|  |  |  |
| --- | --- | --- |
| `http://nowinki.mech.pk.edu.pl/img/uploaded/1435565202arilo90.jpg | Politechnika Krakowska im. Tadeusza Kościuszki  Wydział Fizyki,  Matematyki  i Informatyki | https://upload.wikimedia.org/wikipedia/commons/thumb/d/d4/Logo_Wydzia%C5%82u_Fizyki,_Matematyki_i_Informatyki_PK.jpg/240px-Logo_Wydzia%C5%82u_Fizyki,_Matematyki_i_Informatyki_PK.jpg |
| **Programowanie równoległe i rozproszone** | | |
| Równoległe działanie silnika do gry w szachy za pomocą MPI w jezyku C# | | |
| **Data oddania:**  17.06.2018 | **Kierunek:** Informatyka Stosowana  **Rok akademicki:** 2017/2018 | **Wykonali:**  Zdobysław Antas, Michał Kisielewski  **Nr albumu:** 124602, 108673 |

Spis treści

[1. Cel i opis projektu 3](#_Toc516861233)

[2. Zastosowane technologie 3](#_Toc516861234)

[2.1 MPI.NET 3](#_Toc516861235)

[2.2 Ten JSON 3](#_Toc516861236)

[3. Interfejs użytkownika 3](#_Toc516861237)

[4. Wyniki 5](#_Toc516861238)

[5. Podsumowanie 8](#_Toc516861239)

# Cel i opis projektu

Celem projektu jest zrównoleglenie silnika do gry w szachy za pomocą standardu MPI w języku C#. Użyty w projekcie silnik do gry w szachy to Chess Core autorstwa Adam’a Berent’a zaimplementowany oryginalnie w technologii .NET Core. W projekcie użyta została implementacja standardu MPI o nazwie MPI.NET.

# Zastosowane technologie

Projekt został wykonany w środowisku Microsoft Visual Studio 2017 w języku C#. Użyte technologie w projekcie to .NET Framework w wersji 4.6.2, MPI.NET w wersji 1.3.0, oraz ???.

## MPI.NET

MPI.NET to implementacja standardu MPI w technologii .NET. Wywodzi się z Indiana University w USA. Implementacja ta pozwala za opracowywanie i uruchamianie programów ze standardem MPI w technologii .NET w C#. Obecnie projekt nie jest już utrzymywany. MPI.NET można zainstalować poprzez np. menadżer pakietów NuGet.

## Ten JSON

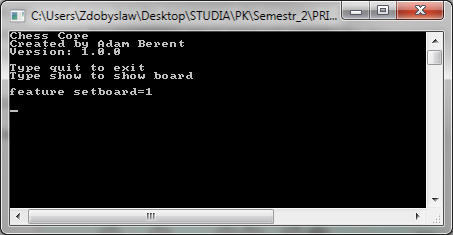
YY

# 2. Opis algorytmu

Algorytm w programie …

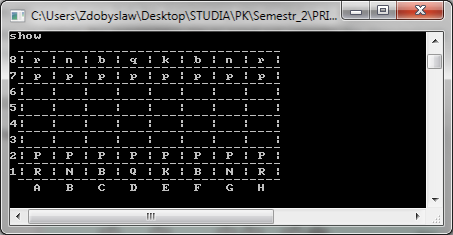
# Interfejs użytkownika

Interfejs dostępny użytkownikowi jest w pełni tekstowy. Poniżej zaprezentowane jest menu startowe silnika do gry Chess Core.



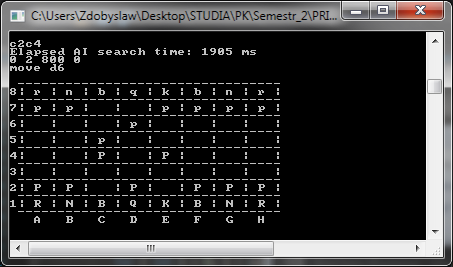
Rys 1. Menu startowe gry w szachy.

Użytkownik po starcie programu ma do dyspozycji 2 komendy – *show* oraz *quit*. Komenda *quit* pozwala na wyjście z programu. Użycie komendy *show* powoduje wyświetlenie planszy. Poniżej znajduje się zrzut ekranu prezentujący ekran startowy programu.



Rys 2. Plansza startowa do gry w szachy.

Na przedstawionej na Rys 2 planszy Wielkimi literami oznaczone są pionki gracza, a małymi literami oznaczone są pionki przeciwnika (algorytmu). W celu wykonania ruchu użytkownik musi podać w konsoli ruch w formacie [a-hA-H][1-8][a-hA-H][1-8], gdzie pierwsza litera oraz cyfra odpowiada współrzędnej figury, którą wykonuje się ruch, a druga cyfra oraz litera odpowiada polu, na które ma się przesunąć wybrana figura. Jeśli przy określaniu ruchu został użyty zrównoleglony algorytm dodatkowo wyświetli się komunikat w postaci *Elapsed AI serach time: <czas szukania ruchu> MS*, który informuje jak długo zajęło algorytmowi wyszukanie rozwiązania. Poniżej została przedstawiona plansza gry po wykonaniu ruchu przez użytkownika oraz algorytm.



Rys 3. Plansza gry po wykonaniu ruchu przez użytkownika oraz algorytm.

# Wyniki

Wyniki ilustrujące przyspieszenie działania zrównoleglonego algorytmu zostały zebrane uruchamiając program na komputerze z 4 rdzeniami fizycznymi. Poniżej zostały przedstawione wyniki dotyczące czasu oraz przyspieszenia działania algorytmu wyszukującego optymalny ruch przeciwnika w zależności od ilości procesów.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **czas [ms]** | | | | |
| **liczba procesów** | **ruch f1c4** | **ruch d2d3** | **ruch g1f3** | **ruch c1e3** | **średnia** |
| 1 | 1041 | 1322 | 1374 | 1245 | 1245.5 |
| 2 | 677 | 840 | 783 | 868 | 792 |
| 3 | 463 | 731 | 601 | 660 | 613.75 |
| 4 | 420 | 516 | 539 | 580 | 513.75 |
| 5 | 439 | 540 | 582 | 645 | 551.5 |
| 6 | 464 | 612 | 612 | 648 | 584 |

Rys 4. Tabela przedstawiająca czas szukania ruchu przeciwnika w zależności od ilości procesów dla kolejnych ruchów gracza.

Rys 5. Wykres przedstawiający czas szukania ruchu przeciwnika w zależności od ilości procesów dla kolejnych ruchów gracza.

Rys 6. Wykres przedstawiający średni czas szukania ruchu przeciwnika w zależności od ilości procesów.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **przyspieszenie [ms]** | | | | |
| **liczba procesów** | **ruch f1c4** | **ruch d2d3** | **ruch g1f3** | **ruch c1e3** | **średnia** |
| 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2 | 1.54 | 1.57 | 1.75 | 1.43 | 1.58 |
| 3 | 2.25 | 1.81 | 2.29 | 1.89 | 2.06 |
| 4 | 2.48 | 2.56 | 2.55 | 2.15 | 2.43 |
| 5 | 2.37 | 2.45 | 2.36 | 1.93 | 2.28 |
| 6 | 2.24 | 2.16 | 2.25 | 1.92 | 2.14 |

Rys 7. Tabela przedstawiająca przyspieszenie szukania ruchu przeciwnika w zależności od ilości procesów dla kolejnych ruchów gracza.

Rys 8. Wykres przedstawiający przyspieszenie obliczeń ruchu przeciwnika w zależności od ilości procesów dla kolejnych ruchów gracza.

Rys 9. Wykres przedstawiający średnie przyspieszenie szukania ruchu przeciwnika w zależności od ilości procesów.

# Podsumowanie

Udało się w projekcie zrealizować zrównoleglenie algorytmu wyszukującego optymalny ruch przeciwnika w grze w szachy. Jednym z największych problemów pojawiających się w projekcie była sama komunikacja międzyprocesowa. Używane algorytmy w silnikach do gry w szachy okazałe się trudniejsze do zrównoleglenia niż algorytmu zrównoleglone w ramach zajęć.