Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное   
учреждение высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, механики и математики

**Отчет по лабораторной работе**

**ИССЛЕДОВАНИЕ РАЗЛИЧНЫХ ИГРОВЫХ АЛГОРИТМОВ ПОИСКА НАИЛУЧШЕГО ХОДА НА ПРИМЕРЕ ИГРЫ «РУССКИЕ ШАШКИ»**

**Выполнили**:студент группы 0823-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Грачева Е. А.

студент группы 0826-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Панов А. А.

Подпись

**Проверил**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород

2017

**Содержание**

Оглавление

[Введение 3](#_Toc482020398)

[Постановказадачи 4](#_Toc482020399)

[Алгоритмы 5](#_Toc482020400)

[Алгоритм полного перебора 5](#_Toc482020401)

[Alpha-Beta с отсечениями 5](#_Toc482020402)

[Форсирование 5](#_Toc482020403)

[Оценочная функция 5](#_Toc482020404)

[Реализация 6](#_Toc482020405)

[Алгоритмическое ядро (С++) 7](#_Toc482020406)

[Графический интерфейс (C#) 9](#_Toc482020407)

[Апробация 10](#_Toc482020408)

[Заключение 11](#_Toc482020409)

[Литература 12](#_Toc482020410)

[Приложение. Фрагменты исходного кода программы 13](#_Toc482020411)

[template<class Data> class Stack 13](#_Toc482020412)

# Введение

В настоящее время в программировании популярно направление, связанное с компьютеризацией различных логических игр. Мотивируется это тем, что, во-первых, компьютерные игры достаточно востребованы; во-вторых, логические игры представляют собой хороший материал для исследования различных алгоритмов поиска наилучшего хода.

# Постановказадачи

Цели работы: исследовать различные алгоритмы поиска лучшего хода в игровой программе, реализовать и повозможности оптимизировать их, а также выбрать наиболее эффективные. В качестве разрабатываемой игровой программы были выбраны “русские шашки”, т.к. игра имеет сравнительно простую логику, но при этом является хорошим примером, иллюстрирующим различные алгоритмы поиска оптимального хода. Сравнение с известными программами, такими как Тундра или Аврора, позволит оценить эффективность наших алгоритмов.

# Алгоритмы

Шашки – игра с конечным числом состояний. Это означает, что, обладая бесконечными вычислительными ресурсами, мы смогли бы найти решение этой игры – то есть такую стратегию следуя которой всегда можно было бы выиграть или по крайней мере сыграть вничью. Каждая позицияpos в шашках это: либо победа для белых, либо победа для черных, либо ничья. Мы можем обозначить это с помощью функции***f(pos)***Если бы у нас был бесконечно быстрый компьютер, мы могли бы вычислить ее следующим образом:

1. Присвоим всем финальным позициям значения –1, 0, 1, в зависимости от исхода игры.
2. Применим рекурсивное правило

f(pos) = max(pos→pos‘) – f(pos‘),где pos→pos‘ обозначает все допустимые ходы из позиции pos. Знак минус используется потому, что игроки делают ходы поочередно, то есть, если позиция p соответствует ходу белых, тогда позиция pos‘ соответствует ходу черных.

Количество [возможных позиций](http://en.wikipedia.org/wiki/Shannon_number) в шашках составляет около 5\*1020. Вычисления такого масштаба выполнить практически невозможно (канадским ученым потребовалось 200 компьютеров и 20 лет вычислений). Следовательно, находить наилучший ход нужно другим способом.

Введем следующее понятие:*оценочная функция* – эточисловая функцияопределенная на множестве позиций и принимающая числовое значение, характеризующее данную позицию. Чем больше это значение, тем выгодней эта позиция.

### Алгоритм полного перебора

Пусть дано некоторое начальное состояние игры:

Сгенерируем все возможные ходы:

### Alpha-Beta с отсечениями

### Форсирование

### Оценочная функция

# Реализация

Проект состоит из двух частей — вычислительного ядра и графического интерфейса. В вычислительной части представлены различные функции поиска лучшего хода, то есть искусственный интеллект для бота. Графическая оболочка предоставляет визуализацию игры, предоставляет настройки бота и дает возможность ходить реальному игроку.

Вычислительное ядро

Графическая оболочка

позиция

лучший ход

Нахождение наилучшего хода требует большого объема вычислений и для обеспечения наилучшей производительности вычислительное ядро написано в виде динамической библиотеки на C++.

Графическая оболочка, написана на C#, так как данный язык предоставляет удобный графический инструментарий.

Итоговое приложение отображает игру с помощью графической оболочки и находит лучший ход с помощью вычислительного ядра.

## Алгоритмическое ядро (С++)

## Графический интерфейс (C#)

Для удобства графическая часть выполнена по схеме MVC (Model, View, Controller). Согласно этой схеме данные разделяются на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо. Также изменение и отображение данных происходит строго по заданной схеме:

* ***Модель*** (*Model*) предоставляет собой данные и реагирует на команды контроллера, изменяя своё состояние;
* ***Представление*** (*View*) отвечает за отображение данных модели пользователю, реагируя на изменения модели;
* ***Контроллер*** (*Controller*) интерпретирует действия пользователя, оповещая модель о необходимости изменений;

View

Controller

Model

I’m update!

user’s change

change date

Модель в проекте представлена классом Game\_model.В нем

***Диаграмма классов:***

# Апробация

# Заключение

# Литература

* Гергель В. П., Лабутина А. А. Учебно-образовательный комплекс по методам программирования //Нижний Новгород: ННГУ им. НИ Лобачевского. – 2007. ☺☺☺ *//да это надо на первое место))))*

# Приложение. Фрагменты исходного кода программы

## template<class Data> class Stack

Stack(int\_size)//конструкторинициализации

{

if (\_size>= 0)

top = -1;

elsethrowNEGATIVE\_INDEX;

size = \_size;

ar = newData[size];

}/\*-------------------------------------------------------------------------\*/

void push(Datanum)

{

if (top + 1 < size)

ar[++top] = num;

elsethrowSTACK\_FULL;

}/\*-------------------------------------------------------------------------\*/

voidquickPush(Datanum)

{

ar[++top] = num;

}/\*-------------------------------------------------------------------------\*/

Datapop()

{

if (top > 0)

return ar[top--];

elsethrowSTACK\_EMPTY;

}/\*-------------------------------------------------------------------------\*/

DataquickPop()

{

return ar[top--];

}/\*-------------------------------------------------------------------------\*/

Data&showTop()

{

return ar[top];

}/\*-------------------------------------------------------------------------\*/

DatagetElem(inti)

{

return ar[i];

}/\*-------------------------------------------------------------------------\*/

boolisEmty()

{

return (top == -1);

}/\*-------------------------------------------------------------------------\*/

boolisFull()

{

return (top + 1 == size);

}/\*-------------------------------------------------------------------------\*/

intgetNumTop()

{

return top;

}/\*-------------------------------------------------------------------------\*/

voidsetTop(int\_top)

{

top = \_top;

}/\*-------------------------------------------------------------------------\*/

~Stack()//деструктор

{

if (ar != 0)

delete[] ar;

}/\*-------------------------------------------------------------------------\*/