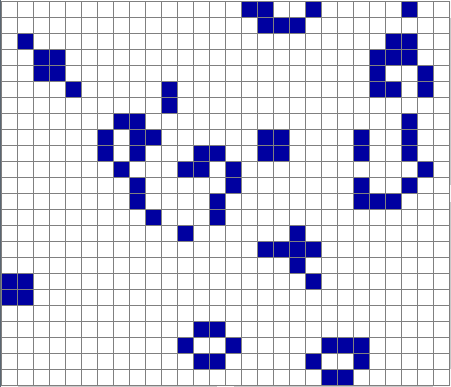
**Dokumentacja projektowa**

**Współbieżna, rozproszona gra life w erlangu**



**Autorzy:**

**Małgorzata Maciurzyńska**

**Rafał Płonka**

**Konrad Seweryn**

**Mateusz Stanaszek**

**Mateusz Ścirka**

# Wprowadzenie teoretyczne

## Cel projektu i wymagania

Celem projektu bylo opracowanie architektury rozproszonego, skalowalnego systemu w erlangu dla gry Life w/g podstawowej reguły Conwaya 23/3.

**Rozmiar planszy** jest kwadratowy będący potęgą 2 począwszy od 256×256 do 16384×16384 (rozmiar 8-14)

Program wykorzystuje **rozproszenie**.

Program uwzględnia, że nie wszystkie węzły będą zawsze dostępne i nie będą znikać w trakcie obliczeń.

Program posiada możliwość **generowania losowych plansz**.

Program posiada wbudowany **benchmark**.[[1]](#footnote-1) (funkcję **test\_time/1** wykonująca **podaną ilość iteracji funkcji** next/0)

Program posiada funkcję **„next/0” , która wylicza następną iterację.**

Program ma możliwość **wczytania planszy z pliku i zapisu do pliku**.[[2]](#footnote-2)

## Reguły gry według Conwaya

Martwa komórka, która ma dokładnie 3 żywych sąsiadów, staje się żywa w następnej jednostce czasu (rodzi się)

Żywa komórka z 2 albo 3 żywymi sąsiadami pozostaje nadal żywa; przy innej liczbie sąsiadów umiera (z "samotności" albo "zatłoczenia").

# Wykorzystywane algorytm

## Dzielenie tablicy na mniejsze

Rozważana jest tablica 2^3 żeby pokazać w sposób przejrzysty działanie algorytmu.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 0 | 1 | 1 | **1** | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **1** | 0 | 1 | 1 | **1** | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Jak widać tablica jest podzielone na 4 mniejsze podtablice, które zostaną wysłane później do policzenia następnej iteracji. Gdyby tablice zostały wysłane w ten sposób:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 1 | 0 | 1 | **1** | | 0 | 1 | 1 | **1** | | 1 | 1 | 0 | **0** | | 0 | 0 | 0 | **1** | | **1** | **1** | **1** | **1** | | |  |  |  |  | | --- | --- | --- | --- | | **1** | 1 | 1 | 1 | | **0** | 1 | 1 | 1 | | **1** | 1 | 0 | 0 | | **0** | 0 | 0 | 1 | | **1** | **1** | **1** | **1** | | |  |  |  |  | | --- | --- | --- | --- | | **1** | **0** | **1** | **1** | | 0 | 1 | 1 | **1** | | 1 | 1 | 0 | **0** | | 0 | 0 | 0 | **1** | | 1 | 1 | 1 | **1** | | |  |  |  |  | | --- | --- | --- | --- | | **1** | **1** | **1** | **1** | | **1** | 1 | 1 | 1 | | **1** | 1 | 0 | 0 | | **0** | 0 | 0 | 1 | | **1** | 1 | 1 | 1 | |

pogrubione kolumny i wiersze nie mogłyby zostać poprawnie obliczone. Dlatego też musi zostać dodany dodatkowy wiersz i kolumna dla wszystkich miejsc złączeń podtablic.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 0 | 1 | 1 | 1 | | 0 | 1 | 1 | 1 | 0 | | 1 | 1 | 0 | 0 | 1 | | 0 | 0 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | 1 | | 1 | 0 | 1 | 1 | 1 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 1 | 1 | 1 | 1 | | 1 | 0 | 1 | 1 | 1 | | 0 | 1 | 1 | 0 | 0 | | 1 | 0 | 0 | 0 | 1 | | 1 | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 1 | 1 | 1 | 1 | | 1 | 0 | 1 | 1 | 1 | | 0 | 1 | 1 | 1 | 1 | | 1 | 1 | 0 | 0 | 1 | | 0 | 0 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | 1 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | | 0 | 1 | 1 | 0 | 0 | | 1 | 0 | 0 | 0 | 1 | | 1 | 1 | 1 | 1 | 1 | |

Są to minimalne wielkości tablic jakie muszą zostać przesłane do dalszych obliczeń. Wraz z tablicami przesyłana jest **informacja o miejscu wklejenia nowo policzonej tablicy (dwie zmienne – x oraz y).**

## Algorytm liczenia podtablicy

Otrzymany fragment tablicy wraz z informacją czy i gdzie należy dodać wiersz/kolumnę zerową:

0000 – kolejne bity oznaczają odpowiednio górny wiersz, prawą kolumnę, dolny wiersz, lewą kolumnę.

W tym przypadku byłoby to zatem 1001.

Algorytm dodaje kolumnę, wiersz zerową w odpowiednie miejsca i rozpoczyna iterację po wszystkich komórkach które nie leżą na krawędzi, a dzięki dodatkowym krawędziom wartości odpowiednich komórek zostaną policzone w sposób prawidłowy.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |

Algorytm zwraca już tylko odpowiedni fragment bez zbytecznych kolumn i wierszy, które to zostaną wysłane do nadzorcy sklejającego tablicę oraz wraz z **informacją o miejscu wklejenia nowo policzonej tablicy**:

|  |  |  |  |
| --- | --- | --- | --- |
| N | N | N | N |
| N | N | N | N |
| N | N | N | N |
| N | N | N | N |
| N | N | N | N |

## Algorytm scalania podtablic

Otrzymany fragment tablicy wraz z informacją gdzie należy u tablicę zostaje następnie umieszczony w odpowiednie miejsce, aż nowa tablica zostaje całkowicie wypełniona.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |
| N | N | N | N | N | N | N | N |

Gdy to nastąpi, tablica albo zostaje ponownie policzona dla kolejnej iteracji albo są zwracane nowe wartości komórek, ponieważ żądana liczba iteracji osiągnęła żądaną wartość.

# Przebieg algorytm

## Warunki wstępne

Użytkownik ma możliwość:

* wygenerowania losowej planszy i zapisania do pliku
* wczytania planszy z pliku na podstawie jego nazwy
* wyświetlenia planszy z pliku (działa tylko dla małych tablic)
* rozpoczęcia symulacji (konkretna liczba iteracji) wraz ze zmierzeniem czasu iteracji

## Działanie

Zachodzą odpowiednie operacje:

* sprawdzenie liczby dostępnych węzłów i na tej podstawie wybranie nadzorcy oraz procesów odpowiedzialnych za liczenie części tablicy
* nadzorca dzieli planszę podzielenie na mniejsze fragmenty i wysyła je do procesów liczących
* nadzorca czeka aż wszystkie procesy przyślą wiadomość z obliczonym fragmentem tablicy i wszystkie fragmenty tablicy zostaną obliczone
* jeżeli ma wykonać kolejną iterację to wraca do punktu 2 w przeciwnym razie idzie dalej
* zwrócenie przez nadzorcę informacji do użytkownika, że zostały wykonane iteracje

## Warunki końcowe

Na koniec zostają wykonane następujące czynności:

* użytkownik zostaje poinformowany o czasie działania algorytmu
* zapisanie do pliku?

Spis Treści

[1. Wprowadzenie teoretyczne 2](#_Toc377089129)

[1.1. Cel projektu i wymagania 2](#_Toc377089130)

[1.2. Reguły gry według Conwaya 2](#_Toc377089131)

[2. Wykorzystywane algorytm 3](#_Toc377089132)

[2.1. Dzielenie tablicy na mniejsze 3](#_Toc377089133)

[2.2. Algorytm liczenia podtablicy 4](#_Toc377089134)

[2.3. Algorytm scalania podtablic 5](#_Toc377089135)

[3. Przebieg algorytm 6](#_Toc377089136)

[3.1. Warunki wstępne 6](#_Toc377089137)

[3.2. Działanie 6](#_Toc377089138)

[3.3. Warunki końcowe 6](#_Toc377089139)

1. <https://erlangcentral.org/wiki/index.php/Measuring_Function_Execution_Ti> [↑](#footnote-ref-1)
2. Plik jest skompresowanym ciągiem zawierającym rozmiar jako pierwszy bajt (2^X, np. 12 oznacza planszę 2^12 na 2^12) oraz wartości poszczególnych komórek (0 lub 1) wierszami [↑](#footnote-ref-2)