

Министерство науки и высшего образования РФ
Брянский государственный технический университет

Кафедра: «Компьютерные технологии и системы»

Дисциплина: «Языки программирования»

КУРСОВАЯ РАБОТА

на тему: «Генетический алгоритм в клеточном автомате»

Выполнил студент гр: 3-21-
ИСТ-итпк-Б

Журавлёв Е. А.

Проверил преподаватель:

Чмыхов Д. В.

Брянск, 2022

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовую работу по дисциплине «Языки программирования»

Студент Журавлёв Е. А.

Группа 3-21-ИСТ-итпк-Б

Тема: разработка демонстрационной программы эвристического алгоритма генетического отбора в клеточном автомате.

Общая формулировка задания:

Необходимо разработать демонстрационную программу эвристического алгоритма генетического отбора на основе клеточного автомата с использованием языка программирования C#.

Требования к пользовательскому интерфейсу:

- программа должна работать в текстовом режиме программы-терминала;
- должна отображаться визуализация клеточного автомата и поясняющей информации;
- в программе должны использоваться элементы управления в виде интерактивного текстового интерфейса;
- должны отображаться текущие результаты отбора.

Требования к функциональным возможностям:

- программа должна предоставлять возможности настройки клеточного автомата;
- интерфейс настройки поведения особей в клеточном автомате;
- удобочитаемые таблицы с результатами отбора и текущими характеристиками популяции;

Руководитель

Чмыхов Д.В.

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	2
ВВЕДЕНИЕ	3
ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	4
2.1 Клеточный автомат	4
2.2 Генетический алгоритм	5
2.3 Текстовой интерфейс пользователя	8
3. ПРАКТИЧЕСКАЯ ЧАСТЬ.	10
3.1 Общее описание алгоритма	10
3.2 Организация программы	14
4. ЗАКЛЮЧЕНИЕ	19
5. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	20
ПРИЛОЖЕНИЕ “Программный код”	Ошибка! Закладка не определена.

1. ВВЕДЕНИЕ

Темой работы является разработка программы на языке программирования C#, которая демонстрирует работу эвристического алгоритма генетического отбора в клеточном автомате.

Актуальность работы заключается в применимости алгоритма генетического отбора к различным системам, в частности в процессе обучения систем искусственного интеллекта на основе Искусственных Нейронных Сетей.

В процессе выполнения работы, её реализации на языке программирования C#, предполагается использование полученных ранее знаний:

- понимания о практическом применении структур данных (стек, дек, очередь) при построении алгоритмов работы клеточного аппарата.
- понимания о типах данных в языке C#, при подборе оптимальных типов для организации больших массивов (например: *sbyte* вместо *integer*)
- использование и построение простых текстовых интерактивных интерфейсов, что является весьма важным навыком.
- использование понятия рекурсивных алгоритмов
- Получение новых навыков.

Целью работы является построение демонстрационной программы генетического отбора, изучение дополнительной литературы о генетических алгоритмах, передового опыта использования генетических алгоритмов. Практически реализовать демонстрацию, в максимально подходящей для этого системе - клеточном автомате.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1 Клеточный автомат

Клеточные автоматы были предложены в работе фон Неймана. Большой интерес к ним вызван тем, что такие автоматы являются универсальной моделью параллельных вычислений подобно тому, как машины Тьюринга являются универсальной моделью для последовательных вычислений.

Клеточный автомат — дискретная динамическая система, представляющая собой совокупность одинаковых клеток, одинаково соединенных между собой. Все клетки образуют так называемую решетку клеточного автомата. Эти решетки могут быть разных типов и отличаться как по размерности, так и по форме клеток. В настоящей работе каждая клетка — это конечный автомат, состояние которого определяется состояниями соседних клеток и, возможно, ее собственным. В клеточных автоматах, как в моделях вычислений, не рассматриваются входные и выходные воздействия.

При реализации клеточные автоматы обычно называют однородными структурами. В общем случае клеточные автоматы обладают следующими свойствами.

- Изменения значений всех клеток происходят одновременно после вычисления нового состояния каждой клетки решетки.
- Решетка однородна - невозможно различить какие-либо две области решетки по ландшафту.
- Взаимодействия локальны. Лишь клетки окрестности (как правило, соседние) способны повлиять на данную клетку.
- Множество состояний клетки конечно.

Наверное самым известной программой - клеточным автоматом в данный момент является всем известная игра “Жизнь” — клеточный автомат, придуманный английским математиком Джоном Конвеем (John Horton Conway) в

1970 году. Описание этой игры было опубликовано в октябрьском выпуске (1970) журнала Scientific American, в рубрике «Математические игры» Мартина Гарднера (Martin Gardner).

Место действия этой игры — «вселенная» — это размеченная на клетки поверхность. Каждая клетка на этой поверхности может находиться в двух состояниях: быть живой или быть мёртвой. Клетка имеет восемь соседей. Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего по таким правилам:

- пустая (мёртвая) клетка ровно с тремя живыми клетками-соседями оживает;
- если у живой клетки есть две или три живые соседки, то эта клетка продолжает жить; в противном случае (если соседок меньше двух или больше трёх) клетка умирает (от «одиночества» или от «перенаселённости»).

Основная идея «Жизни» состоит в том, чтобы, начав с какого-нибудь простого расположения особей (организмов), расставленных по одной в клетке, проследить за эволюцией исходной позиции под действием «генетических законов» Конуэя, которые управляют рождением, гибелью и выживанием фишек. Конуэй тщательно подбирал свои правила, и долго проверял их «на практике».

2.2 Генетический алгоритм

Генетический алгоритм — это эвристический алгоритм¹ поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием

¹ Эвристический алгоритм (эвристика) — алгоритм решения задачи, включающий практический метод, не являющийся гарантированно точным или оптимальным, но достаточный для решения поставленной задачи. Позволяет ускорить решение задачи в тех случаях, когда точное решение не может быть найдено.

механизмов, аналогичных естественному отбору в природе. Является разновидностью эволюционных вычислений, с помощью которых решаются оптимизационные задачи с использованием методов естественной эволюции, таких как наследование, мутации, отбор и кроссинговер. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе, когда более сильные особи из популяции переживают более слабых и производят следующее поколение особей. В глубоком обучении генетические алгоритмы применяются для задачи оптимизации параметров нейросети.

Естественный отбор:

Процесс естественного отбора начинается с выбора сильных особей из популяции. Их потомство наследует характеристики родителей и является частью следующего поколения особей. Если оба родителя сильные, то их потомство будет сильнее родителей. Это итеративный процесс и завершается, когда найдены наиболее сильные особи. Эта идея применяется для задачи поиска. Рассматривается набор решений для проблемы и отбирается набор лучших из них.

Процесс обучения генетического алгоритма делится на 5 этапов:

1. Начальная популяция
2. Функция силы особи
3. Отбор наиболее сильных решений
4. Обмен характеристиками между двумя особями
5. Мутация
6. Новая итерация с созданием начальной популяции

Процесс начинается с набора особей, который называется популяцией. Каждая особь — это решение проблемы, которая была поставлена. Особь характеризуется набором параметров (переменных), которые называют генами. Гены объединены в одну строку и формируют хромосому — решение задачи.

В генетическом алгоритме набор генов особи представлен в виде бинарной строки. Закодированная комбинация генов называется хромосомой.

Популяция, хромосомы и гены:

Функция силы

Функция силы определяет, насколько сильна отдельная особь. Сила определяется как способность особи конкурировать с остальными особями по заданной метрике. Функция присваивает каждой особи уровень силы. Вероятность того, что особь будет выбрана для производства следующей популяции, основывается на уровне силы особи.

Отбор

Идея отбора заключается в том, чтобы отобрать наиболее сильных особей и передать их гены следующему поколению особей. N пар особей (родители) отбирается на основании их силы.

Скращивание

Скращивание — это основная часть генетического алгоритма. Для каждой пары родителей. Рандомно выбирается точка в бинарной строке хромосомы, до которой особи обмениваются генами. После этого модифицированные особи называются потомством.

Точка скращивания

Потомство создается через процесс обмена генами родителей до случайно заданной позиции в строке. Ниже можно увидеть, как родители обмениваются генами, когда точка скращивания = 3.

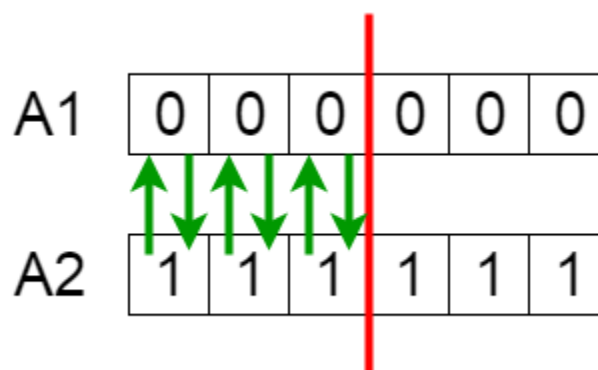


Рис. Точка скрещивания

После обмена генами между родителями потомство добавляется в новую популяцию.

Мутация

Какая-то часть генов будет маловероятной. Чтобы поддерживать разнообразие популяции, отдельно прописывается процесс мутации новых особей.

Before Mutation

A5	1	1	1	0	0	0
----	---	---	---	---	---	---

After Mutation

A5	1	1	0	1	1	0
----	---	---	---	---	---	---

Рис. Пример мутации особи

Завершение алгоритма

Алгоритм завершает работу, когда популяция сошлась, то есть не производит потомство, которое значительно отличается от предыдущего поколения. Когда алгоритм сошелся, на выходе получается набор оптимальных решений заданной проблемы.

2.3 Текстовый интерфейс пользователя

Текстовый пользовательский интерфейс — разновидность интерфейса пользователя, использующая при вводе-выводе и представлении информации исключительно набор буквенно-цифровых символов и символов псевдографики. Характеризуется малой требовательностью к ресурсам аппаратуры ввода-вывода (в частности, памяти) и высокой скоростью отображения информации. Появился на одном из начальных этапов развития вычислительной техники, при развитии возможностей аппаратуры, нацеленной на реализацию появившегося ранее интерфейса командной строки, который, в свою очередь, является наследником использования телетайпов в качестве интерфейса вычислительной техники.

Интерфейс командной строки имеет ряд преимуществ в простоте использования перед графическим интерфейсом, поэтому программы с текстовым интерфейсом создаются и используются по сей день, особенно в специфических сферах и на маломощном оборудовании.

Недостатком подобного типа интерфейса является ограниченность изобразительных средств по причине ограниченности количества символов, включенных в состав шрифта, предоставляемого аппаратурой.

Программы с текстовым интерфейсом могут имитировать оконный интерфейс, чему особенно способствует применение псевдографических символов.

3. ПРАКТИЧЕСКАЯ ЧАСТЬ.

3.1 Общее описание алгоритма

Клеточный автомат представляет из себя топологически поверхность тора, (согласно свойствам клеточных автоматов описанных в 2.1) задан в виде двумерного массива байтов (world). Отображение производится в текстовой консоли символами UTF-16. При запуске программы происходит определение размеров текстовой консоли и предварительное заполнение массива мира “яблоками”. При помощи разработанных для данной программы редакторов выбираются параметры объекта змейка и производится запуск основного цикла программы.

Редактор матрицы поведения Змейки

	0	1	2	3	4	5	6	7	8	9	10
0											
1											
2									-1		
3						-5					
4			-3			-99 -1					
5				-5	-99		-99	-5			
6			-1			-99					
7						-5					
8			-1								
9											
1											

Рис. Пример состояния в редакторе матрицы поведения змейки

С точки зрения модели ООП - каждая особь представляет из себя объект с задаваемой моделью поведения, используя типизированные файлы с “матрицами весов”, на основании которых принимается решение о последующем ходе экземпляра конкретной “змейки”. Таковых две: “веса для препятствий” (определяет вес каждой клетки) и “веса для яблок”.

Для удобного редактирования матриц поведения был разработан *специальный редактор*, и в виде инкапсулированного класса добавлен в программу в виде отдельного пространства имён.

Редактор матрицы поведения Змейки

	0	1	2	3	4	5	6	7	8	9	10
0	1 1	1 1 -5	1 1	1 1	1	1	1	1 1	1 1	1 1	1 1
1	1 1	1 1	1 1	1 1	1	1	1	1 1 1	1 1	1 1	1 1
2	1 1	1 1	1 1	1 1	1	1	1	1 1 3	1 1	1 1	1 1
3	1 1	1 1	1 1	2 2	2	2	2	2 2	1 1	1 1	1 1
4	1	1	1	2 2	2	5 -1	2	2 2	1 1	1	1
5	1	1	1	2	5		5	2	1	1	1
6	1	1	1	2 2	2	5	2	2 2	1	1	1
7	1 1	1 1	1 1	2 2	2	2	2	-2 2 2	1 1	1 1	1 1
8	1 1	1 1	1 1	1 1	1	1	1	1 1	1 1	1 1	1 1
9	1	1 1	1 1	1 1	1	1	1	1 1	1 1	1 1	1 1
10	1 1	1 1	1 1	1 1	1 1	1	1 1	1 1	1 1	1 1	1 1

Рис. Пример состояния в редакторе матрицы поведения змейки

Методы особи предусматривают её: инициализацию, загрузку матриц поведения, выполнение взвешенного хода, изменение длины змейки при встрече с яблоком, удаление змейки при встрече с другой змейкой или фрагментом своего “тела”, производство нового объекта змейки при достижении условной длины (по умолчанию 15 клеток), случайное изменение матриц поведения (мутации), корректное удаление змейки с сохранением её результатов.

Главный цикл подразумевает выполнение хода каждой особи, дополнение мира новыми “яблоками” на каждой итерации. Численность особей - змеек регулируется количеством и плотностью “яблок” в клеточном автомате. Устанавливается равновесие в популяции.

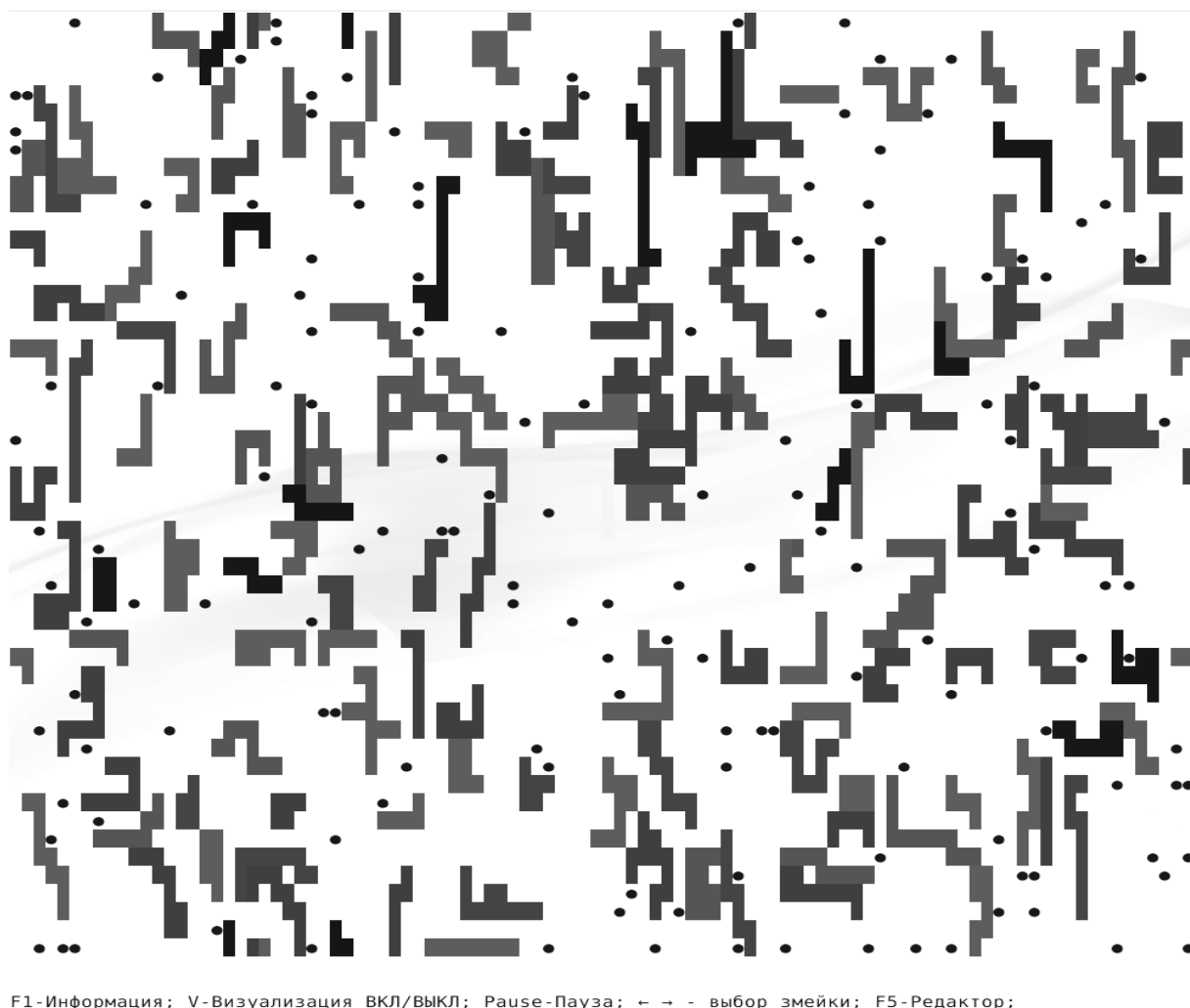


Рис. Пример установившегося равновесия в главном цикле клеточного автомата

Главный цикл клеточного автомата выполняется определенное (100 000 по умолчанию) количество итераций. После чего происходит анализ сортированных списков змеек по количеству “потомков”, и по количеству ходов, пока змейка была активна.

Таблица результатов отбора:				
№	Дата	Возраст	Потомков	Мутаций
1.	255	65	1	1
2.	387	81	0	0
3.	394	185	3	1
4.	416	191	3	1
5.	467	243	3	0
6.	495	382	4	0
7.	567	457	5	1
8.	600	458	4	1
9.	734	554	3	0
10.	824	824	10	0
11.	942	885	13	1
12.	1942	1027	13	5
13.	3634	1384	19	1
14.	19809	1437	20	11
15.	24751	1570	17	13
16.	35200	1740	20	21
17.	38766	1830	15	26
18.	510	345	5	0
19.	586	284	6	1
20.	725	373	7	0
21.	1902	873	14	1

Рис. Таблица результатов отбора особей

Лучшие 20 экземпляров (по-умолчанию) после *отбора* запускаются в чистый клеточный автомат и начинается следующий цикл *отбора*.

Для управления клеточным автоматом был разработан текстовый интерфейс пользователя, в виде подсказок об управляющих клавишах, элементы управления курсором для редактирования матриц поведения особей, их загрузки.

Для некоторого ускорения вычислений в клеточном автомате предусмотрена возможность отключения его визуализации, однако в программе не предусмотрено разделение на потоки, поэтому вычислительные ресурсы не задействованы в полной мере в любом случае.

Опишем основные шаги генетического алгоритма:

Начальная популяция: представляет из себя единственную особь с предварительно настроенными матрицами поведения.

Функция силы особи задаётся совокупностью количества совершённых ходов и произведённых потомков, с последующей сортировкой особей по указанным признакам.

Отбор наиболее сильных решений производится прямо в процессе работы главного цикла в виде структуры данных очередь (на самом деле использовано две очереди, чтобы избежать ненужных сортировок), куда заносятся максимально результативные объекты.

Обмен характеристиками между двумя особями в данной реализации отсутствует.

Мутация задаётся в виде случайного изменения элемента матрицы поведения. Эмпирически установлена 1 мутация в среднем на 10 новых особей.

Новая итерация с созданием начальной популяции, как уже отмечалось выше, организуется путём отбора особей и старта нового главного цикла клеточного автомата с отобранными особями. На самом деле выбранный алгоритм и организация клеточного автомата подразумевают “обнуление” автомата и запуск нового цикла с “сильнейшими” особями на любой его итерации.

Таким образом все основные шаги генетического отбора реализованы.

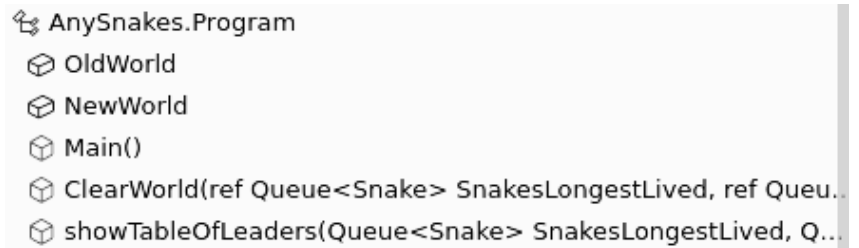
3.2 Организация программы

Точка входа в программу - метод **Main()**, который вызывает инициализацию клеточного автомата описанного полем **NewWorld**.

Поле **OldWorld** используется для визуализации. С целью оптимизации взаимодействия с программой-терминалом (консолью) рассчитываются “на лету”, в процессе формирования очередного кадра с состоянием клеточного автомата

только лишь изменяемые позиции курсора путем сравнения полей *NewWorld* и *OldWorld*.

Метод *ClearWorld()*
используется для переноса
отобранных особей в
новый цикл.



Метод
showTableOfLeaders() - выводит в консоль перечень отобранных особей.

В главном цикле автомата существование всех особей происходит в очередях.

Коллекция активных особей:

List<Snake> Snakes

Коллекция особей считающиеся **погибшими** в главном цикле:

List<Snake> DiedSnakesCollector

Коллекция особей, ГОТОВЫХ К **размножению**:

List<Snake> CanBornSnakesCollector

Коллекция самых **долгоживущих** особей:

Queue<Snake> SnakesLongestLived

Коллекция особей с самым большим количеством **потомков**:

Queue<Snake> SnakesMostProlific

Выбранный подход позволяет наиболее оптимально обращаться с памятью, задействованной в автомате. Эффективно использовать сборщик мусора платформы .NET и избежать дополнительных сортировок. Весь отбор происходит “налету” в процессе работы клеточного автомата.

Класс Snake описывает поведение и качества особи, включает в себя следующие методы и поля:

Name - поле, в котором формируется имя особи из мутаций её матриц относительно оригинальной матрицы особи общего предка.

MatrixOfFear, MatrixOfWish - поля, содержащие многомерные массивы, определяющие вес ближайших клеток, как цели для особи.

Body - поле типа Очередь (queue) из кортежа координат, определяющих положение каждого участка особи в клеточном автомате. Структура данных очередь использована обоснованно: первой приходит голова особи, она же первой и выходит в автомат при расчете нового хода.

ThinkedMove - поле, кортеж из координат в клеточном автомате, для следующего хода.

isAlive - поле булевого типа, определяющее активность особи.

Age - целочисленное поле, содержащее возраст особи в ходах.

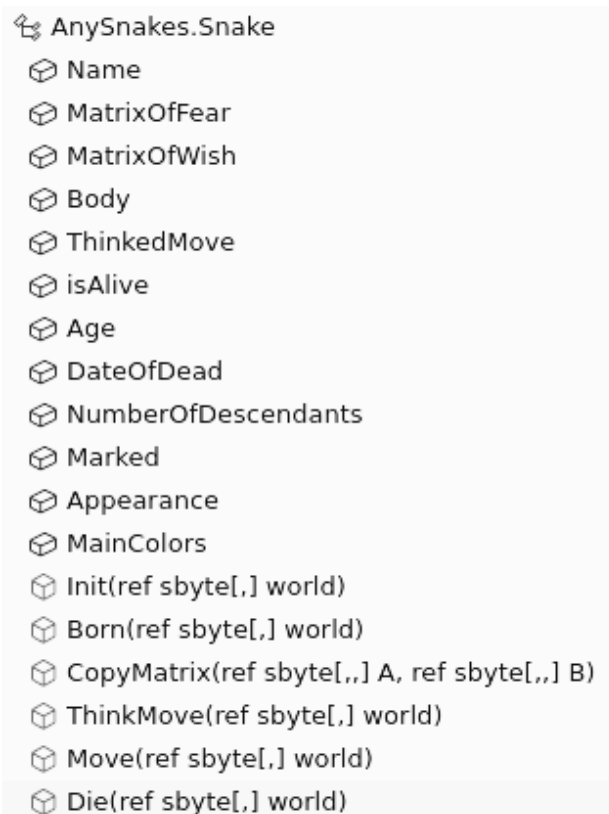
DateOfDead - целочисленное поле, ход в главном цикле, когда особь потеряла активность.

NumberOfDescendants - целочисленное поле - количество потомков.

MainColors - поле enum, перечисление всех цветов, используемых для отображения особей в автомате.

Appearance - поле, выбираемое случайно из MainColors, в момент появления особи с мутацией. Служит исключительно для визуализации.

Init - метод, инициализация новой змейки с координатами головы.



```
AnySnakes.Snake
  Name
  MatrixOfFear
  MatrixOfWish
  Body
  ThinkedMove
  isAlive
  Age
  DateOfDead
  NumberOfDescendants
  Marked
  Appearance
  MainColors
  Init(ref sbyte[,] world)
  Born(ref sbyte[,] world)
  CopyMatrix(ref sbyte[,] A, ref sbyte[,] B)
  ThinkMove(ref sbyte[,] world)
  Move(ref sbyte[,] world)
  Die(ref sbyte[,] world)
```

Born - метод, создание объекта новой Змейки от текущего предка, мутация при копировании данных. Данный метод порождает новый объект класса **snake**.

CopyMatrix - Копирование многомерного массива из А(источник) в В(целевой)

ThinkMove - метод, вычисление хода в клеточном автомате по текущему положению.

Move - метод, выполнение хода змейки, по полю **thoughtMove**.

Die - метод, уборка “мусора” в клеточном автомате, фиксация значений в соответствующих полях

Класс Apples - управляет положением целей - “яблок” для особей “змеек”.

```
AnySnakes.Apples
  Calculate(ref sbyte[,] world)
  Add(ref sbyte[,] world, int count = 1)
  Create(ref sbyte[,] world, int maximum)
```

Create, Add - методы, заполнение целями с случайным положением в текущем состоянии клеточного автомата

Calculate - метод, подсчёт текущего количества целей - “яблок” в автомате.

Класс **SnakeEditor** - класс, предоставляющий текстовый пользовательский интерфейс для управления параметрами особей.

```
SnakeEditors.SnakeEditor
  initListEdit(Queue<Snake> SnakesLongestLived, Queue<Snake> SnakesMostProlific)
  showTable(ref List<Snake> Snakes, int activeInput)
  Loop(Queue<Snake> SnakesLongestLived, Queue<Snake> SnakesMostProlific)
  showTable(ref Snake W, ref Dictionary<string, string> Sheet, int activeInput)
  LoadCaptions(ref Snake W, out Dictionary<string, string> Sheet)
  Change(ref Dictionary<string, string> Sheet, string Key, ref Snake snake)
  Loop(ref Snake W)
  SelectFile(string FileExt, bool write = true)
  Load(string FileName, ref Snake snake)
  Save(string FileName, ref Snake snake)
```

Класс инкапсулирован, для передачи управления программой достаточно вызвать метод `loop` (главный цикл текстового пользовательского интерфейса). Так

как изменяются важные параметры - продолжать выполнение главного цикла основной программы клеточного автомата не имеет смысла. Управление выполнением полностью передаётся интерфейсу. Подробное рассмотрение текстового интерфейса выходит за рамки целей данной работы, однако исходный код можно увидеть в приложении.

Класс *MatrixEditor* концептуально организован как и SnakeEditor. Инкапсулирован. Передача управления происходит так же - вызовом метода loop. Предоставляет возможность редактирования матриц поведения особей в ручном режиме. Загружать и сохранять сериализованные файлы с матрицами.

```
SnakeEditors.MatrixEditor
  showTable(sbyte[,] table, int i = 11, int j = 11, int k = 4)
  Loop(ref sbyte[,] table, int i = 11, int j = 11)
  Edit(sbyte[,] table, int i, int j)
  Save(sbyte[,] table, string FileName)
  Load(ref sbyte[,] table, string FileName)
```

4. ЗАКЛЮЧЕНИЕ

В результате проведённых опытов с генетическим отбором в клеточном автомате были выявлены следующие особенности: генетический отбор очень чувствителен к качеству начальных настроек особи. Кроме того в результате наблюдений, выявлено получение мутаций, заметно ухудшающие результативность особей вплоть до полного вымирания популяции. Таким образом длительный отбор речь идёт о большом количестве главных циклов (эпох), не делает поведение особи более результативным, а приводит её к некоторому усреднённому значению, достаточному для параметров отбора. Начинать отбор, заведомо имея эффективное решение задачи, - весьма сомнительная стратегия, т.к. алгоритм требует значительных вычислительных мощностей (а как следствие затраты энергии и времени) даже для решения простых задач.

В результате выполнения данной работы был получен опыт использования клеточных автоматов, проработаны основные моменты связанные с разработкой и применением алгоритма эвристического поиска. Был получен опыт работы с текстовыми пользовательскими интерфейсами и клеточными автоматами.

Все результаты, несомненно будут использованы в дальнейших реализациях задач.

5. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Варшавский В.И., Мараховский В.Б., Песчанский В.А., Розенблюм Л.Я. Однородные структуры. М., 1973.
2. Марков А. А., Нагорный Н. М., Теория алгоритмов, М.: Наука, 1984.
3. Парамонов С., “Играем с генетическими алгоритмами”, Хабр, 2014
4. Тоффоли Т., Марголус Н., Машины клеточных автоматов, М.: Мир, 1991
5. Хляпов А., “Поиграем в Жизнь”, <https://habr.com/ru/post/63848/> , Хабр, 2009
6. Шалыто А., Туккель Н. От тьюрингова программирования к автоматному, Мир ПК. 2002. №2.
7. Шилдт Г. - С# 4.0. Полное руководство, Виллиамс, 2015
8. Яшин А., “Жизнь на частицах”, <https://habr.com/ru/post/442128/> , Хабр, 2019
9. E. Alba, B. Dorronsoro, Cellular Genetic Algorithms, Springer, 2008