Информатика

Ткаченко Кирилл Станиславович
Инженер 1-й категории,
ФГАОУ ВО «Севастопольский государственный университет»

Разработка игры «Жизнь» по Конвею на школьном алгоритмическом языке

Предлагается разработанная на школьном алгоритмическом языке реализация игры «Жизнь» по Конвею — одного из наиболее известных и распространенных клеточных автоматов. Приводится полный исходный текст и результаты работы. Программа может быть полезна всем изучающим программирование и школьный алгоритмический язык, подпрограммы и обработку двумерных массивов.

В настоящее время активно используются системы, имитирующие реальные объекты. Для имитации можно использовать разнообразные приемы, одним из которых является применение клеточных автоматов. Одним из первых и наиболее распространенных клеточных автоматов является игра «Жизнь», впервые предложенная Конвеем [1, 2]. Для этой игры имеется чрезвычайно много программных реализаций на языках высокого и низкого уровня [3]. В настоящей работе будет предложена программная реализация этого клеточного автомата на школьном алгоритмическом языке по аналогии с работой [4].

Прежде чем приступить непосредственно к описанию разработанной программы, стоит описать правила функционирования игры «Жизнь». Игра происходит на некотором прямоугольном поле. Каждая клетка поля может находиться в двух состояниях: в ней есть «жизнь» или нет «жизни». Новое состояние поля определяется после изменения состояний отдельных клеток. Если в клетке нет «жизни» и число соседей равно трем, то в клетке появляется «жизнь». Если в клетке есть «жизнь», но число соседей меньше двух либо больше трех, то в клетке «жизнь» исчезает. В противном случае состояние клетки остается неизменным.

В начале программы задаются следующие целочисленные параметры: КоличествоПоколений — количество поколений, в течение которых будет происходить работа конечного автомата, КоличествоСтолбцов — количество столбцов в игровом поле, КоличествоСтрок — количество строк в игровом поле (граничные

строки и столбцы не будут рассматриваться конечным автоматом), КакРедко — как редко при начальном псевдослучайном заполнении игрового поля клетка будет «живой» (вероятность выбора состояния «живой» клетки):

```
цел КоличествоПоколений = 100;
цел КоличествоСтолбцов = 25;
цел КоличествоСтрок = 10;
вещ КакРедко = 0.15;
```

Головным алгоритмом является Потенциал (изображенный на рисунке 1). В этом алгоритме имеются следующие целочисленные двумерные массивы и переменные, а именно: Поле — игровое поле для конечного автомата, КопияПоля — его временная копия, А, Б и В — счетчики циклов, К — количество соседей у клетки:

```
алг Потенциал
нач
  цел таб Поле[1 : КоличествоСтрок, 1 : КоличествоСтолбцов];
  цел таб КопияПоля[1 : КоличествоСтрок, 1 : КоличествоСтолбцов];
  цел А, Б, В;
  цел К;
```

Вначале происходит инициализация игрового поля, создается его временная копия, поле сообщается пользователю:

```
  ИнициализацияПоля(Поле);
  КопироватьПоле(Поле, КопияПоля);
  ВыводПоля(Поле);
  вывод нс;
```

Для всех поколений:

```
  нц для В от 1 до КоличествоПоколений
```

По всем строкам и столбцам, кроме граничных:

```
    нц для Б от 2 до КоличествоСтрок - 1
      нц для А от 2 до КоличествоСтолбцов - 1
```

Подсчитывается количество соседей у клетки:

```
        К := Поле[Б - 1, А - 1] + Поле[Б - 1, А] + Поле[Б - 1, А + 1];
        К := К + Поле[Б, А - 1] + Поле[Б, А + 1];
        К := К + Поле[Б + 1, А - 1] + Поле[Б + 1, А] + Поле[Б + 1, А + 1];
```

По правилам конечного автомата определяется новое состояние клетки. Если в клетке нет «жизни» и число соседей равно трем, то в клетке появляется «жизнь». Если в клетке есть «жизнь», но число соседей меньше двух либо больше трех, то в клетке «жизнь» исчезает. В противном случае, состояние клетки остается неизменным.

```
        КопияПоля[Б, А] := Поле[Б, А];
        если (Поле[Б, А] = 0) и (К = 3)
          то
            КопияПоля[Б, А] := 1;
```

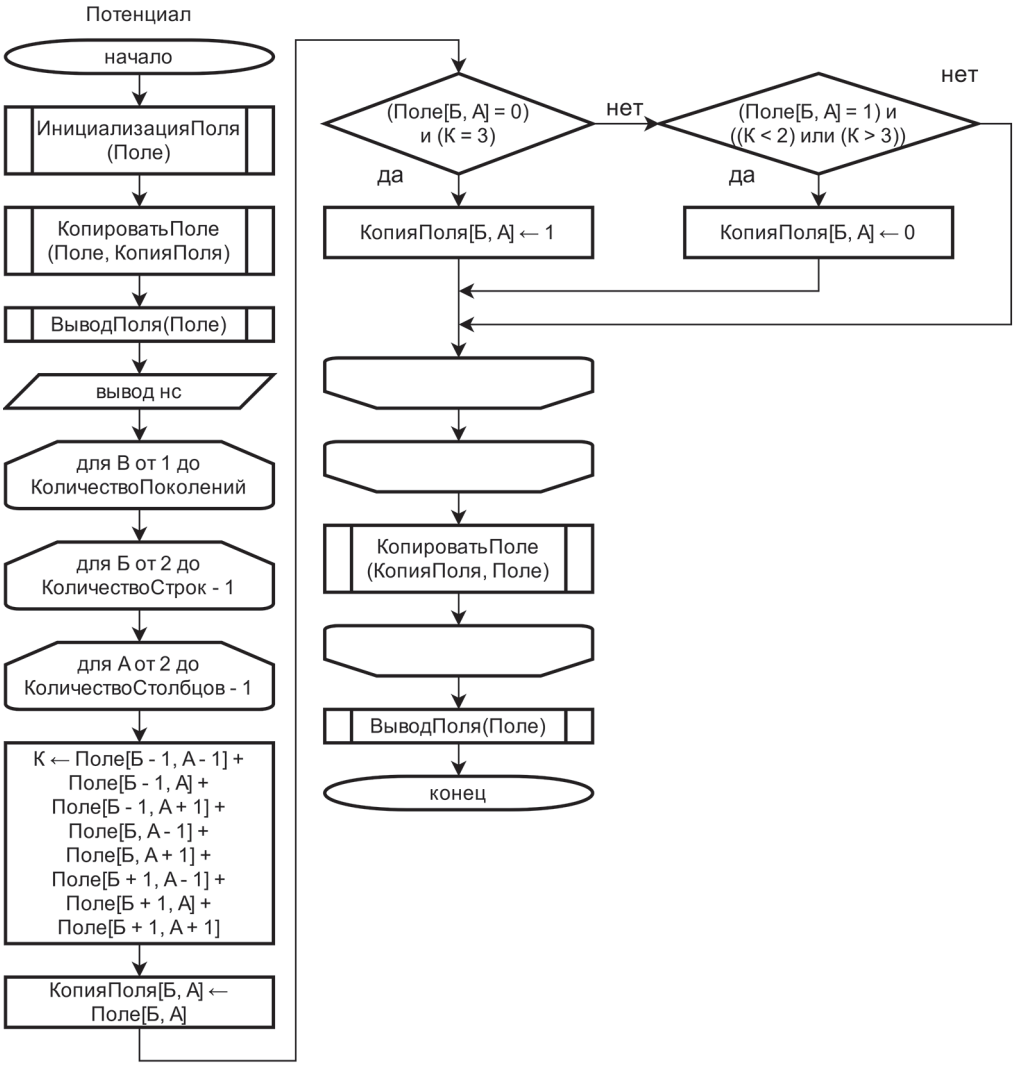

Рис. 1. Схема головного алгоритма

```
        иначе
          если (Поле[Б, А] = 1) и ((К < 2) или (К > 3))
            то
              КопияПоля[Б, А] := 0;
          все
      все
    кц
  кц
```

Временная копия поля замещает само поле:

```
  КопироватьПоле(КопияПоля, Поле);
кц
```

Итоговое поле сообщается пользователю:

```
  ВыводПоля(Поле);
кон
```

Алгоритм ИнициализацияПоля заполняет начальными значениями двумерный массив — свой аргумент Поле. В этом алгоритме используются целочисленные счетчики циклов А и Б:

```
алг ИнициализацияПоля(аргрез цел таб Поле[1 : КоличествоСтрок,
1 : КоличествоСтолбцов])
нач
  цел А, Б;
```

Вначале для всех строк и столбцов состояние клетки является не «живая»:

```
  нц для Б от 1 до КоличествоСтрок
    нц для А от 1 до КоличествоСтолбцов
      Поле[Б, А] := 0;
    кц
  кц
```

Затем для всех строк и столбцов, если выбранное из отрезка [0, 1] псевдослучайное число не превосходит КакРедко, то состояние клетки становится «живая»:

```
  нц для Б от 2 до КоличествоСтрок - 1
    нц для А от 2 до КоличествоСтолбцов - 1
      если rnd(1) <= КакРедко
        то
          Поле[Б, А] := 1;
      все
    кц
  кц
кон
```

Алгоритм ВыводПоля сообщает аргумент Поле пользователю. Целочисленные А и Б – счетчики циклов:

```
алг ВыводПоля(арг цел таб Поле[1 : КоличествоСтрок, 1 : Количе-
ствоСтолбцов])
нач
  цел А, Б;
```

По всем строкам и столбцам поле, если клетка является «живой», то сообщается “О”, иначе — точка, после каждой строки выводится терминатор:

```
  нц для Б от 1 до КоличествоСтрок
    нц для А от 1 до КоличествоСтолбцов
      если Поле[Б, А] = 1
        то
          вывод "О";
        иначе
          вывод ".";
      все
    кц
```

```
    вывод нс;
  кц
кон
```

Алгоритм КопироватьПоле копирует аргумент Поле в КопияПоля. Переменные А и Б — целочисленные счетчики циклов:

```
алг КопироватьПоле(арг цел таб Поле[1 : КоличествоСтрок, 1 : КоличествоСтолбцов], аргрез цел таб КопияПоля[1 : КоличествоСтрок, 1 : КоличествоСтолбцов])
нач
  цел А, Б;
```

По всем строкам и столбцам состояния клеток копируются:

```
  нц для Б от 1 до КоличествоСтрок
    нц для А от 1 до КоличествоСтолбцов
      КопияПоля[Б, А] := Поле[Б, А];
    кц
  кц
кон
```

Полный исходный текст программы находится в приложении А, результаты работы — Б, среда с отработавшей программой — на рисунке 2.

```
 1  цел КоличествоПоколений = 100;
 2  цел КоличествоСтолбцов = 25;
 3  цел КоличествоСтрок = 10;
 4  вещ КакРедко = 0.15;
 5
 6  алг Потенциал
 7  нач
 8  . цел таб Поле[1 : КоличествоСтрок, 1 : КоличествоСтолбцов];
 9  . цел таб КопияПоля[1 : КоличествоСтрок, 1 : КоличествоСтолбцов];
10  . цел А, Б, В;
11  . цел К;
12  .
13  . ИнициализацияПоля(Поле);
14  . КопироватьПоле(Поле, КопияПоля);
15  . ВыводПоля(Поле);
16  . вывод нс;
17  .
18  . нц для В от 1 до КоличествоПоколений
19  . . нц для Б от 2 до КоличествоСтрок - 1
20  . . . нц для А от 2 до КоличествоСтолбцов - 1
21  . . . . К := Поле[Б - 1, А - 1] + Поле[Б - 1, А] + Поле[Б - 1, А + 1];
```

```
.........................
.............0...........
.............0...........
.............0...........
.........................
.........................
........0................
.......0.0...............
.......00................
.........................
```

Анализ Выполнено шагов: 230948

Рис. 2. Среда разработки с отработавшей программой

Полученная программа найдет свое место при обучении алгоритмам и основам школьного алгоритмического языка, в частности, обработке двумерных массивов и работе с подпрограммами.

Литература

1.Игра «Жизнь» // Википедия. URL: https://ru.wikipedia.org/wiki/Игра_«Жизнь» (дата обращения: 09.09.2020).

2.Conway's Game of Life // Wikipedia. URL: https://en.wikipedia.org/wiki/Conway's_Game_of_Life (дата обращения: 09.09.2020).

3.Conway's Game of Life // Rosetta Code. URL: http://rosettacode.org/wiki/Conway's_Game_of_Life (дата обращения: 08.09.2020).

4.Ткаченко К.С. «Муравей Лэнгтона» на школьном алгоритмическом языке / К.С. Ткаченко // Журнал для старшеклассников и учителей «Потенциал»: «Математика. Физика. Информатика». Июнь № 06, 2018, ISSN 1814-6422. С. 49–55.

Приложение А
Полный исходный текст программы

```
цел КоличествоПоколений = 100;
цел КоличествоСтолбцов = 25;
цел КоличествоСтрок = 10;
вещ КакРедко = 0.15;

алг Потенциал
нач
  цел таб Поле[1 : КоличествоСтрок, 1 : КоличествоСтолбцов];
  цел таб КопияПоля[1 : КоличествоСтрок, 1 : КоличествоСтолбцов];
  цел А, Б, В;
  цел К;

  ИнициализацияПоля(Поле);
  КопироватьПоле(Поле, КопияПоля);
  ВыводПоля(Поле);
  вывод нс;
  нц для В от 1 до КоличествоПоколений
    нц для Б от 2 до КоличествоСтрок - 1
      нц для А от 2 до КоличествоСтолбцов - 1
        К := Поле[Б - 1, А - 1] + Поле[Б - 1, А] + Поле[Б - 1, А + 1];
        К := К + Поле[Б, А - 1] + Поле[Б, А + 1];
        К := К + Поле[Б + 1, А - 1] + Поле[Б + 1, А] + Поле[Б + 1, А + 1];
```

```
        КопияПоля[Б, А] := Поле[Б, А];
        если (Поле[Б, А] = 0) и (К = 3)
          то
            КопияПоля[Б, А] := 1;
          иначе
            если (Поле[Б, А] = 1) и ((К < 2) или (К > 3))
              то
                КопияПоля[Б, А] := 0;
            все
        все
      кц
    кц

    КопироватьПоле(КопияПоля, Поле);
  кц

  ВыводПоля(Поле);
кон

алг ИнициализацияПоля(аргрез цел таб Поле[1 : КоличествоСтрок,
1 : КоличествоСтолбцов])
нач
  цел А, Б;

  нц для Б от 1 до КоличествоСтрок
    нц для А от 1 до КоличествоСтолбцов
      Поле[Б, А] := 0;
    кц
  кц

  нц для Б от 2 до КоличествоСтрок - 1
    нц для А от 2 до КоличествоСтолбцов - 1
      если rnd(1) <= КакРедко
        то
          Поле[Б, А] := 1;
      все
    кц
  кц
кон
алг ВыводПоля(арг цел таб Поле[1 : КоличествоСтрок, 1 : Количе-
ствоСтолбцов])
нач
  цел А, Б;

  нц для Б от 1 до КоличествоСтрок
    нц для А от 1 до КоличествоСтолбцов
      если Поле[Б, А] = 1
```

```
        то
          вывод "0";
        иначе
          вывод ".";
      все
    кц
    вывод нс;
  кц
кон

алг КопироватьПоле(арг цел таб Поле[1 : КоличествоСтрок, 1 :
КоличествоСтолбцов], аргрез цел таб КопияПоля[1 : Количество-
Строк, 1 : КоличествоСтолбцов])
нач
  цел А, Б;

  нц для Б от 1 до КоличествоСтрок
    нц для А от 1 до КоличествоСтолбцов
      КопияПоля[Б, А] := Поле[Б, А];
    кц
  кц
кон
```

Приложение Б
Результаты работы программы

```
.........................
...........0.00.0........
..................0.0.....
.0.......00...0..........
...0.............0.......
....0...0.0..............
..............0..........
.0......0.............0..
.......00..0......0..0...
.........................

.........................
..............0..........
..............0..........
..............0..........
.........................
.........................
........0................
.......0.0...............
.......00................
.........................
```