Agnieszka Janowicz

***Gra w życie***

***Dokumentacja***

**Zasady gry**

Istnieją dwie populacje – A i B. Celem gry jest osiągnięcie dominacji przez jedną z populacji.

Są trzy rodzaje osobników każdej populacji – dziecko, dorosły oraz starszy. Osobniki różnych rodzajów różnią się od siebie siłą.

Komórka w szachownicy może być w jednym z dwóch stanów: żywym lub martwym.

Zmiany w komórce są determinowane przez stan komórek sąsiadujących według poniższych zasad:

1. Jeżeli osobnik dowolnej populacji nie ma sąsiedztwie zadnej żywej komórki dowolnej populacji, umiera.
2. Jeśli osobnik populacji A (obojętnego rodzaju) znajduje się w sąsiedztwie x osobników A oraz y osobników B, gdzie y>x, to dany osobnik umiera, a żywe komórki populacji B starzeją się (dziecko dorasta i staje się dorosłym, dorosły staje się starszy; starszy pozostaje w tym samym wieku). (Dla osobnika populacji B analogicznie)
3. Jeśli osobnik populacji A (obojętnego rodzaju) znajduje się w sąsiedztwie x osobników A oraz y osobników B, gdzie x=y, o przewadze decyduje rodzaj osobników. Najsilniejsze są osobniki starsze, później dorosłe, a najslabsze są dzieci. Jeśli w sąsiedztwie danego osobnika jest przewaga starszych osobników innej populacji, sytuacja jest analogiczna do punktu drugiego. Jeżeli w sąsiedztwie znajduje się taka sama liczba osobników tego samego rodzaju, dany osobnik przeżywa, a komórki sąsiadujące nie zmieniają stanu ani rodzaju.
4. Jeżeli martwe pole jest w sąsiedztwie 4 lub więcej osobników populacji A oraz w sąsiedztwie nie ma osobników populacji B, ożywa jako dziecko populacji A (podobnie dla analogicznej sytuacji).
5. Martwe pole w sąsiedztwie mniej niż 4 osobników jednej populacji pozostaje martwe.
6. Martwe pole w sąsiedztwie 4 osobników populacji A i 4 osobników populacji B ożywa jako dziecko populacji, z której w sąsiedztwie było więcej silniejszych osobników. Jeśli była ich taka sama liczba, pozostaje martwe.
7. Jeśli osobnik populacji A jest w sąsiedztwie 4 lub więcej osobników ze swojej populacji, wsród których są starsi, oraz w sąsiedztwie nie ma osobników populacji B, umiera. Jeżeli zas wsród nich nie ma starszych, to osobnik ten pozostaje żywy i starzeje się.

Sterowanie:

* **strzałki** - sterowanie kursorem
* **ENTER** - zmiana stanu komorki
* **F1** - wygenerowanie losowej planszy
* **F2** - jedno przejscie automatu
* **F3** – autoiteracja – wykonywanie cyklów aż jedna z populacji osiągnie dominację
* **F10** - wyjscie

Program składa się z 4 plików nagłówkowych : osobniki.h, gra.h, gui.h, main.h i 4 odpowiadających im plików źródłowych.

1. **Osobniki.h, osobniki.cpp**

Zawierają: deklaracje i definicje klas: osobnik, istota, lSasiadow.

osobnik

istota

W programie wykorzystywane są dwie klasy dotyczące przechowywania informacji o stanie i typie komórki – **osobnik** i dziedzicząca od niej klasa **istota**. Klasa osobnik może być użyta do implementacji pierwotnej wersji gry w życie, zawierającej jedynie martwe lub żywe komórki (bez podziału na populacje i rodzaje osobników). Klasa istota rozbudowuje ją o dane dotyczące populacji i rodzaju, oraz o funkcje wykorzystywane do zmiany stanu komórki.

Stan żywy w programie jest oznaczany liczbą 1, a martwy – 0. Ponadto są 3 stany posrednie: stan oznaczany liczbą 2 oznacza zamianę ze stanu żywego na marwy, 3 – stan posredni oznaczający zamianę z martwej komórki na komórkę zawierającą osobnika populacji A, a 4 – stan oznaczający zamianę martwej komórki na osobnika populacji B.

Klasa **lSasiadow** zlicza sąsiadów komórki z uwzględnieniem typu populacji i rodzaju osobników. Posiada zmienne służące do przechowywania liczby sąsiadów każdego typu (int Adziecko, int Adorosly, int Astarszy, Bdziecko, Bdorosly, Bstarszy) oraz do ogólnej liczby osobników jednej populacji (int Aall, Ball). Ponadto posiada 8elementową tablicę wskaźników służącą do przechowywania wskaźników na żywe komórki.

**Class osobnik**

**int stan;** - przechowuje informację o stanie komórki; 0 oznacza martwą komórkę, 1 - żywą

**class istota**

int populacja; - informacja o typie populacji – 1 oznacza populację A, 2 – populację B

int rodz; informacja o rodzaju osobnika; 0 – dziecko, 1 – dorosły, 2 - starszy

void starzenie(); - funkcja wykorzystywana do zmiany rodzaju osobnika na starszy (silniejszy)

**class lSasiadow**

**int Adziecko; int Adorosly; int Astarszy; int Bdziecko; int Bdorosly; int Bstarszy; int Aall; int Ball; -** przechowują liczbę osobników sąsiadujących konkretnego typu

istota \*sasiad[8]; - tablica wskaźników przechowująca wskaźniki na żywe komórki sasiadujące (maksymalnie 8)

void licz(); - funkcja służąca do zliczania sąsiadów danej komórki

1. **Gra.h, gra.cpp**

Zawierają funkcje służące do implementacji zasad gry w życie. Funkcje zawarte są w klasie **szachownica**, która została stworzona do operacji przeprowadzanych na planszy.

**Class szachownica**

**int rozmiar;** - przechowuje rozmiar planszy

**void wyzeruj\_plansze(std::vector< std::vector <istota> >&);** - dezaktywuje wszystkie komórki na planszy

**void losowe( std::vector< std::vector <istota> >&);** - wypełnia dowolne miejsca osobnikami losowego rodzaju

**lSasiadow policz\_sasiadow(unsigned short, unsigned short, std::vector< std::vector <istota> > &);** - funkcja służąca do zliczania sąsiadów danej komórki planszy

**void zmien\_stan\_komorek(std::vector< std::vector <istota> >&);** - służy do miany stanu komórki w zależnosci od sąsiadujących pól, na podstawie zasad gry w życie

**void ruch(std::vector< std::vector <istota> > &plansza);** - funkcja odpowiadająca za losowe przemieszczanie się osobników o jedno pole w jednym cyklu symulacji

Poza klasą znajduje się funkcja

**void\* watek\_autoiteracja(void \*arg)** - Funkcja wykonywana w osobnym watku, odpowiedzialna za automatyczna zmiane stanow automatu w podanych odstepach czasu

1. **Gui.h, gui.cpp**

Zawierają funkcje slużące do obsługi trybu graficznego.

**void print\_in\_middle(WINDOW \*win, int starty, int startx, int width, char \*string, chtype color);** - Wydruk podanego tekstu na srodku podanego obszaru w danym oknie.  
Parametry funkcji:

|  |  |  |
| --- | --- | --- |
|  | *\*win* | Wskaznik na okno (NULL dla glownego okna) |
|  | *starty* | Offset Y, od ktorego obliczac srodek (0 gdy w obecnym miejscu) |
|  | *startx* | Offset X, od ktorego obliczac srodek (0 gdy w obecnym miejscu) |
|  | *width* | Maksymalna szerokosc przewidziana na dany tekst (gdy 0, to przyjmujemy 80) |
|  | *string* | Tekst do wydrukowania |
|  | *color* | Zestaw kolorow, w jakim wydrukowac napis |

**WINDOW \*stworz\_okno(int width, int height, int startx, int starty, char\* title);** - Tworzy okno ncurses z obramowaniem.  
Parametry:  
width Szerokosc okna   
height Wysokosc okna   
startx Odleglosc od lewej krawedzi, w jakiej powstanie okno (-1 = na srodku)   
starty Odleglosc od gornej krawedzi, w jakiej powstanie okno (-1 = na srodku)   
title Tytul okna

**void rozmiar\_planszy\_form();** - Wyswietla okno sluzace do podania rozmiaru planszy.

**void autoiteracja\_form();**

**void rysuj\_plansze(std::vector< std::vector <istota> >&);** - Rysuje cala plansze na podstawie wartosci w wektorze **plansza**.

**void rysuj\_pasek();** - Rysuje gorny pasek menu.

**void kursor\_na\_miejsce();**  - Przywraca kursor po kazdym rysowaniu na miejsce na planszy, na ktorym ostatnio zostawil go uzytkownik.

**void komunikat(const char\*, ...); -**  drukuje komunikat pod plansza na wzor funkcji printf()

**void komunikat2(const char\*, ...); -** drukuje komunikat pod plansza na wzor funkcji printf(). Funkcja analogiczna do poprzedniej; komunikat nie znika

**void\* watek\_komunikat\_wyczysc(void\*);** - usuwa komunikat stworzony przez funkcję komunikat

**void\* watek\_komunikat\_wyczysc\_bez\_czekania(void\*); -**usuwa komunikat stworzony przez funkcję komunikat2

**Zmienne globalne:**

**extern unsigned short rozmiar** - rozmiar planszy

**extern unsigned short current\_x** - obecne polozenie kursora – X

**extern unsigned short current\_y** - obecne polozenie kursora - Y

**extern unsigned int autoiteracja** - 0, jesli wylaczona jest automatyczna iteracja. W przeciwnym wypadku czas w milisekudach, co jaki zmieniac stan komorek

**extern unsigned short nieskonczona\_petla**

**extern unsigned int iteracja** - numer kolejnej iteracji

**extern pthread\_t pwatek-** watek wykorzystywany na potrzeby autoiteracji

**extern int liczbaA** – liczba osobników populacji A

**extern int liczbaB** – liczba osobników populacji B

**extern szachownica pole**

**extern pthread\_t pkomunikat\_watek** - watek, ktory czeka 3 sekundy i czysci wyswietlony komunikat

**extern unsigned short pkomunikat\_watek\_running** – informuje czy watek komunikatu jest uruchomiony

**Zdefiniowane kolory:**

**KOMA\_CP** - para kolorow uzywana do oznaczania zywych komorek populacji A

**KOMB\_CP** - para kolorow uzywana do oznaczania zywych komorek populacji A

**KOMN\_CP** – para kolorow używana do oznaczania martwych komórek

**FORM\_CP** – para kolorow używana do okien

**FIELD\_CP** – para kolorów używana do pól tekstowych