

Sprawozdanie z Projektu  
*The Game of Life*

Daniel Ślusarczyk i Jakub Łaba

07.04.2021

### **Streszczenie**

Dokument stanowi sprawozdanie z projektu "*The Game of Life*", z wyszczególnionym opisem aktualnego stanu projektu, zmian względem specyfikacji oraz wnioskami.

# Spis treści

1	Zarys Projektu . . . . .	1
2	Struktura programu . . . . .	1
	2.1 Struktura katalogów . . . . .	1
	2.2 Struktura modułów . . . . .	1
3	Kompilacja programu . . . . .	2
4	Przykładowe wyniki działania programu . . . . .	3
5	Zmiany względem specyfikacji . . . . .	11
6	Podsumowanie współpracy . . . . .	14
7	Podsumowanie projektu . . . . .	14
8	Wnioski . . . . .	14

## 1 Zarys Projektu

Projekt „The Game of Life” zakłada stworzenie programu emulującego działanie automatu komórkowego Johna Conwaya – Grę w Życie (ang. The Game of Life), oraz napisanie dokumentacji opisującej stronę funkcjonalną i implementacyjną programu. Działanie gry jest następujące: na prostokątnej planszy znajdują się komórki żywe oraz martwe. Program przeprowadza określoną liczbę generacji, w których komórki ożywają/obumierają na następujących zasadach:

- **Komórki żywe** – pozostają żywe w kolejnej generacji, jeżeli posiadają dokładnie 2 lub 3 żywych sąsiadów, w każdym innym przypadku umierają.
- **Komórki martwe** – ożywają w kolejnej generacji, jeżeli posiadają dokładnie trzech żywych sąsiadów, w przeciwnym przypadku pozostają martwe.

Komórki znajdujące się poza planszą są uznawane za permanentnie martwe.

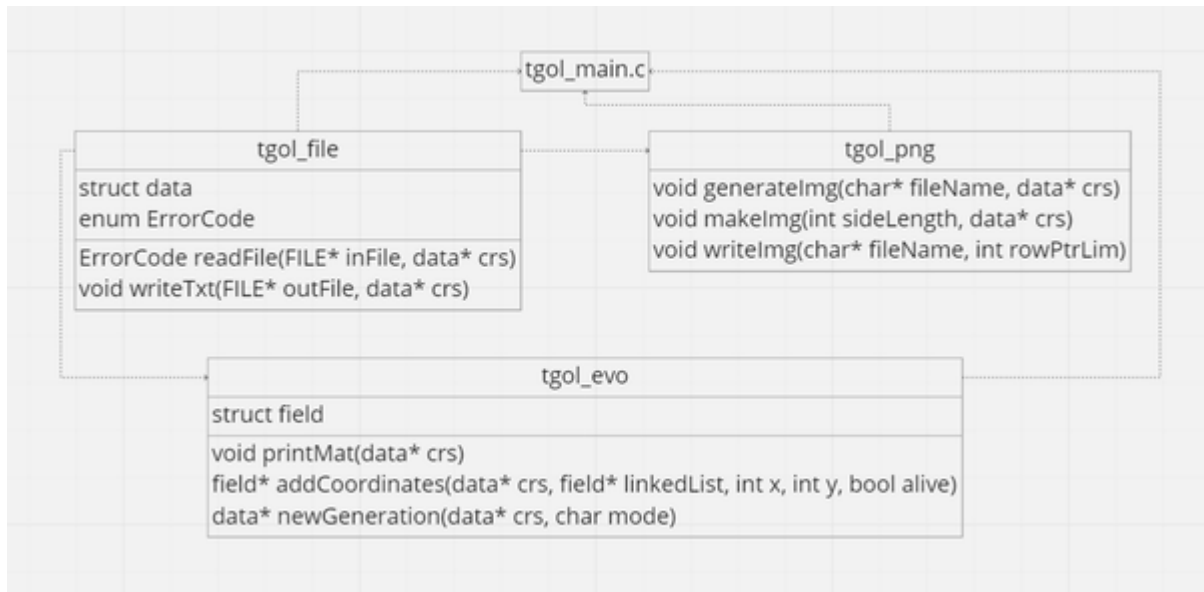
## 2 Struktura programu

### 2.1 Struktura katalogów

Projekt został podzielony na 3 katalogi dla zachowania porządku w jego plikach. Katalog *kod* zawiera cały kod źródłowy projektu oraz skrypt makefile. Katalog *dokumenty* – całą dokumentację projektu, natomiast *input* przykładowe pliki wejściowe do uruchomienia programu.

### 2.2 Struktura modułów

Program składa się z trzech modułów: *tgol\_file*, *tgol\_evo*, *tgol\_png* oraz pliku *tgol\_main.c*, zawierającego funkcję `main` całego programu oraz kluczowe makra. Każdy z modułów składa się z dwóch plików: `.h` oraz `.c`, o odpowiedniej nazwie. W każdym z modułów plik `.h` zawiera prototypy funkcji oraz ewentualne definicje struktur, stałe oraz makra. Większość obsługi błędów znajduje się bezpośrednio w funkcji `main`, z tego względu, że gdyby zostały przeniesione do zewnętrznych funkcji, niezbędne byłoby przekazywanie dużej ilości parametrów – byłoby to niezbyt czytelne. Drugą możliwością byłoby zadeklarowanie owych zmiennych jako globalne, jednak duża ilość zmiennych globalnych nie jest dobrą praktyką. Moduł *tgol\_file* zawiera funkcje odpowiadającą za wczytanie zawartości plików tekstowych do macierzy w formacie CRS oraz za zapis końcowego stanu planszy do plików tekstowych. Zawiera również definicje kodów błędów oraz pomocnicze struktury, pozwalające na wyświetlanie komunikatów o błędach oraz sprawdzanie flag. Moduł *tgol\_evo* zawiera funkcje związane z przeprowadzaniem ewolucji – tj. wyświetlanie macierzy na podstawie CRS na ekran oraz przekształcanie CRS zgodnie z jednym z dwóch sąsiedztw, oraz zasadami "The Game of Life". Ponadto w module znajdują się funkcje pomocnicze, dla tych dwóch głównych funkcjonalności. Moduł *tgol\_png* zawiera funkcje związane z generowaniem plików graficznych w formacie `.png` na podstawie macierzy w formacie CRS.



Rysunek 1: Diagram modułów

### 3 Kompilacja programu

Skrypt makefile został wyposażony w 4 opcje:

- **Domyślna kompilacja – make**

Kompiluje program w wersji dla użytkownika z zastosowaniem sąsiedztwa Moore’a.

- **Kompilacja z sąsiedztwem von Neumanna – make neumann**

Kompiluje plik w wersji dla użytkownika z zastosowaniem sąsiedztwa von Neumanna.

- **Kompilacja w trybie debug – make debug**

Kompiluje program z dodatkowymi informacjami, przydatnymi do debugowania oraz flagą -g, aby umożliwić analizę za pomocą programów *gdb* oraz *valgrind*.

- **Skasowanie plików – make clean**

Kasuje plik wykonywalny oraz wszystkie pliki obiektowe.

## 4 Przykładowe wyniki działania programu

Przedstawione poniżej wyniki programu prezentują różne kombinacje flag z różnymi plikami w celu zarysowania możliwości działania programu.

- Podstawowe uruchomienie

### Plik wejściowy

Rysunek 2 przedstawia plik wejściowy zawierający 5 punktów rozmieszczonych na planszy 10 na 10 tzw. *szybowiec*.



Rysunek 2: Plik wejściowy glider.txt

### Wywołanie:

```
./TGOL ../input/glider.txt 3
```

### Wynik działania

Rysunek 3 przedstawia wynik działania programu zawierającego zadaną liczbę generacji wypisanych jedną pod drugim.



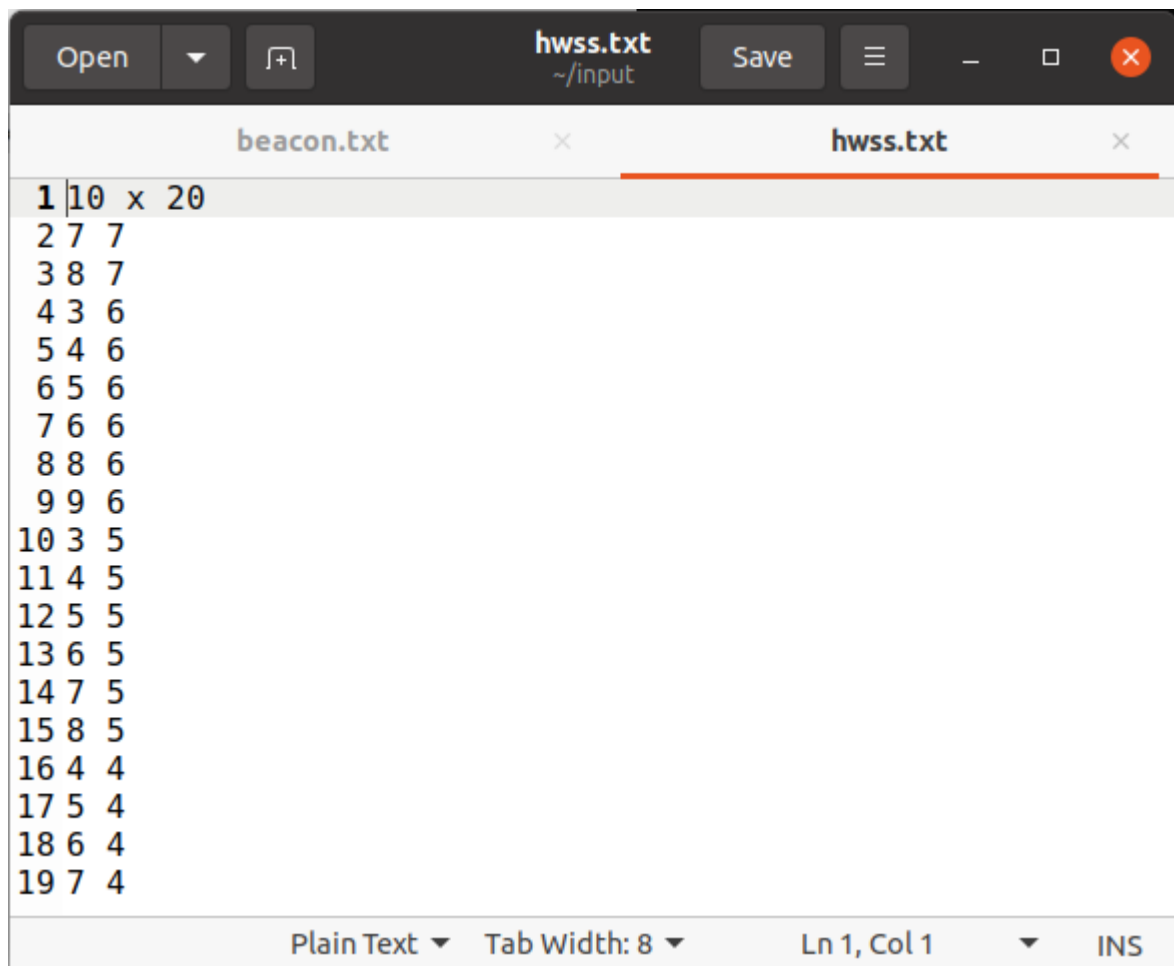




- Uruchomienie z flagą `-refresh` z parametrem oraz flagą `-sbs`

#### Plik wejściowy

Rysunek 7 przedstawia plik wejściowy zawierający 18 punktów rozmieszczonych na planszy 10 na 20.



Rysunek 7: Plik wejściowy hwss.txt

#### Wywołanie:

```
./TGOL ../input/hwss.txt 10 -refresh 0.5 -sbs
```

#### Wynik działania

Rysunki 7, 8 i 9 przedstawiają wynik działania programu, który w pierwszym etapie informuje o częstotliwości odświeżania, a następnie w każdym kolejnym etapie wyświetla kolejne generacje, czekając na przycisk.

```
wczytano czas odświeżenia ekranu: 0.50s
wciśnij ENTER aby kontynuować
```

Rysunek 8: Wynik działania programu w etapie pierwszym

```
GENERACJA NUMER: 1
. . . . .
. . . . .
. . . . .
. . . X X X X X . . . . .
. . X . . . . X . . . . .
. . . . . . . X . . . . .
. . X . . . . X . . . . .
. . . . X X . . . . .
. . . . .
. . . . .
. . . . .
wciśnij ENTER aby kontynuować
```

Rysunek 9: Wynik działania programu w etapie drugim

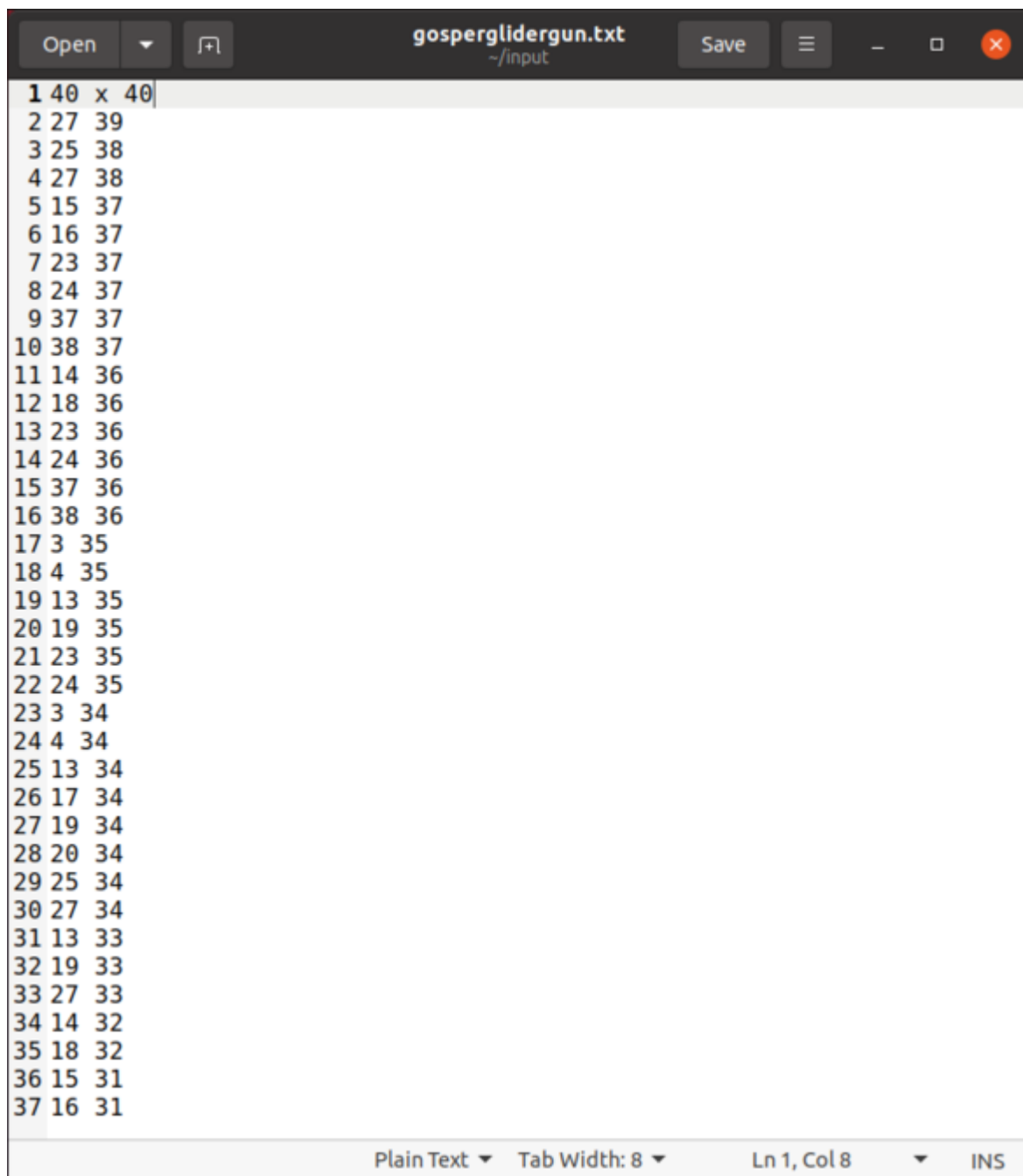
```
GENERACJA NUMER: 2
. . . . .
. . . . .
. . . . X X X X . . . . .
. . . X X X X X X . . . . .
. . . X X X X . X X . . . . .
. . . . . . X X . . . . .
. . . . .
. . . . .
. . . . .
. . . . .
wciśnij ENTER aby kontynuować
```

Rysunek 10: Wynik działania programu w etapie trzecim

- Uruchomienie z flagą `-refresh` z parametrem oraz flagą `-save`

#### Plik wejściowy

Rysunek 11 przedstawia plik wejściowy zawierający 36 punktów rozmieszczonych na planszy 40 na 40.



```
1 40 x 40
2 27 39
3 25 38
4 27 38
5 15 37
6 16 37
7 23 37
8 24 37
9 37 37
10 38 37
11 14 36
12 18 36
13 23 36
14 24 36
15 37 36
16 38 36
17 3 35
18 4 35
19 13 35
20 19 35
21 23 35
22 24 35
23 3 34
24 4 34
25 13 34
26 17 34
27 19 34
28 20 34
29 25 34
30 27 34
31 13 33
32 19 33
33 27 33
34 14 32
35 18 32
36 15 31
37 16 31
```

Rysunek 11: Plik wejściowy gosperglidergun.txt

#### Wywołanie:

```
./TGOL ../input/gosperglidergun.txt 100 -refresh 0.05 -save out
```

## Wynik działania

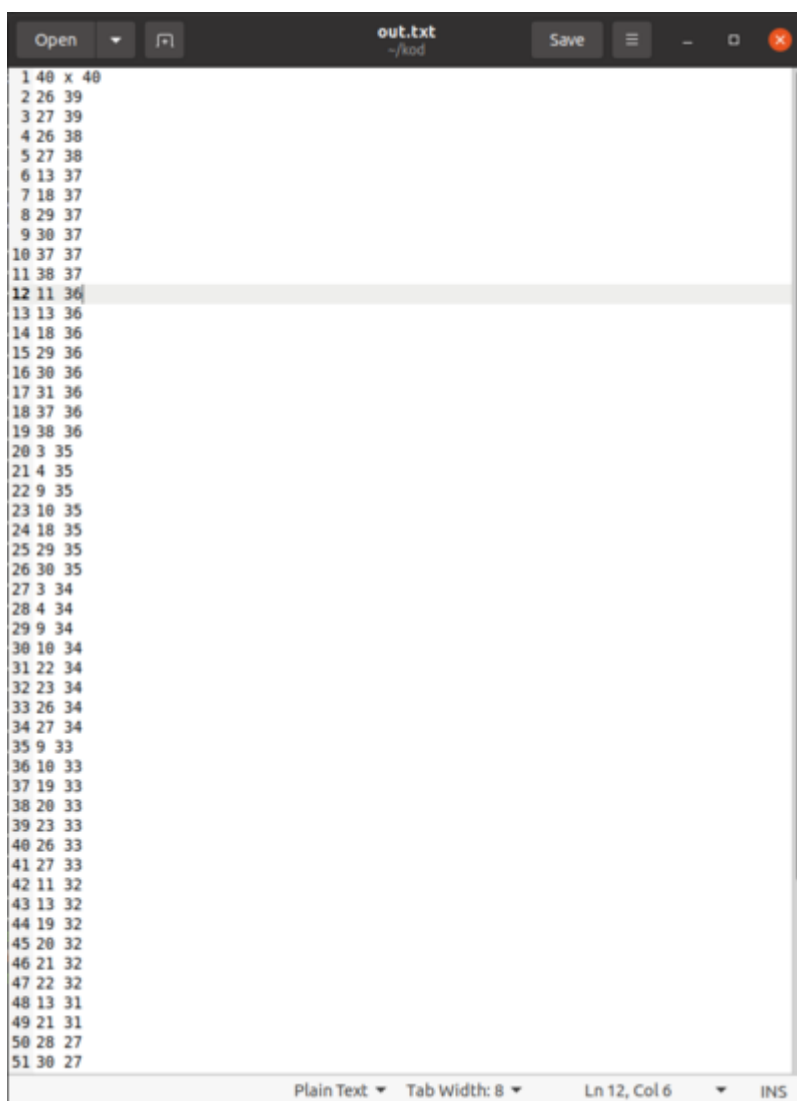
Rysunki 12 i 13 przedstawiają wynik działania programu, który w pierwszym etapie informuje o wybraniu podanej częstotliwości odświeżania, a następnie wyświetla kolejne generacje w podanym odstępie, kończąc wyświetlanie generacji w drugim etapie. W ostatnim kroku program generuje plik tekstowy z rozszerzeniem *.txt* (Rysunek 14) i plik graficzny z rozszerzeniem *.png* (Rysunek 15) na podstawie ostatniej generacji.

```
wczytano czas odświeżenia ekranu: 0.05s  
wciśnij ENTER aby kontynuować
```

Rysunek 12: Wynik działa programu w etapie pierwszym

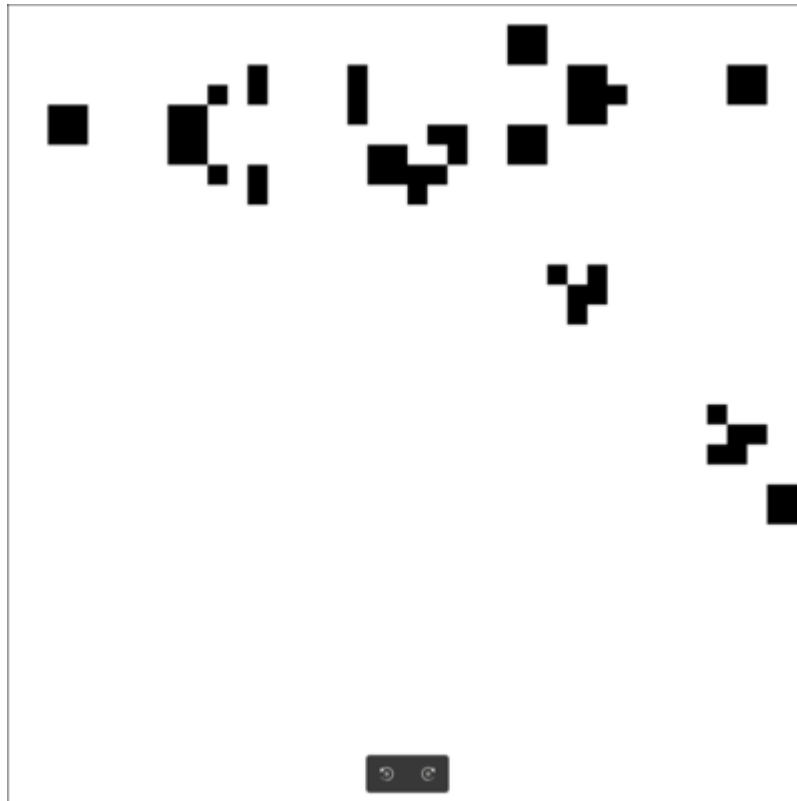


Rysunek 13: Wynik działa programu w etapie drugim



```
1 40 x 40
2 26 39
3 27 39
4 26 38
5 27 38
6 13 37
7 18 37
8 29 37
9 30 37
10 37 37
11 38 37
12 11 36
13 13 36
14 18 36
15 29 36
16 30 36
17 31 36
18 37 36
19 38 36
20 3 35
21 4 35
22 9 35
23 10 35
24 18 35
25 29 35
26 30 35
27 3 34
28 4 34
29 9 34
30 10 34
31 22 34
32 23 34
33 26 34
34 27 34
35 9 33
36 10 33
37 19 33
38 20 33
39 23 33
40 26 33
41 27 33
42 11 32
43 13 32
44 19 32
45 20 32
46 21 32
47 22 32
48 13 31
49 21 31
50 28 27
51 30 27
```

Rysunek 14: Wynik działania programu –plik tekstowy



Rysunek 15: Wynik działania programu –plik graficzny

## 5 Zmiany względem specyfikacji

Założeniem projektu od samego początku powstawania było zachowanie ścisłości między pisanym programem, a specyfikacją funkcjonalną i implementacyjną. Proces powstawania elementów projektu zweryfikował dokładnie wszystkie założone idee i spowodował zmianę następujących punktów w poszczególnych dokumentach:

### Zmiana formatu danych wejściowych

Program potrzebuje danych z informacjami o stanie wejściowej planszy (generacji 0), oraz o liczbie generacji, którą użytkownik chce wygenerować. Plik wejściowy powinien zawierać informacje sformatowane następująco:

```
[row] x [col]
```

```
[x1] [y1]
```

```
[x2] [y2]
```

```
...
```

```
[xn] [yn]
```

Gdzie:

- `row` – liczba rzędów planszy
- `col` – liczba kolumn planszy

- $x_1, x_2, \dots, x_n$  – współrzędne x-owe kolejnych żywych komórek
- $y_1, y_2, \dots, y_n$  – współrzędne y-owe kolejnych żywych komórek

**Uwaga:** kolejność podawania współrzędnych żywych komórek jest ustalona od lewej do prawej i od góry do dołu i tylko w takiej kolejności jest akceptowana. W przypadku niespełnienia tego warunku, lub innych błędów w formacie są zwracane odpowiednie kody błędów. Współrzędne określa się względem klasycznego 2-wymiarowego kartezjańskiego układu współrzędnych, a punkty nie mogą się powtarzać.

### **Przykład**

Chcąc wczytać planszę 2x3 z jedną żywą komórką w lewym górnym rogu, oraz jedną w prawym dolnym rogu, plik powinien wyglądać w następujący sposób:

```
2 x 3
1 2
3 1
```

### **Zmiana działania flagi -save**

**-save** –program wywołany z tą flagą zapisze do pliku wynikowego ostatnią przeprowadzoną generację, sformatowaną identycznie jak plik wejściowy (dzięki temu można użyć pliku wynikowego jako wejścia do kolejnego uruchomienia programu) oraz zdjęcie w formacie *png* ostatniej generacji. Nazwę pliku wynikowego należy podać bezpośrednio po flagie **-save**, w przypadku chęci zapisania pliku w innym folderze, należy podać jego nazwę razem z pożądaną ścieżką

### **Usunięcie błędów INPUT\_NOT\_INT, INPUT\_SHORT, INPUT\_XY**

Błędy INPUT\_NOT\_INT, INPUT\_SHORT, INPUT\_XY w wyniku nowego formatu pliku wejściowego straciły użyteczność.

### **Dodanie błędów INPUT\_INCORRECT, INPUT\_DIMS, INPUT\_XY\_LIMIT, INPUT\_INCORRECT\_ORDER**

Wprowadzenie nowego formatu wymusiło wprowadzenie następujących błędów, które są zwracane w przypadku nieprawidłowego pliku wejściowego zgodnie z tabelą:

Komunikat	Przyczyna
INPUT_INCORRECT	W pliku wejściowym liczba współrzędnych x nie jest równa y, plik zawiera znaki niebędące liczbami naturalnymi, lub plik nie zawiera informacji o żadnych punktach
INPUT_DIMS	W pliku wejściowym znajduje się niewłaściwie wymiary planszy (niezgodny z formatem lub brak wymiaru)
INPUT_XY_LIMIT	W pliku wejściowym podano zbyt wielkie wartości współrzędny względem deklarowanych wymiarów
INPUT_INCORRECT_ORDER	W pliku wejściowym zastosowano niepoprawną kolejność współrzędnych lub wystąpiło powtórzenie



## 6 Podsumowanie współpracy

Współpraca w obrębie projektu przebiegała sprawnie i bez większych problemów. Wartym odnotowania współczynnikiem, który wpływał korzystnie na efektywność współpracy, było korzystanie systemu kontroli wersji. Z pomocą tego narzędzia wymiana aktualnymi wersjami programu i zarządzanie całą strukturą programu odbywało się efektywnie. Pomocne były też częste rozmowy z wyjaśnieniami zastosowanych rozwiązań, wspólne uzgadnianie podziału pracy i podobieństwo tworzenia kodu na poziomie conceptualnym. Jedynym czynnikiem czasami negatywnie wpływającym na współpracę było modyfikowanie kodu, którego nie jest się autorem. Mogło to powodować nieścisłości w kodzie i utratę pewności do prawidłowego działania danego fragmentu kodu. Podsumowując, współpraca projektowa była na zadowalającym poziomie pomimo drobnych problemów i pozwalała na wspólne uczenie się nowych rozwiązań.

## 7 Podsumowanie projektu

Projekt "The Game of Life" został w całości przygotowany od 24.02.2021 do 07.04.2021. W ramach niego powstał działający program *TGOL*, makefile, specyfikacja funkcjonalna, specyfikacja implementacyjna oraz sprawozdanie końcowe. Przygotowane rozwiązanie implementuje różne flagi pomocne podczas analizowania kolejnych generacji, posiada prostą i intuicyjną obsługę oraz pozwala na efektywne sprawdzanie kolejnych generacji za sprawą zastosowania metody przechowywania macierzy rzadkich CRS. Dodatkowo przy wywołaniu możliwe jest użycie flagi *-save*, która pozwala na zapis ostatniej generacji w postaci *png*. Program *TGOL* został przetestowany pod kątem różnych błędów i nietypowych plików wejściowych. W konsekwencji jego działanie powinno być w znaczącej liczbie przypadków zgodne z oczekiwaniami.

## 8 Wnioski

Projekt "The Game of Life" jest zadaniem, który pozwala na różnorodne podejście do problemu i zaimplementowanie rozwiązań, które znacząco wpłyną na działanie programu. W przypadku *TGOL*, czyli oprogramowania, które może analizować duże zbiory danych pomocne, jest zastosowanie narzędzia pozwalającego maksymalnie zmniejszyć używaną pamięć. Użyta w programie metoda *CRS* pozwoliła ograniczyć ten problem, ale jednocześnie negatywnie wpłynęła na czytelność i przejrzystość kodu. Ponadto przytoczony sposób reprezentacji macierzy rzadkiej utrudnia odwołanie się do elementów zerowych. Drugim istotnym aspektem jest ograniczenie do zera tzw. wycieków pamięci, które w przypadku zastosowania języka C bywają ciężkie do wykrycia i naprawienia. Dla omawianego programu niezbędne okazało się użycie dodatkowych programów raportujących takie niezamierzone użycie pamięci.