short y) { beatY = y; }

//Проверки на возможность удара

bool canhit();//Для всей доски

bool CanHitChecker(short i, short j);//Для одной шашки

bool CanHitKing(short i, short j);//Для одной дамки

};

Файл «Ch.cpp» містить реалізації методів класу «ch».

Файл «Ch.cpp»:

Підключення "Ch.h" де описано клас ch.

#include "Ch.h"

Щоб застосувати MessageBox треба узяти простір імен System та System::Windows::Forms.

using namespace System;

using namespace System::Windows::Forms;

Далі – реалізація методів класу ch, які було описано вище.

Файл «MyForm.cpp»

У файлі «MyForm.cpp» спочатку підключаються файли «MyForm.h» та «Ch.h», а також об’являються простори імен.

#include "MyForm.h"

#include "Ch.h"

using namespace System;

using namespace System::Windows::Forms;

У файлі «MyForm.cpp» містяться методи класу MyForm, а також функція main(), з якої починається уся програма:

[STAThreadAttribute]// Точка входа

void main(void) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

//Создание формы и запуск приложения.

Checkers::MyForm form;

Application::Run(% form);

}

Також у формі об’являються загальні дані:

//Данные

Об’єкт класу ch, з яким ми будемо працювати

ch game;

Змінна status, яка передається у методи класу ch та назад, вказуючи що робити після виклику методів класу ch.

short status = 0;

Далі – реалізація обробників кнопок, яка вже була описана вище.

Файл «MyForm.h»

Спочатку у файлі об’являються диретиви препроцесора, функція яких – скорочення для більш швидкого написання коду.

#define WCHECKERTEX "WChecker.png"//текстура белой шашки

#define BCHECKERTEX "BChecker.png"//текстура черной шашки

#define WKINGTEX "WKing.png"//текстура белой дамки

#define BKINGTEX "BKing.png"//текстура черной дамки

#define VOIDTEX "Void.png"//текстура пустоты

#define POS game.Getdesk//Сокращение метода получения значения клетки на доске из элемента класса ch.

Потім описується простір імен проекту

namespace Checkers {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

А в ньому – сам клас MyForm.

public ref class MyForm : public System::Windows::Forms::Form

{

Далі містяться дані про усі елементи форми(Тобто описується ), а також прототипи обробників натискань на кнопкки, які було приведено вище.

1. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

# Інтерфейс користувача

Інтерфейс користувача – це кнопки на дошці та кнопки на панелі справа. Користувач може натискати на будь-яку кнопку у будь-який момент.

# Інструкція користувача

Програма призначена для користування 2 гравцями, тому що у шашках грають лише удвох. Після запуску програми виникне вікно, яке має 4 кнопки справа та дошку. Одразу ж після запуску можна починати гру та робити перший хід.

Для того, щоб зробити хід натисніть 1 раз на клітинку, з якої ви хочете пересунути фігуру а потім натисніть на клітинку, на яку ви хочете зробити хід. Якщо зробити бажаний хід неможливо, програма видасть вікно з повідомленням, після його закриття гра продовжується.

Щоб дізнатись правила гри, натисніть кнопку «Правила».

Щоб сдатись, натисніть кнопку «Сдаться».

Щоб об’явити нічию, натисніть кнопку «Ничья».

Щоб вийти з гри та закрити програму, натисніть кнопку «Выход» або хрестик у верхньому правому углу вікна.

ВИСНОВОК

У результаті курсової роботи вдалось зробити програму – комп’ютерну гру «Шашки» з мінімальним функціоналом. Вона має мінімальний функціонал, а тому дуже далека від ідеалу. Її можна використовувати як розвагу або тренажер уваги та логічного мислення. Результат роботи не має значного наукового або соціального значення в цілому, але може бути використана як приклад при вивченні Windows Forms, або при розробці більш досконалої аналогічної програми.

СПИСОК ДЖЕРЕЛ

1. Руководство по программированию в Windows Forms [Електронний ресурс] – Режим доступу до ресурсу: <https://metanit.com/sharp/windowsforms/>.
2. Windows Forms overview [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/framework/winforms/windows-forms-overview>.
3. Введение в Windows Forms — пишем первую программу [Електронний ресурс] – Режим доступу до ресурсу: <https://code-live.ru/post/first-windows-form/>.
4. Control.Click Event [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.click?view=netframework-4.7.2>
5. Элемент управления Label (Windows Forms) [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/controls/label-control-windows-forms>