**Министерство науки и высшего образования**

**Российской Федерации**

**ФЕДЕРАЛЬНОЕ Государственное Автономное**

**образовательное учреждение высшего образования**

**«новосибирский национальный исследовательский государственный университет»**

**Факультет информационных технологий**

Кафедра параллельных вычислений

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) Программная инженерия и компьютерные науки

**ОТЧЕТ**

**о прохождении учебной практики, научно-исследовательской работы (получение первичных навыков научно-исследовательской работы)**

(указывается наименование практики)

**Обучающегося** Трубицыной Юлии Сергеевны **группы №** 19202 **курса 3**

(Ф.И.О. полностью)

**Тема задания**: Реализация двумерного клеточного автомата, покрывающего область паттерном домино.

**Место прохождения практики:** Федеральное государственное бюджетное учреждение науки Институт вычислительной математики и математической геофизики Сибирского отделения Российской академии наук, 630090, г. Новосибирск, проспект Академика Лаврентьева, 6

**Сроки прохождения практики:** с 06.10.2021 г. по 22.12.2021 г.

**Руководитель практики   
от профильной организации** Киреев С.Е., н.с. ИВМиМГ СО РАН \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Руководитель практики от НГУ** Маркова В.П., доц. каф. ПВ ФИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Руководитель ВКР** Малышкин Виктор Эммануилович зав. каф. ПВ ФИТ, д.т.н., профессор

**Оценка по итогам защиты отчета:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(неудовлетворительно, удовлетворительно, хорошо, отлично)

**Отчет заслушан на заседании кафедры** **параллельных вычислений**

**протокол \_\_\_\_\_\_\_\_\_от** «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_г.

Новосибирск 2021

**Содержание**

[Введение 3](#_Toc90920384)

[1. Теоретическая часть 4](#_Toc90920385)

[2. Практическая часть 6](#_Toc90920386)

[2.1 Описание реализации 6](#_Toc90920387)

[2.2 Тестирование программы 7](#_Toc90920388)

[Заключение 8](#_Toc90920389)

[Список литературы 9](#_Toc90920390)

# **Введение**

Практика проходила в Лаборатории синтеза параллельных программ Института вычислительной математики и математической геофизики Сибирского отделения РАН (ИВМиМГ СО РАН).

Одним из научных направлений лаборатории является разработка клеточно-автоматных моделей для исследования различных процессов. Область клеточно-атомарного моделирования является очень важной для изучения сложных систем в различных научных сферах. Главная особенность таких моделей заключается в локальности взаимодействий, т.е. изменение состояния клетки зависит не только от ее текущего состояния, но и от состояния соседних клеток. Именно эта особенность делает клеточно-автоматную модель мощным инструментом для исследования поведения сложных систем, а также позволяет распараллелить программную реализацию такой модели.

Целью моей работы была реализация алгоритма покрытия поля плитками домино, описанного в статьях [2-3].

Также в начале работы были поставлены следующие задачи:

1. Ознакомиться с терминологией по теме клеточных автоматов.
2. Ознакомиться с текстами статей о минимальном и максимально покрытии поля плитками домино.
3. Написать программу на основе алгоритма, предложенного в статьях.
4. Протестировать полученную программу.

# **1. Теоретическая часть**

Задача заключается в том, чтобы покрыть все поле плитками домино. Для работы мне понадобилось изучить некоторые материалы по теме клеточно-автоматного моделирования. Следует обозначить несколько важных определений:

Клеточный автомат – это множество одинаковых простых вычислителей, которые в модели представлены парами (u, x), называемыми клетками, где u ∈ A – состояние клетки из алфавита A, x ∈ X – имя клетки, часто задаваемое вектором x = (i, j, k) из конечного множества координат d-мерного дискретного пространства X [1, с. 2]. Состояния клеток изменяются с помощью функции перехода [1, c. 2]

Следующим важным понятием является итерация [1, c. 3]. Итерация может выполняться разными способами, которые называются режимами функционирования. Основными являются синхронный и асинхронный режимы. В алгоритме, представленном ниже, используется именно асинхронный режим функционирования.

Для описания алгоритма решения поставленной задачи также важно ввести следующие определения:

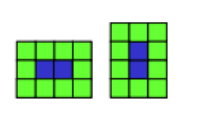
**** Плитка домино - двумерный массив, состоящий из ядра и оболочки размером 3 х 4. Оболочка помечается 0 (зеленый), а ядро 1 (синий). (см. Рис.1)

Рис.1. Плитка домино. Первая плитка домино расположена горизонтально, вторая – вертикально.

Метка – выделенная клетка плитки домино (см. Рис. 2), которая используется при прикладывании плитки домино к окну для последующего сравнения (см. Рис 3).

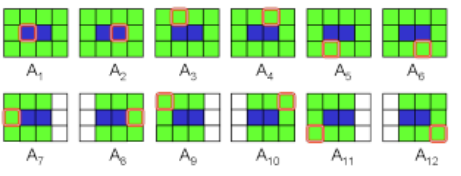
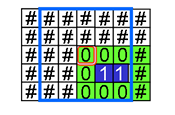


Рис.2. Представление плиток домино с метками (красный квадрат). Всего представлено 12 плиток, т.к. остальные можно получить поворотом на 90 градусов, как показано на Рис.1.

Окно – двумерный массив размером 5 х 5, где центром является клетка поля, которая была выбрана случайным образом (см. Рис. 3).

Рис.3. На рисунке представлен процесс сравнения окна (синий квадрат) и одной из плиток домино. Плитка домино накладывается на окно так, что метка сопоставляется с его центром, и далее происходит сравнение.

**2. Практическая часть**

# **2.1 Описание реализации**

Ниже представлен алгоритм одной итерации [1, c. 2].

1. **function** ITER (pointsInField)
2. **for** 0 **to** pointsInField **do**
3. *x* ← *rnd*()
4. *y* ← *rnd*()
5. *centerWindow* ← *field*[*x*][*y*]
6. *window* ← *makeWindow*(*centerWindow*)
7. **for each** *dominoTile* **in** *arrayDominoTile* **do**
8. **if** (*pointMatchToDominoTile*[[1]](#footnote-1)(*window*, *centerWindow*, *dominoTile*) = *true* & *isHull*(*centerWindow*) = *true*) **then**
9. *hits* ← *hits* + 1
10. **end if**
11. **if** (*pointMatchToDominoTile*(*window*, *centerWindow*, *dominoTile*) = *true* & *isCore*(*centerWindow*) = *true*) **then**
12. *hits* ← 100
13. **end if**
14. **end for**
15. **if** (*hits* > 0 & *hits* ≠ 100) **then**
16. *state* ← 0
17. *field*[*x*][*y*] ← *state*
18. **else** **if** (*hits* = 100) **then**
19. *state* ← 1
20. *field*[*x*][*y*] ← *state*
21. **else if** (*rnd*() < *probability*[[2]](#footnote-2)) **then**
22. *state* ← *rnd*()
23. **end if**
24. **end for**
25. **end function**

Алгоритм повторяется до тех пор, пока поле не покроется плитками домино.

В моей работе используется большое количество двумерных массивов, поэтому для реализации алгоритма был выбран язык C++, т.к. он позволяет оптимизировать программы на низком уровне, что потребуется в ходе дальнейшей работы.

На вход программы подается размер поля (ширина и длина), а также вероятность, обозначенная в алгоритме как *probability*.

В программе поле представлено классом *Field*, где находится массив клеток. Клетка описана классом *Cell*. Он имеет поля *state* (состояние клетки), *hits* (число попаданий клетки в плитку домино) и *coord* (координаты клетки на поле). На начальном этапе исполнения программы происходит инициализация поля методом *init\_field* следующим образом: поле заполняется случайный образом значениями 0 и 1, а также к нему добавляются границы (в выводе программы значение представлено “#”). После происходит создание 12 плиток домино c помощью метода *fill\_template*, расположенных по горизонтали. Плитки домино находятся в массиве. Каждый элемент такого массива является объектом класса *Template*. Полями этого класса являются *single\_template* (плитка домино) и *center* (метка).

Основной цикл программы представлен в классе *Runnable*. В методе evolve находятся 2 цикла, описывающие эволюцию [2-3] поля. Он использует несколько вспомогательных методов, таких как *fill\_window*, *count\_hits* и *change\_state.* В первый метод подаются случайным образом сгенерированные координаты клетки, на основе которых он заполняет окно. Для подсчета попаданий точки в плиту домино используется метод *count\_hits*. В нем происходит сравнение плиток с помощью метода *compare\_with\_template*, расположенных как горизонтально, и после метод возвращает количество попаданий клетки в плитки домино. На основе значения, которое вернул метод *count\_hits*, метод *change\_state* меняет состояние клетки в соответствии с алгоритмом.

# **2.2** **Тестирование программы**

Последним этапом стала проверка программы на корректность работы и ее тестирования на поле 7 x 7 с разными начальными данными. Пример входных и выходных данных приведен ниже:

|  |  |
| --- | --- |
| Рис.4 Входные данные поля. | Рис.5. Выходные данные поля. |

Во время тестирования программы было замечено, что поведение эволюции [2 - 3] в зависимости от вероятности очень похоже на поведение, описанное в предложенных на изучение статьях [3, c. 8].

# **Заключение**

За время прохождения практики были выполнены все поставленные задачи. Итогом стала программа, которая покрывает все поле плитками домино. Она была протестирована и проверена на корректность.

Во время реализации данной программы я встретилась с некоторыми трудностями. Основная из них заключалась в большом объеме работы с двумерными массивами, из-за чего отлаживание программы занимало большую часть времени.

Практика помогла мне более детально изучить предметную область исследования и дала стимул для дальнейшей работы над данной темой.

# **Список литературы**

1. [О.Л.Бандман. Клеточно-автоматное моделирование процесса просачивания жидкости через пористый материал // Параллельные вычислительные технологии-2013, Сборник трудов Международной научной конференции ПаВТ-2013, ЮрГУ, 1–5 апреля 2013 г., г. Челябинск . – С. 278-287](http://ssd1.sscc.ru:9780/bandman/PaVT-2013.pdf)
2. Hoffmann, R., Désérable, D., Seredyński, F. A cellular automata rule placing a maximal number of dominoes in the square and diamond // *J Supercomput* **77,**9069–9087 (2021).
3. Hoffmann R., Désérable D., Seredyński F. (2021) Minimal Covering of the Space by Domino Tiles. In: Malyshkin V. (eds) Parallel Computing Technologies. PaCT 2021. Lecture Notes in Computer Science, vol 12942. Springer, Cham. DOI: 10.1007/978-3-030-86359-3\_35

1. Функция сопоставления плиток домино с соседними клетками центра окна. Функция возвращает истину, если центр окна и его соседние точки совпали с плиткой домино (сопоставление плитки домино и окна показано на рис. 3). [↑](#footnote-ref-1)
2. Вероятность нужна для продвижения эволюции [2 - 3], если не произошло попадание центра окна в плитку домино. [↑](#footnote-ref-2)