МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ /В. І. Шинкаренко/

«18» грудня 2020 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Моделювання бієктивних відображень фракталів різної природи**

Theme **Modeling of bijective mappings of fractals of different origin nature**

Керівник дипломної роботи проф. _____ В. І. Шинкаренко

Нормоконтролер доц. _____ О. С. Куроп'ятник

Студент групи ПЗ1921 _____ Р. Р. Чигір

Student Chyhir Robert

Дніпро – 2020

Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

Факультет Комп'ютерних технологій і систем кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ проф. Шинкаренко В.І.
(підпис)
«18» грудня 2020 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС Магістр
(освітній ступень)

студента групи (ПЗ1921) 961-М Чигіра Роберта Романовича
(номер групи) (ПІБ)

1 Тема дипломної роботи: Моделювання бієктивних відображень фракталів різної природи затверджена наказом по університету від «10» жовтня 2019 р. № 779 ст.

2 Термін подання студентом закінченого проекту «16» грудня 2020 р.

3 Вихідні дані до дипломного проекту _____

4 Зміст пояснювальної записки (перелік питань до розробки) проведення дослідження наявності фрактальних властивостей бієктивних відображеннях фракталів, моделювання конструкторів конструктивно-продуційних граматик, розробка інструментарію для створення продуційних конструкторів.

5 Перелік демонстраційного матеріалу презентація на тему моделювання часових рядів та графічних зображень L-систем на основі конструкторів конструктивно-продуційних граматик; відео демонстрація роботи розробленого програмного інструментарію для проведення досліджень.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	доц. Гненний М.В.	30.10.20	02.12.20
Охорона праці та безпека в надзвичайних ситуаціях	ст. викладач Музикін М.І.	26.11.20	14.12.20

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва розділів дипломного проекту	Термін виконання розділів проекту (роботи)	Примітка
1	Вступ	15.11.2019 – 14.12.2019	
2	Огляд літератури	16.12.2019 – 27.12.2019	
3	Постановка задачі, технічне завдання	03.01.2020 – 07.02.2020	
4	Створення тестової програми	10.02.2020 – 02.10.2020	
5	Перші тестування	05.10.2020 – 23.10.2020	30%
6	Аналіз результатів	26.10.2020 – 30.10.2020	
7	Розрахунок економічних показників	02.11.2020 – 06.11.2020	
8	Охорона праці	09.11.2020 – 13.11.2020	60%
9	Оформлення пояснювальної записки	16.11.2020 – 07.12.2020	
10	Демонстраційні матеріали	08.12.2020 – 15.12.2020	100%

Дата видачі завдання «10» жовтня 2019 р.

Керівник дипломного проекту _____ (підпис) Шинкаренко В. І. (ПІБ)

Завдання прийняв до виконання _____ (підпис) Чигір Р. Р. (ПІБ)

РЕФЕРАТ

Об’єктом дослідження є моделювання фрактальних числових рядів та їх графічних відображень на основі конструктивно-продуційних граматик.

Предметом дослідження є порівняння фрактальних бієкцій граматик.

Метою даної роботи є розробка інструментарію для створення конструкторів конструктивно-продуційних граматик та моделювання на їх основі числових рядів.

Методи дослідження: аналіз різних відображень класичних фракталів з порівнянням отриманих даних для подальшого дослідження.

Результати та їх новизна: унікальність програмного продукту для створення конструкторів конструктивно-продуційних граматик.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та 4 додатків.

Вступ – описується сутність роботи та її актуальність (4 сторінки).

Першому розділ – аналіз проблеми та огляд інструментів (14 сторінок).

Другий розділ – висвітлено обґрунтування напрямку дослідження(15 сторінок).

Третій розділ – описано процес проектування і розробки інструментального програмного забезпечення для досліджень (11 сторінок).

Четвертому розділ – дослідження моделювання відображень конструктивно-продуційних граматик (16 сторінок).

П’ятий розділ – розкриті питання охорони та безпеки праці в надзвичайних ситуаціях (7 сторінок).

Додатки – технічне завдання, робочий проект, 6 доповідей, 2 тези, 3 свідоцтва про авторське право, 2 звіти про науково-дослідницьку роботу.

Таблиць – 13 , рисунків – 39, діаграм – 4, бібліографія – 70 джерел.

Ключові слова: конструктивно-продукційна структура; конструктори; числові ряди; фрактали; L-система

ЗМІСТ

ВСТУП

Споконвіку людину оточують різні природні явища та процеси. Серцебиття, блискавки чи коливання хвиль – це прояви процесів, чиї фрактальні властивості [1] здебільш можливо побачити лише через самоподібність [2] в їх графічних відображеннях. Адже багато закономірностей, що лежать в основі фрактальності цих фігур до сих пір не були уніфіковані в об’єктні моделі.

Тому можливість створювати моделі на основі явищ дає змогу заглибитися у процес аналізу, поглибити знання з цих процесів, що у довготривалому процесі вивчення та дослідження дозволяє передбачати послідовність розвинення цих явищ.

Одним з методів дослідження є порівняння об’єктів та їх даних, а порівняння моделей, розроблених на одних і тих самих даних (тобто, бієктивні відображення) [3], дозволяють помітити риси, що притаманні описаним процесам, не залежно від реалізації відображення.

Актуальність роботи. Увесь світ можливо представити за допомогою об’єктних моделей. Описуючи їх атрибути можливо виявити закономірності в їх структурі. Одним з підходів до опису явищ є опис використовуючи конструктивно-продуційні граматики [4].

Цей підхід, що набув відомість у останні 10 років, за допомогою означення чотирьох складових [5, 6, 7] – спеціалізації, інтерпретації, конкретизації та реалізації – дозволяє описувати об’єкти та явища не тільки з вираженими фрактальними характеристиками, а і з часозалежним розгортуванням.

Дослідження конструктивно-продуційного підходу розглядалися у використані дослідження фракталів, часових рядів та природних процесів, таких як створення блискавок у статичному та динамічному грозовому фронті, що ставить дослідження у даному напрямі актуальними.

На основі підходу були розроблені програмні додатки, що розглядаються проблеми аналізу та дослідження погодних явищ, що представляє приклади розгляду дослідження різних форм представлення даних граматики.

Порівнюючи результати продукції одних і тих самих граматик з бієктивним зображенням результатів можливо припускати о фрактальних властивостях у різних форм зображень.

Також, існування фрактальних властивостей у різних відображень може дозволити продовжити дослідження у створені більшої кількості форм відображень, що можуть мати теж властивості фракталів за транзитивністю між формами.

Об'єкт дослідження. Об'єктом дослідження є моделювання фрактальних зображень на основі розгортання конструкторів конструктивно-продуційних граматик, що відображають класичні фрактали.

Предмет дослідження. Предметом дослідження є уніфікація процесу опису конструкторів конструктивно-продуційних граматик.

Мета і завдання дослідження. Метою даної роботи є створення програмного забезпечення для формування конструкторів конструктивно-продуційних граматик, а саме:

- створення простих та алгоритмічних конструкторів;
- наповнення конструкторів інформацією о спеціалізації, інтерпретації та конкретизації;
- створення реалізації конструкторів на основі їх спеціалізації, інтерпретації та конкретизації;
- задання алгоритмів інтерпретування на основі процесу програмування;
- аналіз результатів моделювання часових рядів та графічних фракталів;
- виявлення фрактальних характеристик у бієктивних відображеннях;
- моделювання фракталів та об'єктів природи на основі граматик.

Наукова новизна. Задання алгоритмів інтерпретування на основі процесу опису алгоритмів використовуючи актуальну форму алгоритмічних мов програмування.

Унікальність методу представлення об'єктів і явищ та інноваційність способу конструювання конструкторів конструктивно-продуційних граматик.

Уніфікація процесу створення та опису продуційних конструкторів за допомогою уніфікуючого програмного забезпечення

Практичне значення. Дослідження використання конструктивно-продукційного моделювання для роботи з фракталами увійшли у звіти з науково-дослідницьких робіт:

- «Конструктивно-продуційне моделювання фракталів», № держреєстрації 0118U004215;
- «Підвищення конкурентоспроможності залізничного транспорту на основі уніфікованих інтелектуальних технологій процесі перевезень та експлуатації парків технічних систем», № держреєстрації 0117U004392.

Апробація результатів дослідження. Процес та результати робити над даним дослідженням доповідалися на засіданнях наукових семінарів кафедри КІТ.

Також були надані доповіді на наступні конференції:

- Міжнародна науково-практична конференція «Інформаційні технології в металургії та машинобудуванні» (Дніпро, 2017 р.);
- XIV міжнародна конференція TAAPSD`2017 (Київ, 2017 р.);
- Всеукраїнська конференція «Проблеми математичного моделювання» (Кам'янське, 2018 р.);
- Міжнародна конференція ISDMCI'2018 (Херсон, 2018 р.);
- XII міжнародна конференція «Современные информационные и коммуникационные технологии на транспорте, в промышленности и образовании» (Дніпро, 2018 р.);
- XIV міжнародна конференція CSIT 2019 (Львів, 2019 р.);
- XV міжнародна конференція CSIT 2020 (Львів, 2020 р.).

Публікації за темою роботи. Публікація за темою була представлена та включена до міжнародного журналу «Information Technologies & Knowledge» том 13, №1 [8].

Також були опубліковані праці у збірниках до наступних конференцій:

- «Варіативність уточнюючих перетворень конструктивно-продукційного моделювання», XIV міжнародна конференція TAAPSD`2017 [9];

- «Конструктивно-продукційне моделювання фракталів», Міжнародна конференція ISDMCI’2018 [10];
- «Конструктивно-продукційне моделювання фрактальних часових рядів на основі L-систем», Всеукраїнська науково-методична конференція «Проблеми математичного моделювання», 2018 рік [11];
- «Modeling of Lightning Flashed in Thunderstorm Front by Constructive Production of Fractal Time Series», Міжнародна науково-технічна конференція CSIT 2019 [12].

Були отримані рішення та свідоцтва на реєстрацію авторського права на наступні комп‘ютерні програми:

- «Моделювання часових рядів із заданими фрактальними властивостями»;
- «Рекурентний аналіз часових рядів породжуваних L-системами»;
- «Моделювання взаємопов‘язаних часових рядів й геометричних фракталів породжуваних в L-системах».

1 АНАЛІЗ ПРОБЛЕМИ ВИЯВЛЕННЯ ФРАКТАЛЬНИХ ВЛАСТИВОСТЕЙ ОБ’ЄКТІВ ТА ЯВИЩ

1.1 Призначення та область застосування

У результаті дослідження має бути створений програмний додаток, що дозволяє на основі конструктивно-продуційного моделювання створювати та описувати інформацію конструктивно-продуційних граматик та створювати графічне відображення описаного процесу.

Моделювання на основі конструктивно-продукційного підходу вже було застосовано при моделюванні об'єктів та процесів різної природи [4, 13], таких як часові ряди, фрактали, блискавки чи грозовий фронт [12].

Застосовуючи даних підхід, можливо моделювати різноманітні процеси з подальшим прогнозуванням наступних дій, внесенням ймовірностей [14] для урізноманітнення можливого просування процесу чи моделювати об’єкти, що представлені у навколишньому світі.

1.2 Формулювання завдання та його складових

Результатом даної роботи повинен бути програмний додаток для уніфікації процесу моделювання на основі конструктивно-продукційного підходу.

Програмний продукт повинен мати наступний функціонал:

- створення простих конструкторів та конструкторів алгоритмів конструктивно-продуційних граматик;
- заповнення інформації о спеціалізації, інтерпретації та конкретизації граматики;
- створення графічного відображення реалізації граматики;
- вивід інформації о значеннях атрибутів після розгорнення реалізації.

1.3 Аналіз існуючих програмних рішень

На момент проведення дослідження при створенні програмного забезпечення для уніфікації створення конструкторів для дослідження фрактальних властивостей часових рядів [15] та L-систем [16] вже були розроблені програмні системи, що дозволяють проводити дослідження.

1.3.1 Огляд програмного додатку «PyTime»

Додаток «PyTime» дозволяю створювати часові ряди, графічні фрактали та будувати графіки рекурентного аналізу та математичного очікування рядів, отриманих на основі інтерпретації розгорненої кінцевої стрічки, отриманої на основі конструктивно-продуційної граматики.

Додаток потребує заповнення інформації о:

- аксиомі (початковій символьній стрічці) [17];
- правилах підстановки;
- математичному очікуванні та дисперсії рівня ряду часового ряду та довжини лінії графічного фракталу;
- математичному очікуванні та дисперсії періоду часового ряду та куту нахилу лінії графічного фракталу;
- числі ітерацій розгорнення чи числі точок ряду;
- значень рекурентного аналізу [18].

Використовуючи описану інформацію, програмний додаток виконує побудову графіку часового ряду, графіку значень математичного очікування [19] кожної отриманої точки ряду, графіку фазового простору [20], рекурентну діаграму [21] та графічний фрактал [22] на основі L-системи.

Користувач може обрати яким чином будуть змінюватися значення параметрів при розгортанні – від початкових введених значень чи будь розраховуватися кожного разу від поточних значень атрибутів.

Програмний додаток має предвстановлені приклади класичних фракталів, що дозволяє пришвидшити розгортання досліджень.

Користувач додатку може зберегти результати моделювання у форматі файлу «docx» та «xlsx» зі значеннями граматики та графічні значення графіків «png».

Інтерфейс програмного забезпечення на прикладі класичного фракталу «Сніжинка Коху» наведені на рисунках 1.1 та 1.2.

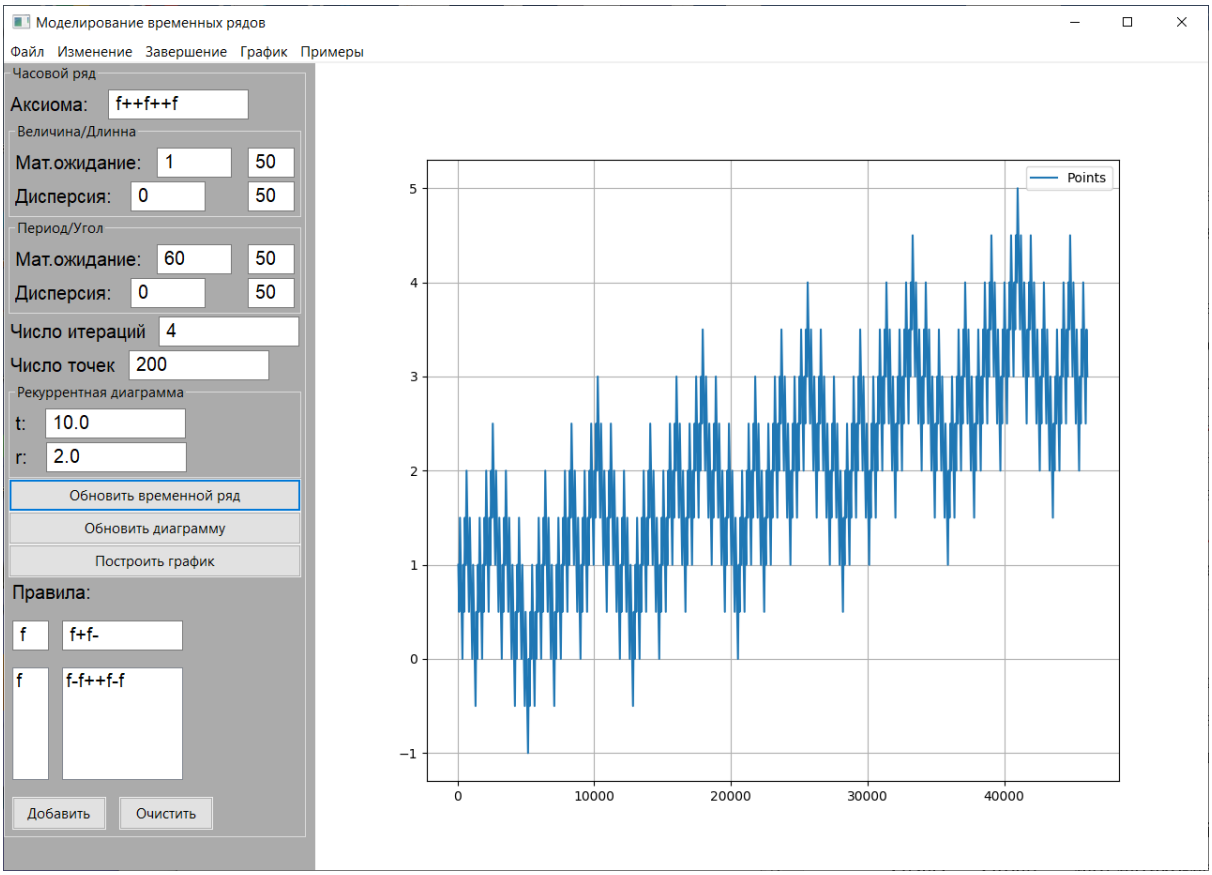


Рисунок 1.1 – Вікно з часовим рядом на основі «Сніжинки Коха»

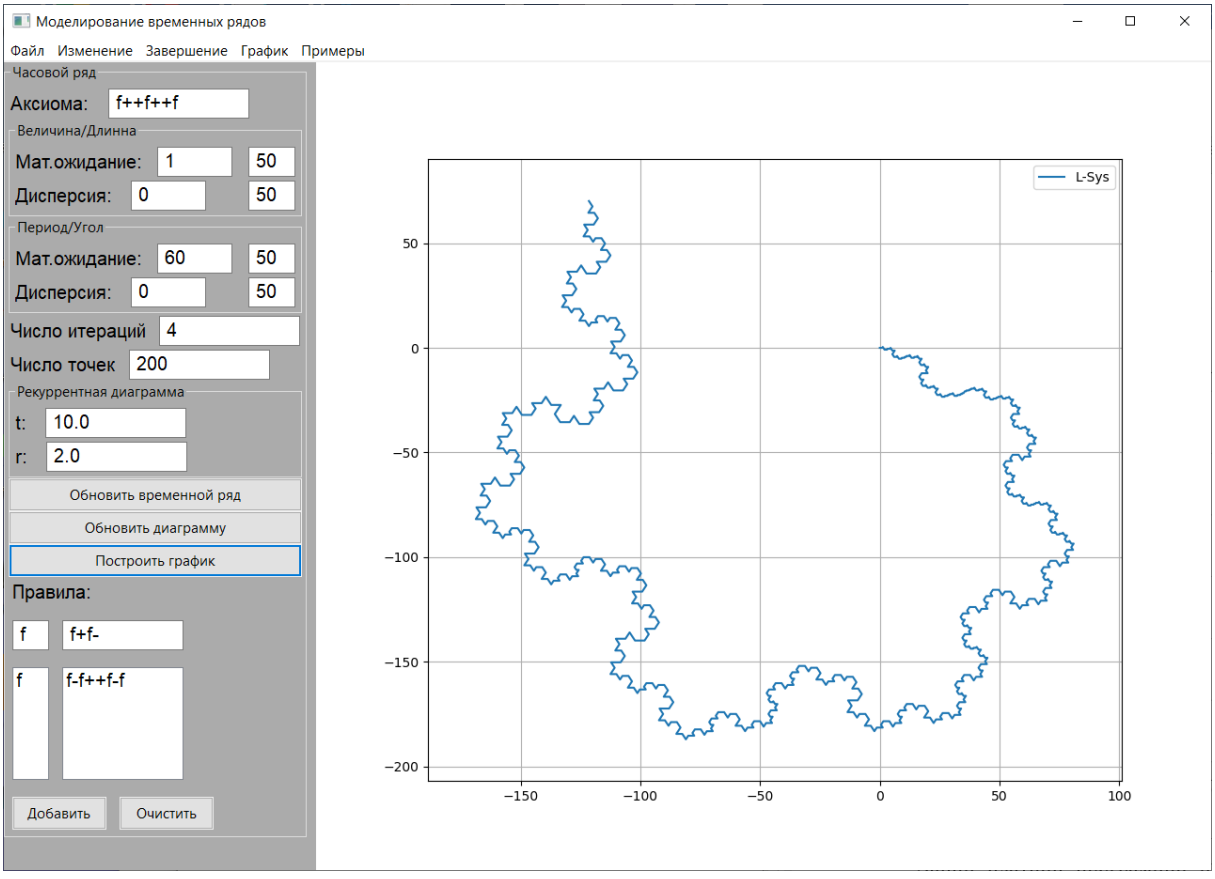


Рисунок 1.2 – Вікно з графічним фракталом на основі «Сніжинки Коха»

1.3.2 Огляд програмного додатку «PyLight»

Програмний додаток є модифікацією програмного забезпечення «PyTime» для дослідження процесів утворення блискавок у грозовому фронті.

Даний програмна система створює часові ряди для положення блискавки та її сили, які у подальшому можуть бути розглянуті з файлів логування інформації та трансльовані у графічне динамічне відображення.

Програма складається з двох вікон:

– вікна опису частин представлення моделі блискавки – положення блискавки уздовж грозового фронту, положення блискавки від грозового фронту та сила одиничної блискавки;
– вікна моделювання процесу утворення блискавок у грозовому фронті.

У другому вікні користувач розставляє точки кривих Безьє [23] та запускає процес відображення блискавок.

В даному програмного додатку є предвстановлена кількість кривих грозового фронту що дорівнює трьом. Кожна крива Безьє, що відображає частину цілого грозового фронту, має початкове положення та кінцеве положення, до якого з рівномірним шагом наближається крива від початкового положення.

Дані параметри фактично описують життєвий цикл грозового фронту – його створення, просування, створення блискавок, що починаються від маленької точки та поширюють світло до певного радіусу, й закінчення шляху просування.

Процес можна зафіксувати, що позволить проаналізувати ділянки грозового фронту на максимальну активність та поширення протягом часу.

Даний програмний додаток використовується лише при дослідженні блискавок та їх активності.

Інтерфейс програмного забезпечення, що демонструє генерування інформації та процес моделювання життєвого циклу грозового фронту наведені на рисунках 1.3 та 1.4.

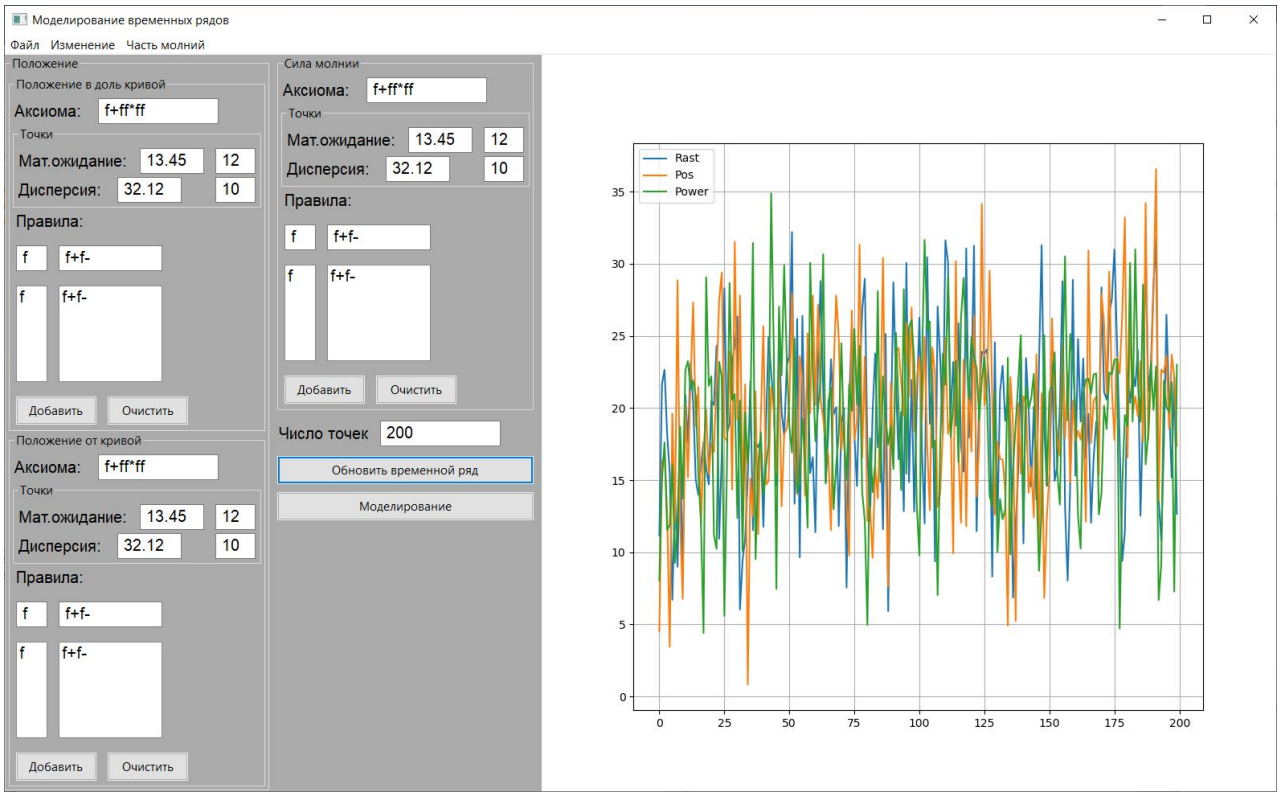


Рисунок 1.3 – Вікно з генерацією молній

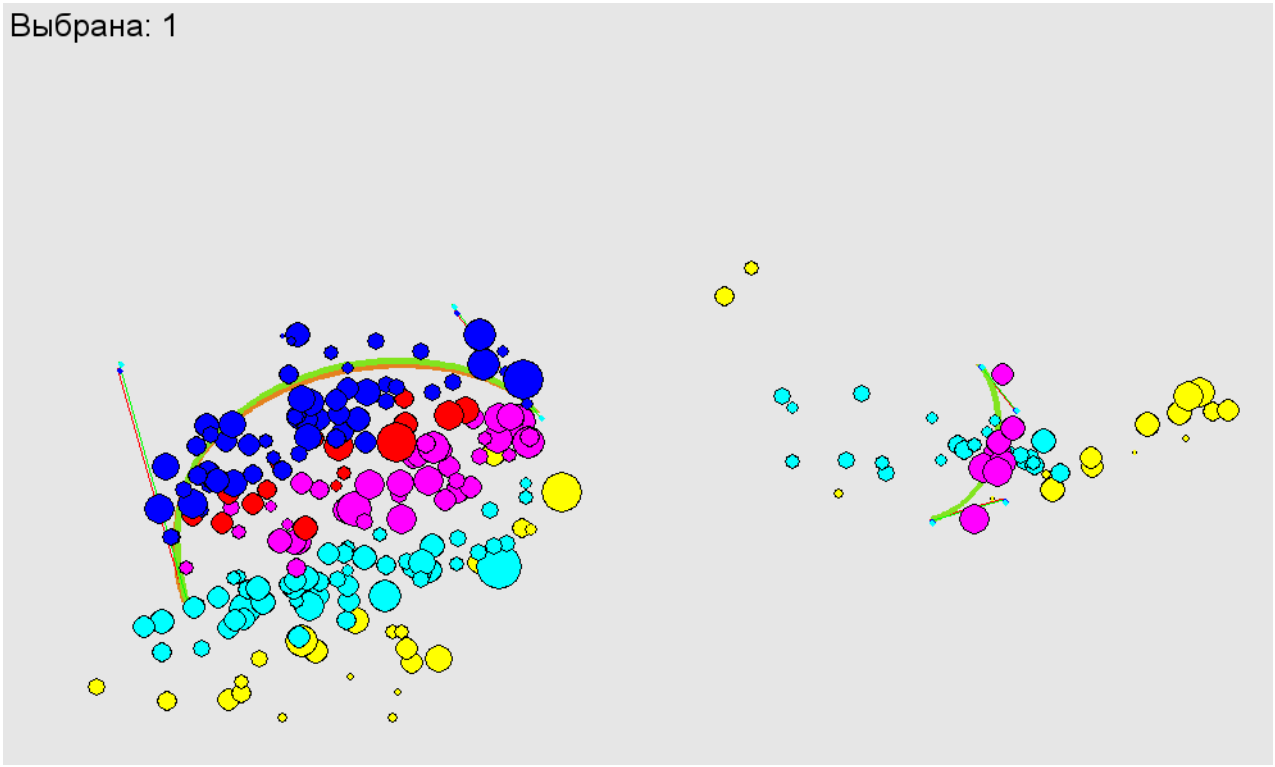


Рисунок 1.4 – Вікно з моделювання процесу утворення блискавок

1.3.3 Огляд мови програмування Python

Розроблені програмні додатки були написані на інтерпретованій об’єктно-орієнтованій мові програмування високого рівня зі строгою динамічною

типізацією [24]. Також ця мова підтримує інші парадигми програмування – функціональне та процедурне [25].

Оскільки це мова високого рівня, то основна задача, яка стає перед програмістом це програмування логіки не вдаючись повного опису структурних елементів самих команд програми.

Це дозволяє писати легкочитаємий, зрозумілий більшості код, що не засмічений додатковою інформацією, а несе лише прямий опис логіки процесу.

Зрозумілий та простий синтаксис, що базується на відступах та простих синтаксичних конструкціях дозволяє людині з базовими знаннями програмування описувати функції, що можуть бути використані у етапі інтерпретування.

1.4 Огляд об’єктів дослідження

1.4.1 Поняття фракталу

Фракталом називають об’єкт, що має у собі прояв характеристики самоподібності у більший (повній) чи меншій самого себе (тобто повторення форми чи будови при виділені підоб’єкту з об’єкту).

Також, іншою умовою можливою умовою класифікації об’єкту як фракталу є наявність дробової фрактальної розмірності в математичній чи графічній моделі об’єкту.

Під фракталом розуміють об’єкт, який точно або наближено співпадає з частиною самого себе. Тому фрактальний об’єкт наділено властивістю самоподібності.

За Фальконом [26] визначають наступний перелік структурових, геометричних чи множинних властивостей об’єкту для означення його терміном «фрактал» [27]:

- дробова фрактальна розмірність;
- ніде не диференційовані функції;
- самоподібність – геометрична множина Q містить копії самої себе в багатьох різних масштабах;
- множина Q має «тонку структуру», яка містить деталі в довільних малих масштабах;
- хоча Q має складну структуру, фактичне визначення Q дуже просте;

– для отримання Q застосовується рекурсивна процедура;
– геометрію Q нелегко описати в класичних термінах, це не геометричне місце точок, які задовольняють деяку просту умову і не множина розв’язків будь-якого простого рівняння;
– проблематично описати локальну геометрію Q ¬ поблизу кожної з її точок є велика кількість інших точок, розділених проміжками різної довжини;
– хоча Q в певному роді достатньо велика множина (незліченна), її розмір не визначається кількісно звичними мірами, такими як довжина, площа, об’єм

При розглядані графічних фракталів (фракталів, що графічно представлені на координатній площині), здебільшого як головну властивість ставлять розраховану дробову фрактальну розмірність отриманої фігури у межах мінімальної виділеної області, але важливо враховувати і елементну самоподібність в об’єкті.

Приймемо наступне означення: детермінованим геометричним фракталом називається фігура, яка може бути сформована конструктивною системою з рекурсивними, постійно застосовними правилами [28].

Таким чином, під це правило можемо віднести усі вище наведені характерні властивості фракталу.

Приклад фрактальних зображень [29] наведено на рис.1.5.

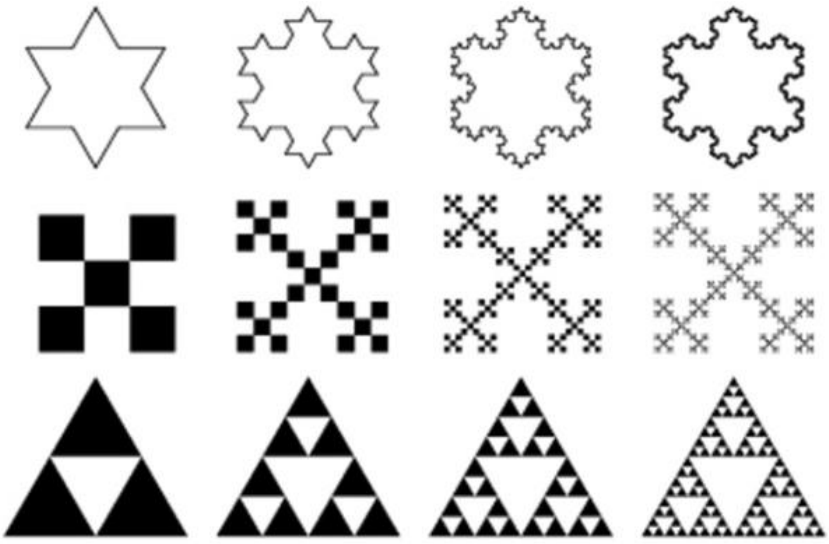


Рисунок 1.5 – Розгорнені приклади фракталів

1.4.2 Поняття L-системи

L-система або система Ліденмайера - це паралельна система переписування і вид формальної граматики. L-система складається з алфавіту символів, які можуть бути використані для створення рядків, набору правил, які задають правила підстановки замість кожного символу певного набору інших символів, початкового рядка («аксіоми»), з якої починається побудова, і механізму перекладу утвореного рядку в геометричні структури. L-системи запропонував і розвивав в 1968 Арістід Ліденмайер, угорський біолог і ботанік з Утрехтського університету. Ліденмайер використовував L-системи для опису поведінки клітин рослин і моделювання процесу розвитку рослини. L-системи використовувалися також для моделювання морфології різних організмів і можуть бути використані для генерації самоподібних фракталів [16].

Прикладом фракталу, створеного моделюванням L-системи наведено на рис.1.6.

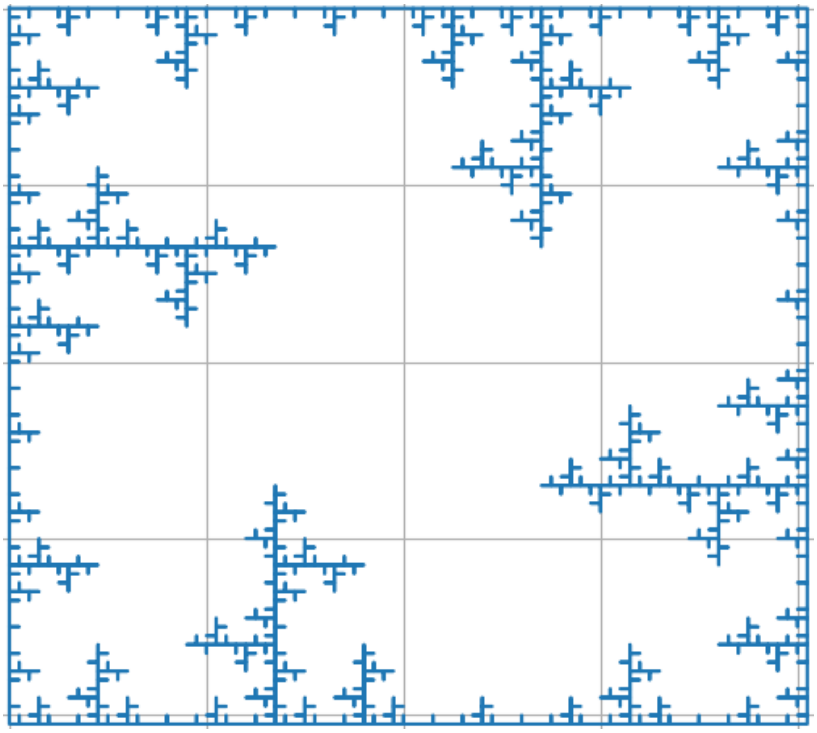


Рисунок 1.6 – Розгорнені приклади фракталів

При описанні L-системи фрактального об’єкту, використовується певний набір початкових атрибутів та функцій інтерпретування:

– початкова символьна стрічка (аксіома);
– набір правил підстановки;
– значення кутів повороту;
– значення довжин лінії;
– значення кольорів ліній;
– набір функцій:
  - проведення ліній;
  - переміщення без малювання лінії;
  - поворот за часовою стрілкою та проти часової стрілки по колу;
  - занесення та винесення значення положення точки до стеку;
– набір поєднань функцій з символами.

У класичному представлені L-систем, прикладами зарезервованих символів з виконуваними функціями є :

– «f» – проведення лінії;
– «F» – переміщення без малювання лінії;
– «+» – поворот за часовою стрілкою по колу;
– «-» – поворот проти часової стрілки по колу.

Але залежно від поєднання символу і дії, пари зарезервованих символів та дій можуть відрізнятися відповідно до реалізації [30].

На основі цих дій описується метод черепашкової графіки, за яким створюються графічні відображення фракталів описаних граматикою.

На рис.1.7 зображено приклад зображення фракталу «Сніжинка Коха» [31] на першій ітерації, створеного методом черепашкової графіки [32].

Черепашкова графіка базується на простих командах, таких як:

– просунутися вперед;
– просунутися назад;
– просунутися ліворуч;
– просунутися праворуч;
– підняти перо;

– опустити перо;
– переміститись в точку з заданими координатами.

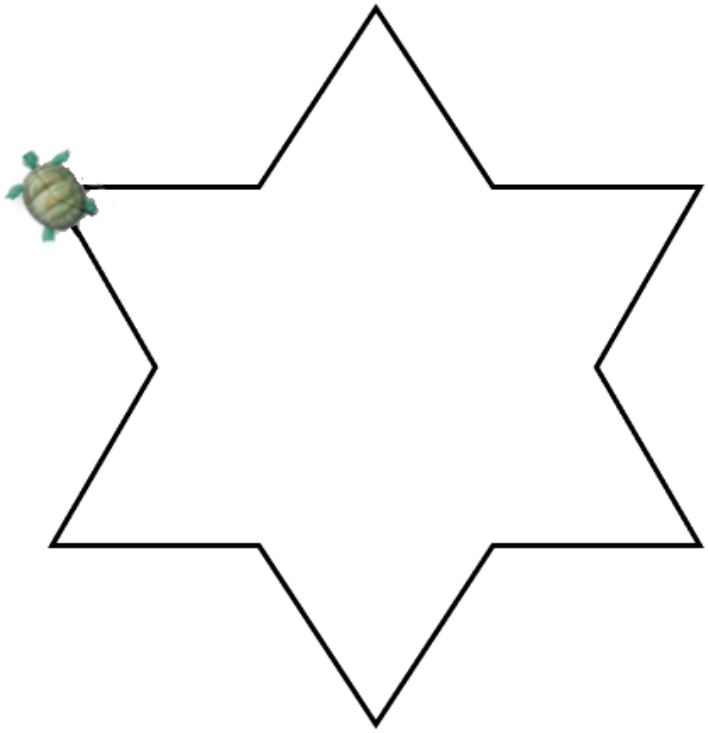


Рисунок 1.7 – «Сніжинка Коха» відображена черепашковою графікою

Через простоту опису створення зображень, створення зображень даним методом використовується при демонструванні послідовностей точок фракталу чи при демонстрації фракталів з невеликою кількості точок.

Головним недоліком даного методу є швидкість малювання відображення програмою. Оскільки, «черепашка» повинна буде виконувати велику кількість послідовних однотипних простих дій, таких як попереміннi дії повороту з просуванням черепашки, кількість команд, які має бути виконано будуть більше кількості символів кінцевої стрічки граматики.

1.4.3 Поняття часового ряду

Часовий ряд (або ряд динаміки) - зібраний в різні моменти часу статистичний матеріал про значення будь-яких параметрів (в найпростішому випадку одного) досліджуваного процесу. Кожна одиниця статистичного матеріалу називається виміром або відліком, також допустимо називати його рівнем на вказаний з ним момент часу. У часовому ряді для кожного відліку має бути зазначено час

вимірювання або номер вимірювання по порядку. Часовий ряд істотно відрізняється від простої вибірки даних, так як при аналізі враховується взаємозв'язок вимірювань з часом [33].

1.4.4 Поняття бієктивного відображення

Біекція - це відображення, яке ставить кожному елементу однієї множини рівно один елемент іншої множини [34].

Прилад бієктивного відображення однієї множини до іншої наведено на рис.1.8.

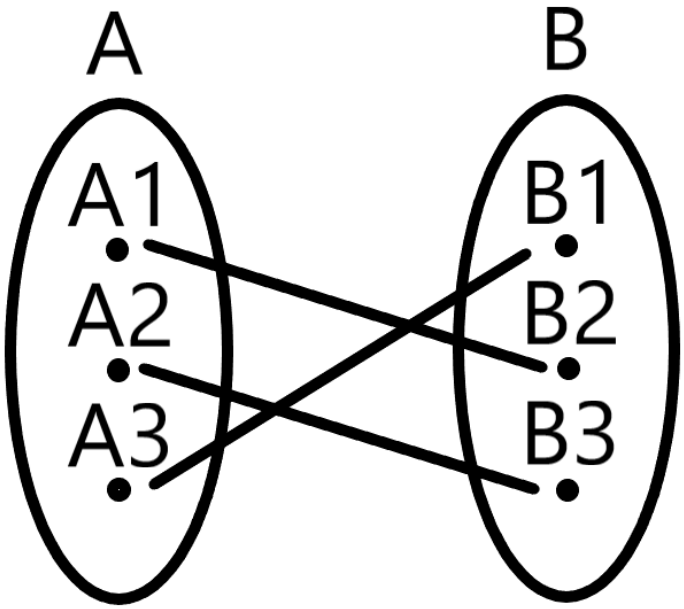


Рисунок 1.8 – Розгорнені приклади фракталів

Бієктивні відображення є прикладом повного перенесення даних та їх кількості однієї множини на іншу. Кінцева множина зберігає кількість атрибутів початкової множини. Тобто, кожному елементу деякої множини B ставиться лише один і тільки один елемент множини A.

Також, окрім бієкції, вирізняють відображення ін’єктивні [35] та сюр’єктивні [36].

Ін’єктивні відображення характеризується тим, що кожному елементу деякої множини B ставиться не більше одного елементу множини A. Графічне відображення ін’єкції наведено на рис.1.9.

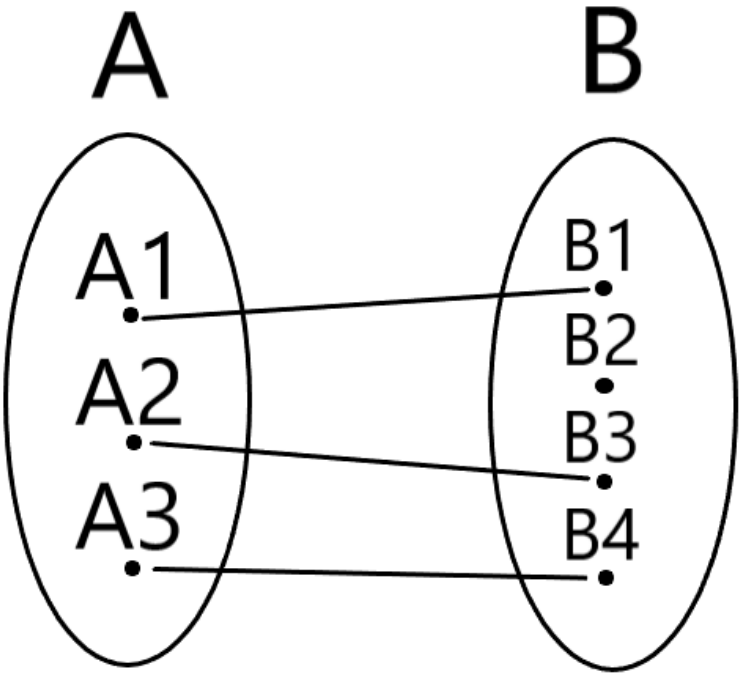


Рисунок 1.9 – Розгорнені приклади фракталів

Сюр’єктивне відображення характеризується тим, що кожному елементу деякої множини B ставиться щонайменше один елемент множини A. Приклад цього відображення наведено на рис.1.10.

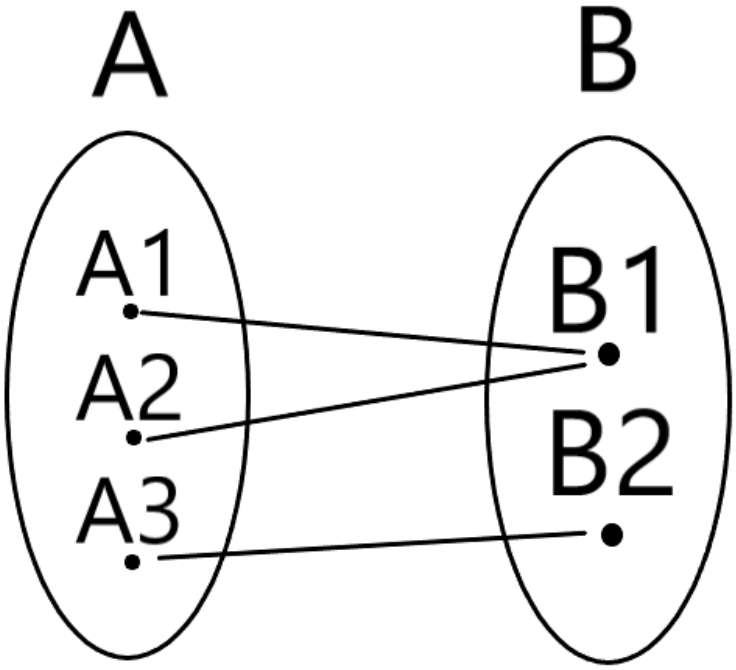


Рисунок 1.10 – Розгорнені приклади фракталів

1.5 Огляд літератури

1.5.1 Огляд елементів теорії фрактальних множин

При дослідженні розглядалися навчальні посібники, що розглядають фрактальну геометрію та теорію фрактальних множин. При ознайомлені з літературою, відзначається представлення прикладів існуючих фрактальних об’єктів, що можуть бути дослідженими, чи приклади існуючих досліджень у даній області знань.

При визначенні фрактальної геометрії, у навчальному посібнику В. С. Секованова [37] представлена інформація про становлення вивчення даної дисципліни на прикладах знань дослідників минулих років у розрізі часу досліджень.

Автор представляє можливі програмні рішення для дослідження, такі як використання L-систем чи афінних перетворень геометричних фігур для представлення фрактальних об’єктів. Також наводяться алгоритми, що можуть бути застосовані при створенні графічних відображень прикладів.

При розгляді даної роботи, важливо звернути увагу на аудиторію, перед якою представляється даний навчальний посібник. Робота має за мету зацікавити студентів та молодих науковців звернути увагу до даної дисципліни, пропонуючи на доступній для більшості мові знання, які можливо використати на практиці.

1.5.1 Огляд вимог до вирішення проблематики

Після огляду літератури та існуючих наукових рішень за даною темою проблематики, були виявлені наступні важливі вимоги до програмних засобів:

- робота конструктора базується на четвірці етапів конструктивно-продуційних граматики;
- задання атрибутивних елементів (атрибутики) [38] модельного об’єкту з описом їх областей значень на основі базових типів даних існуючих рішень алгоритмічних мов програмування;
- можливість графічного представлення продукції конструктивного процесу з вибором інтерпретуємої інформації з конструктора об’єкту (побудова двовимірних графіків на координатній площині);

– описання функцій інтерпретування за допомогою існуючих мов описання алгоритмічних дій та структур.

Висновки до першого розділу

У розділі була розглянута мета та доцільність розвитку наукових досліджень у даному напрямку.

Розглянуті основні знання, поняття яких мають доцільність при дослідженні у роботі. На основі часових рядів та L-систем розглядаються варіанти можливих відображень фракталів. Ставиться питання розгляду множин даних, що можуть бути отримані при реалізації моделювання.

Були розглянуті програмні додатки, які використовувалися у попередніх дослідженнях за цією темою, що маються на сьогодення корисне використання у розгляді конструктивно-продуційного моделювання. Додатки «PyTime» та «PyLight» ставить початкову точку з досліджень використання продуційних конструкторів при описі об’єктів та навколишніх явищ, але вони не можуть використовуватися за межами предметних областей до яких були розроблені.

Також, були розглянуті літературні знання з теорії фрактальної геометрії, на основі яких були розроблені вимоги до програмних засобів. У літературі підіймаються не лише питання досліджень з даної дисципліни, а й поглиблення знань з існуючих досліджень та реалзацій.

2 ОБҐРУНТУВАННЯ НАПРЯМКУ ДОСЛІДЖЕННЯ КОНСТРУКТИВНО-ПРОДУЦІЙНИХ ГРАМАТИК

2.1 Формування основної мети та завдання дослідження

Дослідження в даній роботі перед собою ставить доведення існування транзитивності властивостей від одного відображення конструктивно-продуційної граматики до іншого відображення, створених шляхом бієктивного перетворення атрибутів з використанням різних інтерпретуючих функцій.

Проблема полягає у обмеженості можливостей моделюючих програмних додатків, тому що вони зосереджують свої можливості лише на певних предметних областях.

При створенні програмного продукту ставилася задача створення програмного продукту, який зможе моделювати графічні зображення на основі конструктивно-продуційних граматик з набором атрибутів визначених користувачем, а трансляція та інтерпретація виконується через запрограмовані функції, які може власноруч описати користувач.

Підводячи вищезазначене, метою дослідження є використовую засоби методи конструктивно-продуційного моделювання виявити транзитивність властивостей фракталів між різними бієктивними відображеннями за допомогою уніфікованого процесу опису систем конструкторів конструктивно-продуційних граматик.

Відповідно до цього, ставиться завдання розробки:

- сутностей конструкторів фракталів на основі мультисимвольних граматик;
- програмного забезпечення, яке уніфікує процес створення й опису конструкторів фракталів та формує дані їх продукції.

2.2 Обґрунтування процесу дослідження

На основі описаних у першому розділі прикладах програмного забезпечення та прикладах існуючих досліджень, проведених на окремих прикладах предметних областей, виконано узагальнення вимог до реалізації систем дослідження конструктивно-продуційних граматик. Керуючись вищевикладеним, формується набір задач, які допомагає вирішити моделювання фракталів та їх подальший аналіз.

На основі цього, створення уніфікуючого інструментарію дає змогу розширити предметні області майбутніх досліджень та створювати демонстраційний матеріал на основі потреб дослідження.

2.3 Обґрунтування використання конструктивно-продуційного моделювання

Фрактальность, як самоподібність у часових рядів, вперше була описана Херстом і нині фрактальні часові ряди почали активно вивчатися в різних прикладних науках, однак питання формального моделювання фрактальних часових рядів мало вивчені [28].

В [5, 6, 7] закладені основи математико-алгоритмічного конструктивізму. На цій основі представляється можливим моделювання і формалізація будь-яких конструктивних процесів з використанням узагальненого конструктора (УК). Для формування конструкцій в рамках УК необхідно виконувати ряд уточнюючих перетворень в наступній послідовності: спеціалізація, інтерпретація, конкретизація, реалізація.

Продукційні L-системи (Lindenmayer system) широко використовуються для моделювання різних біологічних систем, процесів комп'ютерної графіки, музики.

$$C_G = \langle M, \Sigma, \Lambda \rangle,$$

де M – неоднорідний носій структури, Σ – сигнатура, що складається з безлічі операцій зв'язування, підстановки і висновку, операцій над атрибутами і відносин підстановки, Λ – інформаційне забезпечення.

Сигнатура Σ складається з безлічі операцій: Ξ - зв'язування, Θ - підстановки і висновку, Φ - операцій над атрибутами. Сигнатура містить також відносини підстановки «→». Таким чином, формально сигнатурою є $\Sigma = \langle \Xi, \Theta, \Phi, \{\rightarrow\} \rangle$ з властивостями: $\Xi \cap \Theta = \emptyset;\ \Xi \cap \Phi = \emptyset;\ \Theta \cap \Phi = \emptyset, \varepsilon \in \Phi$.

Сигнатура складається з імен операцій $\{\otimes_j\}$, що містять набір атрибутів w_i, та представляється як ${}_w\otimes \in \Sigma$.

Назвемо L-структурою наступну спеціалізацію УК:

$$C_G = \langle M, \Sigma, \Lambda \rangle_s \mapsto C_{LS} = \langle M_{LS}, \Sigma_{LS}, \Lambda_{LS} \rangle$$

Носій M_{LS} структури C_{LS} складається з конструктивних елементів з набором атрибутів, які пов'язані зі статичними, динамічними, або складовими властивостями елементів. Сигнатура Σ_{LS} складається з імен операцій, що володіють набором атрибутів, і складається з безлічі операцій: зв'язування, підстановки і висновку, операцій над атрибутами.

В результаті функціонування УК відбувається формування і перетворення конструкцій з використанням операцій, що задаються правилами: зв'язування, підстановки, виведення та ін.

Призначення конструктивно-продуційної структури (КПС) [] полягає у формуванні множин конструкцій за допомогою операцій зв'язування, підстановки, виведення та ін. Операцій, що задаються правилами аксіоматики.

Для формування конструкцій необхідно виконувати ряд уточнюючих перетворень узагальненої КПС [5, 6, 7]:

- **спеціалізація** визначає предметну область: семантичну природу носія, кінцеве множину операцій і їх семантику, атрибутику операцій, порядок їх виконання і обмеження на правила підстановки;
- **інтерпретація** полягає в зв'язуванні операцій сигнатури з алгоритмами виконання деякої алгоритмічної структури. При інтерпретації виконується зв'язування інформаційної моделі способу побудови конструкцій і моделі виконавця;
- **конкретизація** КПС полягає в розширенні аксіоматики множиною правил продукцій, завданні конкретних множин нетермінальних і термінальних символів з їх атрибутами і, при необхідності, значень атрибутів;
- **реалізація** КПС полягає в формуванні конструкції з елементів носія КПС шляхом виконання алгоритмів, пов'язаних з операціями сигнатури. Реалізація можлива тільки для попередньо спеціалізованої, інтерпретованої і конкретизованої КПС.

Згідно аксіоматиці УКПС формою ${}_{w}l$ з атрибутом w називається набір терміналів і нетерміналів, що об'єднуються операціями зв'язування. Конструкцією називається форма, яка містить тільки термінали.

Правила підстановки мають вигляд $\psi_r: \langle s_r, g_r \rangle \in \Psi$, де s_r – відношення підстановки, g_r – набір операцій над атрибутами. Відношення підстановки – двомісне відношення з атрибутами ${}_{w_i}l_i \; {}_{w}\rightarrow {}_{w_j}l_j$. Для форми ${}_{w_l}l_l = {}_{w_0}\oplus (_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_h}l_h, \ldots, {}_{w_k}l_k)$ і доступного відношення підстановки ${}_{w_h}l_h \; {}_{w_p}\rightarrow {}_{w_q}l_q$ такого ${}_{w_h}l_h \prec {}_{w_l}l_l$, що (${}_{w_h}l_h$ є частиною ${}_{w_l}l_l$), результатом тримісної операції підстановки $w_p \Rightarrow ({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_l}l_l)$ буде форма ${}_{w_l^*}l_l^* = {}_{w_0}\oplus (_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_q}l_q, \ldots, {}_{w_k}l_k)$, де $\oplus$– будь-яка операція зв'язування з Σ.

Операція часткового виведення ${}_{\text{тах}n,\vec{q},m}l^* = (| \Rightarrow (\Psi, {}_{\text{тах}n,\vec{q},m}l))$ полягає у:

- виборі одного з доступних правил підстановки ${}_{\vec{d}}\psi_r: \langle s_r, g_r \rangle \in \Psi$, з відношеннями підстановки s_r і виконанні на його основі операцій підстановки, де $\vec{d}$ – вектор доступності правил. Доступність правила ${}_{\vec{d}}\psi_r$ визначається значенням вектора доступності: якщо $d_r = 1$ правило доступне, якщо $d_r = 0$ – правило не доступне;
- виконанні операцій над атрибутами g_r.

Порядок застосування операції над атрибутами в процесі виконання операції часткового виведення задається атрибутом τ_j, де $\tau_j \in I$, $I = \{\tau_0, \tau_1\}$, $I \subset M_{KAC}$, τ_0 – операція над атрибутом виконується перед операцією підстановки, τ_1 – після операції підстановки.

Операція повного виведення (або просто виведення) полягає в послідовному виконанні операції часткового виведення, починаючи з початкового нетермінала і закінчуючи конструкцією, що задовольняє умові закінчення виведення.

Умовою закінчення виведення є відсутність нетерміналів в формі.

Перетворення КПС дозволяють застосовувати їх для різних предметних областей.

Підхід на основі КПС може бути використаний для формалізації побудови детермінованих та стохастичних геометричних фракталів, часових рядів з фрактальними властивостями.

2.4 Обґрунтування моделювання графічних фракталів на основі L-систем

Конструктивно-продуційний підхід до моделювання фракталів дозволяє встановити відповідність між мультісімвольнимі і лінійними плоскими геометричними фракталами, детермінованими і стохастичними. Це розширює можливості конструювання останніх з метою вивчення їх властивостей і можливостей моделювання об'єктів реального світу. Засоби конструктивно-продуційного моделювання отримали розвиток у вигляді сімейства параметричних конструкторів, що дозволяє варіювати можливості конструкторів.

Розглянемо моделювання відомих детермінованих та стохастичних двовимірних фракталів та фрактально-подібних конструкцій в рамках конструктивно-продуційного підходу з використанням конструктора $C = \langle M, \Sigma, \Lambda \rangle$.

Нехай носій M спеціалізованого конструктора $C = \langle M, \Sigma, \Lambda \rangle_S \mapsto C_{LS} = \langle M_{LS}, \Sigma_{LS}, \Lambda_{LS} \rangle$ містить термінальні символи $\{f, y, x\} \in M$ та наступні атрибути конструювання:

- M_x – математичне очікування довжини побудованого на площині відрізка;
- dM_x – зміна величини математичного очікування;
- D_x – дисперсія довжини побудованого відрізка;
- α – кут повороту побудованого відрізка;
- $d\alpha$ – дисперсія кута повороту побудованого відрізка.

Сигнатура операцій складається з $\{+, -\} \in \Sigma$. Інформаційне забезпечення Λ складається з онтології конструктивно-продуційного моделювання, яка доповнена онтологією L-систем і відомими поняттями «площина», «відрізок», «дійсне число», «координати», «кут», та іншими, які дозволяють оперувати як з лінійними геометричними фігурами, так і з дійсними числами. Правила конструювання містять початкову аксіому f і правила підстановки $y \rightarrow \ldots$, $x \rightarrow \ldots$. Обмеженням виступає правило заміни $\rightarrow \varepsilon$ на пустий символ, якщо жодне з правил заміни не виконується.

Відповідність символів конструктора операціям L-систем представлено у табл.2.1.

Таблиця 2.1- Відповідність множин символів операціям L-систем

Символ	Операція
*	Збільшити математичне очікування довжини відрізка
:	Зменшити математичне очікування довжини відрізка
\	Збільшити дисперсію довжини відрізка
/	Зменшити дисперсію довжини відрізка
+	Поворот на додатній кут
-	Зменшити дисперсію кута
>	Збільшити додатній кут
<	Зменшити додатній кут
!+	Збільшити від'ємний кут
!-	Зменшити від'ємний кут

Інтерпретація конструктора доповнює онтологію зв'язками операцій та терміналів з відповідними алгоритмами їх реалізації:

$$(A_j|_{+}^{\tilde{+}} \hookleftarrow +);\ (A_j|_{-}^{\tilde{-}} \hookleftarrow -);\ (A_j|_{f}^{\tilde{f}} \hookleftarrow f);\ (A_j|_{y+}^{\tilde{y}} \hookleftarrow y).$$

Будемо вважати, що алгоритмічна структура C_A має всі необхідні базові алгоритми та алгоритм A_i, який будує зображення вхідного символа.

Результатом послідовного виконання операцій конструктора є мультисимвольний ланцюг з властивістю самоподібності, а в подальшому графічне зображення фракталу, як реалізація алгоритму графічного зображення символів побудованої мультисимвольної послідовності.

Нижче наведено побудови в рамках конструктивно-продуційного підходу відомих фрактальних множин: фрактал «Сніжинка Коха», фрактал «дракон Хартера – Хейтуея» [39], фрактал «Крива Коха» [40], фрактал «Крива Госпера» [41], фрактал «Льодовий фрактал» [42], фрактал «Крива Серпинського» [43].

2.4.1 Фрактал «Сніжинка Коха».

Для фракталу «Сніжинка Коха» граматика позначається:

- Аксіома: f++f++f;
- Правила підстановки: f→f-f++f-f.

Мультисимвольний ланцюг має вигляд: f-f++f-f-f-f++f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++…

Детермінований фрактал «сніжинка Коха», представлений на рис.2.1, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 60°$.

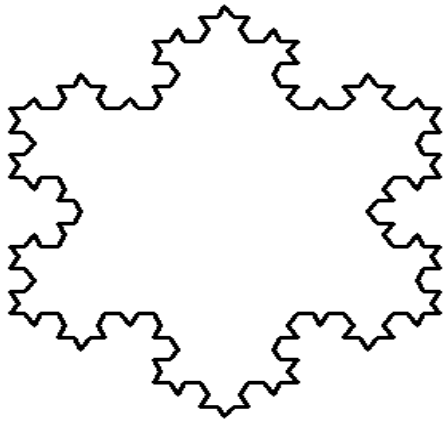


Рисунок 2.1 – Реалізація конструктора детермінованого фракталу «сніжинка Коха»

Стохастичний фрактал «сніжинка Коха», представлений на рис.2.2, є реалізацією конструктора з параметрами $M_x = 1,\ dM_x = 0{,}5,\ D_x = 0{,}5,\ dDx = 0{,}25,\ \alpha = 60°,\ d\alpha = 30°$.

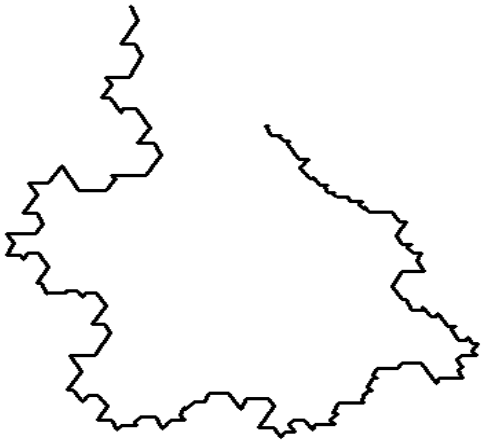


Рисунок 2.2 – Реалізація конструктора стохастичного фракталу «сніжинка Коха»

2.4.2 Фрактал «Дракон Хартера-Хейтуея»

Для фракталу «Дракон Хартера-Хейтуея» граматика позначається:

- Аксіома: fx;
- Правила підстановки: y→-fx-y; x→x+yf+.

Мультисимвольний ланцюг має вигляд: fx+yf++-fx-yf++-fx+yf+--fx-yf++-fx+yf++-fx-yf+--fx+yf+--fx-yf++-fx+yf++-fx-yf++-fx+yf+--fx-yf+--fx+yf++-fx-yf+--fx+yf+--…

Детермінований фрактал «Дракон Хартера-Хейтуея», представлений на рис.2.3, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 90°, d\alpha = 0$.

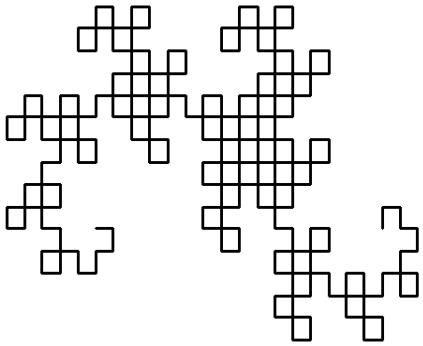


Рисунок 2.3 – Реалізація конструктора детермінованого фракталу «Дракон Хартера-Хейтуея»

Стохастичний фрактал «сніжинка Коха», представлений на рис.2.4, є реалізацією конструктора з параметрами $M_x = 1,\ dM_x = 0{,}5,\ D_x = 0{,}5,\ dDx = 0{,}25,\ \alpha = 90°,\ d\alpha = 20°$.

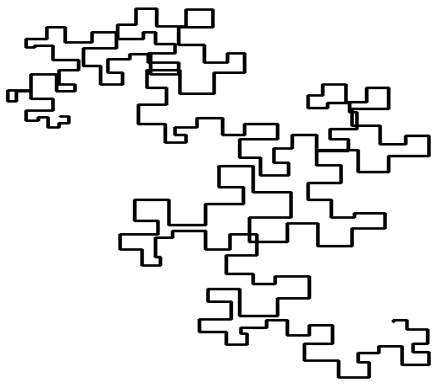


Рисунок 2.4 – Реалізація конструктора стохастичного фракталу «Дракон Хартера-Хейтуея»

2.4.3 Фрактал «Крива Коха»

Для фракталу «Крива Коха» граматика позначається:

- Аксіома: f;
- Правила підстановки: f→f-f++f-f.

Мультисимвольний ланцюг має вигляд: f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f-f-f++f-f++f-f++ …

Детермінований фрактал «Крива Коха», представлений на рис.2.5, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 60°$.

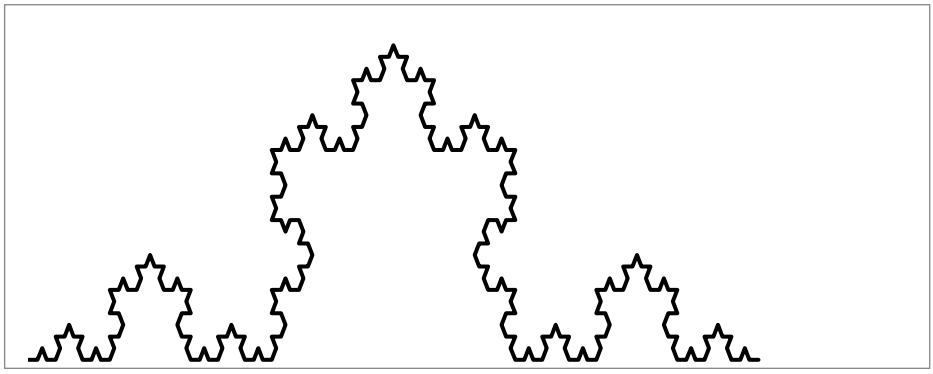


Рисунок 2.5 – Реалізація конструктора детермінованого фракталу «Крива Коха»

Стохастичний фрактал «Крива Коха», представлений на рис.2.6, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0{,}5,\ D_x = 0{,}5,\ dDx = 0{,}25,\ \alpha = 60°,\ d\alpha = 30°$.

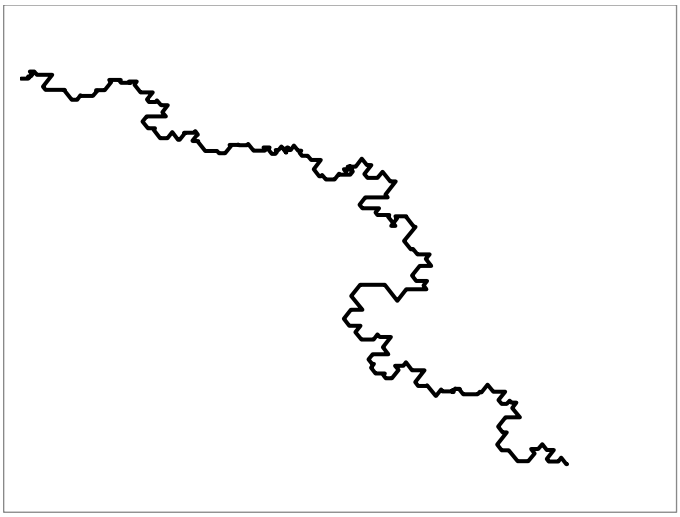


Рисунок 2.6 – Реалізація конструктора стохастичного фракталу «Крива Коха»

2.4.4 Фрактал «Крива Госпера»

Для фракталу «Крива Госпера» граматика позначається:

- Аксіома: xf;
- Правила підстановки: y->-fx+yfyf++yf+fx--fx-y, x->x+yf++yf-fx--fxfx-yf+ .

Мультисимвольний ланцюг має вигляд x+yf++yf-fx--fxfx-yf++-fx+yfyf++yf+fx--fx-yf++-fx+yfyf++yf+fx--fx-yf-fx+yf++yf-fx--fxfx-yf+--fx+yf++yf-fx--fxfx-yf+fx+yf++yf-fx--fxfx-y…

Детермінований фрактал «Крива Госпера», представлений на рис.2.7, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 60°$.

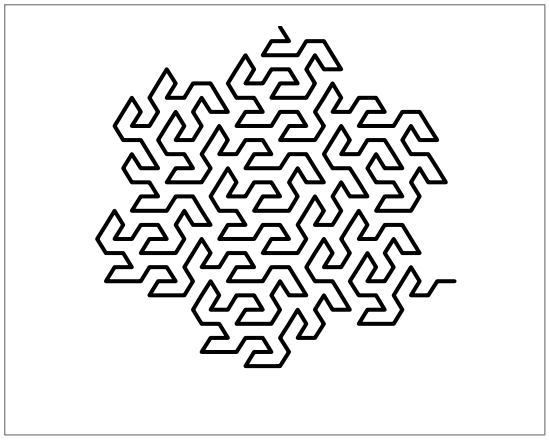


Рисунок 2.7 – Реалізація конструктора детермінованого фракталу «Крива Госпера»

Стохастичний фрактал «Крива Госпера», представлений на рис.2.8, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0{,}5,\ D_x = 0{,}5,\ dDx = 0{,}25,\ \alpha = 60°,\ d\alpha = 30°$.

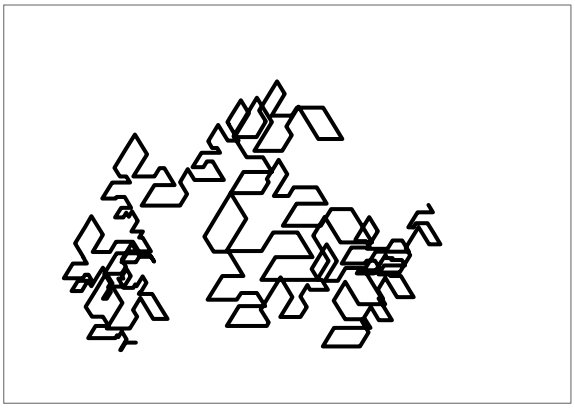


Рисунок 2.8 – Реалізація конструктора стохастичного фракталу «Крива Госпера»

2.4.5 Фрактал «Льодовий фрактал»

Для фракталу «Льодовий фрактал» граматика позначається:

- Аксіома: f+f+f+f;
- Правила підстановки: f→ ff+f++f+f.

Мультисимвольний ланцюг має вигляд: ff+f++f+fff+f++f+f+ff+f++f+f++f f+f++f+f+ff+f++f+fff+f++f+fff+f++f+f+ff+f++f+f++ff+f++f+f+ff+f++f+f+ff+f …

Детермінований фрактал «Льодовий фрактал», представлений на рис.2.9, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 90°$.

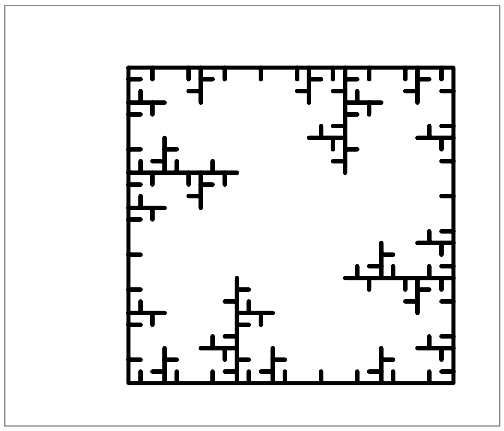


Рисунок 2.9 – Реалізація конструктора детермінованого фракталу «Льодовий фрактал»

Стохастичний фрактал «Льодовий фрактал», представлений на рис.2.10, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0{,}5,\ D_x = 0{,}5,\ dDx = 0{,}25,\ \alpha = 90°,\ d\alpha = 45°$.

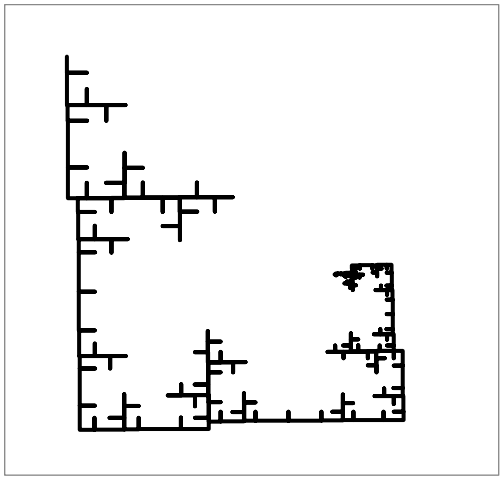


Рисунок 2.10 – Реалізація конструктора стохастичного фракталу «Льодовий фрактал»

2.4.6 Фрактал «Крива Серпинського»

Для фракталу «Крива Серпинського» граматика позначається:

- Аксіома: f+xf+f+xf;
- Правила підстановки: x→ xf-f+f-xf+f+xf-f+f-x.

Мультисимвольний ланцюг має вигляд: f+xf-f+f-xf+f+xf-f+f-xf-f+f-xf-f+f-xf+f+xf-f+f-xf+f+xf-f+f-xf+f+xf-f+f-xf-f+f-xf+f+xf-f+f-xf-f+f-xf-f+f-xf+f+xf-f+f-xf-f+f-xf-f+f-xf+f+xf…

Детермінований фрактал «Крива Серпинського», представлений на рис.2.11, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 90°$.

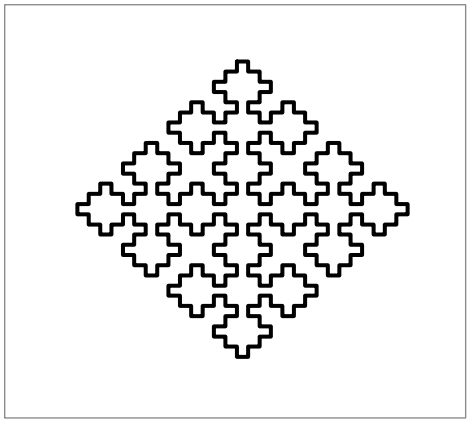


Рисунок 2.11 – Реалізація конструктора детермінованого фракталу «Крива Серпинського»

Стохастичний фрактал «Крива Серпинського», представлений на рисунок 2.12, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0{,}5,\ D_x = 0{,}5,\ dDx = 0{,}25,\ \alpha = 90°,\ d\alpha = 45°$.

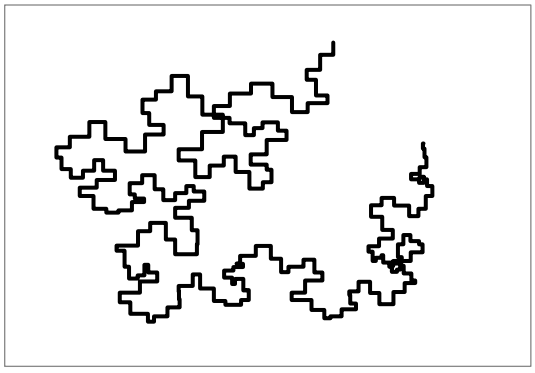


Рисунок 2.12 – Реалізація конструктора стохастичного фракталу «Крива Серпинського»

2.4.7 Фрактал «Трикутники»

Для фракталу «Трикутники» граматика позначається:

- Аксіома: f+f+f;
- Правила підстановки: f→ f-f+f.

Мультисимвольний ланцюг має вигляд: f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f+f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f …

Детермінований фрактал «Трикутники», представлений на рис.2.13, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 120°$.

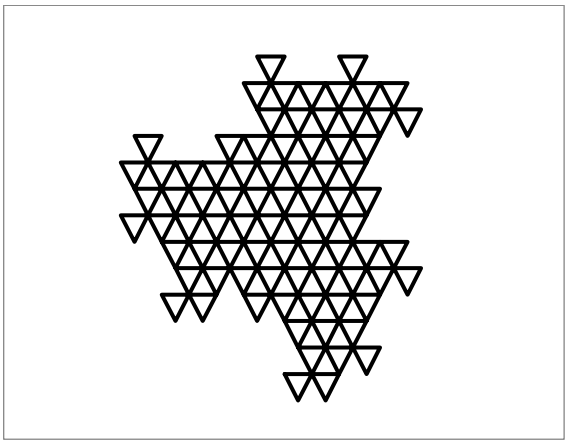


Рисунок 2.13 – Реалізація конструктора детермінованого фракталу «Трикутники»

Стохастичний фрактал «Трикутники», представлений на рис.2.14, є реалізацією конструктора с параметрами $M_x = 1,\ dM_x = 0{,}5,\ D_x = 0{,}5,\ dDx = 0{,}25,\ \alpha = 120°,\ d\alpha = 60°$.

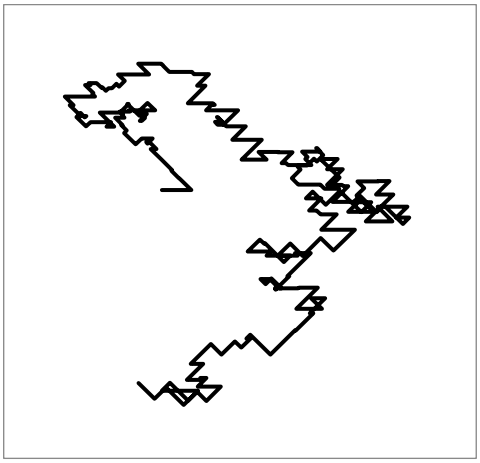


Рисунок 2.14 – Реалізація конструктора стохастичного фракталу «Трикутники»

2.5 Обґрунтування моделювання часових рядів на основі L-систем

Розглянемо можливість застосування конструктивно-продукційного підходу до моделювання фрактальних часових рядів на основі L-систем.

Нехай носій M_K узагальненого конструктора ${}_K C$ містить термінали $\{f\} \in M_K$ та атрибути конструювання $M_x, dM_x, D_x, \alpha, \alpha t$; сигнатура операцій $\{+,-\} \in \sum K$; інформаційне забезпечення Λ_K включає: онтологію з Λ_G, доповнену онтологією L-систем, правила конструювання: аксіому і правила підстановки, обмеження: $n = 8, M_x = 1, dM_x = 0, D_x = 0, \alpha = 90, \alpha t = 90$.

На першому кроці конструктор формує мультисимвольні ланцюги, які мають самоподібні властивості. Уточнюючі перетворення конструктора мають вигляд:

$${}_{K,S}C =< M_{S_1}, \sum S_1, \Lambda_{S_1} > \mapsto <_{K,S_1} C, C_A >_I \mapsto_{K,S_1,I_1} C =< M_{I_1}, \sum I_1, \Lambda_{I_1} >_R \mapsto$$

${}_{K,S_1,I_1,R_1}C$. Спеціалізація конструктора: онтологію в Λ_{S_1} доповнена семантикою елементів носія та операцій. Їх значення являють собою зображення відповідних символів $\{f,+,-\}$. f – показник необхідності обчислення значення часового ряду, операції $+,-$ змінюють значення M_x на величину dM_x.

Інтерпретація конструктора:

$$(A_k|^{x(t_i),i}_{M_x,dM_x,i} \hookleftarrow f);\ (A_{k+1}|^{M_x}_{M_x,dM_x} \hookleftarrow +);\ (A_{k+2}|^{M_x}_{M_x,dM_x} \hookleftarrow -),$$

де A_k – видає нормально розподілене випадкове число як значення часового ряду $x(t_i)$ з математичним очікуванням M_x і дисперсією dM_x, а також збільшує лічильник i на одиницю, $A_{k+1},\ A_{k+2}$ – збільшують чи зменшують (відповідно) M_x на величину dM_x.

Розглянемо конкретний приклад конструктора. Якщо початкова аксіома має вигляд f++f++f , правила підстановки f→f-f++f--f , тоді конструктор будує мультисимвольний рядок f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-.

Визначаємо значення атрибутів конструювання:

- початкове математичне очікування величини та зміни математичного очікування відповідно $M_x = 1, dM_x = 0{,}5$;

- початкова дисперсія величини і зміна величини дисперсії відповідно $D_x = 0, dD_x = 0$;
- початкове математичне очікування періоду і зміни періоду $\alpha = 60, \alpha t = 30$.

Реалізацією конструктора є детермінований фрактальний часовий ряд, представлений на рис.2.15. Фрактальність визначається методом побудови ряду і добре простежується на зображенні.

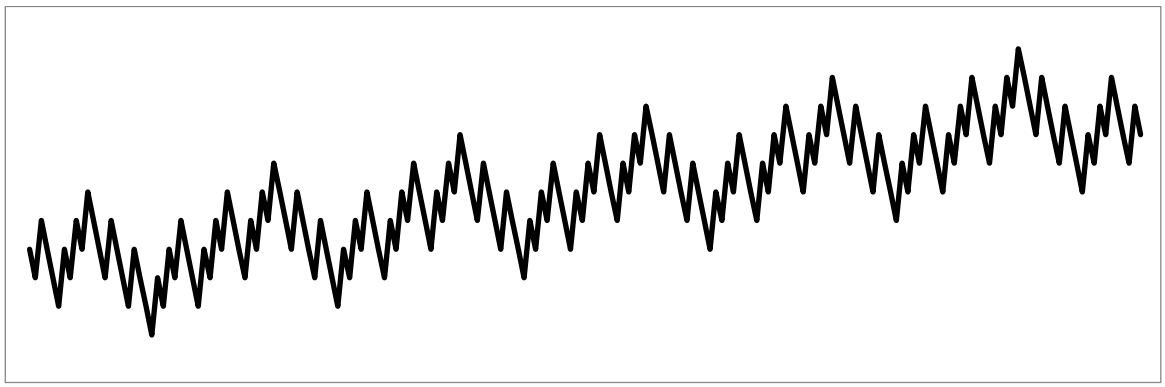


Рисунок 2.15 – Часовий ряд з нульовою дисперсією

Побудований конструктор дозволяє будувати часові фрактальні ряди, які обтяжені стохастичними шумами. На рис.2.16 представлено фрактальний часовий ряд, побудований попереднім конструктором, в якому стохастичні шуми$D_x = 0{,}5, dD_x = 0{,}25$.

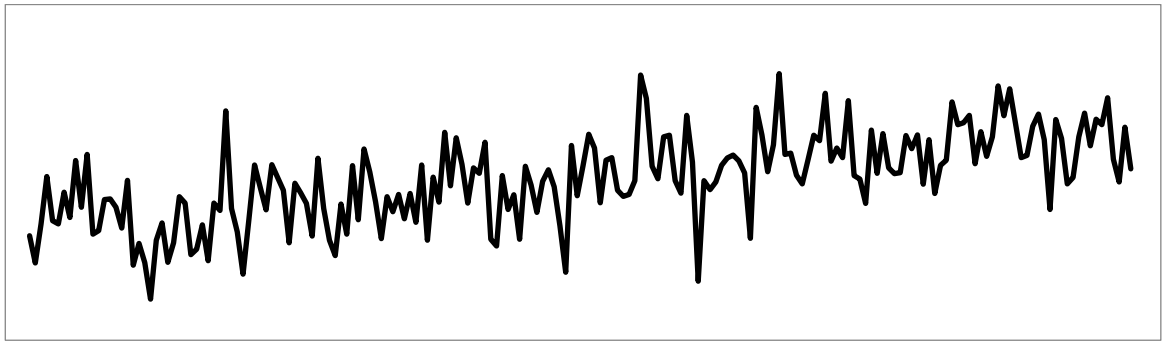


Рисунок 2.16 – Часовий ряд зі стохастичними шумами

Висновки до другого розділу

У розділі був обґрунтований напрям дослідження. Були сформовані мета та завдання дослідження.

Був визначений процес дослідження на основі конструктивно-продукційного моделювання, що застосовується при описі процесі, об’єктів та явищ різного предметного напрямку.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНТРУМЕНТАЛЬНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Зовнішнє проектування

3.1.1 Вхідні данні

Вхідними даними програми є:

- обраний тип конструктору конструктивно-продуційної граматики – простий конструктор чи конструктор алгоритмів;
- набір атрибутів конструктору з типами даних цих атрибутів (серед типів «int» (ціле число зі знаком), «float» (дробове число зі знаком), «bool» (логічний тип даних), «list» (масив значень), «dict» (асоціативний масив), «str» (текстове значення);
- тип графічного інтерпретування даних конструктора – у вигляді часового ряду чи у вигляді L-системи;
- атрибут конструктора, що буде відповідати масиву значень для осі абсцис;
- атрибут конструктора, що буде відповідати масиву значень для осі ординат;
- описані функції інтерпретування конструкторів алгоритмів на мові програмування Python з використанням стандартних бібліотечних засобів мови;
- описані форми інтерпретування простого конструктора, що складаються з обраного символу інтерпретування, вибраного алгоритмічного конструктора, вибраної функції інтерпретування з цього конструктора, перелічених атрибутів на вхід до функції та атрибуту для занесення продукції функції;
- описані початкові значення атрибутів простого конструктора в межах областей значень обраних типів даних.

3.1.2 Вихідні дані

Результатом роботи програми є наступні вихідні дані:

- розгорнута кінцева стрічка сформована на базі аксіоми та правил підстановки через ітераційний підхід;

- кінцевий стан усіх атрибутів простого конструктора у текстовому представлені у форматі «<Атрибут>: <Значення атрибуту>»;
- графічне представлення інтерпретування продукції простого конструктора у вигляді двовимірного графіку.

3.2 Початковий архітектури програмного забезпечення

При розробці програмного забезпечення у процесі розробки можна виділити основні етапи проектування:

- проектування архітектури програмного забезпечення;
- деталізація компонентів структурних частин програми;
- опис базових сутностей;
- описання сценаріїв роботи з програмним забезпеченням.

На етапі проектування програмного забезпечення обирається архітектура, яка буде реалізовуватися.

Найкращим вибором з варіантів архітектурних рішень при реалізації програми, що вирішує проблему уніфікації створення продуційних конструкторів вважатимемо монолітну архітектуру [44].

Основними позитивними елементами, за які була обрана така архітектура вважаються [45]:

- простота розробки програмної системи – багатий вибір інструментів та існуючих програмних компоненті, які можуть бути інтегровані у систему, дозволяють полегшити процес розробки. Крім того, всі дії виконуються з одним каталогом, що спрощує розробку;
- простота розгортання програмної системи – завдяки монолітному ядру не потрібно розгортати зміни або оновлення окремо, оскільки вони можуть бути зроблені зміною версії програмного забезпечення. Це полегшує створення нового функціоналу та можливість налагодження програмної системи;
- зменшення кількості компонентних проблем – більшість додатків залежать від безлічі міжкомпонентних завдань, як відправка даних та очікування отримання результатів обробки. Монолітні додатки набагато

легше враховують ці питання завдяки своїй єдиній кодової базі. До цих завдань простіше підключати компоненти, коли все працює в єдиному додатку;

- покращення продуктивність програмного забезпечення –монолітні додатки зазвичай більш продуктивні, ніж додатки на основі мікросервісів [46]. Монолітні додатки забезпечують більш швидкий зв'язок між програмними компонентами завдяки загальному коду і пам'яті.

Реалізацію архітектури у середовищі цілей програмного продукту наведено на рис.3.1.

Рисунок 3.1 – Архітектура програмного забезпечення

Також, до цієї архітектури описується базовий сценарій взаємодії користувача та системи [47]. Діаграма послідовності взаємодії користувача та логіки обробки конструкторів [48] наведено на рис.3.2.

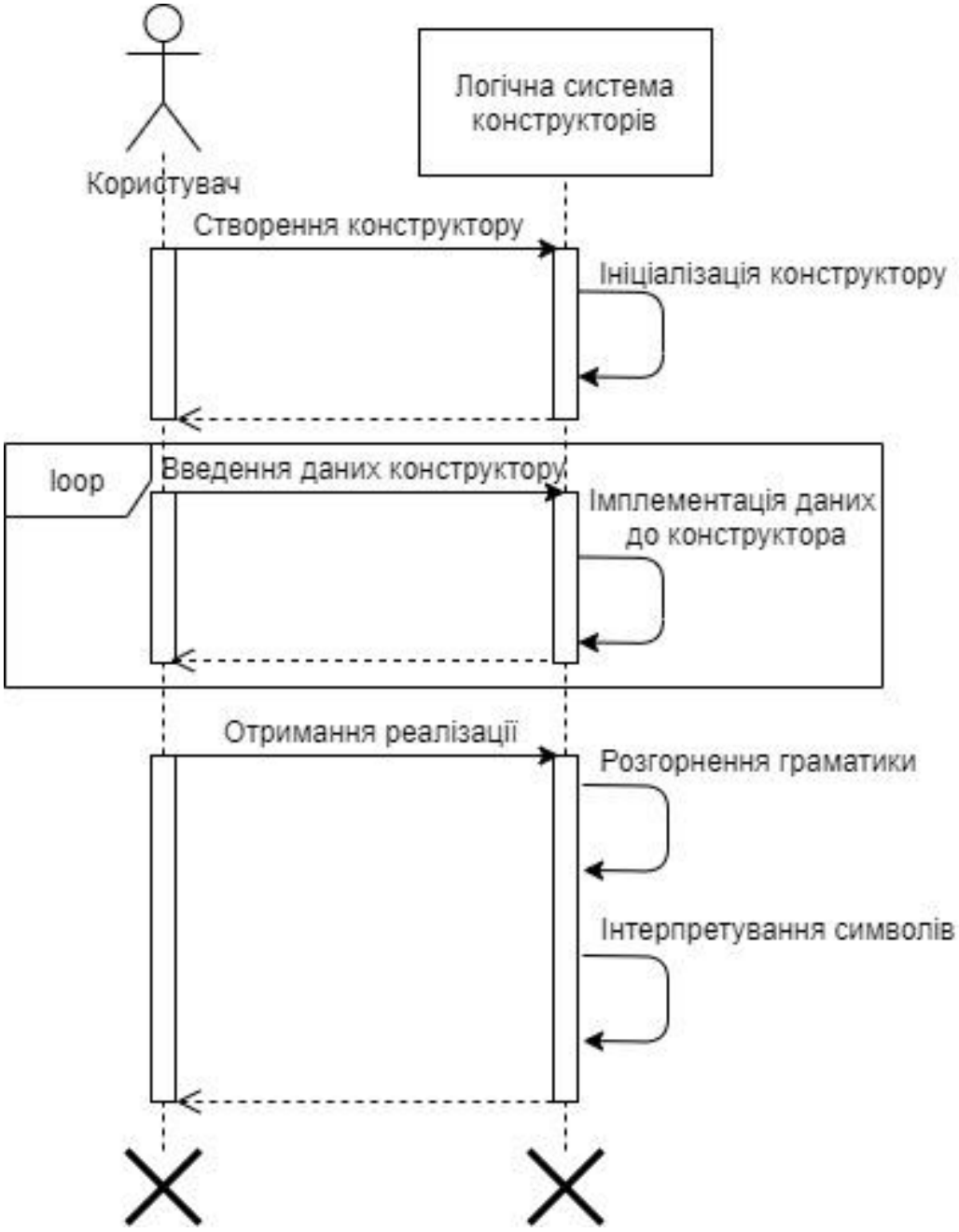


Рисунок 3.2 – Діаграма послідовності взаємодії користувача з системою моделювання конструкторів

При деталізації архітектури виділяються основні складники системи, а саме:

- інтерфейс взаємодії з користувачем;
- компонент обробки конструкторів;
- базова модель конструктора.

Деталізуючи взаємодію користувача з програмною систему на основі розширення на складові частини, отримуємо більш детально прописаний сценарій моделювання конструкторів.

Деталізована діаграма послідовності наведена на рис.3.3.

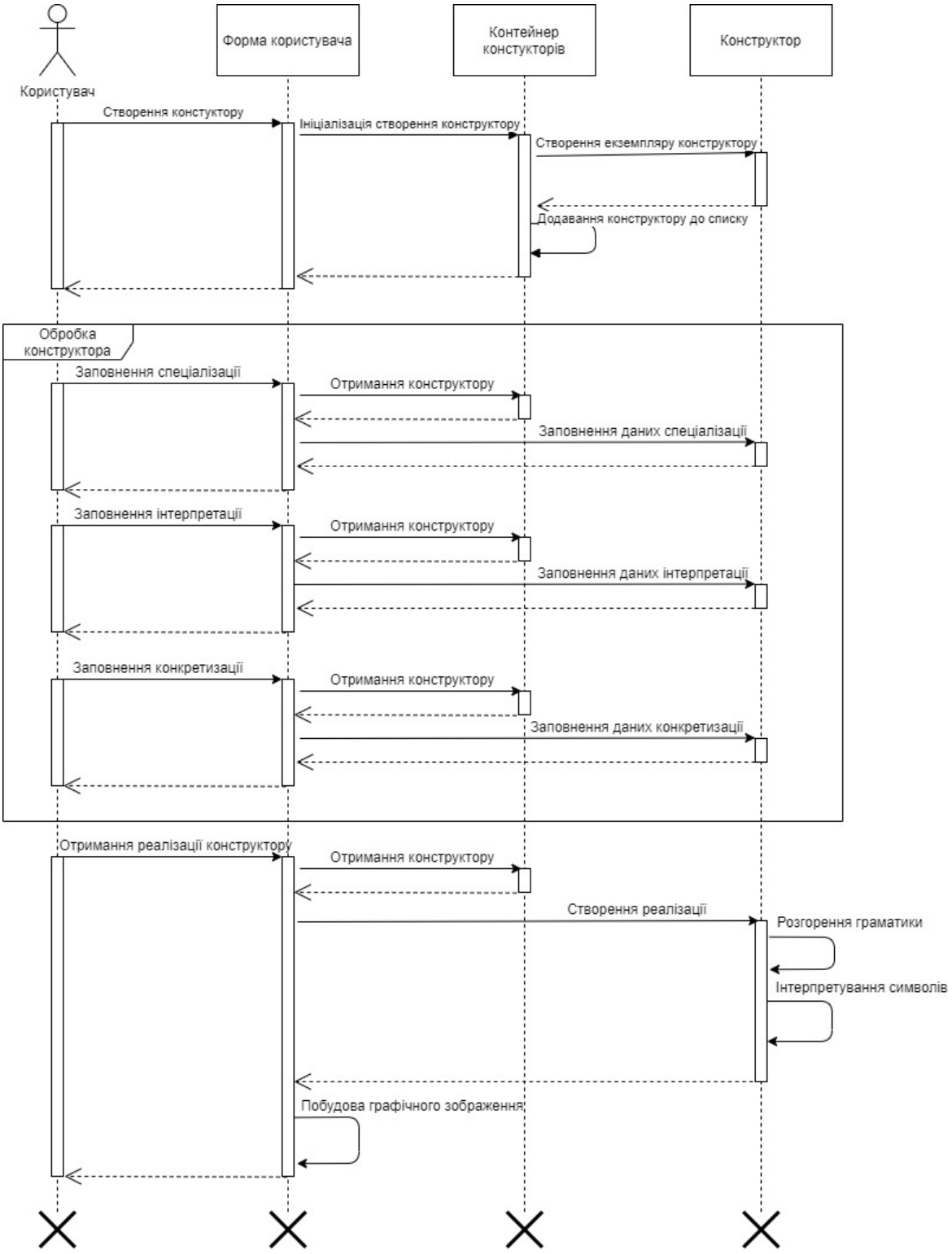


Рисунок 3.3 – Діаграма послідовності взаємодії користувача с програмною системою з виділенням компонентів

3.3 Опис базових сутностей

На основі цілей представлених в даній роботі, виділяється за основну робочу сутність [49] модель конструктору.

Сутність конструктору являє собою базову модель датакласу, що зберігає у собі основу інформацію про характеристики конструктору. Конструктор зберігає інформацію, розділену на чотири частини:

- спеціалізація;
- інтерпретація;
- конкретизація;
- реалізація.

Інформація спеціалізації представляє собою множину атрибутів конструктору з обмеженнями на збережену інформацію в них. На основі описаних атрибутів, в спеціалізації зберігається форма представлення та інформація о даних для відображення.

До інтерпретації конструктору належать підключений набір функцій та інформація інтерпретування символів граматики. Інтерпретування символів виконується через зв’язування символів та функцій, а також опис вхідних та вихідних даних виконання функції.

Конкретизація конструктору виконується у описі початкових значення атрибутів відповідно до їх обмежень створених у спеціалізації конструктору.

Кінцевою частиною конструктору є створення реалізації. Спочатку, конструктор на основі атрибуту правил підстановок утворює на основі правил лівосторонніх граматик кінцеву символьну стрічку. Після розгорнення стрічки, починається інтерпретування символів на основі правил інтерпретування описаних в інтерпретації конструктору. Завершення реалізації характеризується означенням масивів даних, що будуть використовуватися при створені графічного відображення.

На основі сутності конструктору описуються сутності простого конструктору та конструктору алгоритмів. Приклад алгоритму роботи з сутностями [50] наведено на рис.3.4.

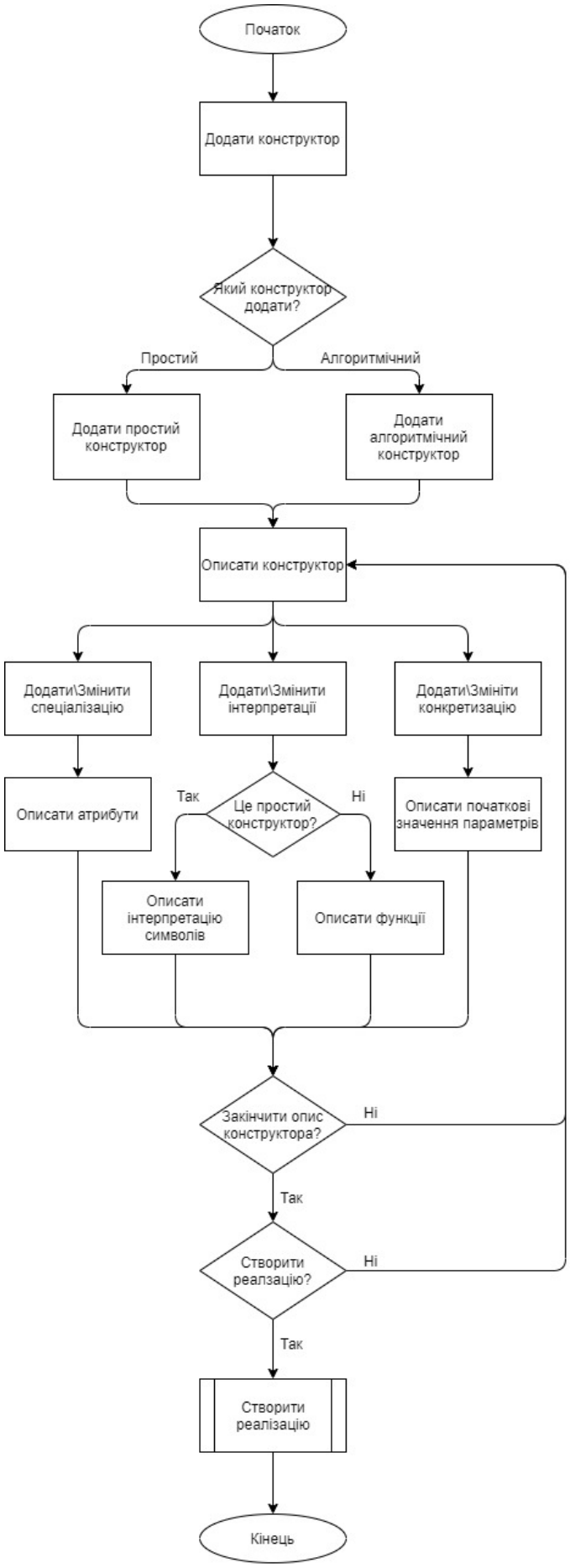


Рисунок 3.4 – Алгоритм створення та реалізації конструктору

Простий конструктор є розширенням конструктору, якому ставиться при ініціалізації базові атрибути, як аксіома чи правила підстановки, а також можливість підключення конструктору алгоритмів до себе, що зберігає описані функції для підключення через їх до символів.

На відміну від простого конструктору, у конструктору алгоритмів спеціалізація динамічно наповнюється на основі атрибути описаних функцій інтерпретування. Також, інтерпретація конструктору потребує прописування функцій, що можуть використовуватися простими конструкторами.

Додатковою сутністю виділяється контейнер конструкторів, що зберігає у собі всі проініціалізовані конструкторами да забезпечує доступ до них.

Описана інформація про логічні сутності може бути згрупована у табличному вигляді та представлена у табл.3.1.

Таблиця 3.1 – Базові сутності та їх функції

Сутність	Інформаційне наповнення	Функціональні особливості
Базовий конструктор	Базові структурні частини	Зберігання атрибутів Зберігання частин конструктору
Простий конструктор	Набір особин (хромосом) одного виду визначеного розміру	Побудова реалізації
Алгоритмічний конструктор	Функції інтерпретування	Зберігання функцій
Контейнер конструкторів	Набір конструкторів	Оперування набором конструкторів Формування конструкторів

До описаних сутностій можливо представлення їх атрибутів та функцій.

Деталізація сутностей [51] наведена на рис.3.5.

Рисунок 3.5 – Базова діаграма класів сутностей та їх зв’язку

3.4 Опис сценаріїв роботи з програмним забезпеченням

При роботі з програмним забезпечення користувачу надаються права створювати конструктори. Програма через діалог з користувачем створює один з видів конструктора:

- простий конструктор
- алгоритмічний конструктор

Конструктор в системі ідентифікується по введеній користувачем назві.

В кінці діалогу створення, конструктор додається до списку конструкторів та відображається в списку доступних конструкторів. Після чого користувач у будь-якій послідовності звертається до складових частин обраного конструктору:

- спеціалізації;
- інтерпретації;
- конкретизації;
- реалізації.

В діалозі спеціалізації користувачеві надаються права створення та видалення атрибутів з конструктору, визначення форми відображення та обрання атрибутів на відображення. Даний діалог доступний лише при роботі з простим конструктором, коли в алгоритмічному конструкторі, дані значення вважаються вже заповненими. Кожен створений атрибут має мати унікальний ідентифікатор-назву для системи та має описаний тип даних, що обмежує область значень, що приймає атрибут.

В діалозі інтерпретації простого конструктору користувач підключає алгоритмічний конструктор, функції з якого будуть транслюватися при інтерпретації символів. Користувач описує четвірки – символ, функції, вхідні значення, вихідне значення. Порядок описаних послідовно функцій інтерпретування до одного символу мають вплив на послідовність їх виконання на етапі реалізації.

В діалозі конкретизації користувач до кожного атрибута, що представлений у спеціалізації конструктору, вводить початкове значення відповідно до обмежень описаних у атрибуту.

При створені реалізації конструктору, перед користувачем відображається отримана розгорнута кінцева символьна стрічка, виводяться усі атрибути конструктора з їх останніми значеннями та графічне відображення.

Висновки до третього розділу

У третьому розділі було розглянуте зовнішнє та внутрішнє проектування програмної системи. Був описаний процес розробки програмного забезпечення, який складається з:

- проектування архітектури програмної системи;
- проектування сценаріїв роботи з програмою;

При проектуванні програмного за основну архітектуру була обрана монолітна архітектура, що дозволяє зосередити програмні можливості та пришвидшити створення реалізації.

Монолітна архітектура має недоліки, але через обмеженість ресурсів та простоту розгорнення та представлення вона покращує подальше використання та можливість модифікації продукту.

При подальших дослідженнях конструктивно-продуційних граматик можливо розгляд архітектури, що базується на розділені чи подійно-орієнтовані архітектурні варіанти.

При визначені програмного забезпечення одним із важливих елементів є створення сценарії роботи з програмним забезпеченням, що представляють основні дії, що будуть демонструвати хід роботи експериментування.

Користувачу надається можливість створювати конструктори двох видів, що зберігаються в єдиному списку. Після створення, будь-який з існуючих конструкторів можливо заповнити інформації, з можливістю подальшої її модифікації. Після заповнення інформаційної частини конструктора, конструктору посилається запит на створення реалізації. Будь який з конструкторів можливо видалити при потребі.

На основі вхідних значень та обраних вихідних значень, були розроблені базові структурні сутності, що будуть використовуватися в програмному забезпеченні при виконані дослідів. Були виділені сутності конструкторів, таких як простий конструктор, що використовується для реалізації конструктивно-продуційного підходу, та алгоритмічний конструктор, що дозволяє створювати функції інтерпретування, їх основні функціональні особливості, а також контейнер конструкторів, який здійснює зберігання та передачу керування конструкторами.

Таким чином, на основі сценарії роботи користувача на основі поставлених задач та базових сутностей конструкторів утворюється повна система умов та задач для розробки програмного забезпечення. Дана система уніфікує створення та роботу з конструкторами продуційних граматик.

4 ДОСЛІДЖЕННЯ БІЄКТИВНИХ ЗОБРАЖЕНЬ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

4.1 Порядок випробування

Використовуючи програмне забезпечення отримане під час дослідження, був розроблено план дослідження моделей процесів описаних продуційними граматиками.

План випробувань складається з наступних пунктів:

– формування базової випробувальної моделі – вибір даних для досліджень;
– розробка функцій інтерпретування даних – описання функцій на мові програмування Python для інтеграції в систему програми;
– побудова моделей на програмному продукті – створення и заповнення інформацією продуційних конструкторів в програмному середовищі;
– порівняння отриманих даних моделей – групування результатів реалізації конструкторів з подальшим аналізом;
– формування аналізу отриманої модельної інформації – описання аналізу експериментів та створення висновків про дослідження.

На початкових етапах формується модель, наповнюється інформацією й вводиться в систему програму для випробувань. Після введення моделі в систему, за допомогою програми формуються дані для подальшого аналізу. Для аналізу отримані графічні зображення для одних й тих самих даних порівнюються та розглядаються на наявність самоподібності.

Основною метою дослідження на основі даного плану є аналіз фрактальних характеристик граматики й отриманих відображень та доказ переходу характеристик при різних відображеннях.

4.2 Формування базової моделі

Для дослідження на фрактальні характеристики за базову модель обирається вже існуючий об’єкт, що має у собі властивості фракталу.

Беручи до уваги, що дослідження основується на розгорненні граматик, за експериментальні моделі обираються моделі, розроблені на базі системи Лінденмайера – «Сніжинка Коха» та «Крива дракону».

«Сніжинка Коха» описується за наступними правилами:

– аксіома – $f++f++f$;
– правила підстановок – $f \to f-f++f-f$;
– значення кута повороту – $60°$;
– початкове значення поточного кута повороту - $0°$;
– початкове значення на координатній площині – $(0, 0)$.

«Крива дракону» описується за наступними правилами:

– аксіома – fx;
– правила підстановок – $x \to x + yf+, y \to -fx - y$;
– значення кута повороту – $90°$;
– початкове значення поточного кута повороту - $0°$;
– початкове значення на координатній площині – $(0, 0)$.

Вигляд «Сніжинки Коха» без стохастичних властивостей наведено на рис.4.1 [52]. Вигляд «Кривої дракона» наведено на рис.4.2 [53].

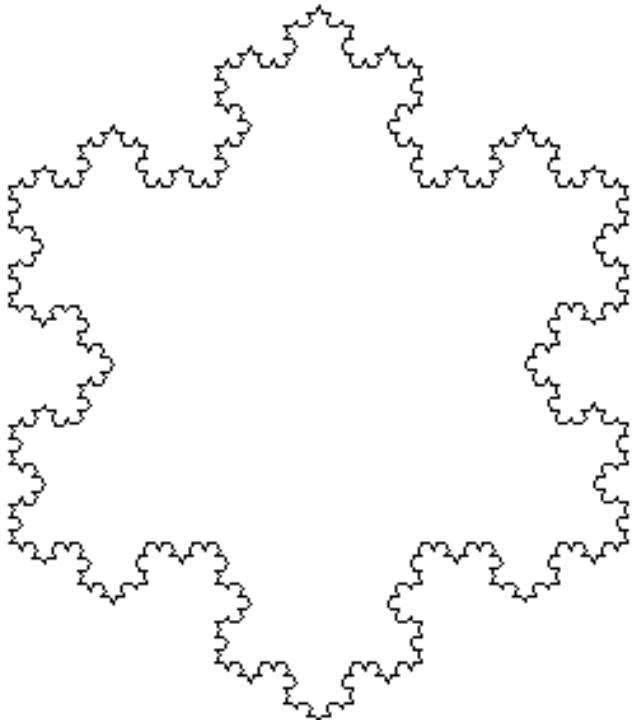


Рисунок 4.1 – «Сніжинка Коха» без стохастичних властивостей

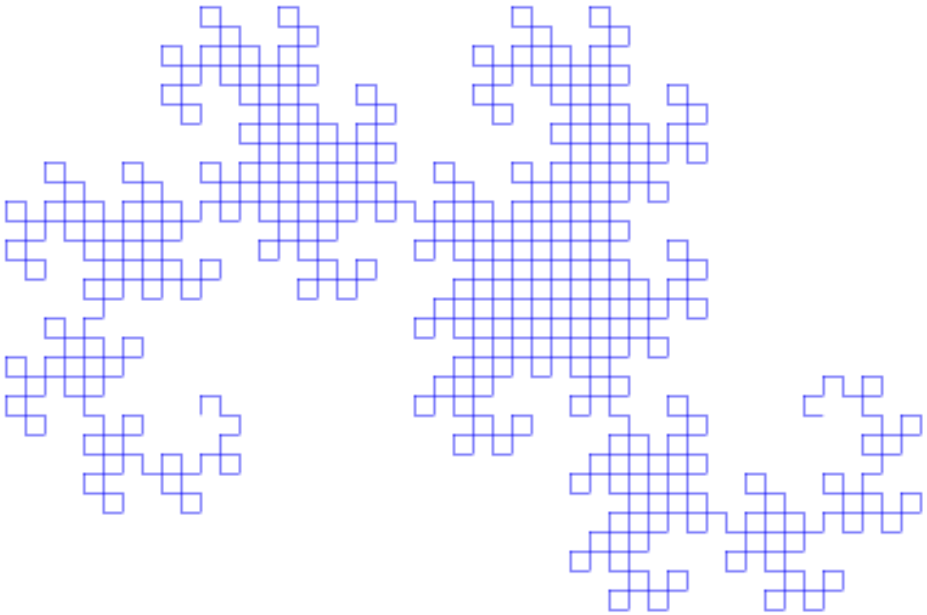


Рисунок 4.2 – «Крива дракону» без стохастичних властивостей

4.3 Розробка функцій інтерпретування даних

На етапі інтерпретації простих конструкторів виконується позначення інтерпретування символів через виконання певних функції, що підключаються до конструктору через додавання конструктору алгоритмів до нього.

Для дослідження був розроблений алгоритмічний конструктор, що містить у собі набір функцій для подальшого використання на етапах інтерпретування конструкторів моделей.

Серед функцій конструктора є допоміжні функції, що можуть бути використані для розрахунків та передачі даних між атрибутами, так і спеціалізовані модельні функції для певних відображень. Прикладом спеціалізованих функції є функції отримання точки на колі певного радіусу за кутом від початку координат [54].

Оскільки розроблений конструктор буде використовуватися при інтерпретації як часового ряду, так і L-системи, то прийнято рішення зберігати функції в одному конструкторі, а не розробляти кожному простому конструктору окремий алгоритмічний конструктор.

Розроблені функції для алгоритмічного конструктору наведено у табл.4.1.

Таблиця 4.1 – Набір функцій інтерпретування

Сутність	Інформаційне наповнення Функціональні особливості
decrement_by_value	def decrement_by_value(var, value): return var - value
increment_by_value	def increment_by_value(var, value): return var + value
simple_return	def simple_return(var): return var
get_point	def get_point(mat_oz, disp): import random return round(random.normalvariate(mat_oz, disp ** 0.5), 3)
rotate_point_x	def rotate_point_x(x, angle, step): import math next_point = x + math.cos(math.radians(angle)) * step return next_point
rotate_point_y	def rotate_point_y(y, angle, step): import math next_point = y + math.sin(math.radians(angle)) * step return next_point

4.4 Побудова моделей на програмному продукті

Для дослідження необхідно створити три конструктори: два прості продуційні конструктори, що відображають модель часового ряду й L-системи, та один конструктор алгоритмів, що містить у собі функції з табл.4.1.

Першим етапом є створення алгоритмічного конструктора ac_time_plot, для подальшого підключення до простих конструкторів.

Результат наповнення конструктору алгоритмів наведено на рис.4.3.

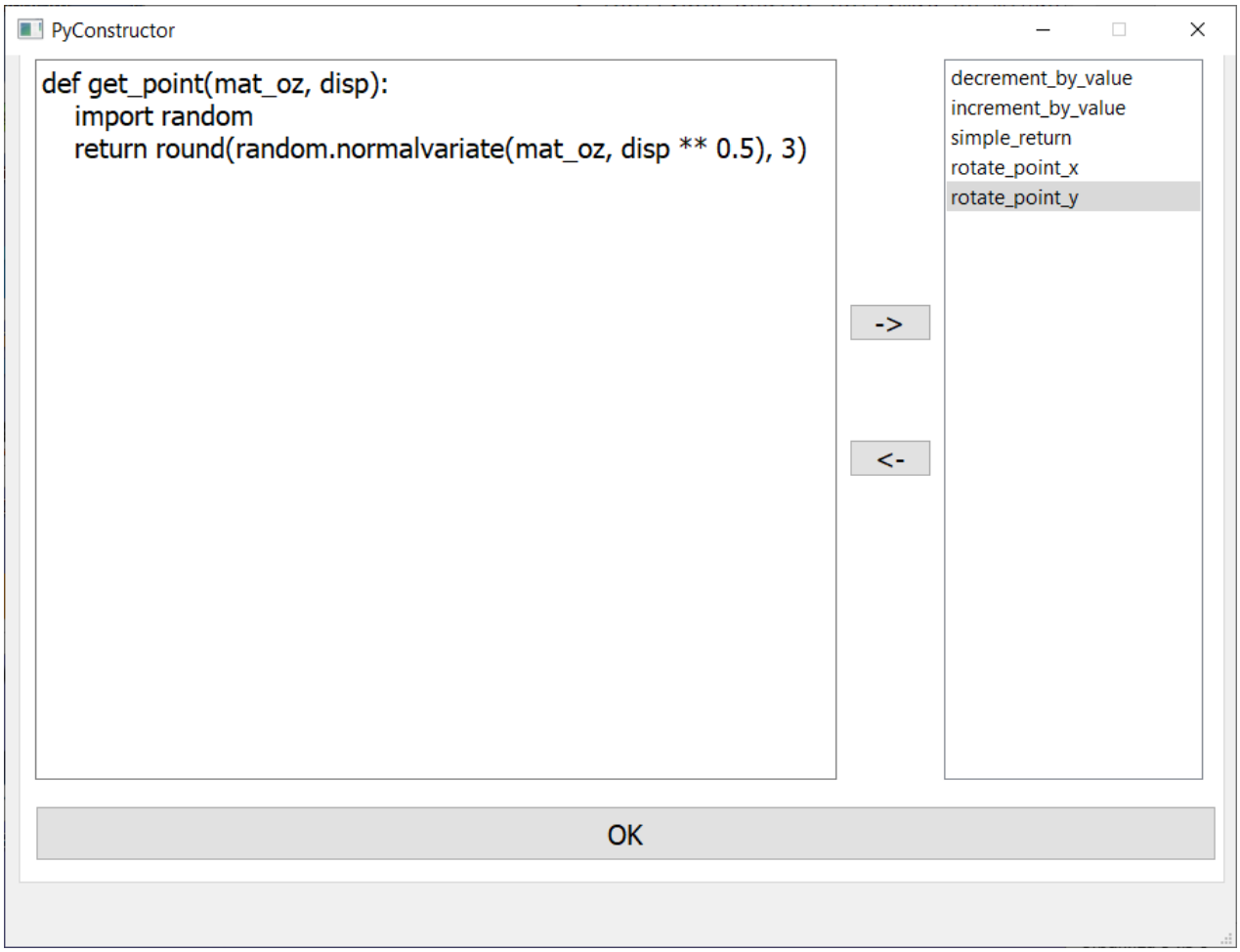


Рисунок 4.3 – «Крива дракону» без стохастичних властивостей

Наступним кроком є створення простих конструкторів time_plot_con (для часового ряду) та l-sys_con (для L-системи).

Спочатку кожному конструктору створюється спеціалізація – треба означити набір атрибутів, що належать до конструктора.

Кожен з простих конструкторів має обов'язковий набір атрибутів, який наведено у табл.4.2.

Таблиця 4.2 – Обов'язкові атрибути конструктора

Назва атрибуту	Тип даних	Значення атрибуту
acsioma	str	Початкова символьна стрічка (аксіома)
rules	dict	Словник правил підстановки
iterator_flag	bool	Флаг типу розгортання
iterations	int	Кількість ітерації розгортання
points	int	Мінімальна кількість точок розгортання

Для продуційного конструктору часового ряду список атрибутів наведено у табл.4.3, для конструктору L-системи – у табл.4.4.

Таблиця 4.3 – Атрибути конструктора часового ряду

Назва атрибуту	Тип даних	Значення атрибуту
x	list	Список точок конструктору по осі абсцис
y	list	Список точок конструктору по осі ординат
mat_oz_level	float	Значення математичного очікування рівня ряду
disp_level	float	Значення дисперсії рівня ряду
current_y	float	Поточне значення періоду
mat_oz_period	float	Значення математичного очікування періоду
disp_period	float	Значення дисперсії періоду
step_period	float	Шаг зміни періоду
step_level	float	Шаг зміни рівня ряду

Таблиця 4.4 – Атрибути конструктора L-системи

Назва атрибуту	Тип даних	Значення атрибуту
x	list	Список точок конструктору по осі абсцис
y	list	Список точок конструктору по осі ординат
mat_oz_len	float	Значення математичного очікування довжини лінії
disp_len	float	Значення дисперсії довжини лінії
step_len	float	Шаг зміни довжини лінії
mat_oz_angle	float	Значення математичного очікування кута повороту
disp_angle	float	Значення дисперсії кута повороту
step_angle	float	Шаг зміни кута повороту
current_angle	float	Поточне значення кута повороту
current_x	float	Поточне значення по осі абсцис
current_y	float	Поточне значення по осі ординат

Для етапу інтерпретації простий конструктор часового ряду і конструктор L-системи використовують алгоритмічний конструктор ac_time_plot.

Для основної дії створення точок чи проведення лінії резервується символ f, а для змінювання поточних значень чи повороту використовується символи $\{+, -\}$ [55].

Означення інтерпретування символів конструктору часового ряду наведено у табл.4.5, означення символів конструктору L-системи – у табл.4.6.

Таблиця 4.5 – Інтерпретування символів конструктору часового ряду

Символ	Виконувана функція	Вхідні атрибути	Вихідний атрибут
f	get_point	mat_oz_level, disp_level	x
f	get_point	mat_oz_period, disp_perion	step_period
f	increment_by_value	current_y, step_period	current_y
f	simple_return	current_y	y
+	increment_by_value	mat_oz_level, step_level	mat_oz_level
-	decrement_by_value	mat_oz_level, step_level	mat_oz_level

Таблиця 4.6 – Інтерпретування символів конструктору L-системи

Символ	Виконувана функція	Вхідні атрибути	Вихідний атрибут
f	get_point	mat_oz_len, disp_len	step_len
f	rotate_point_x	current_x, current_angle, step_len	current_x
f	rotate_point_y	current_y, current_angle, step_len	current_y
f	simple_return	current_x	x
f	simple_return	current_y	y
+	get_point	mat_oz_angle, disp_angle	step_angle
+	increment_by_value	current_angle, step_angle	current_angle
-	get_point	mat_oz_angle, disp_angle	step_angle
-	decrement_by_value	current_angle, step_angle	current_angle

Для кожного атрибуту необхідно позначити початкові значення з якими конструктор почне створювати реалізацію.

Початкові значення атрибутів обираються відповідно до описаних моделей.

Конкретизація атрибутів конструктору часового ряду наведено у табл.4.7, а для конструктору L-системи у табл.4.8.

Таблиця 4.7 – Початкові значення атрибутів конструктору часового ряду

Назва атрибуту	Значення атрибуту
x	[0]
y	[0]
mat_oz_level	60.0 – для «Сніжинки Коха» 90.0 – для «Кривої дракону»
disp_level	0.0
current_y	0.0
mat_oz_period	1.0
disp_period	0.0
step_period	0.0
step_level	60.0 – для «Сніжинки Коха» 90.0 – для «Кривої дракону»
acsioma	f++f++f – для «Сніжинки Коха» fx – для «Кривої дракону»
rules	{«f»:[«f-f++f-f»]} – для «Сніжинки Коха» {"x": ["x+yf+"], "y": ["-fx-y"]} – для «Кривої дракону»
iterator_flag	true
iterations	3
points	100

Таблиця 4.8 – Початкові значення атрибутів конструктору L-системи

Назва атрибуту	Значення атрибуту
x	[0]
y	[0]
mat_oz_len	1.0
disp_len	0.0
step_len	0.0
mat_oz_angle	60.0 – для «Сніжинки Коха» 90.0 – для «Кривої дракону»
disp_angle	0.0
step_angle	0.0
current_angle	0.0
current_x	0.0
current_y	0.0
acsioma	f++f++f – для «Сніжинки Коха» fx – для «Кривої дракону»
rules	{«f»:[«f-f++f-f»]} – для «Сніжинки Коха» {"x": ["x+yf+"], "y": ["-fx-y"]} – для «Кривої дракону»
iterator_flag	true
iterations	3
points	100

Для подальшого порівняння визначаються бієктивні зіставлення атрибутів.

Для при дослідженні заставляються значення рівня ряду до значень куту повороту, а період часового ряду до довжини відрізку L-системи.

Також, за параметр, що наділяється стохастичними властивостями [56] ставиться рівень ряду (у часового ряду) та кут повороту (у L-системи). Для отримання стохастичніх моделей, виставляється значення disp_level та disp_angle рівними 10.

У табл.4.9 наведено бієктивне зіставлення [57] атрибутів конструкторів часового ряду і L-системи.

Таблиця 4.9 – Зіставлення атрибутів конструкторів

Атрибут конструктору часового ряду	Атрибут конструктору L-системи
mat_oz_level	mat_oz_angle
disp_level	disp_angle
mat_oz_period	mat_oz_len
disp_period	disp_len

4.5 Побудова моделей на програмному продукті

Для кожної моделі у результаті розгорнення конструктору у реалізації отримуємо значення кінцевої символьної стрічки та атрибутів, а також графічне відображення.

При дослідженні розглядаються графічні відображення у виглядів часового ряду та графічного двовимірного фракталу отриманого. Кожне відображення розглядається у двох варіантах: за статичним створенням значень множин та зі стохастичним створенням.

Отримані графічні результати реалізації моделей наведено на рисунках 4.3-4.10.

При розгляді відображень помітно, що у часових рядів помітні незначні відхилення у точок отриманих стохастично від статичних значень. Форма графіків часових рядів практично однакова, але за незначні відхилення при розгляді графічних фракталів, мали великий вплив на зміну форми, яка все рівно має чітко виражено самоподобу.

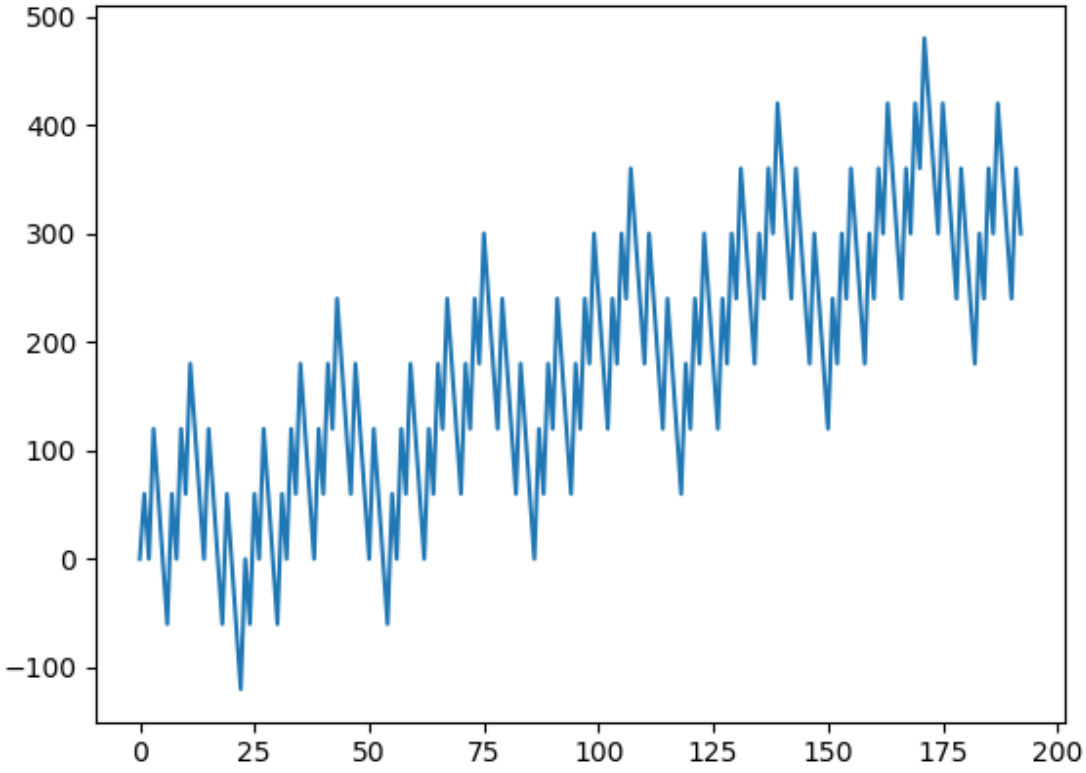


Рисунок 4.3 – Реалізований часовий ряд фракталу «Сніжинка Коха» без стохастичних властивостей

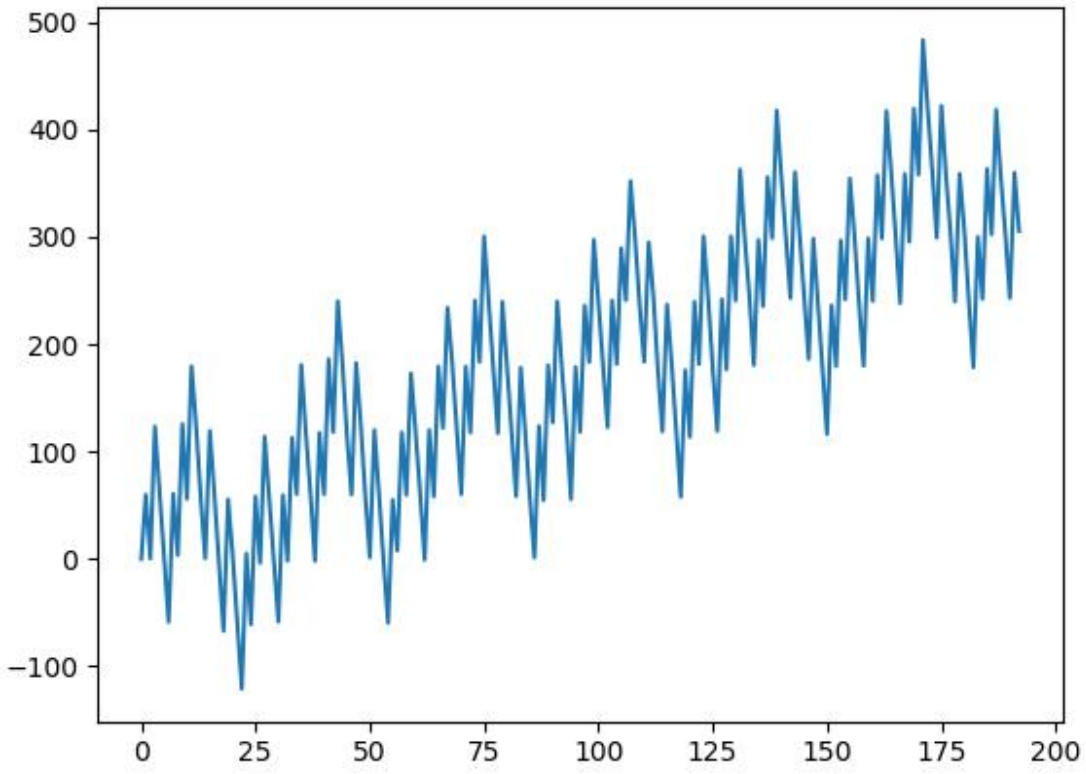


Рисунок 4.4 – Реалізований часовий ряд фракталу «Сніжинка Коха» зі стохастичними властивостями

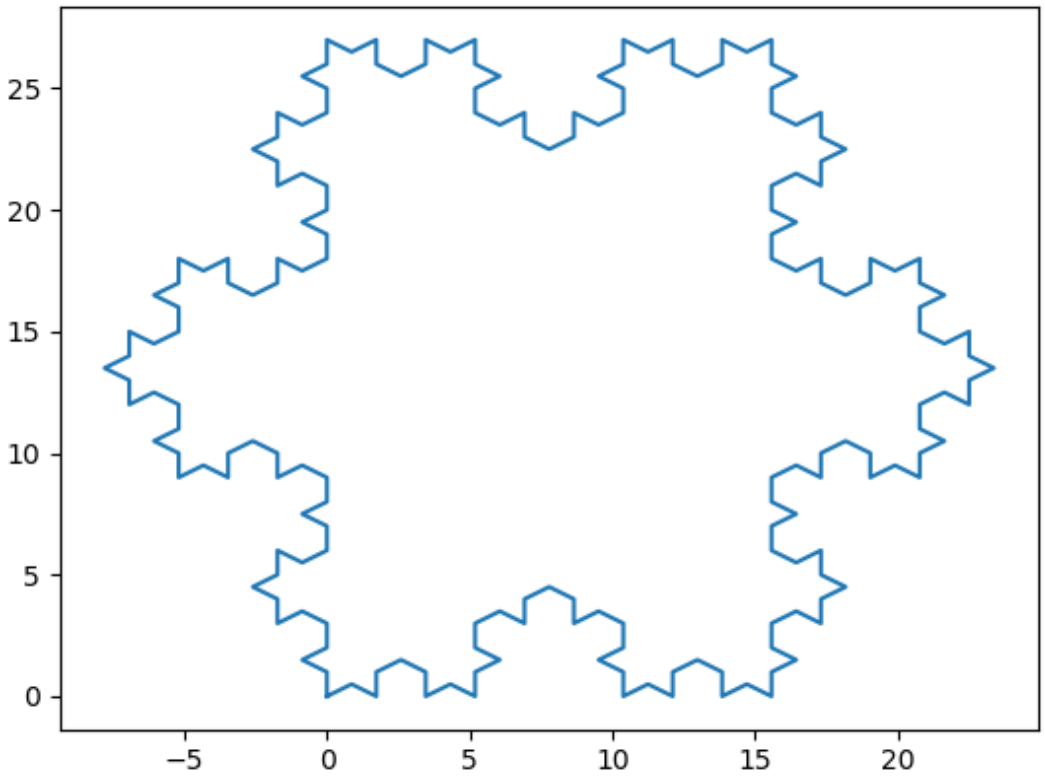


Рисунок 4.5 – Реалізоване графічне відображення фракталу «Сніжинка Коха» зі стохастичними властивостями

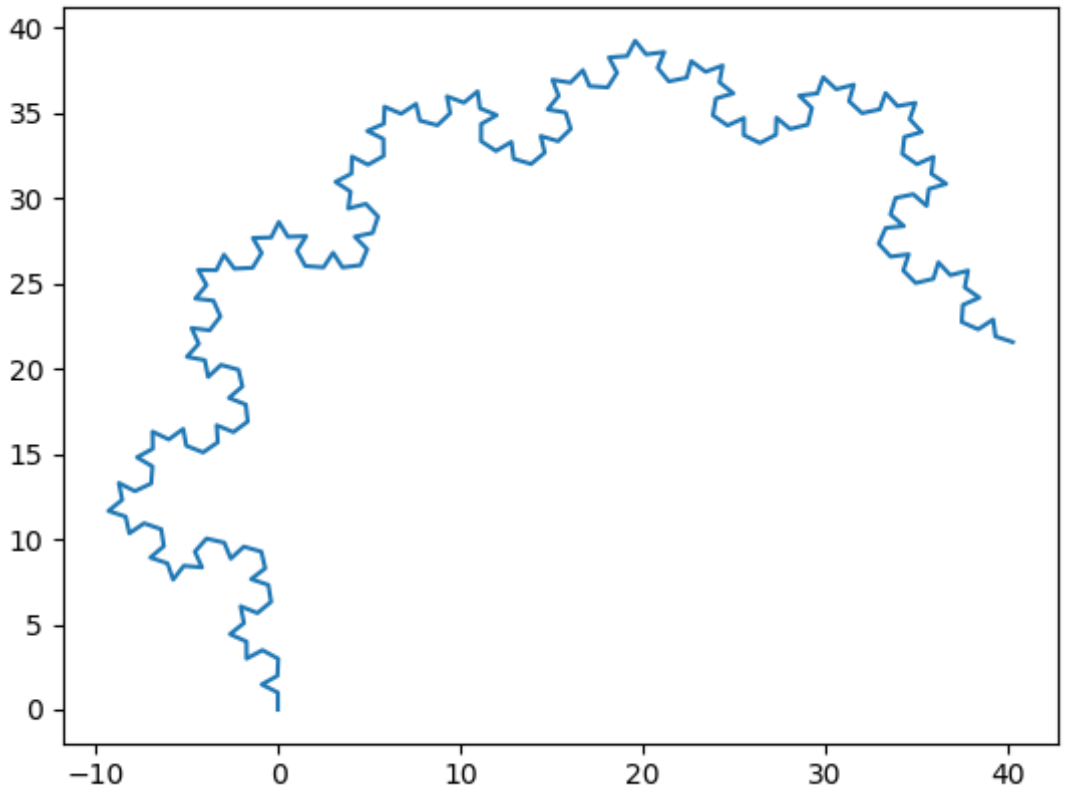


Рисунок 4.6 – Реалізоване графічне відображення фракталу «Сніжинка Коха» без стохастичних властивостей

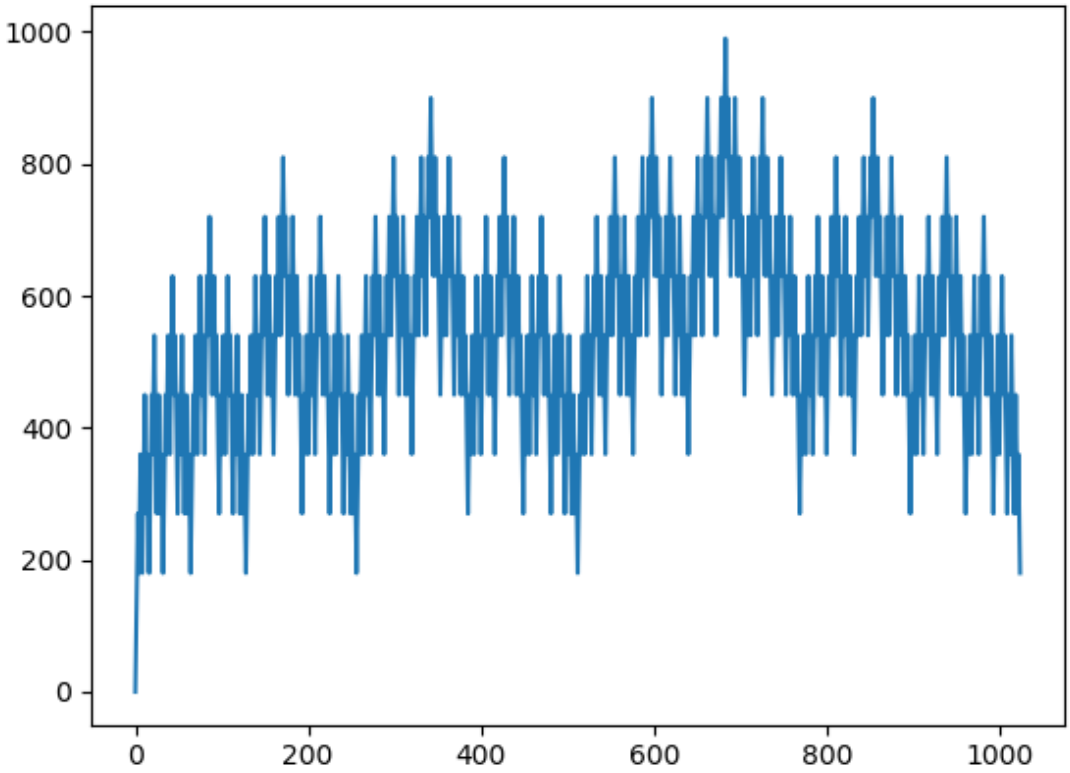


Рисунок 4.7 – Реалізований часовий ряд фракталу «Крива дракону» без стохастичних властивостей

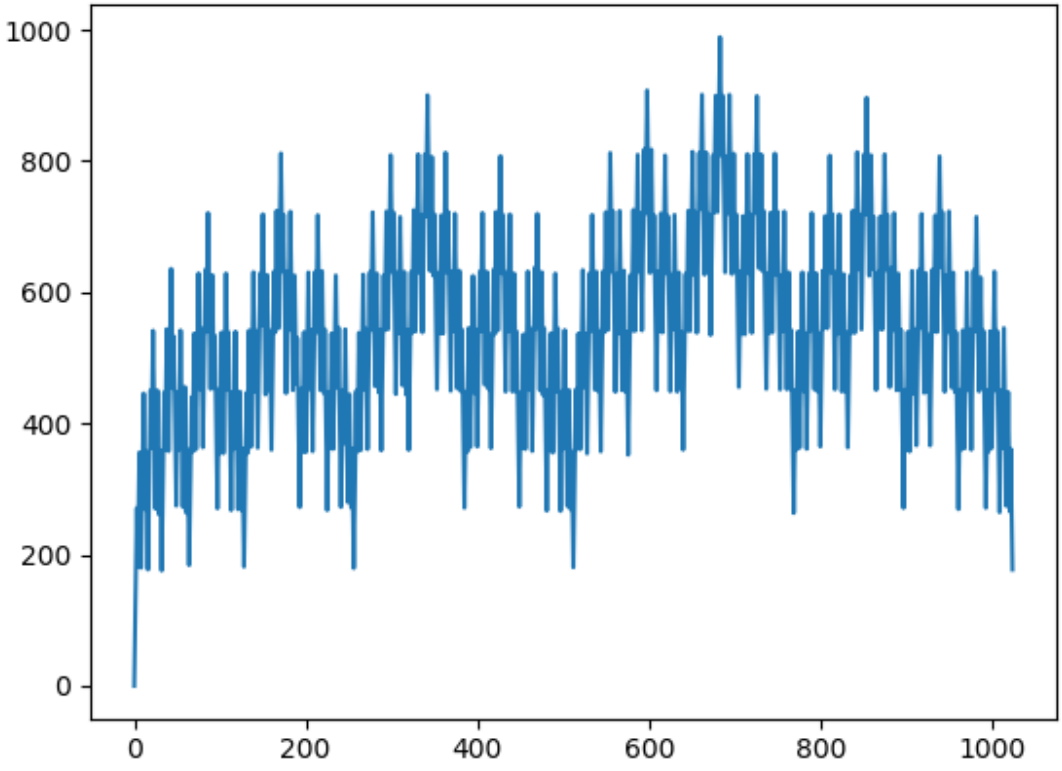


Рисунок 4.8 – Реалізований часовий ряд фракталу «Крива дракону» зі стохастичними властивостями

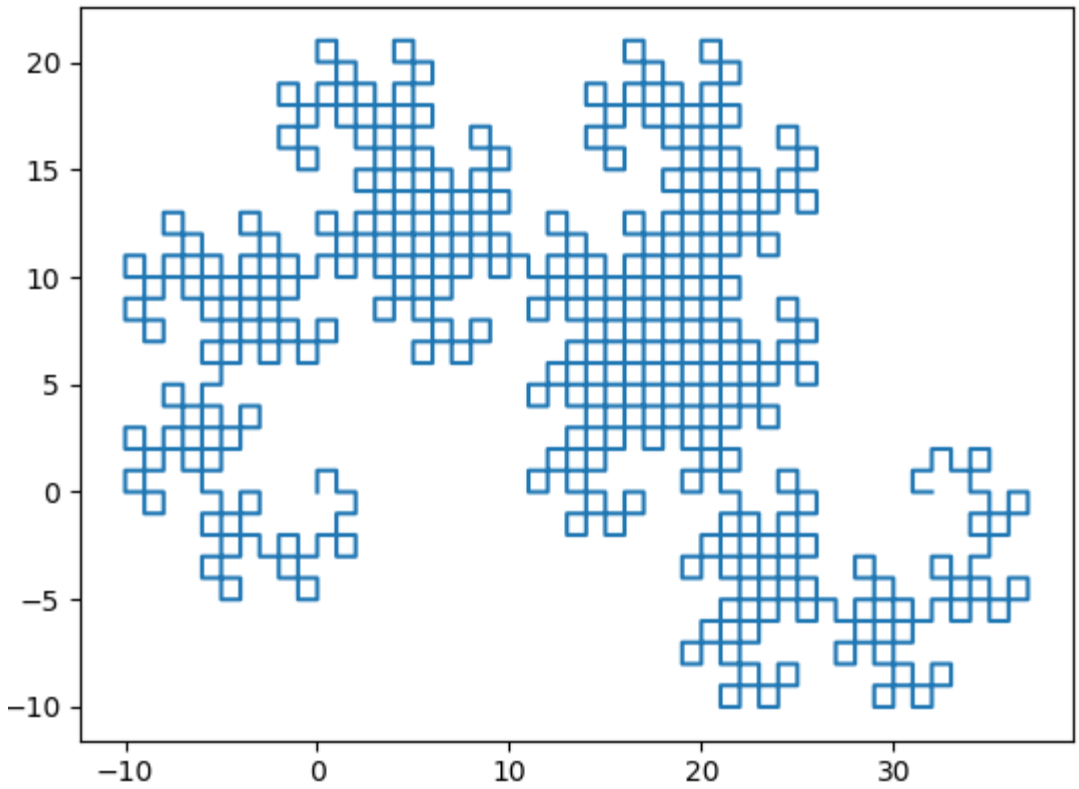


Рисунок 4.9 – Реалізоване графічне відображення фракталу «Крива дракону» зі стохастичними властивостями

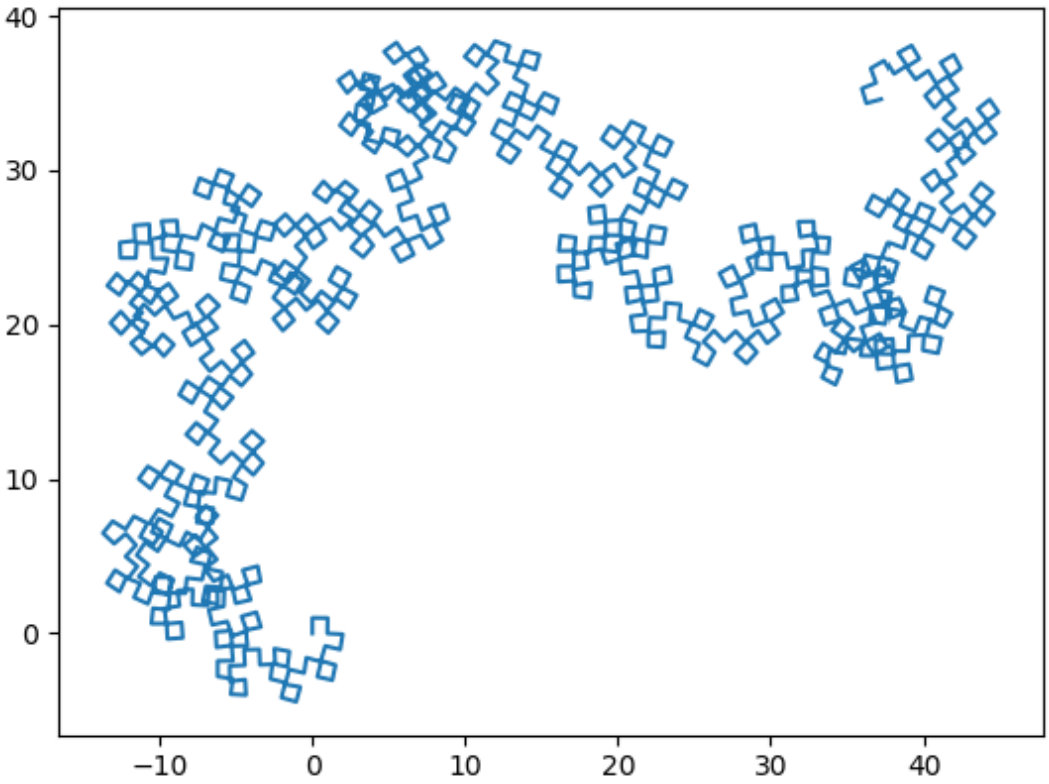


Рисунок 4.10 – Реалізоване графічне відображення фракталу «Крива дракону» без стохастичних властивостей

Кінцева стрічка «Кривої дракону» – fx+yf++-fx-yf++-fx+yf+--fx-yf++-fx+yf++-fx-yf+--fx+yf+--fx-yf++-fx+yf++-fx-yf++-fx+yf+--fx-yf+--fx+yf++-fx-yf+--fx+yf+--fx-yf++-…

Кінцева стрічка «Сніжинки Коха» - f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f…

4.6 Формування аналізу отриманої модельної інформації

Аналіз отриманих результатів можна розділи на аналіз кінцевої стрічки граматики та на аналіз графічних зображень.

Розглядаючи кінцеву стрічку, її розгортання за допомогою правил підстановок, можна зробити висновок о існуванні властивості самоподібності через повторення конструкції правил у частинах стрічки.

При розгляді графічних зображень, факт існування фрактальниї властивостей у класичних фракталах «Сніжинка Коха» та «Крива дракону» не ставиться під сумнів. З цього, отримані часові ряди через створення бієктивного відображення, теж має властивість самоподібності як і зображення L-системи чи у вигляді кінцевої стрічки граматики. Тому можна припускати о існуванні транзитивності фрактальних властивості між бієктивними відображеннями.

Розглядаючи стохастичні та не стохастичні форми можливо помітити схожість зображень. Оскільки, для створення точок використовується закон нормального розподілення, можна побачити, що фігури все рівно залишаються у рамках початкової структури, що базується на інтерпретації символів розгорнутої граматики.

Висновок до четвертого розділу

У четвертому розділі був описаний процес процес випробувань та дослідження бієктивних відображень моделей фракталів. Був послідовно пройдений процес моделювання зображень на основі продуційних конструкторів – від ідеї та створення моделей до отримання результатів реалізації.

Кожен з етапів був детально описаний та приведені дані я можливого повторення експериментів для подальших досліджень на інших процесах чи моделях.

У наслідку випробувань були отримані дані, на основі яких можна вважати о транзитивності властивостей фракталу між різними формами представлення такими як графічний фрактал та часовий ряд.

Оскільки, кожна з форм може мати фрактальні властивості, то ґрунтуючись на існуванні властивості самоподібності у формі символьної стрічки та форми представлення у вигляді геометричної фігури форма часового ряду теж має фрактальність.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Формування аналізу отриманої модельної інформації

Вимоги до виконання робіт на робочому місці регламентовані згідно до Закону України «Про охорону праці» від 16.10.20 р. прийнятий від 14 жовтня 1992 року №2695-12 [58]. Положення цього закону регламентують правила та вимоги до безпеки на робочому місці програміста-розробника, що умовно поділяються на санітарні та експлуатації електронних пристроїв.

До правил, норм та вимог відносять:

– Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99 від 01.12.99 р. [59] затверджені постановою Головного державного санітарного лікаря України від 01.12.99 р. №37;
– Державні санітарні норми виробничої загальної та локальної вібрації ДСН 3.3.6.039-99 від 01.12.99 р. [60] затверджені постановою Головного державного санітарного лікаря України від 01.12.99 р. №39;
– Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 від 10.12.98 р. [61] затверджені постановою Головного державного лікаря України від 10.12.98 р. №7;
– Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 від 01.12.99 р. [62] затверджені постановою Головного державного лікаря України від 01.12.99 р. №42;
– Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями НПАОП 0.00-7.15-18 від 14.02.18 р. [63] затверджені наказом Міністерства соціальної політики України "Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями" від 14.02.18 № 207;

- Загальні вимоги стосовно забезпечення роботодавцями охорони праці працівників НПАОП 0.00-7.11-12 від 25.01.12 р. [64] затверджені наказом Міністерства надзвичайних ситуацій України від 25.01.12 №67;
- Державні санітарні норми та правила «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» від 08.04.14 р. [65] прийняті наказом Міністерством охорони здоров'я України від 08.04.14 №248;
- ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки від 14.10.13 р. [66] затверджений наказом від 14.10.13 р. №1231;
- ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення від 21.07.03 р. [67] затверджений наказом від 21.07.03 р. №126;
- ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища від 28.10.04 р. [68] затверджений наказом від 28.10.04 р. №237.

Основними вимогами безпеки при виконанні робіт програмістами-розробниками можна вважати:

- вимогу до зменшення шкідливого випромінювання від електронних терміналів та електронних пристроїв;
- вимогу про надання необхідного рівня освітлення на робочому місці;
- вимогу про дотримання відповідного рівня мікроклімату на робочому місці;
- вимогу про надання безпечного, ергономічного робочого місця з необхідним вільним простором.

Працівник має право на отримання безпечних умов роботи з дотриманням необхідних мінімальних санітарних норм на підприємстві.

5.2 Шкідливі виробничі фактори на робочому місці

При роботі за електронними дисплеями та з електронними пристроями на програміста-розробника можуть діяти шкідливі виробничі умови, які виникають у наслідок недостатньо організації безпеки та санітарних норм на робочому місці.

До основних шкідливих факторів, що виникають перед людиною в рамках даного робочого місця є:

– недостатня або надмірна освітленість робочого місця;
– високий рівень шуму;
– високий рівень вібрації;
– високий рівень електромагнітного випромінювання;
– високий рівень статичної електрики;
– високий рівень теплового випромінювання;
– невідповідність норм ергономіки робочого місця до стандартів;
– порушення норм показників мікроклімату.

При виконанні роботи програмісту-розробнику надається приміщення розмірами:

– довжина: 5 м;
– ширина: 3 м;
– висота: 2.5 м;
– площа приміщення: 15 м^2.

5.2.1 Освітлення

Роботу за електроними дисплеями та електропристроями класифікуємо як для IV розряду зорових робіт.

Мінімальна освітленість робочого приміщення складає від 300 лк до 500 лк. Для освітлення робочого місця використовуються люмінесцентні лампи.

Для визначення необхідної та мінімальної кількості ламп для освітлення робочої поверхні використовуємо метод світлового потоку:

$$F = \frac{E * K * S * Z}{\eta},$$

де F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк (E = 300 Лк);

S – площа освітлюваного приміщення (у нашому випадку S=15 м^2);

Z – відношення середньої освітленості до мінімальної (Z =1,1);

K – коефіцієнт запасу (K = 1,5);

η – коефіцієнт використання світлового потоку.

Для визначення коефіцієнту використання світлового потоку, розрахуємо індекс приміщення:

$$I = \frac{S}{h(A+B)},$$

де S – площа приміщення, S = 15 м^2;

h – розрахункова висота підвісу ламп, h = 2,4 м;

A – ширина приміщення, A = 3 м;

B – довжина приміщення, B = 5 м.

Підставивши значення до формули отримуємо, що:

$$I = \frac{15}{2{,}4 * (3+5)} = 0{,}78$$

За індексом приміщення за табличними значеннями ДБН В.2.5-28-2018 "Природне і штучне освітлення" [69] отримуємо $\eta = 0{,}28$. За цим значенням розрахуємо світловий потік:

$$F = \frac{300 * 1{,}5 * 15 * 1{,}1}{0{,}28} = 26500$$

Отже, $F = 26500$ Лм, що дає розрахувати кількість ламп для освітлення.

Для освітлення візьмемо стандартні лампи типу ЛБ 40-1 зі світловим потоком $F = 4320$ Лм.

Для розрахунку кількості ламп скористаємося формулою:

$$N = \frac{F}{F_л},$$

де N – кількість ламп, що визначається;

F - світловий потік (F = 26500 Лм);

Fл- світловий потік лампи (Fл = 4320 Лм).

За цими даними:

$$N = \frac{26500}{4320} = 6.$$

Отже, для мінімального освітлення необхідно 6 працюючих ламп.

Схема робочого приміщення з розташуванням ламп наведена на рис.5.1.

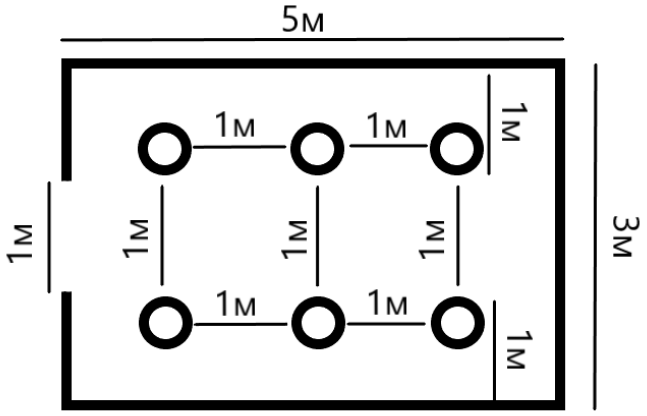


Рисунок 5.1 – Схема розташування ламп освітлення

5.2.2 Мікроклімат

Мікрокліматом приміщення називається сукупність показників внутрішніх фізичних показників середовища приміщення.

Показниками мікроклімату є:

– температура повітря в приміщенні;

– відносна вологість;

– швидкість руху повітря.

У табл.5.1 наведені показники мікроклімату в робочому приміщенні в залежності від періоду року.

Параметри мікроклімату розглядаються в рамках постачання свіжого повітря, що циркулює в межах робочого приміщення. Значення нормальних об’ємів циркуляції повітря наведено у табл.5.2.

Таблиця 5.1 - Параметри мікроклімату для приміщень з ПК

Період року	Параметри мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22- 24º С
	Відносна вологість	40-60%
	Швидкість руху повітря	До 0.1 м/с
Теплий	Температура повітря в приміщенні	23-25º
	Відносна пологість	40-60%
	Швидкість руху повітря	До 0.1-0.2 м/с

Таблиця 5.2 – Норми подачі свіжого повітря в приміщення з ПК

Характеристика приміщення	Об’ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об’єм до 20 $м^3$ на людину	Не менше 30
20-40 $м^3$ на людину	Не менше 20
Більше 40 $м^3$ на людину	Може бути використана природна вентиляція

5.2.3 Шум та вібрація

При розгляданні шкідливої дії шуму та вібрації на внутрішній стан людини слід розрізняти чинники цих явищ.

Шум викликається будь-яким процесом, у якого його звук перевищує певний рівень та викликане негативні відчуття в організмі людини.

Шум нормується відповідно до

Вібрація утворюється внаслідок фізичної дії об’єктів та процесів (будівництво, пересування, тощо), та може викликати дисонанс в організмі людини.

Шум і вібрація нормується відповідно до значень наведених у [59].

5.3 Вимоги безпеки при виконанні робіт на робочому місці

Головною небезпекою програміста-розробника на робочому місті можна вважати ураження електрострумом, що призводить до електротравматизму.

При виникненні ситуації з ураженням працівника від електротехніки, треба керуватися порядком дій описаних в документі «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою» від 16.06.14 р. [70] затвердженим наказом Міністерства охорони здоров'я України від 16.06.14 р. №398.

Спершу, треба переконатися у відсутності небезпеки. Наприклад, переконатися у тому, що електроприлад, що причинив ураження струмом, виключений та не становить більше загрози. Якщо прилад продовжує становити загрозу та є безпечний спосіб припинення дії загрози (вимкнення електрики чи вимкнення електропристрою від електромережі за допомогою електронепровідного засобу).

Після, треба оцінити стан постраждалого. Якщо постраждалий має проблеми з диханням треба провести штучне дихання та непрямий масаж серця. В цей час, необхідно викликати бригаду медичної допомоги.

Під час оцінки стану, треба звернути увагу на можливе місце опіку від електроструму.

При знаходженні місця опіку, треба накласти на це місце стерильну пов'язку.

Якщо під час нагляду за постраждалим до прибуття бригади невідкладної допомоги його стан погіршився, треба виконати повторний виклик бригади зателефонувавши на гарячу лінію.

Висновки до п‘ятого розділу

У розділі були описані основні питання стосовно безпеки на робочому місці.

Були розглянути основні нормативи, правила та вимоги щодо становлення безпечного робочого процесу для програміста розробника та приведені для подальшого використання у при розробці програмного забезпечення.

Були визначені показники мікроклімату та освітлення для нормалізації процесу за робочим місцем. Також, була наведена інструкція щодо наданні домедичної допомоги та діях при уражені електричним струмом.

ВИСНОВКИ

У результаті роботи над дипломним проектом було проведення дослідження виявлення фрактальних властивостей конструктивно-продуційних граматик у бієктивних відображень фракталів.

Були проведені досліди на класичних фракталах, таких як «Сніжинка Коха» та «Крива дракона», що мають виражені фрактальні властивості, існування яких не підлягає сумнівам. Серед властивостей, що розглядалися, важливу роль відіграє ознака самоподібності елементів фракталу у підфрактальних частинах.

Класичні фрактали, що розглядалися, були перетворені з вигляду L-системи у представлення конструктивно-продуційної граматики, що дозволило описати різні інтерпретуючі функції для отримання різних бієктивних зображень.

На початковому етапі дослідження були визначені сутності

У результаті досліджень були отримані графічні відображення у вигляді часових рядів та графічних двовимірних фракталів, у яких існує зв’язок між значеннями рівня ряду та кутом повороту лінії фігури.

Дані множин, отримані у реалізації конструкторів, створювалися зі застосування статичних та стохастичних функцій генерування даних. При створені даних використовувалося нормальне розподілення величин випадкових величин, що дозволяло надавати даним більший чи менший інтервальний розбіг.

У зображень, отриманих даних шляхом, були помічені виражені ознаки самоподібності, що є достатньою ознакою при ґрунтуванні доказу існування фрактальних властивостей у моделей.

Оскільки, самі моделі є класичними фракталами та отримані результуючі дані теж мають виражені фрактальні властивості, можливо робити припущення про існування транзитивності фрактальних властивостей при створенні бієктивних відображень, оскільки дані, що використовуються при відображені графічних зображені у кожному з випадків є одними й тими самими.

Дана робота описує послідовне проведення дослідів, що дає змогу повторного проведення дослідів та продовження дослідження при створенні додаткових

інтерпретуючих функцій, що дозволяються отримувати різні варіанти бієктивних зображень.

Програмний продукт, що був отриманий під час дослідницької роботи, є унікальним уніфікуючим інструментальним засобом, що може використовуватися у наступних дослідженнях при моделюванні інших об’єктів та природних процесів, які можуть бути описані за допомогою контсруктивно-продуційного підходу.

Таким чином, за результатами дослідження можна зробити висновки:

– класичні фрактали зберігають властивість самоподібності у різних відображеннях;
– конструктивно-продуційними структурами можливо описати не тільки класичні фрактали, а й описувати явища природи.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. «Фрактал» [Ел. ресурс]. Available: https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%BB. [Дата звертання: 17.11.2020].
2. «Самоподобие» [Ел. ресурс]. Available: https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%B8%D0%B5. [Дата звертання: 17.11.2020].
3. «Биекция» [Ел. ресурс]. Available: https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B5%D0%BA%D1%86%D0%B8%D1%8F. [Дата звертання: 17.11.2020].
4. Shynkarenko V.I. and Ilman V.M., “Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part II. Refining Transformations”. Cybernetics and Systems Analysis, 50(6), 2014, 829 – 841. doi: 10.1007/s10559-014-9674-9; “Part I. Generalized Formal Constructive-Synthesizing Structure”. Cybernetics and Systems Analysis, vol. 50(5), 2014, pp. 665 – 662. doi: 10.1007/s10559-014-9655-z
5. Ільман В. М. Структурний підхід до проблеми відтворення граматик / В. М. Ільман, В. І. Шинкаренко // Проблемы программирования. – 2007 – №1. – С. 5-16.
6. Shynkarenko, V. I. Constructive-Synthesizing Structures and Their Grammatical Interpretations. I. Generalized Formal Constructive-Synthesizing Structure / V. I. Shynkarenko, V. M. Ilman. // Cybernetics and Systems Analysis. – 2014. – Vol. 50. – Issue 5. – P. 655-662.
7. Шинкаренко В.И. Конструктивно-продукционные структуры и их грамматические интерпретации. II. Уточняющие преобразования. / В.И. Шинкаренко, В.М. Ильман // Кибернетика и системный анализ. – 2014 –№6 – С. 15 – 28.
8. Шинкаренко В.И., Литвиненко К.В., Чигирь Р.Р. “Конструктивное соответствие мультисимвольных и линейных геометрических фракталов”, International Journal “Informationa Technologies & Knowledge”, vol. 1(13), 2019, pp. 76-99.

9. Шинкаренко В. І., Литвиненко К. В., Чигір Р. Р., Жадан А. А., Варіативність уточнюючих перетворень конструктивно-продукційного моделювання / В. І. Шинкаренко, К. В. Литвиненко, Р. Р. Чигір, А. А. Жадан // XIV міжнародна конференція TAAPSD`2017. – 2017. – С. 225-230.
10. Шинкаренко В. І., Литвиненко К. В., Чигір Р. Р., Жадан А. А., Конструктивно-продукційне моделювання фракталів / В. І. Шинкаренко, К. В. Литвиненко, Р. Р. Чигір, А. А. Жадан // Міжнародна конференція ISDMCI'2018. – 2018. – С. 289-291.
11. Шинкаренко В. І., Литвиненко К. В., Чигір Р. Р., Жадан А. А., Конструктивно-продукційне моделювання фрактальних часових рядів на основі L-систем / В. І. Шинкаренко, К. В. Литвиненко, Р. Р. Чигір, А. А. Жадан // Всеукраїнська науково-методична конференція «Проблеми математичного моделювання». – 2018. – С. 161-164.
12. Shynkarenko V., Lytvynenko K., Chyhir R. and Nikitina I., "Modeling of Lightning Flashes in Thunderstorm Front by Constructive Production of Fractal Time Series". In: Advances in Intelligent Systems and Computing IV, vol. 1080, Springer, 2019, pp. 173-185/ doi: 10.1007/978-3-030-33695-0_13
13. Skalozub V., Ilman V. and Shynkarenko V., "Development of ontological support of constructive-synthesizing modeling of information systems", Eastern-European Journal of Enterprise Technologies, vol. 6, issue 4 (90), 2017, pp. 58–69. doi: 10.15587/1729-4061.2017.119497
14. «Вероятностное распределение» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Probability_distribution. [Дата звертання: 20.11.2020].
15. «Сравнение фрактальных характеристик временных рядов экономических показателей» [Ел. ресурс]. Available: https://www.science-education.ru/ru/article/view?id=15974. [Дата звертання: 20.11.2020].
16. «L-системы» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/L-system. [Дата звертання: 20.11.2020].

17. «Построение эквивалентной праворекурсивной КС-грамматики» [Ел. ресурс]. Available: https://studfile.net/preview/5354865/. [Дата звертання: 17.11.2020].
18. «Анализ и распознавание реализации сигналов, обладающих фрактальными свойствами» [Ел. ресурс]. Available: http://www.tsatu.edu.ua/vmf/wp-content/uploads/sites/17/11.pdf. [Дата звертання: 17.11.2020].
19. «Математическое ожидание» [Ел. ресурс]. Available: https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BE%D0%B6%D0%B8%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5. [Дата звертання: 17.11.2020].
20. «Фазовое пространство» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Phase_space. [Дата звертання: 08.11.2020].
21. «Нелинейная динамика и анализ временных рядов – обзор метода Recurrence plots» [Ел. ресурс]. Available: https://habr.com/ru/post/145805/. [Дата звертання: 08.11.2020].
22. «Фрактал» [Ел. ресурс]. Available: https://vlab.wikia.org/ru/wiki/%D0%A4%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%BB. [Дата звертання: 08.11.2020].
23. «Кривая Безье» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/B%C3%A9zier_curve. [Дата звертання: 08.11.2020].
24. «Python – мова програмування» [Ел. ресурс]. Available: https://ru.wikipedia.org/wiki/Python. [Дата звертання: 08.11.2020].
25. «Python – мова програмування» [Ел. ресурс]. Available: https://www.python.org/. [Дата звертання: 08.11.2020].
26. Falconer K. Fractal Geometry: Mathematical Foundations and Applications. John Wiley & Sons. 1999. p 288.
27. «Fractal Foundation» [Ел. ресурс]. Available: https://fractalfoundation.org/resources/what-are-fractals/. [Дата звертання: 24.11.2020].
28. Звіт з науково-дослідної роботи «Конструктивно-продуційне моделювання фракталів» 2018 р., м.Дніпро.

29. «Фракталы в архитектуре» [Ел. ресурс]. Available: http://www.berlogos.ru/article/fraktaly-v-arhitekture/. [Дата звертання: 24.11.2020].
30. «Естественные модели параллельных вычислений. Лекция 3 :: L-системы» [Ел. ресурс]. Available: http://hpc-education.ru/files/lectures/2011/ershov/ershov_2011_slides03.pdf. [Дата звертання: 24.11.2020].
31. «Снежинка Коха» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Koch_snowflake. [Дата звертання: 24.11.2020].
32. «Черепашкова графіка» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Turtle_graphics. [Дата звертання: 24.11.2020].
33. «Временной ряд» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Time_series. [Дата звертання: 24.11.2020].
34. «Биекция» [Ел. ресурс]. Available: https://math.wikia.org/ru/wiki/%D0%91%D0%B8%D0%B5%D0%BA%D1%86%D0%B8%D1%8F. [Дата звертання: 24.11.2020].
35. «Инъекция» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Injective_function. [Дата звертання: 24.11.2020].
36. «Сюръекция» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Surjective_function. [Дата звертання: 24.11.2020].
37. Секованов В.С. «Элементы теории фрактальных множеств: Учебное пособие. Изд. 5-е, перераб. и доп.» // В.С. Секанов // Москва: Книжный дом «ЛИБРОКОМ», 2013. – 248 с
38. «Атрибутные транслирующие грамматики» [Ел. ресурс]. Available: https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%BD%D1%8B%D0%B5_%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B8%D0%B5_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8. [Дата звертання: 17.11.2020].
39. «Крива дракону» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Dragon_curve [Дата звертання: 08.11.2020].

40. «Кривая Коха» [Ел. ресурс]. Available: https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%B2%D0%B0%D1%8F_%D0%9A%D0%BE%D1%85%D0%B0 [Дата звертання: 08.11.2020].
41. «Кривая Госпера» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Gosper_curve [Дата звертання: 08.11.2020].
42. «Ледяной фрактал» [Ел. ресурс]. Available: https://mathworld.wolfram.com/IceFractal.html [Дата звертання: 08.11.2020].
43. «Кривая Серпинского» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Sierpi%C5%84ski_curve [Дата звертання: 08.11.2020].
44. «Монолитная vs Микросервисная архитектура» [Ел. ресурс]. Available: https://proglib.io/p/monolitnaya-vs-mikroservisnaya-arhitektura-2019-09-16. [Дата звертання: 08.11.2020].
45. «Лучшая архитектура для MVP: монолит, SOA, микросервисы или бессерверная?.. Часть 1» [Ел. ресурс]. Available: https://habr.com/ru/company/otus/blog/476024/. [Дата звертання: 08.11.2020].
46. «Микросервисная архитектура» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Microservices. [Дата звертання: 08.11.2020].
47. «UML» [Ел. ресурс]. Available: https://prog-cpp.ru/uml-classes/. [Дата звертання: 08.11.2020].
48. «Диаграмма последовательности» [Ел. ресурс]. Available: https://flexberry.github.io/ru/fd_sequence-diagram.html. [Дата звертання: 08.11.2020].
49. «Объект» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Object_(computer_science). [Дата звертання: 08.11.2020].
50. «Блок-схема» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Flowchart. [Дата звертання: 08.11.2020].
51. «Диаграмма классов» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Class_diagram. [Дата звертання: 08.11.2020].
52. «Turtle графика» [Ел. ресурс]. Available: http://nsft.narod.ru/Fractals/turtle.htm. [Дата звертання: 08.11.2020].

53. «Диаграмма классов» [Ел. ресурс]. Available: https://ru.wikipedia.org/wiki/L-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0#/media/%D0%A4%D0%B0%D0%B9%D0%BB:Dragon_curve_L-system.svg. [Дата звертання: 08.11.2020].
54. «Окружность» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Circle. [Дата звертання: 08.11.2020].
55. «Глава 44. L-системы» [Ел. ресурс]. Available: http://mech.math.msu.su/~shvetz/54/inf/perl-problems/chLSystems.xhtml. [Дата звертання: 08.11.2020].
56. «Нормальное распределение» [Ел. ресурс]. Available: https://en.wikipedia.org/wiki/Normal_distribution. [Дата звертання: 08.11.2020].
57. «Лекция 2: отображения и соответствия» [Ел. ресурс]. Available: https://mipt.ru/upload/ab4/lecture-2-functions-arphn7cdyy7.pdf. [Дата звертання: 08.11.2020].
58. «Закону України «Про охорону праці» прийнятий від 14 жовтня 1992 року № 2695-12»
59. «ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» від 01.12.99 р.»
60. «ДСН 3.3.6.039-99 «Державні санітарні норми виробничої загальної та локальної вібрації» від 01.12.99 р.»
61. «ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» від 10.12.98 р.»
62. «ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» від 01.12.99 р.»
63. «НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.18 р.»
64. «НПАОП 0.00-7.11-12 «Загальні вимоги стосовно забезпечення роботодавцями охорони праці працівників» від 25.01.12 р.»

65. «Державні санітарні норми та правила «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» від 08.04.14 р.»
66. «ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки від 14.10.13 р.»
67. «ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення від 21.07.03 р.»
68. «ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища від 28.10.04 р.»
69. «ДБН В.2.5-28-2006 «Природне і штучне освітлення» від 03.10.18 р.»
70. «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою від 16.06.14 р. затверджений наказом №398 від 16.06.14 р.»

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Б.Є. Боднар

СИСТЕМА МОДЕЛЮВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01194-01-ЛЗ

Завідувач кафедри КІТ
проф. В.І. Шинкаренко

Керівник розробки
проф. В.І. Шинкаренко

Виконавець
студент групи ПЗ1921
Р.Р. Чигір

Нормоконтролер
доц. О.С. Куроп'ятник

2020

ЗАТВЕРДЖЕНО
1116130.01194-01

СИСТЕМА МОДЕЛЮВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

Технічне завдання
1116130.01194-01
Аркушів 21

2020

АНОТАЦІЯ

Документ 1116130.01194-01 «Система моделювання продуційних конструкторів. Технічне завдання» відноситься до програмної документації дипломного проекту.

У даному документі представлено призначення програмного забезпечення, область застосування, основні вимоги, стадії та строки розробки проекту, технічні та техніко економічні показники, що пред’являються до програмного продукту.

ЗМІСТ

ВСТУП

Програма «Система моделювання продуційних конструкторів» призначена для створення продуційних атрибутивних граматик й візуалізації їх продукції.

Споконвік людину окружає різні природні явища та процеси. Багато з них має закономірності у собі та мати риси повторення частинних дій , тому можливо зробити припущення про фрактальні властивості, притаманні цим процесам. Можливість створювати моделі на основі явищ дає змогу заглибитися у процес аналізу, що у довготривалому процесі вивчення та дослідження дозволяє передбачати наступні дії цих явищ.

Одним з методів дослідження є порівняння об’єктів та їх даних, а порівняння моделей, розроблених на одних і тих самих даних (тобто, бієктивні відображення), дозволяють помітити риси, що притаманні описаним процесам, не залежно від реалізації відображення.

Для побудови відображень можна обирати різні варіанти вихідних даних, але текстові граматики дають змогу отримати буквені рядки, які можливо наділити атрибутикою для подальшої реалізації та створити бієктивні відображення на основі єдиного рядка.

Основна термінологія: КОНСТРУКТОР, ПРОДУКЦІЯ, БІЄКТИВНІ ВІДОБРАЖЕННЯ, ФРАКТАЛ, МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ.

Причинами дослідження є відсутність аналогічних досліджень у відкритому просторі та Україні.

Область застосування: моделювання природних явищ на базі науково-дослідницьких центрів та інститутів.

1 ПІДСТАВА ДО РОЗРОБКИ

Підставою для розробки є наказ № 779 ст. «Про призначення керівників та затвердження тем магістерських робіт» факультету «Комп’ютерні технології і системи» за спеціальністю № 121 «Інженерія програмного забезпечення», затверджений ректором Дніпровський національного університету залізничного транспорту імені академіка В. Лазаряна проф. Пшінька О. М. від 10.10.2019 р.

Тема дипломного проекту: Моделювання бієктивних відображень фракталів різної природи.

Керівник дипломного проекту — проф. Шинкаренко В. І.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Функціональне призначення розробки:

– створення простих та алгоритмічних конструкторів на основі конструктивно-продуційних граматик;
– заповнення інформації об’єкті моделювання шляхом заповнення спеціалізації, інтерпретації та конкретизації граматики;
– створення графічних відображень даних шляхом транслювання даних реалізації конструктору;
– отримання інформації про значення атрибутів конструктора на момент реалізації.

2.2 Експлуатаційне призначення

Експлуатаційне призначення розробки:

– проведення досліджень на виявлення фрактальних властивостей у граматиках та їх продукції за допомогою аналізу їх на основі порівняння даних реалізації;
– опис об’єктів, явищ та процесів природи за допомогою параметричних конструкторів для створення моделей та подальшого їх аналізу;
– створення прогнозів майбутньої поведінки модельних явищ та процесів для можливого контролю над ними.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Основними функціональними вимогами є:

– збереження інформації про продуційні конструктори у системі з відновленням їх при створенні нової сессії роботи з програмним продуктом;
– відображення інформації реалізації конструктору у вигляді графічного двовимірного зображення та у вигляді списку значень атрибутів граматики;
– створення продуційних простих конструкторів та конструкторів алгоритмів;
– заповнення інформації про спеціалізацію, інтерпретацію та конкретизацію конструкторів з можливістю модифікації даних.

Вхідні данні:

– значення атрибутів: назва атрибуту, тип даних що позначає область значень, початкове значення;
– форма відображення графічного зображення;
– значення масивів даних для відображення;
– функції інтерпретування;
– значення інтерпретування символів.

Вихідні данні:

– графічне відображення реалізації конструктору у вигляді двовимірного графіку;
– кінцеві значення атрибутів конструктору;
– значення кінцевої символьної стрічки граматики.

3.2 Вимоги до надійності

Вимогами до надійності є:

– програма повинна працювати стабільно не допускаючи більше однієї відмови на 2000 запусків системи;
– програма повинна не допускати пошкодження даних під час роботи з нею.

3.3 Умови експлуатації

Дане програмне рішення може використовуватись в умовах, відповідних до умов, які описані в документу [1].

Для нормального функціонування без конфігураційних збоїв слід дотримуватися наступних умов до середовища, у якому розгортається програмний продукт:

– ЕОМ, які використовуються для роботи програмного продукту, повинні відповідати чинним вимогам та стандартам в Україні, нормативних актами з охорони праці [2];
– на ЕОМ повинен бути встановлений інтерпритатор мови програмування Python версії 3.7;
– стан технічних засобів повинен задовільнять відповідним нормам та вимогам;
– для роботи з програмним продуктом необхідно завчасно ознайомитися з керівництвом користувача з роботи з програмою.

3.4 Вимоги до складу та параметрів технічних засобів

Мінімальна конфігурація програмного комп’ютеру повинна становити:

– місце на жорсткому диску: 256 Мб;
– операційна пам’ять: 1 Гб;
– процесор: Intel Core 2 Q6600 Q6600 @ 2.40 Ггц;
– маніпулятори: клавіатура, комп’ютерна мишка;

– для встановлення програмного забезпечення комп’ютер повинен бути обладнаним CD/DVD приводом або роз’ємом USB 3.0/USB 4.0 залежно від місця зберігання програми.
– операційна система: Linux або Windows.

3.5 Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційної та програмної сумісності: операційна система Microsoft Windows 8\8.1\10 або операційна система Linux (дистрибутив Ubuntu) та інтерпритатор мови програмування Python версії 3.7 для відповідної операційної системи.

3.6 Вимоги до маркування та упаковки

Маркування програмного продукту повинно відповідати штампу на рис. 3.1:

СИСТЕМА МОДЕЛЮВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ Розробник: Чигір Р. Р. ДНУЗТ, КІТ 2019

Рисунок 3.1 – Маркувальний штамп

3.7 Вимоги до транспортування і зберігання

Транспортування програмного продукту можливо за допомогою портативних носіїв збереження інформації таких як USB-накопичувачів та CD/DVD-дисків.

Термін зберігання працездатної копії програмного додатку на носію інформації обумовлений терміном придатності самого носія.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна складатися з:

– технічного завдання;

– робочого проекту.

Робочій проект повинен складатися з:

– специфікації;

– тексту програми;

– опису програми;

– опису застосування;

– керівництва користувача. Керівництво зі створення продуційних конструкторів.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів ГОСТ 19.101-77 «Єдина система програмної документації. Види програм та програмних документів» [3].

5 ВИЗНАЧЕННЯ ВИТРАТ НА ПРОЕКТУВАННЯ ПРОГРАМИ

Основною метою розробки обґрунтування (ТЕО) є надання фінансової оцінки передбачених витрат необхідних для отримання корисного результату. Також разом із витратами проводиться оцінка прибутковості проекту. На основі двох раніше зазначених компонентів у якості підсумку визначається економічна доцільність розробки ПЗ та його впровадження.

Початковим етапом розрахунку величини трудових витрат розробників є оцінка розміру програмного забезпечення. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості (кількісний або якісний).

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях .

$$E = a \cdot S^b \cdot EAF \tag{5.1}$$

де E – витрати праці на проект (в людино-місяцях);

Sb – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

Припустимо, що розмір програмного коду програмного засобу – 3000 рядків:

$$E = 2{,}4 \cdot 3^{1{,}05} \cdot 1 = 7{,}6 \tag{5.2}$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 7,6 людино-місяці.

Основними статтями витрат прийняті:

– основна заробітна плата;
– відрахування на соціальні потреби;
– накладні витрати;
– витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати за одну годину.

Середня заробітна плата Python Junior Software Engineer станом на червень 2020 рік становить 18 853 грн [1].

Розрахунок заробітної платні проводиться по формі табл. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	
1	інженер-програміст	18853	1	7,6	143283

Описаний в проекті програмний продукт буде розроблений одним програмістом в період з 10.02.20 до 02.10.20, що складає 170 днів або 35 робочих тижнів. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 136,6 грн/год [2]. Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} \cdot N_{\text{тиж}} \cdot N_{\text{год}}, \quad (5.5)$$

де $N_{\text{чол}}$ – кількість виконавців, *чол*;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 35 \cdot 40 = 1400 \text{ чол/год}. \quad (5.6)$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (5.7)$$

Де $t_{\text{розробки}}$ – витрати праці у чол/год;

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, приймається 0,75.

ОЗП складається з:

$$ОЗП = 1400 \cdot 136{,}6 \cdot 0{,}75 = 143430 \text{ грн.} \quad (5.8)$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати (Єдиний соціальний внесок становить 22% від окладу працівника [3]):

$$C_{соц} = \frac{ОЗП \cdot 22\%}{100\%}$$

$$C_{соц} = \frac{143430 \cdot 22\%}{100\%} = 31555 \text{ грн.} \quad (5.9)$$

Отримані результати за (8) та (9) підсумовуються. Вони складають 31555 грн. та визначають основні прямі витрати.

Накладні витрати враховують загально господарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30 – 40%) від суми прямих витрат:

$$С_{накл} = \frac{(ОЗП + С_{соц}) \cdot 35\%}{100\%}; \quad (5.10)$$

$$С_{накл} = \frac{(143430 + 31555) \cdot 35\%}{100\%} = 81756 \text{ грн.} \quad (5.11)$$

На протязі усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

– вартість витратних матеріалів;
– витрати на ремонт;
– заробітна плата ремонтника;
– оренда приміщення;
– додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;

– амортизаційні витрати на персональний комп'ютер і програмне забезпечення;

– витрати на електроенергію ($С_{ел}$), які визначаються за формулою:

$$С_{ел} = P \cdot B \cdot T_{розр}, \quad (5.12)$$

де P – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год;

B – вартість 1 кВт/година, складає 1,68 грн [4];

$T_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{ел} = 0{,}45 \cdot 1{,}68 \cdot 1400 = 1059 \text{ грн.} \quad (5.13)$$

Витрати на витратні матеріали ($С_{вм}$) протягом всього терміну експлуатації приблизно 10% від вартості комп’ютеру. Вартість робочої станції приймається 18 000 грн., термін експлуатації – 5 років. За робочу станцію приймається ноутбук Asus Lenovo V14-IIL 82C400XGRA [5]. Отже, можна визначити ці витрати за період створення програмного засобу:

$$С_{вм} = В_{ком} \cdot \frac{N_Д}{N_{експ} \cdot 365} \cdot \frac{10\%}{100\%}, \quad (5.14)$$

де $B_{ком}$ – вартість персонального комп’ютеру;

$N_д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп’ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{вм} = 18000 \cdot \frac{170}{5 \cdot 365} \cdot \frac{10}{100} = 168 \text{ грн.} \quad (5.15)$$

Заробітна плата ремонтника ($C_{рем}$) визначена наступним чином: на ремонт 50 комп’ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 18 853 грн [1]. Тоді в перерахунку на один комп’ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{рем} = \frac{C_{рем}'}{N_{ком}} \cdot T_{міс}, \quad (5.16)$$

де $C'_{\text{рем}}$ – середньомісячна заробітна плата;

$N_{\text{КОМ}}$– кількість комп'ютерів на одного ремонтника.

$T_{\text{міс}}$ – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{18853}{50} \cdot 7{,}6 = 3017 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби ($C_{\text{ком}}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{KOM} = C_{BM} = 168 \text{ грн.} \quad (5.17)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АКП} = \text{В}_{KOM} \cdot \frac{N_{\text{Д}}}{N_{\text{експ}} \cdot 365}; \quad (5.18)$$

$$\text{АКП} = 18000 \cdot \frac{2}{3 \cdot 12} = 1000$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийнятий термін морального старіння для Windows 5 років, то амортизаційні відрахування на програмне забезпечення дорівнюють його вартості.

Для функціонування персонального комп'ютера використовувалася операційна система Windows 10 Professional [6].

$$\text{АВ}_{\text{windows}} = 5999 \cdot \frac{2}{5 \cdot 12} = 200 \text{ грн}$$

В результаті було отримано суму амортизаційних витрат на програмне забезпечення, яке дорівнює 200 грн.

Додаткові витрати ($C_{\text{дод}}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера-програміст, тобто 9426,5 гривень на місяць.

Оренду приміщень приймемо рівною 2250 гривень на місяць відповідно до реальної пропозиції (за 15 м.кв.)[7].

Сумарні експлуатаційні витрати на один персональний комп'ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АПК} + \text{АПО} + C_{\text{ор}} + C_{\text{дод}}; \quad (5.19)$$

$$C_{\text{експ}} = 1059 + 168 + 3017 + 1000 + 200 + 17100 + 9427 = 31971 \text{ грн.} \quad (5.20)$$

Результати розрахунків зведено у табл. 5.2.

Таблиця 5.2 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	1059
Вартість витратних матеріалів	168
Витрати на ремонт	3017
Амортизація персонального комп'ютера	1000
Амортизація програмного забезпечення	200
Оренда приміщення	17100
Додаткові витрати	9427
Всього	**31971**

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \quad (5.21)$$

$$C_{\text{розробки}} = 143430 + 31555 + 81756 + 31971 = 288712 \text{ грн.} \quad (5.22)$$

Розрахунок витрат зведено у табл. 5.3.

Таблиця 5.3 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	143430
Відрахування на соціальні потреби	31555
Накладні витрати	81756
Експлуатаційні витрати	31971
Всього	**288712**

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 288712 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Етапи та стадії розробки програмного додатку відображене у табл. 6.1.

Таблиця 6.1 – Стадії та етапи розробки

Стадії розробки	Етапи розробки	Терміни виконання
1.Технічне завдання (ТЗ)	Постановка задачі	16.12.2019 – 27.12.2019
	Огляд літератури та аналіз аналогів	10.02.2020 – 21.02.2020
	Розробка структур вхідних і вихідних даних	24.02.2020 – 20.03.2020
	Визначення вимог до програми. Вибір та обґрунтування мови програмування	23.03.2020 – 03.04.2020
	Узгодження та затвердження ТЗ	06.04.2020 – 10.04.2020
2. Робочий проект	Розробка та програмування логіки програми	13.04.2020 – 08.05.2020
	Розробка і реалізація інтерфейсу користувача	11.05.2020 – 22.05.2020
	Відлагодження програми	25.05.2020 – 11.09.2020
	Розробка, узгодження та затвердження програмної документації	14.09.2019 – 07.12.2020
3. Впровадження	Підготовка і передача програми та програмної документації замовнику	08.12.2020 – 15.12.2020

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Контроль здійснюється за допомогою виконання набору тестів з метою знаходження помилок в програмі та його специфікації. Контроль виконання роботи забезпечується керівником розробки.

Прийом програми здійснюється уповноваженою комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. «ДСанПіН 3.3.2-007-98. «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» від 10.12.98 р.».
2. «ДСН 3.3.6.042-99. «Санітарні норми мікроклімату виробничих приміщень» від 01.12.99 р.».
3. «ГОСТ 19.101-77. «Виды програм и программных документов» від 20.05.77 р.».
4. "DOU: Сообщество программистов," [Ел. ресурс]. Available: https://jobs.dou.ua/salaries/#period=jun2020&city=Kyiv&title=Junior%20Software%20Engineer&language=Python&spec=&exp1=0&exp2=10. [Дата звернення: 11.2020].
5. "Kadrof.ru: сайт об удаленной работе, фрилансе," [Ел. ресурс]. Available: https://www.kadrof.ru/articles/46641. [Дата звернення: 11.2020].
6. Єдиний соціальний внесок," [Ел. ресурс]. Available: https://index.minfin.com.ua/ua/labour/social/. [Дата звернення: 11.2020].
7. "Тарифи для населення на червень 2020," [Ел. ресурс]. Available: https://ooek.od.ua/pictures/pdf_files/%D0%A2%D0%B0%D1%80%D0%B8%D1%84%D0%B8_%D0%B4%D0%BB%D1%8F_%D0%BD%D0%B0%D1%81%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_%D0%BD%D0%B0_%D1%87%D0%B5%D1%80%D0%B2%D0%B5%D0%BD%D1%8C_2020.pdf [Дата звернення: 11.2020].
8. "Comfy – Інтернет магазин побутової техніки та електроніки" [Ел. ресурс]. Available: https://comfy.ua/noutbuk-lenovo-v14-iil-82c400xgra-iron-grey.html. [Дата звернення: 11.2020].
9. "Microsoft - Official Home Page," [Ел. ресурс]. Available: https://www.microsoft.com/uk-ua/p/windows-10-

pro/df77x4d43rkt/48DN?rtc=1&activetab=pivot%3aoverviewtab.
[Дата звернення: 11.2020].

10. "ОЛХ – сервіс оголошень," [Ел. ресурс]. Available: https://www.olx.ua/obyavlenie/sdam-ofis-pl-15m-pr-kirova-novyy-remont-IDJXOuY.html#fd7f062b72;promoted.
[Дата звернення: 11.2020].

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Б.Є. Боднар

СИСТЕМА МОДЕЛЮВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

Робочий проект
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01194-01-ЛЗ

Завідувач кафедри КІТ
проф. В.І. Шинкаренко

Керівник розробки
проф. В.І. Шинкаренко

Виконавець
студент групи ПЗ1921
Р.Р. Чигір

Нормоконтролер
доц. О.С. Куроп'ятник

2020

ЗАТВЕРДЖЕНО
1116130.01194-01-ЛЗ

СИСТЕМА МОДЕЛЮВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

Специфікація

1116130.01194-01
Аркушів 2

2020

1116130.01194-01

Позначення	Найменування Документація	Примітки
01116130.01194-01-ЛЗ	Лист затвердження	
01116130.01194-01-ЛЗ	Лист затвердження	
01116130.01194-01 13 01	Опис програми	
01116130.01194-01 І3 01	Керівництво користувача. Керівництво зі створення продуційних конструкторів	
01116130.01194-01 12 01	Текст програми	

ЗАТВЕРДЖЕНО
1116130.01194-01-ЛЗ

СИСТЕМА МОДЕЛЮВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

Опис програми
1116130.01194-01 13 01
Аркушів 14

2020

1116130.01194-01

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Назва продукту: «Система моделювання продуційних конструкторів».

Призначення: моделювання параметричних текстових граматик, створення різних відображень на основі єдиних даних та подальше порівняння й аналіз отриманих відображень.

Реалізація: програма реалізована на мові Python 3 у програмному середовищі PyCharm 2020 з використанням технології QT середовища PyQt 5 Design для використання на операційних системах Windows 8\8.1\10 та Linux дистрибутив Ubuntu.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціональне призначення розробки:

– створення простих продуційних конструкторів та конструкторів алгоритмів;
– отримання результатів реалізації конструкторів у вигляді графічного зображення двовимірного графіку та списку значень атрибутів конструктору;
– введення інформації спеціалізації, інтерпретації та конкретизації конструктору досліджуваної моделі.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм програми

Алгоритм створення конструктору та проходження всього його життєвого циклу відображено на рис 3.1.

Конструктор у своєму життєвому циклу проходить наступні дії:

- створення конструктора – користувач створює конструктор та заповню базову інформацію про нього назва для ідентифікації та задає вид конструктору (простий чи алгоритмічний);
- заповнення специфікації – користувач задає базовий набор параметрів (текстове іменування, тип значень), форму відображення (тип відображення та відповідність відображення до існуючих параметрів);
- етап інтерпретації – користувач задає відповідність виконання функції (або ряду функцій), її вхідні та вихідні значення до обробки певного символу (тобто, ставиться відповідність символ до послідовності роботи описаних функцій);
- етап конкретизації – користувач описує початкові значення описаних параметрів на етапі специфікації та список правил виводу для символів
- створення реалізації – інформація про конструктор спочатку проходить етап формування мультисимвольної стрічки на основі правил виводу, після чого виконується інтерпретування символів на основі інформації етапу
- відображення реалізації – створюється графічне відображення реалізації конструктору та відображення атрибутів конструктору.

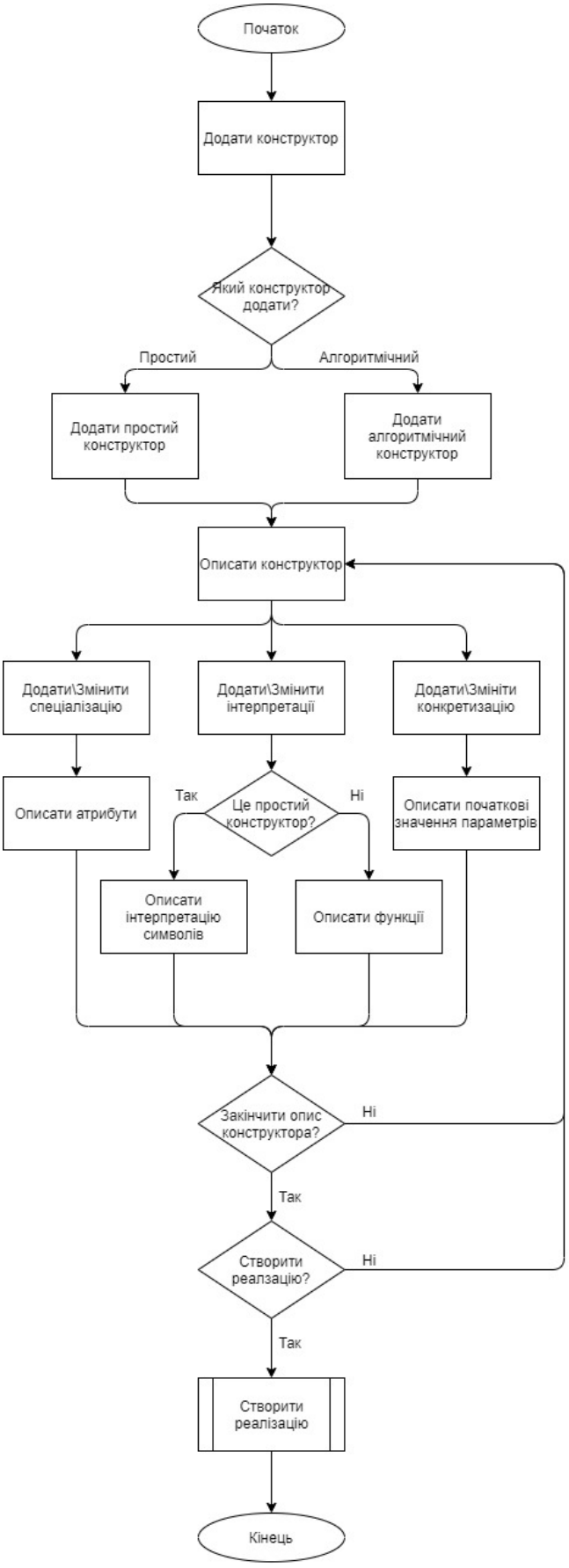


Рисунок 3.1 – Алгоритм створення конструктора

Деталізуючи взаємодію користувача з програмною систему, отримуємо більш детально прописаний сценарій моделювання конструкторів.

Деталізована діаграма послідовності наведена на рис.3.2.

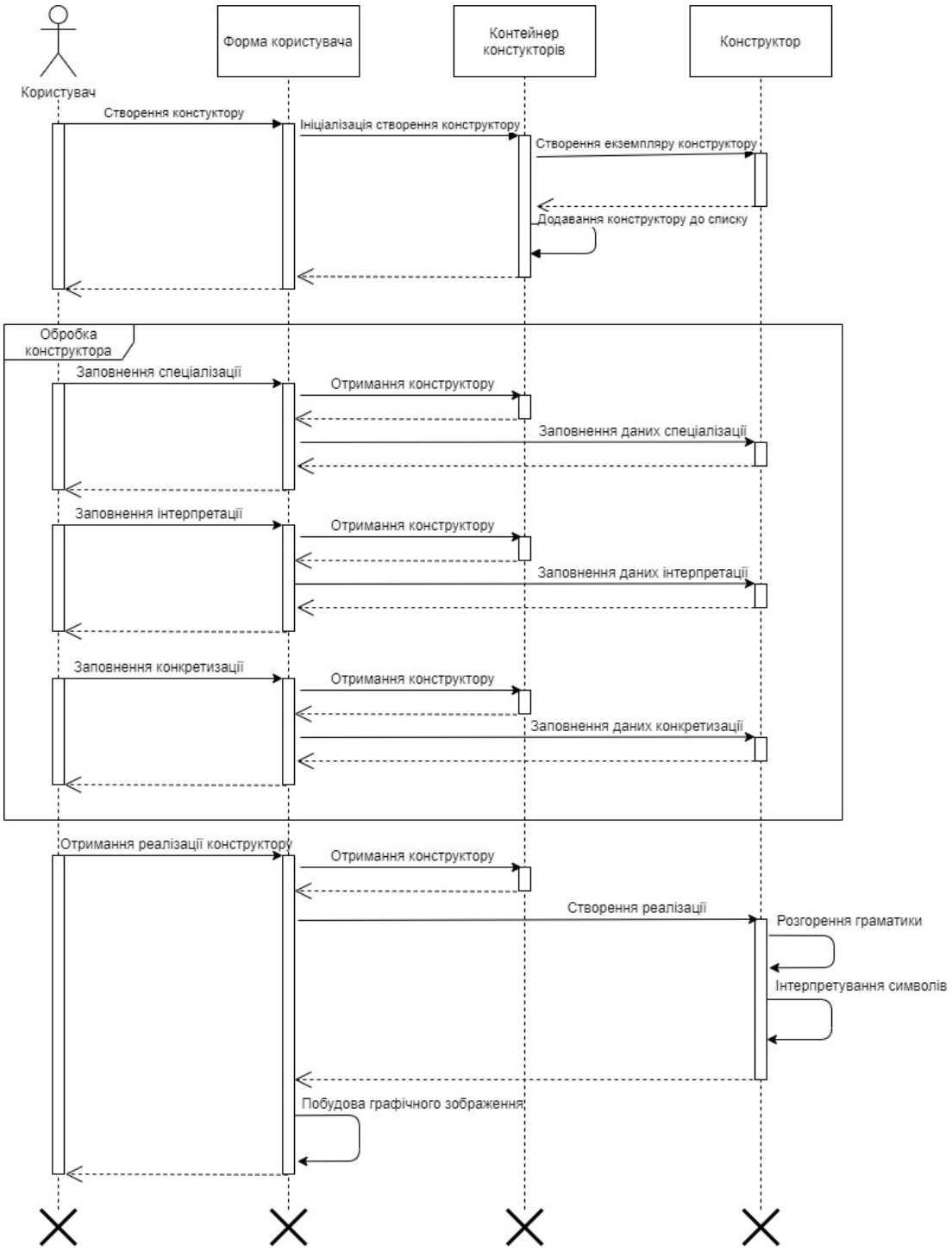


Рисунок 3.2 – Діаграма послідовності взаємодії користувача с програмною системою з виділенням компонентів

На основі життєвого циклу конструктору описуються сутності простого конструктору та конструктору алгоритмів.

Простий конструктор є видом конструктору, якому ставиться при ініціалізації базові атрибути, як аксіома чи правила підстановки, а також можливість підключення конструктору алгоритмів до себе, що зберігає описані функції для підключення через їх до символів.

На відміну від простого конструктору, у конструктору алгоритмів спеціалізація динамічно наповнюється на основі атрибути описаних функцій інтерпретування. Також, інтерпретація конструктору потребує прописування функцій, що можуть використовуватися простими конструкторами.

Додатковою сутністю виділяється контейнер конструкторів, що зберігає у собі всі проініціалізовані конструкторами да забезпечує доступ до них.

Описана інформація про логічні сутності може бути згрупована у табличному вигляді та представлена у табл.3.1.

Таблиця 3.1 – Базові сутності та їх функції

Сутність	Інформаційне наповнення	Функціональні особливості
Базовий конструктор	Базові структурні частини	Зберігання атрибутів Зберігання частин конструктору
Простий конструктор	Набір особин (хромосом) одного виду визначеного розміру	Побудова реалізації
Алгоритмічний конструктор	Функції інтерпретування	Зберігання функцій
Контейнер конструкторів	Набір конструкторів	Оперування набором конструкторів Формування конструкторів

3.2 Використані методи

Програма використовує алгоритм розгорнення лівосторонніх текстових граматик.

Додаткові методи користувач створює за допомогою додавання до конструктору алгоритмів код на мові програмування Python.

3.3 Структура програми з описом функцій складових частин і зв’язків

Програма складається з:

– app.py – файл, що запускає програму;
– form.py – файл, що містить інтерфейс програми;
– constructors_form.py – файл, що містить логіку роботи програмного інтерфейсу;
– constructor_domain.py – файл, що містить сутності конструктору;
– constructors.pkl – файл сессії.

Зв’язки між класами, представленими в файлах представлені на рис.3.3.

3.4 Зв’язки програми з іншими програмами

Розроблюваний програмний продукт не має зв’язків з іншими програмами та працює автономно, використовуючи функціонал стандартної бібліотека мови Python.

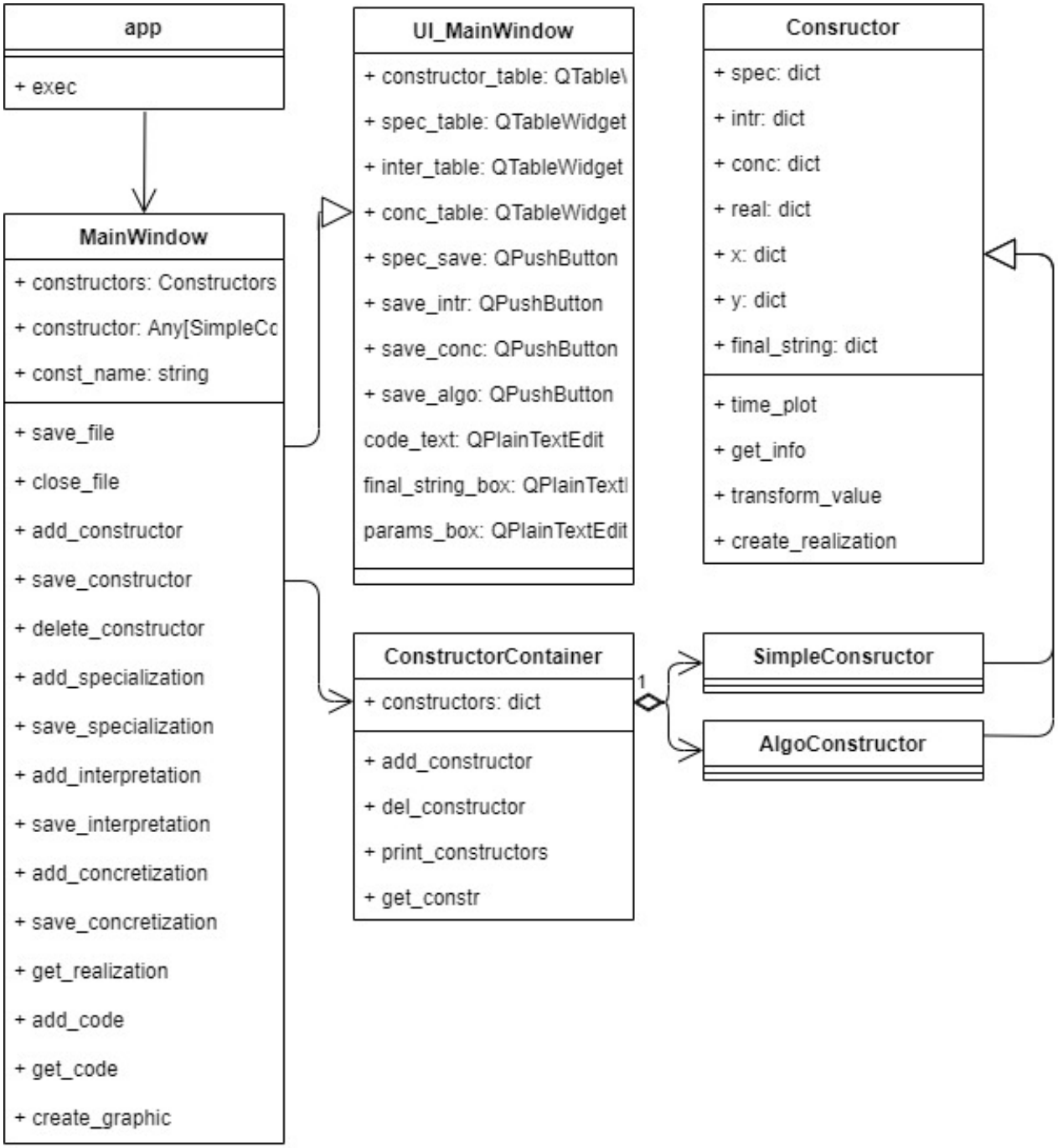


Рисунок 3.3 – Взаємодія між класами програми

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для експлуатації необхідно програмний комп’ютер з операційною системою Microsoft Windows 8\8.1\10 або операційна система Linux (дистрибутив Ubuntu).

Мінімальна конфігурація програмного комп’ютера:

- Місце на жорсткому диску: 256 Мб;
- Операційна пам’ять: 1 Гб;
- Процесор: Intel Core 2 Q6600 @ 2.40 Ггц;
- Маніпулятори: клавіатура, комп’ютерна мишка;
- Для встановлення програмного забезпечення комп’ютер повинен бути обладнаним CD/DVD приводом.

5 ВИКЛИК І ЗАВАНТАЖЕННЯ

Для запуску програмного продукту «Система моделювання продуційних конструкторів» необхідно виконати інтерпретування файлу app.py засобами Python 3.7.

Після запуску програми перед користувачем предстане інтерфейс головного меню програми. Інтерфейс головного меню наведено на рис.5.1.

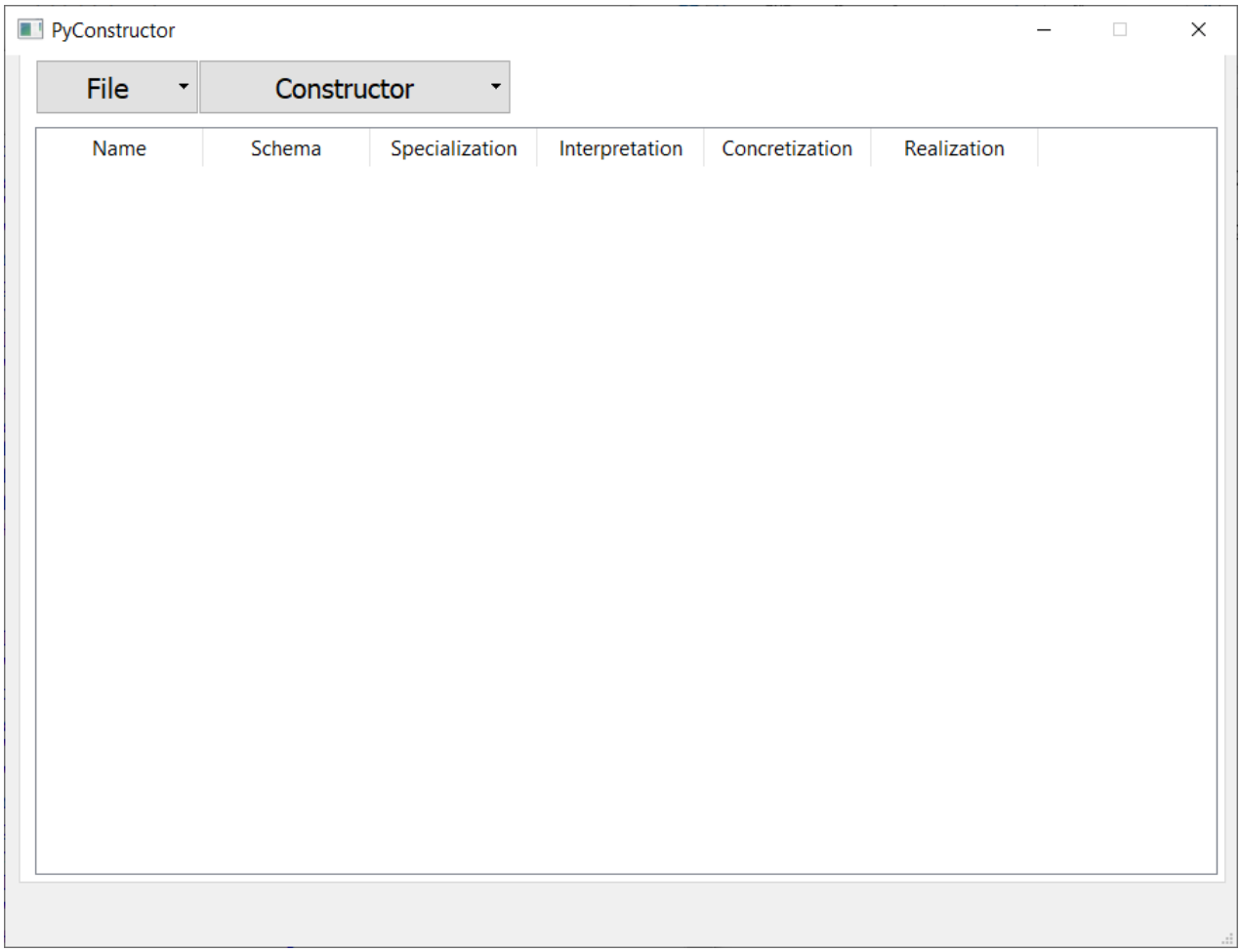


Рисунок 5.1 – Головне меню програми

6 ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними параметрами є параметри конструктора, отримані від користувача під час створення й проходження діалогових вікон опису спеціалізації, інтерпретації та конкретизація. Вхідними даними програми є:

- назва конструктора – символьна стрічка;
- тип конструктора – вибір між варіантами «Simple» та «Algorithms»
- назви атрибутів конструктора – символьна стрічка;
- типи даних відповідних атрибутів конструктора описані на етапі спеціалізації – символьна стрічка з варіантів «int», «float», «str», «bool», «list», «dict»;
- тип форми відображення – вибір між варіантами «Linear» та «Dots»;
- значення форми відображення на осі абсцис та ординат – символьна стрічка;
- символ інтерпретування – символьна стрічка;
- назва функції інтерпретування – символьна стрічка;
- вхідні параметри функції інтерпретування – символьна стрічка з назв атрибутів через кому;
- вихідний параметр функції інтерпретування – символьна стрічка;
- функція інтерпретування – символьна стрічка;
- початкове значення атрибуту – символьна стрічка;

Результатом роботи програми значення продукції конструктору.

Вихідні дані надаються після проходження етапу реалізації. Вихідні даними програми є:

- графічне відображення інтерпретування даних конструктора у вигляді двовимірного графіку;
- розгорнута кінцева стрічка символів конструктору;
- перелік атрибутів конструктору з їх значеннями формату «<Атрибут>: <Значення атрибуту>».

7 ОПИС ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ

7.1 Опис станів програми

Стани програми наведено у табл. 7.1.

Таблиця 7.1 – Стани програми

№ стану	Назва стану	Опис стану	Рекомендовані дії
1	Завантаження програми	Програма завантажується в пам’ять системи та запускається.	Дождатися відображення графічного інтерфейсу програми.
2	Відкрите головне меню (користувач не заповнив параметри)	Програма відкрита. Очікуються дії від користувача (заповнити параметри).	Почати роботу з програмою.
3	Заповнення параметрів програми	Користувач проводить роботу з програмою, вводячи дані у відповідні елементи інтерфейсу.	Вводити запропоновані параметри.
4	Відкрите головне меню (користувач заповнив параметри)	Програма відкрита. Очікуються дії від користувача (змінити записи, виконати обробку).	Перевірити коректність параметрів. Розпочати обробку параметрів.
5	Програма обробляє дані	Програма обробляє дані	Дочекатися обробки параметрів.
6	Програма створила відображення	Програма демонструє відображення конструктора	Ознайомитися з отриманими результатами

7.2 Опис переходів між станами програми

Схема переходів програми представлена на рис. 7.1. Вийти з програми можливо під час усіх станів крім стану 1 та стану 5.

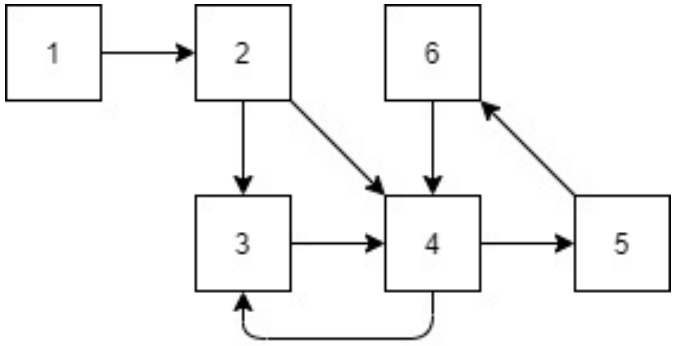


Рисунок 7.1 – Схема переходів між станами програми

7.3 Опис керування діалогом

Етапи роботи з програмою:

– створення та модифікація конструкторів;
– створення відображення конструкторів;
– запис інформації сесії.

На рис.4.1 представлено головне меню програми після її запуску.

Для створення нового конструктору треба перейти «Consructor»-«Add constructor», після чого відобразяться елементи опису початкових параметрів конструктора.

Після створення нового конструктора (обрати дію «ОК»), новий конструктор з’явиться у таблиці існуючих конструкторів сесії. Для подальшої роботи з конструкторами треба обрати один з існуючих конструкторі (натиснути на одну з комірок строки певного конструктора) й обрати один з пунктів меню «Construcor».

Пункт «Specialization» відкриє діалог заповнення інформації спеціалізації конструктора: задання параметрів конструктора (заповнення таблиці параметрів) та обрання форми відображення (вибір в діалозі однієї з можливих форм відображень й вказання параметрів для відображення).

Пункт «Interpretation» відкриє діалог заповнення інтерпретації символів мультисимвольної стрічки під час її інтерпретування (заповнення таблиці символів).

Для алгоритмічного конструктора відкриється діалог додавання програмних функцій інтерпретування до конструктору.

Пункт «Concretization» відкриє діалог заповнення початкових значень параметрів описаних на етапі спеціалізації (заповнення таблиці початкових значень параметрів).

Пункт «Realization» відобразить результати обробки кожного з конструкторів.

7.4 Формування екранів

Програма має головне вікно (рис 4.1) та екран роботи з графічним відображенням (рис 7.2).

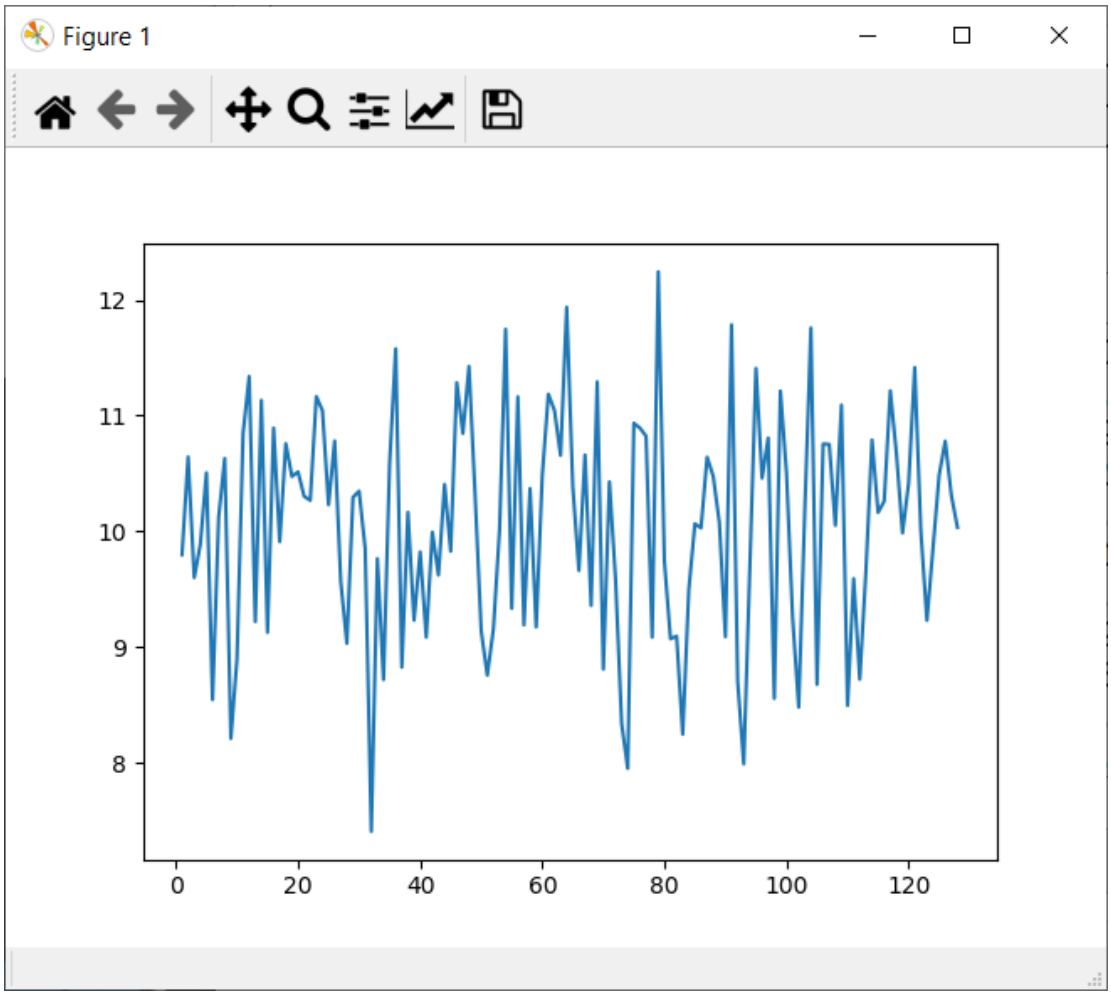


Рисунок 7.2 – Вікно налаштувань графічного відображення

8 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

У разі необхідності обробки продуційних конструкторів необхідно відправити список вхідних параметрів та опис конструкторів на електрону скриньку fractals@productconstructors.gmail.com.

– прийом даних на обробку здійснюється з 8:00 до 10:00 по буднях (оператор відділу прийому);
– обробка отриманих даних здійснюється з 10:00 до 14:00 по буднях (інженер з обробки даних);
– формування та обробка конструкторів здійснюється з 14:00 до 18:00 по буднях (оператор програми);
– відправлення отриманих результатів з обробки конструкторів здійснюється з 18:00 до 20:00 (оператор відділу прийому).

9 ПОВІДОМЛЕННЯ

У табл. 9.1 наведені повідомлення користувачу, що можуть з’явитися під час роботи з програмою. В програмі є лише повідомлення для користувача.

Таблиця 9.1 – Повідомлення програми

Текст повідомлення	Опис ситуації	Рекомендовані дії
Please, define function	Користувач не ввів текст функції.	Ввести текст функції.
Algorithms constructor not required specialization.	Користувач хоче заповнити етап спеціалізації алгоритмічного конструктору.	Обрати іншу дію.
Please, fill field “X”	Користувач не заповнив поле «X» спеціалізації простого конструктору.	Заповнити даними поле «X».
Please, fill field “Y”	Користувач не заповнив поле «Y» спеціалізації простого конструктору.	Заповнити даними поле «Y».
Please, fill attributes with build-in types	Користувач заповнив тип атрибуту не одним з можливих типів.	Заповнити типи даних атрибутів дійсними типами.

ЗАТВЕРДЖЕНО
1116130.01194-01-ЛЗ

СИСТЕМА МОДЕЛЮВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

Керівництво користувача. Керівництво зі створення продуційних конструкторів.
1116130.01194-01 І3 01
Аркушів 11

2020

ЗМІСТ

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

1.1 Функціонал програмного додатку

Функціонал програмного додатку включає:

- створення простих продуційних конструкторів та конструкторів алгоритмів;
- модифікацію та видалення створених конструкторів;
- наповнення інформацією про атрибути та форму відображення спеціалізацію конструктору;
- описання інтерпретування символів інтерпретації конструктору;
- заповнення інформації про початкові значення атрибутів конкретизації конструкторів;
- описання функцій на мові програмування Python інтерпретації конструктору алгоритмів;
- створення реалізації конструктору у вигляді графічного відображення двовимірного графіку;
- вивід значень атрибутів після реалізації конструктору;
- відображення розгорненої кінцевої символьної стрічки граматики.

1.2 Вимоги до складу і параметрів технічних засобів

Вимоги до складу і параметрів технічних засобів:

- місце на жорсткому диску: 256 Мб;
- операційна пам’ять: 1 Гб;
- процесор: Intel Core 2 Q6600 Q6600 @ 2.40 Ггц;
- маніпулятори: клавіатура, комп’ютерна мишка;
- для встановлення програмного забезпечення комп’ютер повинен бути обладнаним CD/DVD приводом або роз’ємом USB 3.0/USB 4.0 залежно від місця зберігання програми.
- операційна система: Linux або Windows.

1.3 Вимоги до інформаційної і програмної сумісності

Вимоги до інформаційної та програмної сумісності: операційна система Microsoft Windows 8\8.1\10 або операційна система Linux (дистрибутив Ubuntu) та інтерпритатор мови програмування Python версії 3.7 для відповідної операційної системи.

2 ПІДГОТОВКА ДО РОБОТИ

Для роботи з даним програмним додатком необхідно мати:

– персональний комп‘ютер з встановленою операційною системою Microsoft Windows 8\8.1\10 або операційна система Linux (дистрибутив Ubuntu);
– інтерпритатор мови програмування Python версії 3.7 для відповідної операційної системи.
– маніпулятори: клавіатура, комп’ютерна мишка;

2.1 Запуск програмного додатку

Для запуску програмного продукту «Система моделювання продуційних конструкторів» необхідно виконати інтерпретування файлу app.py засобами Python 3.7. Після запуску перед користувач буде відображено головне вікно програми.

3 ОПИС ОПЕРАЦІЙ

Після запуску програми перед користувачем предстане інтерфейс головного меню програми. Інтерфейс головного меню наведено на рис. 3.1.

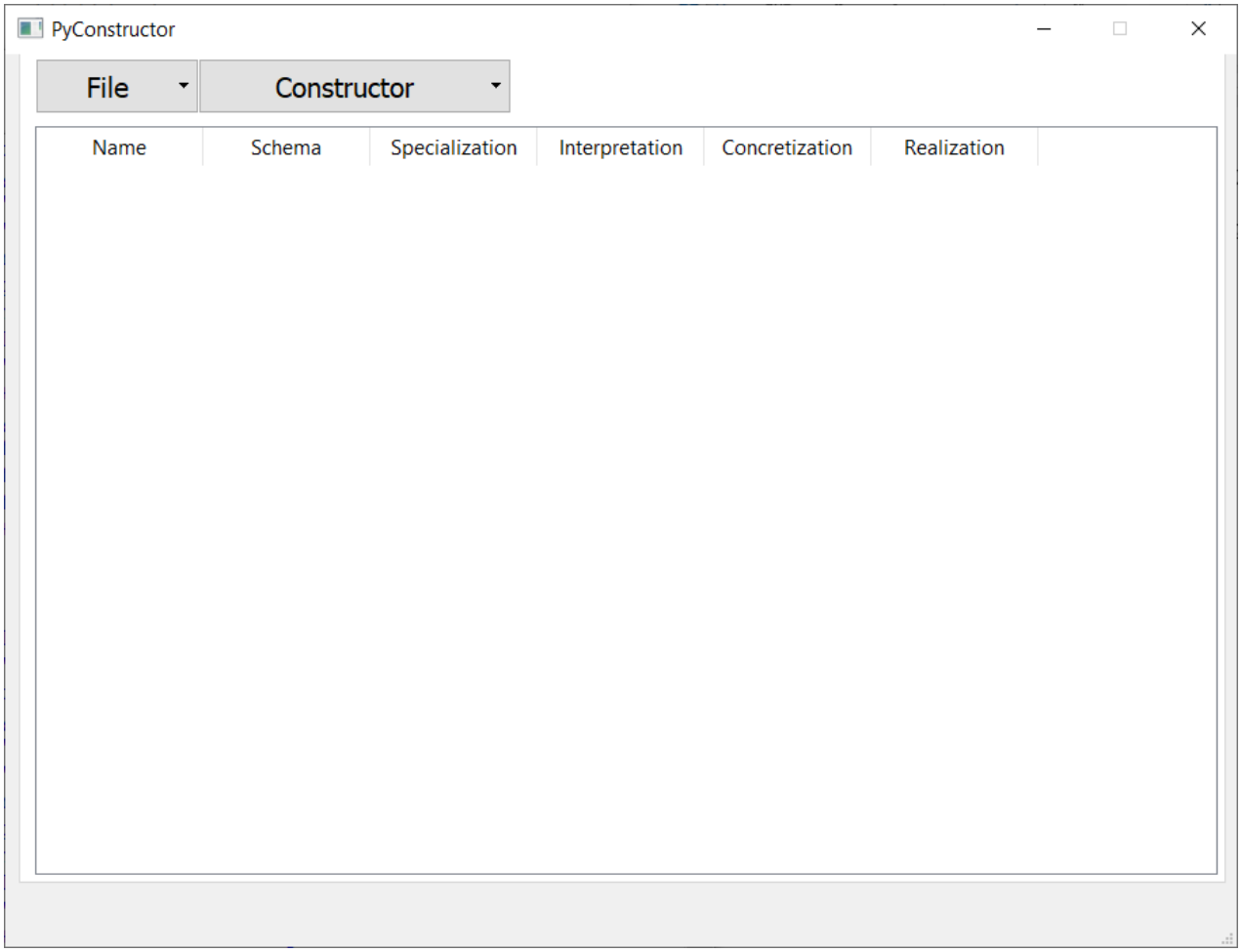


Рисунок 3.1 – Головне меню програми

Для створення нового конструктору перейдіть «Consructor»-«Add constructor», після чого відобразяться елементи опису початкових параметрів конструктора (рисунок 3.2).

Після створення нового конструктора (обрати дію «ОК»), новий конструктор з’явиться у таблиці існуючих конструкторів сесії. Для подальшої роботи з конструкторами оберіть один з існуючих конструкторі (натиснути на одну з комірок строки певного конструктора) й оберіть один з пунктів меню «Construcor».

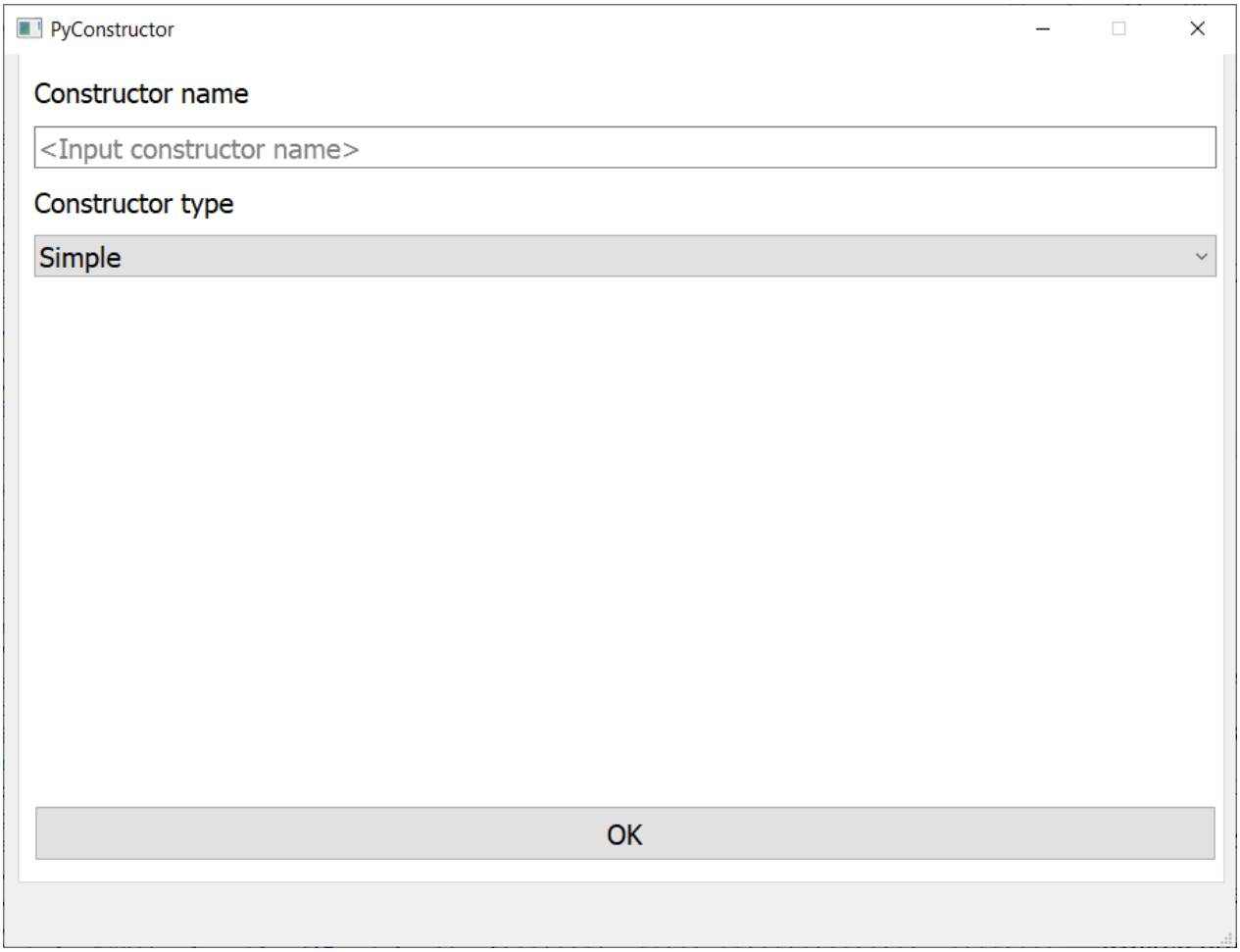


Рисунок 3.2 – Діалог створення нового конструктору

Пункт «Specialization» відкриє діалог заповнення інформації спеціалізації конструктора: задання параметрів конструктора (заповнення таблиці параметрів) та обрання форми відображення (вибір в діалозі однієї з можливих форм відображень й вказання параметрів для відображення) (рисунок 3.3).

Створіть необхідну кількість атрибутів конструктора, антиснувши кнопку «+» та заповніть інформацію про назви атрибутів та типи даних, які вони приймають.

Якщо якийсь з атрибутів не потрібен – натиснить кнопку «-» та видаліть атрибут.

Заповніть інформацію про відображення значень на формі відображення для осі ординат та абсцис.

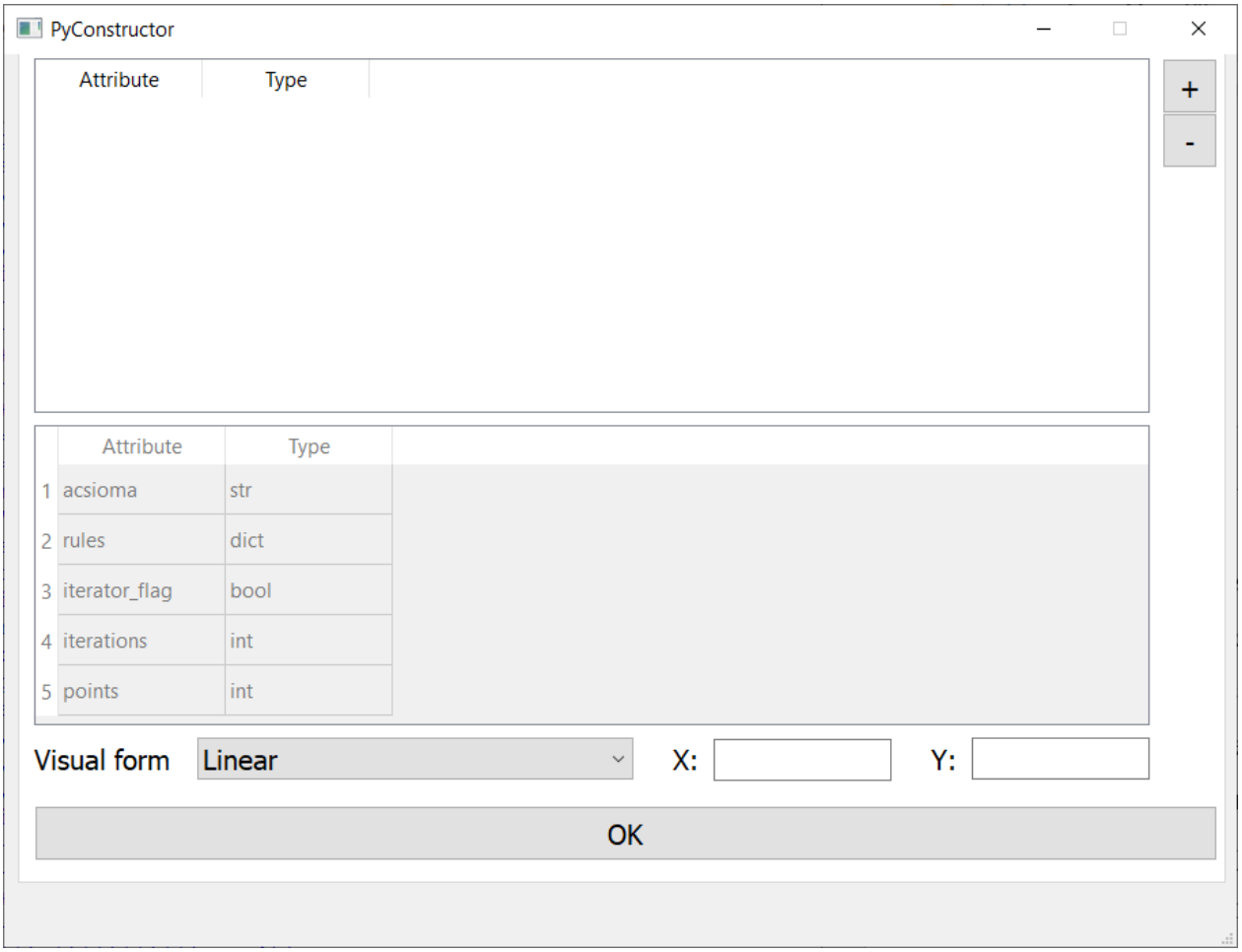


Рисунок 3.3 – Діалог заповнення інформації спеціалізації

Пункт «Interpretation» відкриє діалог заповнення інтерпретації символів мультисимвольної стрічки під час її інтерпретування (заповнення таблиці символів). Для алгоритмічного конструктора відкриється діалог додавання програмних функцій інтерпретування до конструктору.

На рис.3.4 представлено вікно опису інтерпретування простого конструктору.

Заповніть інформацію про інтерпретацію символів, додаючи за допомогою кнопки «+». Видалення непотрібних строк інформації відбувається за допомогою кнопки «-».

Описання функцій інтерпретування алгоритмічного конструктору представлено на рис.3.5. Заповніть поле тексом на мові програмування Python та додайте до конструктору за допомогою стрілок. Для зміни доданої функції використовується кнопка обратної стрілки.

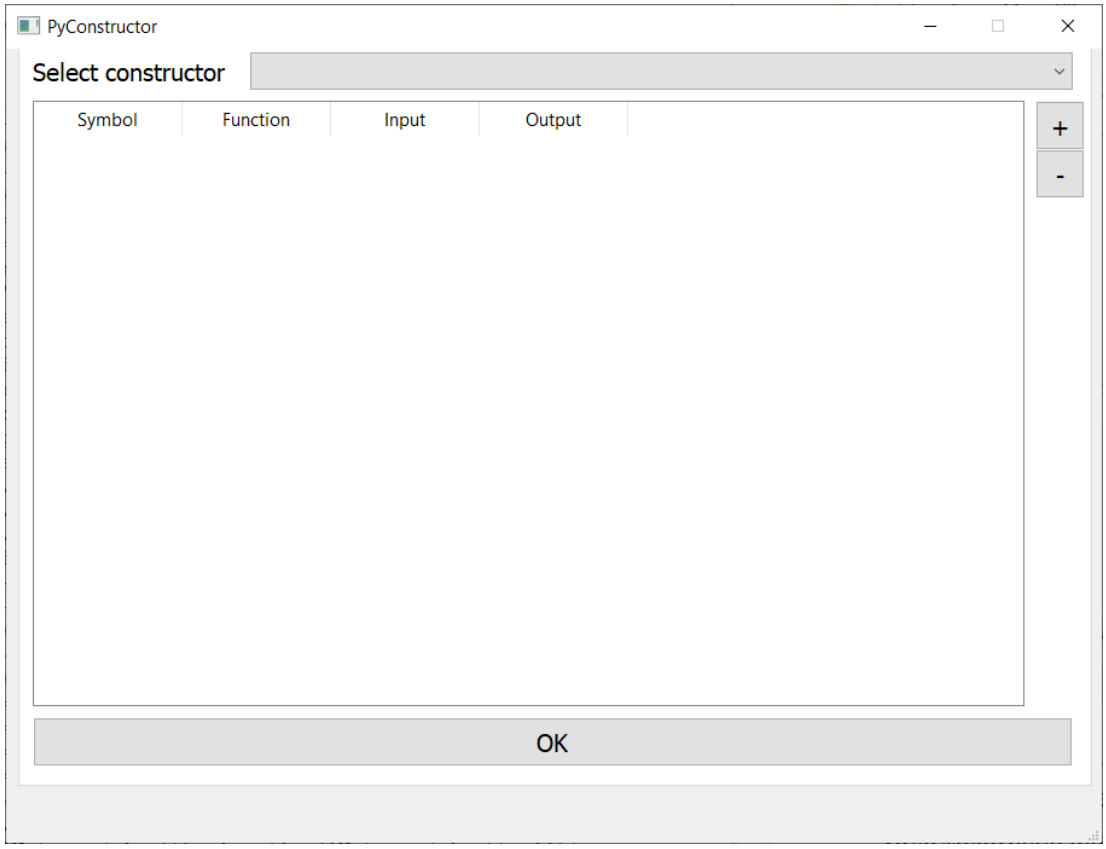


Рисунок 3.4 – Діалог заповнення інформації інтерпретування символів

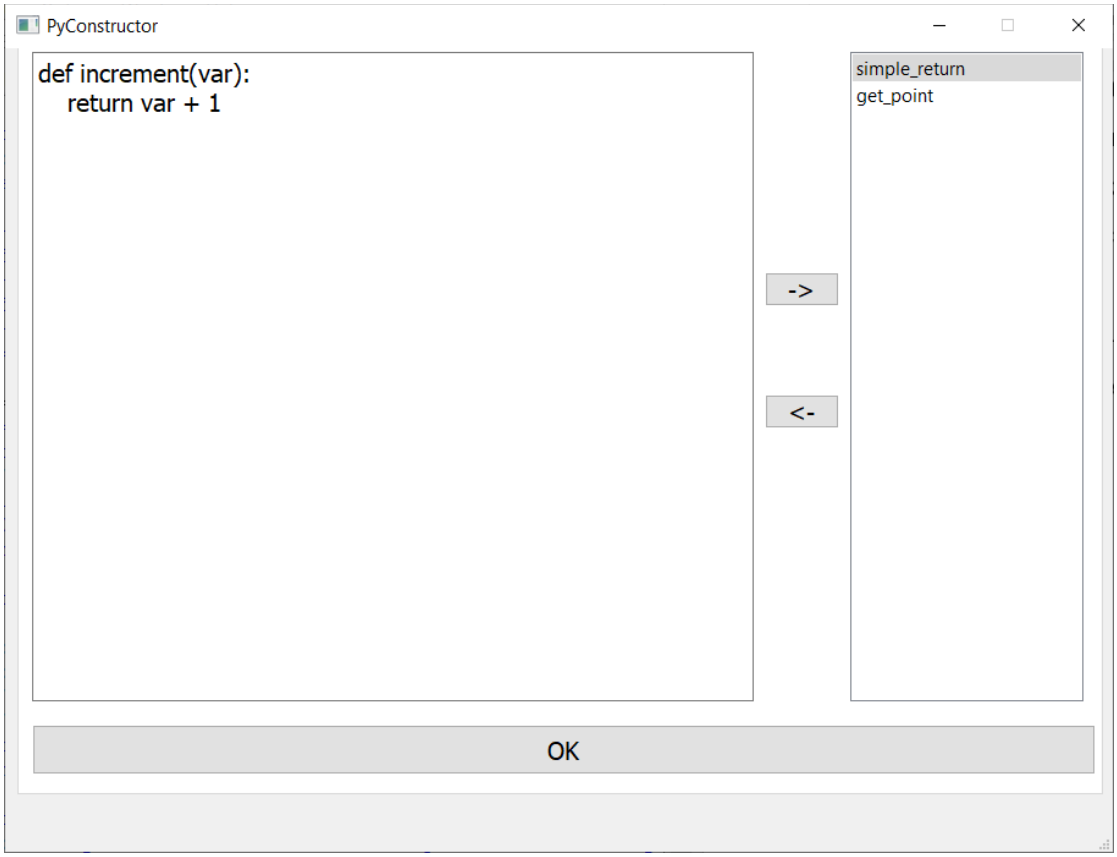


Рисунок 3.5 – Діалог опису функцій алгоритмічного конструктору

Пункт «Concretization» відкриє діалог заповнення початкових значень параметрів описаних на етапі спеціалізації (заповнення таблиці початкових значень параметрів), що наведено на рис.3.6.

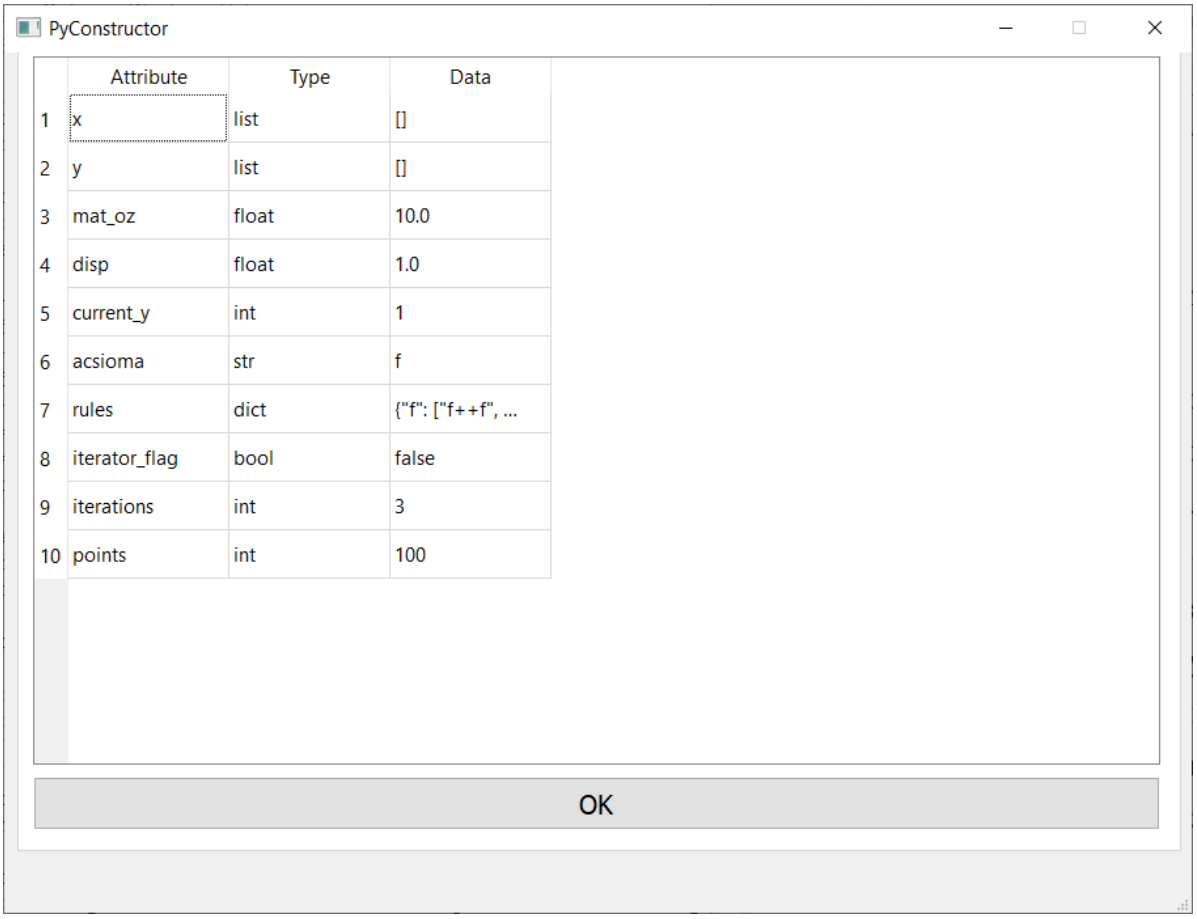


Рисунок 3.6 – Діалог конкретизації атрибутів конструктору

Пункт «Realization» відобразить результати обробки кожного з конструкторів. Перед користувачем буде представлена інформацію про атрибути та кінцеву стрічку конструктору (рисунок 3.7) та вікно графічного відображення (рисунок 3.7).

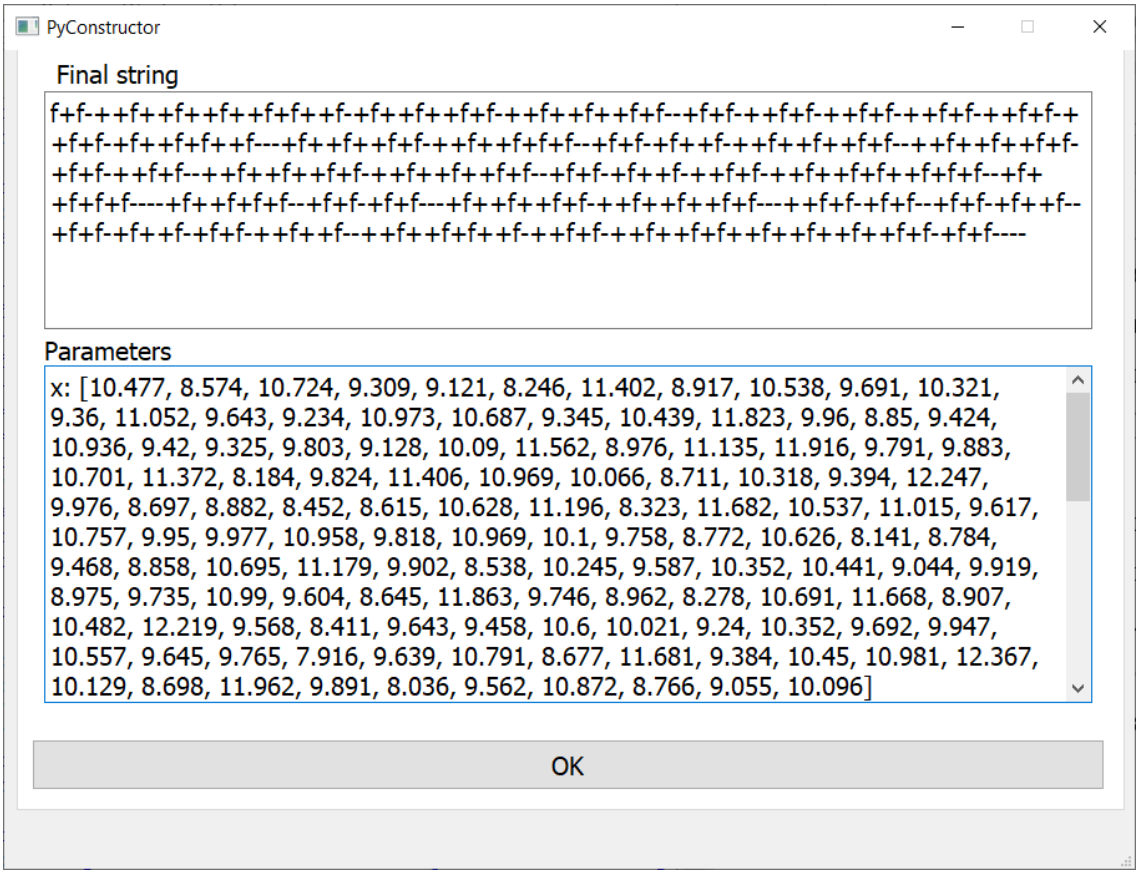


Рисунок 3.7 – Відображення значень атрибутів контсруктору

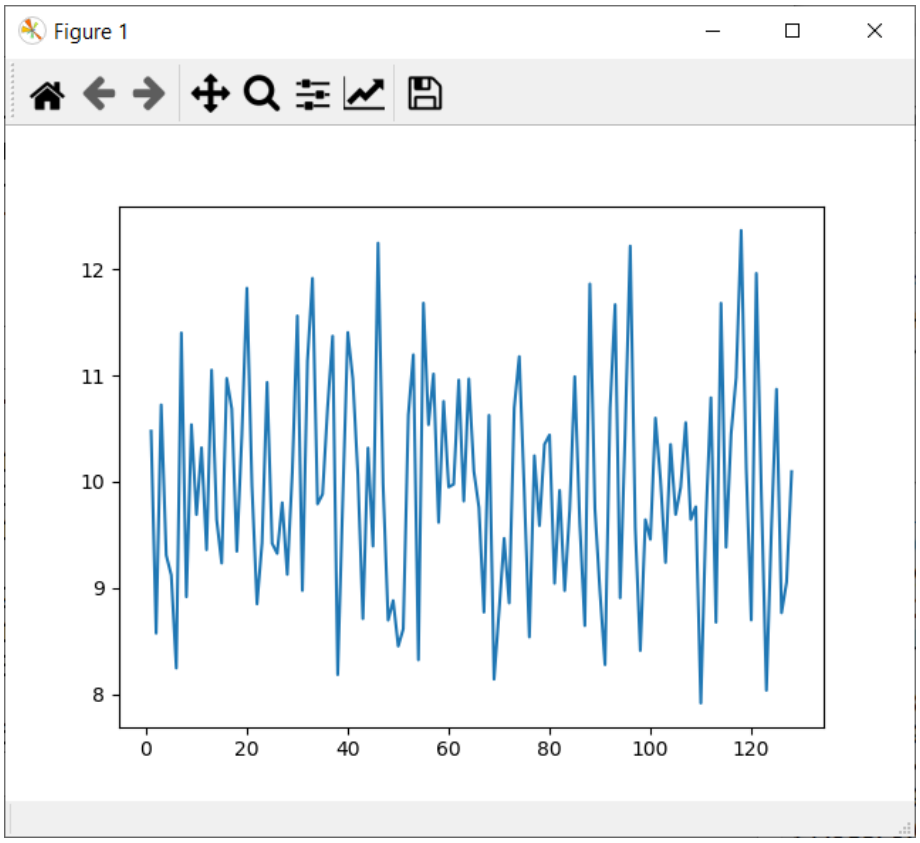


Рисунок 3.8 – Форма представлення графічного відображення

ЗАТВЕРДЖЕНО
1116130.01194-01 ЛЗ

СИСТЕМА МОДЕЛВАННЯ ПРОДУЦІЙНИХ КОНСТРУКТОРІВ

Текст програми
1116130.01194-01 12 01
Аркушів 22

2020

ЗМІСТ

1 СТРУКТУРА ПРОГРАММИ

Програма складається з:

– app.py – файл, що запускає програму;
– form.py – файл, що містить інтерфейс програми;
– constructors_form.py – файл, що містить логіку роботи програмного інтерфейсу;
– constructor_domain.py – файл, що містить сутності конструктору;
– constructors.pkl – файл сеccії.

2 ТЕКСТ ПРОГРАММИ

app.py

```
import sys

from PyQt5 import QtWidgets

from constructors_forms import MainWindow

app = QtWidgets.QApplication([])
application = MainWindow()
application.show()

sys.exit(app.exec())
```

form.py

```
from PyQt5 import QtCore, QtGui, QtWidgets



class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(907, 657)
        font = QtGui.QFont()
        font.setPointSize(12)
        MainWindow.setFont(font)
        self.centralwidget =
QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.tabWidget =
QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setGeometry(QtCore.QRect(10, -40,
891, 651))
        self.tabWidget.setObjectName("tabWidget")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.constructor_table =
QtWidgets.QTableWidget(self.tab)
        self.constructor_table.setGeometry(QtCore.QRect(10,
60, 871, 551))

self.constructor_table.setObjectName("constructor_table")
        self.constructor_table.setColumnCount(6)
        self.constructor_table.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.constructor_table.setHorizontalHeaderItem(0,
item)
        item = QtWidgets.QTableWidgetItem()
        self.constructor_table.setHorizontalHeaderItem(1,
item)
        item = QtWidgets.QTableWidgetItem()
        self.constructor_table.setHorizontalHeaderItem(2,
item)
        item = QtWidgets.QTableWidgetItem()
        self.constructor_table.setHorizontalHeaderItem(3,
item)
        item = QtWidgets.QTableWidgetItem()
        self.constructor_table.setHorizontalHeaderItem(4,
item)
        item = QtWidgets.QTableWidgetItem()
        self.constructor_table.setHorizontalHeaderItem(5,
item)
        self.menu_button = QtWidgets.QPushButton(self.tab)
        self.menu_button.setGeometry(QtCore.QRect(130,
10, 231, 41))
        self.menu_button.setObjectName("menu_button")
        self.file_button = QtWidgets.QPushButton(self.tab)
        self.file_button.setGeometry(QtCore.QRect(10, 10,
121, 41))
        self.file_button.setObjectName("file_button")
        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.name_text = QtWidgets.QLineEdit(self.tab_2)
        self.name_text.setGeometry(QtCore.QRect(10, 60,
871, 31))
```

```
        self.name_text.setText("")
        self.name_text.setObjectName("name_text")
        self.save_new = QtWidgets.QPushButton(self.tab_2)
        self.save_new.setGeometry(QtCore.QRect(10, 560,
871, 41))
        self.save_new.setObjectName("save_new")
        self.label_3 = QtWidgets.QLabel(self.tab_2)
        self.label_3.setGeometry(QtCore.QRect(10, 100, 231,
31))
        self.label_3.setObjectName("label_3")
        self.label = QtWidgets.QLabel(self.tab_2)
        self.label.setGeometry(QtCore.QRect(10, 20, 1021,
30))
        self.label.setObjectName("label")
        self.type_box = QtWidgets.QComboBox(self.tab_2)
        self.type_box.setEnabled(True)
        self.type_box.setGeometry(QtCore.QRect(10, 140,
871, 31))
        self.type_box.setObjectName("type_box")
        self.type_box.addItem("")
        self.type_box.addItem("")
        self.tabWidget.addTab(self.tab_2, "")
        self.tab_4 = QtWidgets.QWidget()
        self.tab_4.setObjectName("tab_4")
        self.spec_add = QtWidgets.QPushButton(self.tab_4)
        self.spec_add.setGeometry(QtCore.QRect(840, 10,
41, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.spec_add.setFont(font)
        self.spec_add.setObjectName("spec_add")
        self.spec_table =
QtWidgets.QTableWidget(self.tab_4)
        self.spec_table.setGeometry(QtCore.QRect(10, 10,
821, 261))
        self.spec_table.setObjectName("spec_table")
        self.spec_table.setColumnCount(2)
        self.spec_table.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.spec_table.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.spec_table.setHorizontalHeaderItem(1, item)
        self.save_spec = QtWidgets.QPushButton(self.tab_4)
        self.save_spec.setGeometry(QtCore.QRect(10, 560,
871, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.save_spec.setFont(font)
        self.save_spec.setObjectName("save_spec")
        self.spec_del = QtWidgets.QPushButton(self.tab_4)
        self.spec_del.setGeometry(QtCore.QRect(840, 50, 41,
41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.spec_del.setFont(font)
        self.spec_del.setObjectName("spec_del")
        self.form_box = QtWidgets.QComboBox(self.tab_4)
        self.form_box.setGeometry(QtCore.QRect(130, 510,
321, 31))
        self.form_box.setObjectName("form_box")
        self.form_box.addItem("")
        self.form_box.addItem("")
        self.label_4 = QtWidgets.QLabel(self.tab_4)
        self.label_4.setGeometry(QtCore.QRect(10, 510, 171,
31))
        self.label_4.setObjectName("label_4")
        self.label_2 = QtWidgets.QLabel(self.tab_4)
        self.label_2.setGeometry(QtCore.QRect(480, 505, 31,
41))
        self.label_2.setObjectName("label_2")
        self.label_6 = QtWidgets.QLabel(self.tab_4)
        self.label_6.setGeometry(QtCore.QRect(670, 510, 31,
31))
        self.label_6.setObjectName("label_6")
        self.x_box = QtWidgets.QLineEdit(self.tab_4)
        self.x_box.setGeometry(QtCore.QRect(510, 511, 131,
31))
        self.x_box.setObjectName("x_box")
        self.y_box = QtWidgets.QLineEdit(self.tab_4)
```

```
        self.y_box.setGeometry(QtCore.QRect(700, 510, 131,
31))
        self.y_box.setObjectName("y_box")
        self.spec_table_2 =
QtWidgets.QTableWidget(self.tab_4)
        self.spec_table_2.setEnabled(False)
        self.spec_table_2.setGeometry(QtCore.QRect(10,
280, 821, 221))
        self.spec_table_2.setObjectName("spec_table_2")
        self.spec_table_2.setColumnCount(2)
        self.spec_table_2.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.spec_table_2.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.spec_table_2.setHorizontalHeaderItem(1, item)
        self.tabWidget.addTab(self.tab_4, "")
        self.tab_6 = QtWidgets.QWidget()
        self.tab_6.setObjectName("tab_6")
        self.label_5 = QtWidgets.QLabel(self.tab_6)
        self.label_5.setGeometry(QtCore.QRect(10, 10, 171,
31))
        self.label_5.setObjectName("label_5")
        self.algo_box = QtWidgets.QComboBox(self.tab_6)
        self.algo_box.setGeometry(QtCore.QRect(190, 10,
681, 31))
        self.algo_box.setObjectName("algo_box")
        self.algo_box.addItem("")
        self.algo_box.setItemText(0, "")
        self.inter_table =
QtWidgets.QTableWidget(self.tab_6)
        self.inter_table.setGeometry(QtCore.QRect(10, 50,
821, 501))
        self.inter_table.setObjectName("inter_table")
        self.inter_table.setColumnCount(4)
        self.inter_table.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.inter_table.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.inter_table.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.inter_table.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.inter_table.setHorizontalHeaderItem(3, item)
        self.save_intr = QtWidgets.QPushButton(self.tab_6)
        self.save_intr.setGeometry(QtCore.QRect(10, 560,
861, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.save_intr.setFont(font)
        self.save_intr.setObjectName("save_intr")
        self.inter_add = QtWidgets.QPushButton(self.tab_6)
        self.inter_add.setGeometry(QtCore.QRect(840, 50,
41, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.inter_add.setFont(font)
        self.inter_add.setObjectName("inter_add")
        self.inter_del = QtWidgets.QPushButton(self.tab_6)
        self.inter_del.setGeometry(QtCore.QRect(840, 90,
41, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.inter_del.setFont(font)
        self.inter_del.setObjectName("inter_del")
        self.tabWidget.addTab(self.tab_6, "")
        self.tab_3 = QtWidgets.QWidget()
        self.tab_3.setObjectName("tab_3")
        self.conc_table =
QtWidgets.QTableWidget(self.tab_3)
        self.conc_table.setGeometry(QtCore.QRect(10, 10,
861, 541))
        self.conc_table.setObjectName("conc_table")
        self.conc_table.setColumnCount(3)
        self.conc_table.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.conc_table.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.conc_table.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.conc_table.setHorizontalHeaderItem(2, item)
```

```
        self.save_conc = QtWidgets.QPushButton(self.tab_3)
        self.save_conc.setGeometry(QtCore.QRect(10, 560,
861, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.save_conc.setFont(font)
        self.save_conc.setObjectName("save_conc")
        self.tabWidget.addTab(self.tab_3, "")
        self.tab_7 = QtWidgets.QWidget()
        self.tab_7.setObjectName("tab_7")
        self.final_string_box =
QtWidgets.QPlainTextEdit(self.tab_7)
        self.final_string_box.setGeometry(QtCore.QRect(20,
40, 841, 191))

self.final_string_box.setObjectName("final_string_box")
        self.label_7 = QtWidgets.QLabel(self.tab_7)
        self.label_7.setGeometry(QtCore.QRect(30, 10, 101,
31))
        self.label_7.setObjectName("label_7")
        self.save_real = QtWidgets.QPushButton(self.tab_7)
        self.save_real.setGeometry(QtCore.QRect(10, 560,
861, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.save_real.setFont(font)
        self.save_real.setObjectName("save_real")
        self.label_8 = QtWidgets.QLabel(self.tab_7)
        self.label_8.setGeometry(QtCore.QRect(20, 240, 121,
16))
        self.label_8.setObjectName("label_8")
        self.params_box =
QtWidgets.QPlainTextEdit(self.tab_7)
        self.params_box.setGeometry(QtCore.QRect(20, 260,
841, 271))
        self.params_box.setObjectName("params_box")
        self.tabWidget.addTab(self.tab_7, "")
        self.tab_5 = QtWidgets.QWidget()
        self.tab_5.setObjectName("tab_5")
        self.code_text =
QtWidgets.QPlainTextEdit(self.tab_5)
        self.code_text.setGeometry(QtCore.QRect(10, 10,
591, 531))
        self.code_text.setPlainText("")
        self.code_text.setObjectName("code_text")
        self.save_algo = QtWidgets.QPushButton(self.tab_5)
        self.save_algo.setGeometry(QtCore.QRect(10, 560,
871, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.save_algo.setFont(font)
        self.save_algo.setObjectName("save_algo")
        self.func_add = QtWidgets.QPushButton(self.tab_5)
        self.func_add.setGeometry(QtCore.QRect(610, 190,
61, 28))
        self.func_add.setObjectName("func_add")
        self.code_list = QtWidgets.QListWidget(self.tab_5)
        self.code_list.setGeometry(QtCore.QRect(680, 10,
191, 531))
        self.code_list.setObjectName("code_list")
        self.func_del = QtWidgets.QPushButton(self.tab_5)
        self.func_del.setGeometry(QtCore.QRect(610, 290,
61, 28))
        self.func_del.setObjectName("func_del")
        self.tabWidget.addTab(self.tab_5, "")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 907,
26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(5)
```

```
QtCore.QMetaObject.connectSlotsByName(MainWindow
)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(_translate("MainWindow",
"PyConstructor"))
        item = self.constructor_table.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Name"))
        item = self.constructor_table.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Schema"))
        item = self.constructor_table.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow",
"Specialization"))
        item = self.constructor_table.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow",
"Interpretation"))
        item = self.constructor_table.horizontalHeaderItem(4)
        item.setText(_translate("MainWindow",
"Concretization"))
        item = self.constructor_table.horizontalHeaderItem(5)
        item.setText(_translate("MainWindow",
"Realization"))
        self.menu_button.setText(_translate("MainWindow",
"Constructor"))
        self.file_button.setText(_translate("MainWindow",
"File"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab
), _translate("MainWindow", "Hub"))

self.name_text.setPlaceholderText(_translate("MainWindo
w", "<Input constructor name>"))
        self.save_new.setText(_translate("MainWindow",
"OK"))
        self.label_3.setText(_translate("MainWindow",
"Constructor type"))
        self.label.setText(_translate("MainWindow",
"Constructor name"))
        self.type_box.setItemText(0,
_translate("MainWindow", "Simple"))
        self.type_box.setItemText(1,
_translate("MainWindow", "Algorithms"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab
_2), _translate("MainWindow", "New"))
        self.spec_add.setText(_translate("MainWindow",
"+"))
        item = self.spec_table.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Attribute"))
        item = self.spec_table.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Type"))
        self.save_spec.setText(_translate("MainWindow",
"OK"))
        self.spec_del.setText(_translate("MainWindow", "-"))
        self.form_box.setItemText(0,
_translate("MainWindow", "Linear"))
        self.form_box.setItemText(1,
_translate("MainWindow", "Dots"))
        self.label_4.setText(_translate("MainWindow",
"Visual form"))
        self.label_2.setText(_translate("MainWindow", "X:"))
        self.label_6.setText(_translate("MainWindow", "Y:"))
        item = self.spec_table_2.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Attribute"))
        item = self.spec_table_2.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Type"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab
_4), _translate("MainWindow", "Specialization"))
        self.label_5.setText(_translate("MainWindow",
"Select constructor"))
        item = self.inter_table.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Symbol"))
        item = self.inter_table.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Function"))
        item = self.inter_table.horizontalHeaderItem(2)
```

```
        item.setText(_translate("MainWindow", "Input"))
        item = self.inter_table.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "Output"))
        self.save_intr.setText(_translate("MainWindow",
"OK"))
        self.inter_add.setText(_translate("MainWindow",
"+"))
        self.inter_del.setText(_translate("MainWindow", "-"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab
_6), _translate("MainWindow", "Interpretation"))
        item = self.conc_table.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Attribute"))
        item = self.conc_table.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Type"))
        item = self.conc_table.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Data"))
        self.save_conc.setText(_translate("MainWindow",
"OK"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab
_3), _translate("MainWindow", "Concretization"))
        self.label_7.setText(_translate("MainWindow", "Final
string"))
        self.save_real.setText(_translate("MainWindow",
"OK"))
        self.label_8.setText(_translate("MainWindow",
"Parameters"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab
_7), _translate("MainWindow", "Realization"))

self.code_text.setPlaceholderText(_translate("MainWindo
w", "<Input constructor description. Optional>"))
        self.save_algo.setText(_translate("MainWindow",
"OK"))
        self.func_add.setText(_translate("MainWindow", "-
>"))
        self.func_del.setText(_translate("MainWindow", "<-
"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab
_5), _translate("MainWindow", "Algo"))
```

constructors_form.py

```
"""
Module for windows classes.
"""
import copy
import pickle
import sys

from PyQt5 import QtWidgets, QtCore, QtGui

from form import Ui_MainWindow as Constr
from constructor_domain import ConstructorsContainer

class MainWindow(QtWidgets.QMainWindow, Constr):
    """
    Main window class.
    """
    def __init__(self, parent=None):
        try:
            super(MainWindow, self).__init__(parent)
            self.setupUi(self)
            self.setFixedSize(self.size())

self.constructor_table.setSelectionMode(QtWidgets.QAbst
ractItemView.SingleSelection)

self.constructor_table.setSelectionBehavior(QtWidgets.QT
ableWidget.SelectRows)


self.spec_table.setSelectionMode(QtWidgets.QAbstractIte
mView.SingleSelection)

self.inter_table.setSelectionMode(QtWidgets.QAbstractIte
mView.SingleSelection)
```

```
        # Done
        file_menu = QtWidgets.QMenu()
        # Done
        saveAction = QtWidgets.QAction('Save', self)
        saveAction.triggered.connect(self.save_file)
        # Done
        closeAction = QtWidgets.QAction('Close', self)
        closeAction.triggered.connect(self.close_file)

        const_menu = QtWidgets.QMenu()
        # Done
        addConstructorAction = QtWidgets.QAction('Add
constructor', self)

addConstructorAction.triggered.connect(self.add_construc
tor)
        # Done
        delConstructorAction =
QtWidgets.QAction('Delete constructor', self)

delConstructorAction.triggered.connect(self.delete_constr
uctor)
        # Done
        specConstructorAction =
QtWidgets.QAction('Specialization', self)

specConstructorAction.triggered.connect(self.add_speciali
zation)
        # Done
        intrConstructorAction =
QtWidgets.QAction('Interpretation', self)

intrConstructorAction.triggered.connect(self.add_interpret
ation)
        # Done
        concConstructorAction =
QtWidgets.QAction('Concretization', self)

concConstructorAction.triggered.connect(self.add_concret
ization)
        # Done
        realConstructorAction =
QtWidgets.QAction('Realization', self)

realConstructorAction.triggered.connect(self.get_realizatio
n)

        const_menu.addAction(addConstructorAction)
        const_menu.addAction(delConstructorAction)
        const_menu.addSeparator()
        const_menu.addAction(specConstructorAction)
        const_menu.addAction(intrConstructorAction)
        const_menu.addAction(concConstructorAction)
        const_menu.addAction(realConstructorAction)

        file_menu.addAction(saveAction)
        file_menu.addAction(closeAction)

        self.file_button.setMenu(file_menu)
        self.menu_button.setMenu(const_menu)

        # Done

self.save_new.clicked.connect(self.save_constructor)
        # Done

self.save_spec.clicked.connect(self.save_specialization)
        # Done

self.save_intr.clicked.connect(self.save_interpretation)
        # Done

self.save_conc.clicked.connect(self.save_concretization)
        # Done

self.save_real.clicked.connect(self.save_realization)
        # Done
```

```
self.save_algo.clicked.connect(self.save_algo_interpretatio
n)

        # Done
        self.spec_add.clicked.connect(self.add_spec_attr)
        # Done
        self.spec_del.clicked.connect(self.delete_spec_attr)

        # Done
        self.inter_add.clicked.connect(self.add_intr_attr)
        # Done
        self.inter_del.clicked.connect(self.delete_intr_attr)

        # Done
        self.func_add.clicked.connect(self.add_code)
        # Done
        self.func_del.clicked.connect(self.get_code)

        self.constructors =
pickle.load(open('constructors.pkl', 'rb'))
        for con_name, con in
self.constructors.constructors.items():
            data = (
                con_name,
                con.get_type(),
            )
            row = self.constructor_table.rowCount()
            self.constructor_table.insertRow(row)
            part_statuses = {"Spec": con.spec_state,
                             "Inter": con.intr_state,
                             "Concr": con.conc_state,
                             "Realz": con.real_state,
                             }
            self.constructor_table.setItem(row, 0,
QtWidgets.QTableWidgetItem(data[0]))
            self.constructor_table.setItem(row, 1,
QtWidgets.QTableWidgetItem(data[1]))
            self.constructor_table.setItem(row, 2,
QtWidgets.QTableWidgetItem(part_statuses["Spec"]))
            self.constructor_table.setItem(row, 3,
QtWidgets.QTableWidgetItem(part_statuses["Inter"]))
            self.constructor_table.setItem(row, 4,
QtWidgets.QTableWidgetItem(part_statuses["Concr"]))
            self.constructor_table.setItem(row, 5,
QtWidgets.QTableWidgetItem(part_statuses["Realz"]))
        self.constructors.print_constructors()

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
0)
    except Exception as e:
        print(e)

  # Done
  def save_file(self):
    try:
        pickle.dump(obj=self.constructors,
file=open('constructors.pkl', 'wb'))
    except Exception as e:
        print(e)

  # Done
  def close_file(self):
    sys.exit()

  # Done
  def add_constructor(self):
    try:
        self.name_text.setText("")
        self.type_box.setCurrentIndex(0)

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
1)
    except Exception as e:
        print(e)

  # Done
  def add_specialization(self):
    try:
        if not self.constructor_table.selectedItems():
```

```
            return
        if self.constructor_table.selectedItems()[2].text()
== "Not required":
            return
        if self.constructor_table.selectedItems()[1].text()
== "Algorithms":
            msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
            msg.setText("Algoritms constuctor not requied
specialization.")
            msg.setWindowTitle("Information")
            msg.exec_()
        else:
            self.const_name =
self.constructor_table.selectedItems()[0].text()
            self.constructor =
self.constructors.get_constr(self.const_name)
            attributes = self.constructor.get_spec()
            x = self.constructor.x
            y = self.constructor.y
            plot = self.constructor.plot
            self.x_box.setText(x)
            self.y_box.setText(y)
            self.form_box.setCurrentIndex(plot)
            build_in_attrs = {"acsioma": "str", "rules":
"dict",
                              "iterator_flag": "bool", "iterations":
"int",
                              "points": "int"}
            for k, v in attributes.items():
                if k in build_in_attrs:
                    continue
                row = self.spec_table.rowCount()
                self.spec_table.insertRow(row)
                self.spec_table.setItem(row, 0,
QtWidgets.QTableWidgetItem(k))
                self.spec_table.setItem(row, 1,
QtWidgets.QTableWidgetItem(v))
            for k, v in build_in_attrs.items():
                row = self.spec_table_2.rowCount()
                self.spec_table_2.insertRow(row)
                self.spec_table_2.setItem(row, 0,
QtWidgets.QTableWidgetItem(k))
                self.spec_table_2.setItem(row, 1,
QtWidgets.QTableWidgetItem(v))

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
2)
        except Exception as e:
            print(e)


    # Done
    def add_interpretation(self):
        try:
            if not self.constructor_table.selectedItems():
                return
            if self.constructor_table.selectedItems()[3].text()
== "Not required":
                return
            self.const_name =
self.constructor_table.selectedItems()[0].text()
            self.constructor =
self.constructors.get_constr(self.const_name)
            if self.constructor_table.selectedItems()[1].text()
== "Algorithms":
                self.code_text.setPlainText("")
                self.codes =
copy.deepcopy(self.constructor.get_intr())
                self.code_list.clear()
                for code in self.codes:
                    self.code_list.addItem(code)

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
6)
            elif self.constructor_table.selectedItems()[1].text()
== "Simple":
                algo_constrs =
self.constructors.get_algorithm_constr_names()
                self.algo_box.clear()
```

```
            for algo in algo_constrs:
                self.algo_box.addItem(algo)
            intr = self.constructor.get_intr()
            for algo in intr:
                for act in intr[algo]:
                    row = self.inter_table.rowCount()
                    self.inter_table.insertRow(row)
                    self.inter_table.setItem(row, 0,
QtWidgets.QTableWidgetItem(algo))
                    self.inter_table.setItem(row, 1,
QtWidgets.QTableWidgetItem(act["function"]))
                    self.inter_table.setItem(row, 2,
QtWidgets.QTableWidgetItem(act["input"]))
                    self.inter_table.setItem(row, 3,
QtWidgets.QTableWidgetItem(act["output"]))

self.algo_box.setCurrentText(self.constructor.algo_intr_na
me)

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
3)
        except Exception as e:
            print(e)

    # Done
    def add_code(self):
        try:
            code_text = self.code_text.toPlainText().strip()
            def_place = code_text.find("def") + 3
            open_place = code_text.find("(")
            if def_place == 2 or open_place == -1:
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, define function.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
            func_name =
code_text[def_place:open_place].strip()
            self.codes[func_name] = code_text
            self.code_text.setPlainText("")

self.code_list.addItem(QtWidgets.QListWidgetItem(func_
name))
        except Exception as e:
            print(e)

    # Done
    def get_code(self):
        try:
            if not self.code_list.selectedItems():
                return
            func_name =
self.code_list.selectedItems()[0].text()
            list_items = self.code_list.selectedItems()
            for item in list_items:
                self.code_list.takeItem(self.code_list.row(item))
            code_text = self.codes[func_name]
            del self.codes[func_name]
            self.code_text.setPlainText(code_text)
        except Exception as e:
            print(e)

    # Done
    def add_concretization(self):
        try:
            if not self.constructor_table.selectedItems():
                return
            if self.constructor_table.selectedItems()[4].text()
== "Not required":
                return
            if self.constructor_table.selectedItems()[2].text() !=
"Ready":
                return
            self.const_name =
self.constructor_table.selectedItems()[0].text()
            self.constructor =
self.constructors.get_constr(self.const_name)
            attributes = self.constructor.get_spec()
```

```
            attributes_conc = self.constructor.get_conc()
            for k, v in attributes.items():
                row = self.conc_table.rowCount()
                self.conc_table.insertRow(row)
                item_1 = QtWidgets.QTableWidgetItem(k)
                item_1.setFlags(QtCore.Qt.ItemIsEnabled)
                item_2 = QtWidgets.QTableWidgetItem(v)
                item_2.setFlags(QtCore.Qt.ItemIsEnabled)
                self.conc_table.setItem(row, 0, item_1)
                self.conc_table.setItem(row, 1, item_2)
                self.conc_table.setItem(row, 2,
QtWidgets.QTableWidgetItem(str(attributes_conc[k]).low
er().replace("'", "")))

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
4)
        except Exception as e:
            print(e)

    # Done
    def get_realization(self):
        try:
            if not self.constructor_table.selectedItems():
                return
            if self.constructor_table.selectedItems()[5].text()
== "Not required":
                return
            self.const_name =
self.constructor_table.selectedItems()[0].text()
            self.constructor =
self.constructors.get_constr(self.const_name)
            output = self.constructor.create_realization()

self.final_string_box.setPlainText(self.constructor.final_str
ing)
            output_string = ""
            for par in output:
                output_string += str(par) + ": " + str(output[par])
+ "\n"
            self.params_box.setPlainText(output_string)
            self.create_graphic(output[self.constructor.x],
output[self.constructor.y], self.constructor.plot)

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
5)
        except Exception as e:
            print(e)

    # Done
    def create_graphic(self, y, x, plot_type):
        try:
            import matplotlib.pyplot as plt
            if plot_type == 0:
                plt.plot(x, y)
            else:
                plt.plot(x, y, 'bo')
            plt.ylabel('some numbers')
            plt.show()
        except Exception as e:
            print(e)

    # Done
    def save_constructor(self):
        try:
            data = (
                self.name_text.text().strip(),
                str(self.type_box.currentText())
            )
            if data[0].strip() == "":
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, write constructor name.")
                msg.setWindowTitle("Information")
                msg.exec_()
            else:
                row = self.constructor_table.rowCount()
                self.constructor_table.insertRow(row)
                part_statuses = {"Spec": "Not required" if
self.type_box.currentText() == "Multi" else ("Ready" if
```

```
self.type_box.currentText() == "Algorithms" else "Not
ready"),
                        "Inter": "Not required" if
self.type_box.currentText() == "Multi" else "Not ready",
                        "Concr": "Not required" if
self.type_box.currentText() == "Multi" or
self.type_box.currentText() == "Algorithms" else "Not
ready",
                        "Realz": "Not required" if
self.type_box.currentText() == "Algorithms" else "Not
ready",
                        }
            self.constructor_table.setItem(row, 0,
QtWidgets.QTableWidgetItem(data[0]))
            self.constructor_table.setItem(row, 1,
QtWidgets.QTableWidgetItem(data[1]))
            self.constructor_table.setItem(row, 2,
QtWidgets.QTableWidgetItem(part_statuses["Spec"]))
            self.constructor_table.setItem(row, 3,
QtWidgets.QTableWidgetItem(part_statuses["Inter"]))
            self.constructor_table.setItem(row, 4,
QtWidgets.QTableWidgetItem(part_statuses["Concr"]))
            self.constructor_table.setItem(row, 5,
QtWidgets.QTableWidgetItem(part_statuses["Realz"]))
            self.constructors.add_constructor(data[0],
data[1])
            self.constructors.print_constructors()

            self.name_text.setText("")
            self.type_box.setCurrentIndex(0)
            self.go_back_to_constructors()
        except Exception as e:
            print(e)

    # Done
    def save_specialization(self):
        try:
            if self.x_box.text().strip() == "":
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, fill field 'X'.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
            elif self.y_box.text().strip() == "":
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, fill field 'Y'.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
            attributes = {}
            for row in range(self.spec_table.rowCount()):
                attributes[self.spec_table.item(row,
0).text().strip()] = self.spec_table.item(row,
1).text().strip().lower()
            for row in range(self.spec_table_2.rowCount()):
                attributes[self.spec_table_2.item(row,
0).text().strip()] =
self.spec_table_2.item(row,1).text().strip().lower()
            if "" in list(attributes.keys()):
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, fill empty attributes.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
            elif "" in list(attributes.values()):
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, fill empty attributes.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
            else:
```

```
            build_in_types = ["int", "str", "list", "float",
"dict", "bool"]
            attributes_values = list(set(attributes.values()))
            for tp in build_in_types:
                try:
                    attributes_values.remove(tp)
                except ValueError:
                    pass
            if attributes_values:
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, fill attributes with build-
in types.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
        self.constructor.set_spec(attributes)
        self.constructor.x = self.x_box.text().strip()
        self.constructor.y = self.y_box.text().strip()
        self.constructor.plot = self.form_box.currentIndex()
        self.constructor.spec_state = "Ready"

        self.constructors.print_constructors()

        self.name_text.setText("")

        while self.spec_table.rowCount() > 0:
            self.spec_table.removeRow(0)
        while self.spec_table_2.rowCount() > 0:
            self.spec_table_2.removeRow(0)
        self.x_box.setText("")
        self.y_box.setText("")
        self.form_box.setCurrentIndex(0)



self.constructor_table.selectedItems()[2].setText("Ready")
        self.go_back_to_constructors()
    except Exception as e:
        print(e)

    # Done
    def save_interpretation(self):
        try:
            self.constructor.algo_intr_name =
self.algo_box.currentText()
            algo_intr =
self.constructors.get_constr(self.algo_box.currentText()).g
et_intr()

self.constructor.set_algo(self.constructors.get_constr(self.a
lgo_box.currentText()).get_intr())
            intr = {}
            for row in range(self.inter_table.rowCount()):
                symbol = self.inter_table.item(row,
0).text().strip()
                func = self.inter_table.item(row, 1).text().strip()
                inp = self.inter_table.item(row, 2).text().strip()
                outp = self.inter_table.item(row, 3).text().strip()
                if symbol not in intr:
                    intr[symbol] = [{"function": func, "input":
inp, "output": outp}]
                else:
                    intr[symbol].append({"function": func,
"input": inp, "output": outp})
            self.constructor.set_intr(intr)

            while self.inter_table.rowCount() > 0:
                self.inter_table.removeRow(0)
            self.constructor.intr_state = "Ready"



self.constructor_table.selectedItems()[3].setText("Ready")
            self.constructors.print_constructors()
            self.go_back_to_constructors()
        except Exception as e:
            print(e)

    # Done
    def save_concretization(self):
```

```
        try:
            attributes = {}
            for row in range(self.conc_table.rowCount()):
                attr_data = self.conc_table.item(row,
2).text().strip().replace("'", "")
                attributes[self.conc_table.item(row,
0).text().strip()] = attr_data
            if "" in list(attributes.values()):
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, fill empty attributes.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
            finish = self.constructor.set_conc(attributes)
            if finish == -1:
                msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Information)
                msg.setText("Please, fill right values.")
                msg.setWindowTitle("Information")
                msg.exec_()
                return
            while self.conc_table.rowCount() > 0:
                self.conc_table.removeRow(0)
            self.constructor.conc_state = "Ready"

self.constructor_table.selectedItems()[4].setText("Ready")
            self.constructors.print_constructors()
            self.go_back_to_constructors()
        except Exception as e:
            print(e)

    # Done
    def add_intr_attr(self):
        try:
            row = self.inter_table.rowCount()
            self.inter_table.insertRow(row)
            self.inter_table.setItem(row, 0,
QtWidgets.QTableWidgetItem(""))
            self.inter_table.setItem(row, 1,
QtWidgets.QTableWidgetItem(""))
            self.inter_table.setItem(row, 2,
QtWidgets.QTableWidgetItem(""))
            self.inter_table.setItem(row, 3,
QtWidgets.QTableWidgetItem(""))
        except Exception as e:
            print(e)

    # Done
    def delete_intr_attr(self):
        try:
            if not self.inter_table.selectedItems():
                return
            row_index =
self.inter_table.selectedItems()[0].row()
            self.inter_table.removeRow(row_index)
        except Exception as e:
            print(e)

    # Done
    def save_algo_interpretation(self):
        try:
            self.constructor.set_intr(self.codes)
            self.constructor.intr_state = "Ready"

self.constructor_table.selectedItems()[3].setText("Ready")
            self.constructors.print_constructors()
            self.go_back_to_constructors()
        except Exception as e:
            print(e)

    # Done
    def save_realization(self):
        try:
            self.constructor.real_state = "Ready"

self.constructor_table.selectedItems()[5].setText("Ready")
```

```
            self.constructors.print_constructors()
            self.go_back_to_constructors()
        except Exception as e:
            print(e)

    # Done
    def add_spec_attr(self):
        try:
            row = self.spec_table.rowCount()
            self.spec_table.insertRow(row)
            self.spec_table.setItem(row, 0,
QtWidgets.QTableWidgetItem(""))
            self.spec_table.setItem(row, 1,
QtWidgets.QTableWidgetItem(""))
        except Exception as e:
            print(e)

    # Done
    def delete_spec_attr(self):
        try:
            if not self.spec_table.selectedItems():
                return
            row_index =
self.spec_table.selectedItems()[0].row()
            self.spec_table.removeRow(row_index)
        except Exception as e:
            print(e)

    # Done
    def go_back_to_constructors(self):

QtWidgets.QTabWidget.setCurrentIndex(self.tabWidget,
0)

    # Done
    def delete_constructor(self):
        try:
            if not self.constructor_table.selectedItems():
                return
            constructor_name =
self.constructor_table.selectedItems()[0].text()
            row_index =
self.constructor_table.selectedItems()[0].row()

self.constructors.del_constructor(constructor_name)
            self.constructor_table.removeRow(row_index)
            self.constructors.print_constructors()
        except Exception as e:
            print(e)
```

constructor_domain.py

```
from pydantic import BaseModel
import simplejson as json
import random
import string
import copy

class Constructor(BaseModel):
    name: str
    _type: str
    spec_state: str = "Not ready"
    intr_state: str = "Not ready"
    conc_state: str = "Not ready"
    real_state: str = "Not ready"
    conc: dict = {}
    intr: dict = {}
    spec: dict = {}
    real: dict = {}
    x: str = ""
    y: str = ""
    final_string: str = ""
    plot: int = 0

    algo_intr_name: str = ""
    algo_intr: dict = {}

    _new_str: str = ""
    _rules: dict = {}
```

```
    points: int = 4
    iterations: int = 1
    _iters_flag: bool = True

    build_in_types: str = {
        "int": 0,
        "float": 0.,
        "str": "",
        "bool": True,
        "dict": {},
        "list": [],
    }

    def time_plot(self):
        u"""Функция построения временного ряда."""
        kol_t = 0
        for i in self.final_string:
            if i in self.intr:  # Если символ есть в словаре
замен
                kol_t += self._parametrs[i][0](i)  # Выполняем
функцию символа
                if not self._iters and kol_t >= self._n: return

    def create_gram(self, data):
        u"""Функция построения грамматики на основе L-
систем."""
        _new_str = data["acsioma"]
        _rules = data["rules"]
        points = data["points"]
        iterations = data["iterations"]
        _iters_flag = data["iterator_flag"]

        b = string.ascii_lowercase
        kol_t = 0
        kol_iter = 1
        old_str = _new_str
        old_kol = kol_t
        while True:
            j = 0
            while j < len(_new_str):
                if _new_str[j] in _rules:
                    j_key = str(_new_str[j])
                    this_choose =
random.choice(_rules[_new_str[j]])
                    _new_str = _new_str[0:j] + this_choose +
_new_str[j + 1::]
                    j += len(this_choose)
                else:
                    j += 1
            if not _iters_flag:
                for let in _new_str:
                    if let in b:
                        kol_t += 1
                if kol_t >= points:
                    return _new_str
                if old_kol == kol_t:
                    return _new_str
                old_kol = kol_t
                kol_t = 0
            else:
                if kol_iter == iterations:
                    return _new_str
                kol_iter += 1
            if old_str == _new_str:
                return _new_str
            else:
                old_str = _new_str

    def get_type(self):
        return self._type

    def get_info(self):
        return {"name": self.name, "type": self._type,
                "spec": (self.spec_state, self.spec),
                "x": self.x, "y": self.y, "plot": self.plot,
                "intr": (self.intr_state, self.intr,
self.algo_intr_name, self.algo_intr),
                "conc": (self.conc_state, self.conc),
                "real": (self.real_state, self.real,
self.final_string),
```

```
            }

    def transform_value(self, value, tp):
        if tp == "bool":
            value = json.loads(value)
            if isinstance(value, bool):
                return value
            raise Exception
        if tp == "int":
            value = json.loads(value)
            if isinstance(value, int):
                return value
            raise Exception
        if tp == "float":
            value = json.loads(value)
            if isinstance(value, float):
                return value
            raise Exception
        if tp == "str":
            return value
        if tp == "dict":
            value = json.loads(value)
            if isinstance(value, dict):
                return value
            raise Exception
        if tp == "list":
            value = json.loads(value)
            if isinstance(value, list):
                return value
            raise Exception

    def set_conc(self, data):
        for k, v in data.items():
            try:
                self.conc[k] = self.transform_value(v,
self.spec[k])
            except:
                print("Wrong with ", k, v)
                return -1
        return 0

    def set_intr(self, data):
        self.intr = {}
        for k, v in data.items():
            self.intr[k] = v

    def set_spec(self, data):
        self.spec = {}
        for k, v in data.items():
            self.spec[k] = v
        self.set_base_values_for_spec()
        self.delete_overvalues()

    def delete_overvalues(self):
        for attr in self.conc.keys():
            if attr not in self.spec.keys():
                del self.conc[attr]

    def set_base_values_for_spec(self):
        for k, v  in self.spec.items():
            if k in self.conc:
                continue
            self.conc[k] = self.build_in_types[v]

    def get_conc(self):
        return self.conc

    def get_intr(self):
        return self.intr

    def set_algo(self, data):
        self.algo_intr = {}
        for k, v in data.items():
            self.algo_intr[k] = v

    def create_plot(self, data_dict):
        for func in self.algo_intr:
            exec(self.algo_intr[func])
        for i in self.final_string:
```

```
        if i in self.intr:  # Если символ есть в словаре замен
            for func in self.intr[i]:
                params = [data_dict[par] for par in func['input'].split(',')]
                params = ",".join([str(par) if not isinstance(par, str) else "'{0}'".format(par) for par in params])
                if self.spec[func['output']] == "list":
                    function_call = f"data_dict['{func['output']}'].append({func['function']}({params}))"
                else:
                    function_call = f"data_dict['{func['output']}'] = {func['function']}({params})"
                exec(function_call)
        return data_dict

    def create_realization(self):
        data_dict = copy.deepcopy(self.conc)
        self.final_string = self.create_gram(data_dict)
        output = self.create_plot(data_dict)
        return output

    def get_spec(self):
        return self.spec

    def get_real(self):
        return self.real_state

class SimpleConstructor(Constructor):
    _type = "Simple"
    conc_state = "Not ready"
    intr_state = "Not ready"
    spec_state = "Not ready"
    real_state = "Not ready"



class MultiConstructor(Constructor):
    _type = "Multi"
    conc_state = "Not required"
    intr_state = "Not required"
    spec_state = "Not required"
    real_state = "Not ready"



class AlgoConstructor(Constructor):
    _type = "Algorithms"
    conc_state = "Not required"
    intr_state = "Not ready"
    spec_state = "Ready"
    real_state = "Not required"

    def set_intr(self, data):
        self.intr = {}
        for k, v in data.items():
            self.intr[k] = v

    def get_intr(self):
        return self.intr



class ConstructorsContainer(BaseModel):
    constructors: dict = {}

    def _get_constructor_by_type(self, const_type):
        constr_map = {
            "Simple": SimpleConstructor,
            "Multi": MultiConstructor,
            "Algorithms": AlgoConstructor,
        }
        return constr_map.get(const_type, SimpleConstructor)

    def add_constructor(self, const_name, const_type):
        self.constructors[const_name] = self._get_constructor_by_type(const_type)(name=const_name)
```

```
    def del_constructor(self, const_name):
        del self.constructors[const_name]

    def get_constr(self, key):
        return self.constructors.get(key)

    def set_specialization(self, const_name, spec_data):
        self.constructors[const_name].set_spec()

    def update_constructor(self, name, key, value):
        self.constructors[name](key=value)

    def get_algorithm_constr_names(self):
        constrs = []
        for cons in self.constructors:
            if isinstance(self.constructors[cons],
AlgoConstructor):
                constrs.append(cons)
        return constrs

    def print_constructors(self):
        print("===========")
        for i, constr in self.constructors.items():
            print(i, constr.get_info())
        print("===========")
```

УКРАЇНА

РІШЕННЯ

про реєстрацію договору, який стосується права автора на твір

Міністерство економічного розвитку і торгівлі України розглянуло заяву

Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна, вул. Ак. Лазаряна, 2, м. Дніпро, 49010

(повне ім'я фізичної або повне офіційне найменування юридичної особи, адреса)

про реєстрацію авторського договору від **11.05.2019**, **№07/19** про передачу (відчуження) майнових прав і прийняло рішення зареєструвати авторський договір, відповідно до якого майнові права на твір

Комп'ютерна програма "Моделювання взаємозв'язаних часових рядів й геометричних фракталів породжуваних в L-системах"; Шинкаренко Віктор Іванович, Чигір Роберт Романович, Жадан Артем Анатолійович

(вид, повна, скорочена (за наявності) назва твору, повне ім'я, псевдонім (за наявності) автора(ів))

передаються(відчужуються)

Шинкаренко Віктор Іванович, вул. Тихвінська, 37, м. Дніпро, 49010; Чигір Роберт Романович, пр-т Героїв, 19, кв. 48, м. Дніпро, 49100 ; Жадан Артем Анатолійович, вул. Софії Ковальської, 86, кв. 62, м. Дніпро, 49108

(повне ім'я фізичної(их) або повне офіційне найменування юридичної(их) особи(осіб), яка(і) передає(ють)(відчужує(ють)) право на твір, адреса)

Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна, вул. Ак. Лазаряна, 2, м. Дніпро, 49010

(повне ім'я фізичної або повне офіційне найменування юридичної особи, якій передається (відчужується) право на твір, адреса)

Повністю

Реєстраційний номер **4369**

Дата реєстрації **31.05.2019**

Державний секретар
Міністерства економічного розвитку
і торгівлі України О. Ю. Перевезенцев

М.П.

УКРАЇНА • UKRAINE • УКРАЇНА • UKRAINE • УКРАЇНА • UKRAINE • УКРАЇНА • UKRAINE • УКРАЇНА • UKRAINE • УКРАЇНА • UKRAINE

УКРАЇНА

СВІДОЦТВО

про реєстрацію авторського права на твір

№ 72579

Комп'ютерна програма "Моделювання часових рядів із заданими фрактальними властивостями"

(вид, назва службового твору)

Автор(и) **Шинкаренко Віктор Іванович, Чигір Роберт Романович, Жадан Артем Анатолійович**

(повне ім'я, псевдонім (за наявності))

Авторські майнові права належать **Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна, вул. Ак. Лазаряна, 2, м. Дніпропетровськ, 49010**

(повне ім'я фізичної та/або повне офіційне найменування юридичної особи, адреса)

Дата реєстрації 27.06.2017

МІНІСТЕРСТВО ЕКОНОМІЧНОГО РОЗВИТКУ І ТОРГІВЛІ УКРАЇНИ ІДЕНТИФІКАЦІЙНИЙ КОД 37508596

М.П.

Державний секретар Міністерства економічного розвитку і торгівлі України О. Ю. Перевезенцев

ПК «Україна». Зам. 17-2005. 2017 р. III кв.

УКРАЇНА

РІШЕННЯ

про реєстрацію договору, який стосується права автора на твір

Міністерство економічного розвитку і торгівлі України розглянуло заяву

Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна, вул. Ак. Лазаряна, 2, м. Дніпро, 49010

(повне ім'я фізичної або повне офіційне найменування юридичної особи, адреса)

про реєстрацію авторського договору від **01.04.2019, №08/19** про передачу (відчуження) майнових прав і прийняло рішення зареєструвати авторський договір, відповідно до якого майнові права на твір

Комп'ютерна програма "Рекурентний аналіз часових рядів породжуваних в L-системах"; Шинкаренко Віктор Іванович, Чигір Роберт Романович, Жадан Артем Анатолійович

(вид, повна, скорочена (за наявності) назва твору, повне ім'я, псевдонім (за наявності) автора(ів))

передаються(відчужуються)

Шинкаренко Віктор Іванович, вул. Тихвінська, 37, м. Дніпро, 49010; Чигір Роберт Романович, пр-т Героїв, 19, кв. 48, м. Дніпро, 49100 ; Жадан Артем Анатолійович, вул. Софії Ковальської, 86, кв. 62, м. Дніпро, 49108

(повне ім'я фізичної(их) або повне офіційне найменування юридичної(их) особи(осіб), яка(і) передає(ють)(відчужує(ють)) право на твір, адреса)

Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна, вул. Ак. Лазаряна, 2, м. Дніпро, 49010

(повне ім'я фізичної або повне офіційне найменування юридичної особи, якій передається (відчужується) право на твір, адреса)

Повністю

Реєстраційний номер **4367**
Дата реєстрації **29.05.2019**

Державний секретар
Міністерства економічного розвитку
і торгівлі України О. Ю. Перевезенцев

М.П.

ПК «Україна». Зам. 19-2002. 2019 р. 1 кв.

Advances in Intelligent Systems and Computing 1080

Natalya Shakhovska
Mykola O. Medykovskyy *Editors*

Advances in Intelligent Systems and Computing IV

Selected Papers from the International Conference on Computer Science and Information Technologies, CSIT 2019, September 17–20, 2019, Lviv, Ukraine

Springer

Modeling of Lightning Flashes in Thunderstorm Front by Constructive Production of Fractal Time Series

Viktor Shynkarenko[1][0000-0001-8738-7225], Kostiantyn Lytvynenko[2][0000-0003-3674-4194], Robert Chyhir[3] and Iryna Nikitina[4]

[1,2,3,4] Dnipro National University of Railway Transport named after academician V. Lazaryan, Dnipro, Ukraine

shinkarenko_vi@ua.fm
kosta11111973@gmail.com
robertchigir@ukr.net
irinasansieva@gmail.com

Abstract. Using the tools of structural-synthesizing modeling, a set of constructors was developed. Implementing parametric multi-character constructors allows to form fractal sequences of characters. Constructor-converter from the character string to time series creates fractal time series, which determine the location, magnitude and decay rate of lightning discharges. Model video images of lightnings in the thunderstorm front are formed in accordance with the implementation of the constructor-assembler. All constructors are developed on the basis of the generalized constructor that was previously presented and repeatedly tested. The model adequacy of the model is confirmed by comparing the video image of the model with the image, what was obtained by NASA satellite. This approach can be the basis for solving the dynamic problems on lightning protection of engineering constructions and civil objects, and development of strategy of aircraft behavior in order to mitigate the risks of lightning strokes in the conditions of movement in the thunderstorm front.

Keywords: L-system, constrictive-synthesizing modeling, fractal, lightning activity, lightning flash, thunderstorm front, time series.

1 Introduction

Standard monitoring and forecasting of hazardous thunderstorm phenomena are carried out in all countries on the basis of a unified program and regulatory documents. Such monitoring includes:

— regular monitoring of qualitative and quantitative indicators of the atmospheric thunderstorm state;
— collection, processing and storage of observations of thunderstorm phenomena;

— creation and maintenance of observational databases.

It becomes possible to conduct modeling experiments in order to develop adequate quantitative predictive models of lightning activity of thunderstorm fronts.

The study of patterns of spatial distribution of thunderstorms is the relevant and practically important problem for solving both the essential tasks of atmospheric electricity and lightning protection of engineering constructions and thunderstorm fire risk of forest areas. One of the sources of data on the spatial distribution of thunderstorms is WWLLN (World Wide Lightning Location Network) [1].

Lightning monitoring was performed by satellites using detectors OTD (Optical Transient Detector) and LIS (Lightning Imaging Sensor). They are recording short bursts of infrared radiation, which arise from the lightning discharge and can be seen from space even in daytime under the clouds.

The main directions of modeling and studying of lightning activity are associated with the study of spreading of currents from clouds to the ground [2], impact of lightning on electrical systems [3].

2 State of Problem

Research on regional and global lightning activity and the global electrical circuit is summarized in scientific research [4]. This area of activity has greatly expanded through observations of lightning by satellite and through increased using of the natural resonances of the Earth–ionosphere cavity.

The complex relationships between lightning and rain yields in convective storms have been studied extensively, with different relationships derived in varying geographical locations and atmospheric conditions [5].

Regional behavior of lightning activity is studied for a long time [6].

The effect of solar variability parameters and meteorological parameters on total lightning flashes and convective rain in two selected regions is studied in the article [7], isolation of zones of the lightning activity in specific geographic areas [8, 9], and their impact on breaking-out of fires [10]. Much lesser number of works deals with the problem of modeling of lightning in the thunderstorm front, which is primarily due to its complexity. Typically, such works are limited to isolation of compact zones (clusters) of lightning formation [11].

In this time the modeling of lightning activity on the base of satellite monitoring of flash rate from convective cell can help severe weather forecasts, improve tornado forecasters and their impending threat to the public.

This paper refers to modeling of lightning activity in the thunderstorm front based on the generated fractal time series which determine the time, coordinates and duration of flashes, and comparison of the model video images to video images received from the satellite.

In the constructive approach to the formation of objects of different nature there is an opportunity to understand the constructive notion of the real technical or nature object.

3 The Main Material

3.1 Constructive modeling tools

Generalized constructor. The basis of constructive-synthesizing modeling is the concept of generalized constructive-synthesizing structure [12-14], or generalized constructor (GC):

$$C = \langle M, \Sigma, \Lambda \rangle, \tag{1}$$

where M – heterogeneous replenishable carrier, Σ – signature of relations and relevant operations, such as linking, substitution, and inference, over attributes, Λ – set of statements of the information support of construction (ISC) including: ontology, purpose, rules, restrictions, terms of starting and completion of construction.

Peculiarities of the constructive-synthesizing modeling are as follows [12-14]: attributiveness of elements and operations, replenishable carrier, model of performer in the form of its basic algorithms, relation of operations to the algorithms of their implementation. Main provisions ontological support of constructive-synthesizing modeling developed in [15].

Ontology of generalized constructor in its informal representation is given in [12, 13]; below we provide its part required for the subsequent presentation.

Signature Σ comprises sets of operations: Ξ – linking, Θ – substitution and inference, Φ – operations over attributes. The signature also contains the relations of substitution "$\rightarrow$".

Operations of linking of constructor elements combine the individual elements into constructions or parts thereof (intermediate forms).

Under the form ${}_{w_i}l$ with the set of attributes w_i we understand:

- ${}_{w_i}l = {}_{w_0}\otimes({}_{w_1}m_1, {}_{w_2}m_2, \text{K}, {}_{w_k}m_k)$ for $\forall {}_{w_i}m_i \in M$;
- ${}_{w_i}l = {}_{w_i}m_j$, if $l = {}_{w_0}\otimes(\varepsilon, \text{K}, \varepsilon, {}_{w_i}m_j, \varepsilon, \text{K}, \varepsilon)$;
- ${}_{w_i}l = {}_{w_0}\otimes({}_{w_1}l_1, {}_{w_2}l_2, \text{K}, {}_{w_k}l_k)$,

where ${}_{w_1}l_1, {}_{w_2}l_2, \text{K}, {}_{w_k}l_k$ – forms, ${}_{w_0}\otimes$ – any linking operation of Ξ with attribute w_o, ε – empty element.

The substitution relation is ${}_{w_i}l_i \rightarrow {}_{w_j}l_j$.

Let it be $s = \langle {}_{w_1}l_1 \rightarrow {}_{w_2}l_2, \ {}_{w_3}l_3 \rightarrow {}_{w_4}l_4, \text{K}, \ {}_{w_m}l_m \rightarrow {}_{w_{m+1}}l_{m+1} \rangle$ – sequence of substitution relations or $s = \varepsilon$, and $g = \langle \oplus_1(w_{1,1}, w_{2,1}, \text{K}, w_{k_1,1}), \ \oplus_2(w_{1,2}, w_{2,2}, \text{K}, w_{k_2,2},), \text{K}, \ \oplus_n(w_{1,n}, w_{2,n}, \text{K}, w_{k_n,n}) \rangle$ – sequence of operations over attributes. Substitution rule is $\psi : \langle s, g \rangle$. Here $\oplus$ is a any operation over attributes ($\oplus \in \Phi$).

A set of substitution rules is $\Psi = \{\psi_i : \langle s_i, g_i \rangle\}$.

Suppose the specified form ${}_{w_i}I = \otimes({}_{w_1}I_1, {}_{w_2}I_2, \text{K}, {}_{w_h}I_h, \text{K}, {}_{w_k}I_k)$ and relation of substitution ${}_{w_h}I_h \to {}_{w_q}I_q$ is such that ${}_{w_h}I_h \text{ p } {}_{w_i}I$ (relation p – contains), then the result of ${}_{w_i}I^*$ trinary operation of substitution $\Rightarrow({}_{w_h}I_h, {}_{w_q}I_q, {}_{w_i}I)$ will be the form ${}_{w_i}I^* = \otimes({}_{w_1}I_1, {}_{w_2}I_2, \text{K}, {}_{w_q}I_q, \text{K}, {}_{w_k}I_k)$ where $\Rightarrow \in \Theta$.

Binary operation of partial inference ${}_{w_i}I^* = {}_{\nu_p}|\Rightarrow(\Psi, {}_{w_i}I)$ $(|\Rightarrow \in \Theta)$ consists in:

- choice of one of available substitution rules $\psi_r : \langle s_r, g_r \rangle$ with the relations of substitution s_r;
- performance of substitution operations on its base;
- performance of operations over attributes g_r.

Operation of full inference $(||\Rightarrow \in \Theta)$ consists in stepwise conversion of forms starting with the initial form and ending with the construction satisfying the condition of inference completion which involves the cyclic performance of partial inference operations. It is a binary operation ${}_{\Delta w_i}I^* = ||\Rightarrow(\Psi, {}_{w_i}I)$ where ${}_{w_i}I \in M$.

The resulting constructions of full inference operations belong to $\Omega(C)$.

With a view to forming the constructions, a number of clarifying transformations are carried out:

— specialization determines the subject area: semantic nature of the carrier, finite set of operations and their semantics, attributes of operations, order of their performance and limitations of substitution rules;
— interpretation binds of signature operations with their execution algorithms, thus connecting the information model of means of constructions' formation and performer model, which generate the constructive system;
— concretization of the constructor expands axiomatics with a set of substitution rules, assigning of specific sets of nonterminal and terminal characters with their attributes and, where appropriate, the attribute values;
— realization, which essence is formation of a set of constructions using carrier elements.

Specialization of generalized constructor on the basis of constructive-synthesizing approach and L-systems [16] can be considered as

$$C = \langle M, \Sigma, \Lambda \rangle \; {}_S\text{a} \quad C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle, \tag{2}$$

where M_L includes the character terminals, as well as intermediate forms and multi-character constructions, Σ_L comprises a single operation, i.e. concatenation of characters and character strings (as a rule, the sign of operation between operands is omitted), Λ_L – information support includes the basics of constructive-synthesizing modeling and peculiar features of L-systems $\Lambda_L = \Lambda \cup \Lambda_1$.

Ontology of ISC Λ_1 includes the above designations and their semantics, notions "character", "concatenation", "character string" and other well-known concepts of multi-character processing, as well as provisions given below.

Partial inference operation $|\Rightarrow(\Psi, {}_{W_i} I)$ is clarified: all permitted operations of substitution of Ψ applicable to terminals of the form ${}_{w_i} I$ are performed, with looking through from left to right, except the recursion.

Initial conditions are given as a character string (axiom).

A set of non-terminal is empty.

The constructive system allows to generate the certain set of constructions (possibly, one) or to perform the check of attribution of specified construction to the above set.

In some cases, it is necessary to form two or more distinct sets of constructions similarly, where the sets of constructions being formed are different, and the processes of their formation have little variability.

In such cases it is advisable to apply parametric constructors. Suppose the family of constructions is a set of constructions characterized by the limited number of provisions of ISC. When determining the family we specify in parentheses the constructor parameters (variable of ISC elements within the family are listed).

Parametric multi-character creator constructors. Concretization of C_L to the level of the family of parametric multi-character constructors gives

$$C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle_K \text{a} \quad C_{MS}(\text{B},\text{P},\text{n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle \tag{3}$$

where B – initial character string (axiom), P – set of substitution rules, n – minimum number of terminals f in output string, $M_{MS} \supset \{f, x, y, p, m, d, u, +, -, /, \backslash\}$, $\Sigma_{MS} = \Xi_{MS} = \ = \{o\}$, o – concatenation operation, $\Lambda_{MS} = \Lambda_L \cup \Lambda_2$. Ontological component Λ_2 includes the above terms and their semantics, as well as provisions below:

- purpose of construction – formation of string of fractal structure;
- substitution rules are set by the parameter P;
- limitations – there are no operations over attributes;
- initial conditions – the axiom is specified by B;
- termination condition – number of terminals f in output string $\geq$ n.

As a result of interpretation, we form the constructive system as a set of two models: constructor and internal performer

$$\begin{aligned} &\langle C_{MS}(\text{B},\text{P},\text{n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle, C_A = \langle M_A, \Sigma_A, \Lambda_A \rangle \rangle_I \text{a} \\ &C_{A,MS}(\text{B},\text{P},\text{n}) = \langle M_{A,MS}, \Sigma_{A,MS}, \Lambda_{A,MS} \rangle, \end{aligned} \tag{4}$$

where C_A – model of the performer in the constructor form capable of executing the basic and constructed algorithms; M_A – a set of basic and constructed algorithms; $\Sigma_A = \{\cdot,:\}$ includes operations of sequential and conditional algorithms execution; ISC Λ_A is given in [14]; $M_{A,MS} = \langle M_{MS}, M_A \rangle$, $\Sigma_{A,MS} = \langle \Sigma_{MS}, \Sigma_A \rangle$, $\Lambda_{A,MS} = \Lambda_{MS} \cup \Lambda_A \cup$ $\{(A_1^0 \mid_{A_i,A_j}^{A_i \cdot A_j} \dashv \cdot), (A_2^0 \mid_{Z_1,Z_2,A_i}^{:A_i} \dashv :), (A_3^0 \mid_{I_i,I_j}^{I_i \circ I_j} \dashv \circ); (A_4 \mid_{I_h,I_e,I_j}^{I_j} \dashv \Rightarrow); (A_5 \mid_{I_i,\Psi}^{I_j} \dashv \mid \Rightarrow);$

$(A_6 \mid_{I_i,\Psi}^{I_j} \dashv \| \Rightarrow)\}$.

Algorithms M_A:

— performing an algorithm composition compilation $A_1^0 \mid_{A_i,A_j}^{A_i \cdot A_j}$ ($A \mid_X^Y$ – an algorithm over data from an input set X with result values from a set Y, A^0 – generating an algorithm), $A_i, A_j \in \Omega(C_{A,MS})$, $A_i \cdot A_j$ – sequential execution of the algorithm A_j after the algorithm A_i;

— conditional execution $A_2^0 \mid_{Z_1,Z_2,A_i}^{:A_i}$, which consists in executing the algorithm A_i under condition $Z_1 \supseteq Z_2$;

— concatenations of chain of symbols $A_3^0 \mid_{I_i,I_j}^{I_i \circ I_j}$, $I_i, I_j \in M_{MS}$;

— executing the substitution operation $\{A_4 \mid_{I_h,I_e,I_i}^{I_j}$, , I_i, I_j, $I_h, I_e \in M_{MS}$, I_i, I_j – the current form in which the substitution operation is performed before and after it is executed, I_h, I_e – the chains in the left and right part of the substitution relation, according to which is executed;

— performing partial and complete output operations $A_5 \mid_{I_i,\Psi}^{I_j}$, $A_6 \mid_{I_i,\Psi}^{I_j}$, $\Psi \subset \Lambda_{MS}$ – a set of rules of substitution.

Constructor-converter from the character string to time series. The family of such constructors

$$C_{TS}(\Omega_i(C_{MS}), M_x, dM_x, D_x, dD_x, m) = \langle M_{TS}, \Sigma_{TS}, \Lambda_{TS} \rangle \quad (5)$$

where $\Omega_i(C_{A,MS})$ – strings obtained as a result of implementation of the constructor $C_{A,MS}$; M_x – initial value of mathematical expectation of the time series value, dM_x – its increment (%), D_x – initial value of dispersion of the time series, dD_x – its increment (%), m – number of time series points, M_{TS} includes a set of terminals $T = \{f, v, x, y, p, m,), \quad d, u, +, -, /, \backslash\}$ nonterminals $N = \{A\}$, $\Sigma_{TS} = \Xi_{TS} \cup \Phi_{TS}$, $\Xi_{TS} = \{0, f\}$, $\Phi_{TS} = \{\wedge, +, -, \times, :, /, \backslash\}$, $\Lambda_{TS} = \Lambda_L \cup \Lambda_3$.

Let's introduce the operations over attributes:

- addition, subtraction, multiplication and division, accordingly, $+(c,a,b)$, $-(c,a,b)$, $\times(c,a,b)$ and $:(c,a,b)$ with operands a, b and result c;
- generation of the random normally distributed number $\wedge(c,a,b)$ with the mathematical expectation a and dispersion b.

ISC Λ_3 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series and the real numbers;
- "o" – relation between adjacent array elements;
- purpose of construction is formation of the time series $v(t)$;
- rules of substitution:
  $\{\langle\langle A \to fA,\ B \to z_i\, o B\rangle,\langle\wedge(v,\mathrm{t}M_x,tD_x),+(t,t,dt),+(i,i,1)\rangle\rangle,$
  $\langle\langle A \to +A\rangle,\langle\times(\mathrm{qM},M_x,dM_x),:(\mathrm{qM},\mathrm{qM},100),+(\mathrm{t}M_x,\mathrm{t}M_x,qM)\rangle\rangle,$
  $\langle\langle A \to -A\rangle,\langle\times(\mathrm{qM},M_x,dM_x),:(\mathrm{qM},\mathrm{qM},100),-(\mathrm{t}M_x,\mathrm{t}M_x,qM)\rangle\rangle,$
  $\langle\langle A \to /A\rangle,\langle\times(\mathrm{q}D,D_x,dD_x),:(\mathrm{q}D,\mathrm{q}D,100),+(\mathrm{t}D_x,\mathrm{t}D_x,qD)\rangle\rangle,$
  $\langle\langle A \to \backslash A\rangle,\langle\times(\mathrm{q}D,D_x,dD_x),:(\mathrm{q}D,\mathrm{q}D,100),-(\mathrm{t}D_x,\mathrm{t}D_x,qD)\rangle\rangle,$
  $\langle\langle A \to xA\rangle,\langle\varepsilon\rangle\rangle,\langle\langle A \to yA\rangle,\langle\varepsilon\rangle\rangle,$
  $\langle\langle A \to pA\rangle,\langle\varepsilon\rangle\rangle,\langle\langle A \to mA\rangle,\langle\varepsilon\rangle\rangle,$
  $\langle\langle A \to dA\rangle,\langle\varepsilon\rangle\rangle,\langle\langle A \to uA\rangle,\langle\varepsilon\rangle\rangle,$
  $\langle\langle A \to \varepsilon\rangle,\langle\varepsilon\rangle\rangle\}$;
- operations over attributes:
  — addition, subtraction, multiplication and division, accordingly, $+(c,a,b)$, $-(c,a,b)$, $\times(c,a,b)$ and $:(c,a,b)$ with operands a, b and result c;
  — generation of the random normally distributed number $\wedge(c,a,b)$ with the mathematical expectation a and dispersion b.
- limitation – the rule $\langle A \to \varepsilon,\langle\varepsilon\rangle\rangle$ is observed if the other ones are not applicable;
- initial conditions – initial terminals A (for multi-character construction), B (for creating time series construction), multi-character construction $\Omega_i(C_{MS})$; initial time $t = 0$, its step $dt = 0.04$ seconds, current value $\mathrm{t}M_x = M_x$, $\mathrm{t}D_x = D_x$, $i = 1$;
- termination condition – observance of the empty rule.

Let's define the constructive system by interpreting C_{TS}:

$$\langle C_{TS}(\Omega_i(C_{MS}),M_x,\mathrm{d}M_x,\mathrm{D}_x,\mathrm{m}) = \langle M_{TS},\Sigma_{TS},\Lambda_{TS}\rangle, C_B = \langle M_B,\Sigma_B,\Lambda_B\rangle\rangle\ {}_I\mathrm{a}$$

$$C_{B,TS}(\Omega_i(\mathrm{C}_{A,MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \mathrm{d}D_x, \mathrm{m}) = \langle M_{B,TS}, \Sigma_{B,TS}, \Lambda_{B,TS} \rangle$$

where $M_B \supset M_A$ and supplemented by algorithms that perform operations on attributes, $\Sigma_B = \Sigma_A, \Lambda_B = \Lambda_A$.

Constructor-assembler. Allows to create a constructive process in the form of a video of the formation of the flashes of lightning based on several time series. Constructor

$$C_{VL}(\mathrm{m}, n, Z_1, Z_2, \mathrm{K}, Z_{3\text{-}n}) = \langle M_{VL}, \Sigma_{VL}, \Lambda_{VL} \rangle \quad (6)$$

where m – number of time series points, M_{VL} includes a set of terminals $T = \{z_{ij}\}$), $(Z_i = [z_{i1}, z_{i2}, \mathrm{K}, z_{im}])$ and computer windows with a picture on it, nonterminals $N = \{A, B\}$, $\Sigma_{VL} = \Xi_{VL} \cup \Phi_{VL}$, $\Xi_{VL} = \{\mathrm{o}, \bullet\}$, $\Phi_{VL} = \{+, >, \vee\}$, $\Lambda_{VL} = \Lambda_L \cup \Lambda_4$.

ISC Λ_4 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series, the real numbers and computer windows;
- "o" – relation between adjacent array elements;
- purpose of construction is video of lightning flashes on computer window;
- rules of substitution:

  $\{\langle\langle A \to z_{i,j}\,\mathrm{o}\,z_{i+1,j}\,\mathrm{o}\,z_{i+2,j}\,\mathrm{o}\,A,\ B \to \bullet(z_{i,j}\,\mathrm{o}\,z_{i+1,j}\,\mathrm{o}\,z_{i+2,j})B\rangle,$

  $\langle +(i,i,3), >(q,i,n), \vee(q,+(i,0,1)), \vee(q,+(j,j,1))\rangle\rangle,$

  $\langle\langle A \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle\};$
- operations over attributes:
  — addition $+(c,a,b)$, with operands a, b and result c;
  — comparison $>(c,a,b)$, if $a > b$ result $c = true$, else – $c = false$;
  — $\vee(c,a)$ – run operations a if $c = true$;
- operation $\bullet(a,b,c)$ – visualization flash of lightning on window in such position: distance along given curve a, distance from given curve b, with imitation flash power c;
- limitation – the rule $\langle A \to \varepsilon, \langle\varepsilon\rangle\rangle$ is observed if the other ones are not applicable;
- initial conditions – initial terminals A (for time series construction), B (for creating video), initial time $t = 0$, its step $dt = 0.04$ seconds, $i = 1$, $j = 1$; given a curve (curves) of thunderstorm front;
- termination condition – observance of the empty rule when $i = 1$, $j > m$.

Let's define the constructive system by interpreting C_{VL}:

$$\left\langle C_{VL}(\mathrm{m}, n, Z_1, Z_2, \mathrm{K}, Z_{3^*n}) = \left\langle M_{VL}, \Sigma_{VL}, \Lambda_{VL} \right\rangle, C_D = \left\langle M_D, \Sigma_D, \Lambda_D \right\rangle \right\rangle_I \mathrm{a}$$

$$C_{D,VL}(\mathrm{m}, n, Z_1, Z_2, \mathrm{K}, Z_{3^*n}) = \left\langle M_{D,VL}, \Sigma_{D,VL}, \Lambda_{D,VL} \right\rangle,$$

where $M_D \supset M_A$ and supplemented by algorithms that perform operations on attributes and operation $\bullet(a,b,c)$, $\Sigma_D = \Sigma_A, \Lambda_D = \Lambda_A$.

3.2 Modeling of lightning activity in thunderstorm front

NASA and National Oceanic and Atmospheric Administration (NOAA) regularly release images from the Geostationary Lightning Mapper (GLM) instrument onboard the GOES-16 satellite. This is the new step in weather-watching capability, because it might see lightning activity above the clouds [17].

Based on the above constructors, the program modeling the lightning activity was developed.

Testing of the developed methods for modeling lightning flashes is performed by comparing with real data obtained from the satellite. Fig.1 represents: frames of satellite video image [17] (first column), the same with the removal of background (relief and cloudiness, second column), with the flashes brought to the regular form (third column).

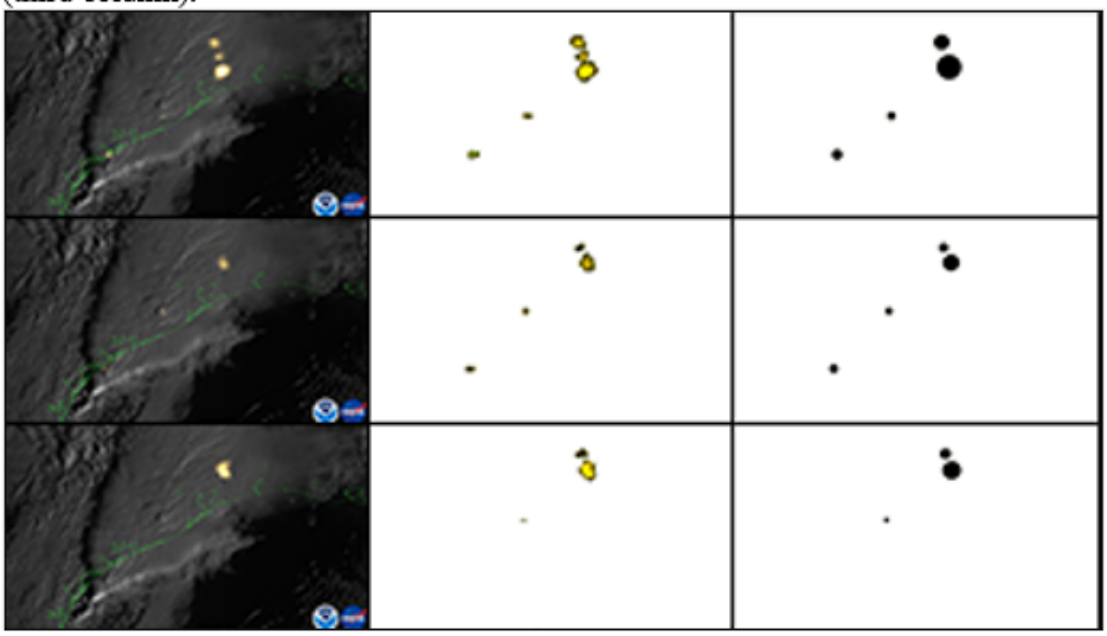


Fig. 1. Separate frames of the original video and flashes of lightning highlighted on them

For this purpose, the original video was converted as follows:

— video of the thunderstorm was divided into frames in the format RRGGBB;
— the blue channel of the images was deleted;

— the image background was removed (only flashes of lightning remain). Each pixel of the *i*-th frame is compared with pixel at the same position of (*i*-1)-th frame. If similarity condition $\frac{\left|C_{green,i} - C_{green_{,i-1}}\right|}{C_{\max}} + \frac{\left|C_{red,i} - C_{red,i-1}\right|}{C_{\max}} \le 0{,}1$ is doesn't true, pixel's color changes into white (here $C_{green,i}$, $C_{red,i}$ – values of green and red pixel's channels on appropriate frames, and $C_{\max}$ – the maximum possible value for each color channel);
— recursive pixel bypassing for extract highlight lightning flashes;
— smoothing out flashes, noise removal and contour and color conversion of flashes are performed;
— as a result, there is two videos of only flashes of lightning produced: a natural form and a model one having form of a circle.

A similar result was obtained as a result of modeling. The model of thunderstorm front is specified by Bezier curve. The pair of constructors $C_{A,MS}(f,\{f \to f+f\},200)$ $_R$a $\Omega_1(C_{A,MS})$ and $C_{A,TS}(\Omega_1(C_{A,MS}),13,50,20,10,200)$ $_R$a $\Omega_1(C_{A,TS})$ forms the time series of the lightning discharge position along the curve $u_S(t) = \Omega_1(C_{A,TS})$ (Fig. 2). The second pair $C_{A,MS}(yxyxy, \{x \to +xf, y \to -yf\},200)$ $_R$a $\Omega_2(C_{A,MS})$ and $C_{A,TS}(\Omega_2(C_{A,MS}),9,5,20,10,200)$ $_R$a $\Omega_2(C_{A,TS})$ – distance from the curve $u_L(t) = \Omega_2(C_{A,TS})$. The third pair $C_{A,MS}(--\backslash\backslash\backslash\backslash\backslash\backslash ppmyuumxmxxmpmmxmpxmff, \{y \to -yf, x \to +xf, f \to ff, p \to +p, m \to -m, d \to \backslash\backslash d, u \to /u\},200)$ $_R$a $\Omega_3(C_{A,MS})$ and $C_{A,TS}(\Omega_3(C_{A,MS}),6, 30,30,15,200)$ $_R$a $\Omega_3(C_{A,TS})$ – value of the discharge $u_R(t) = \Omega_3(C_{A,TS})$. Constructed three time series (Fig. 3).

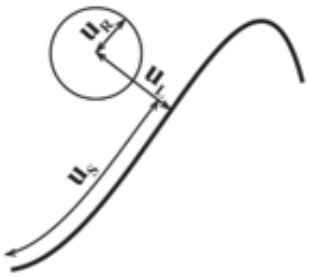
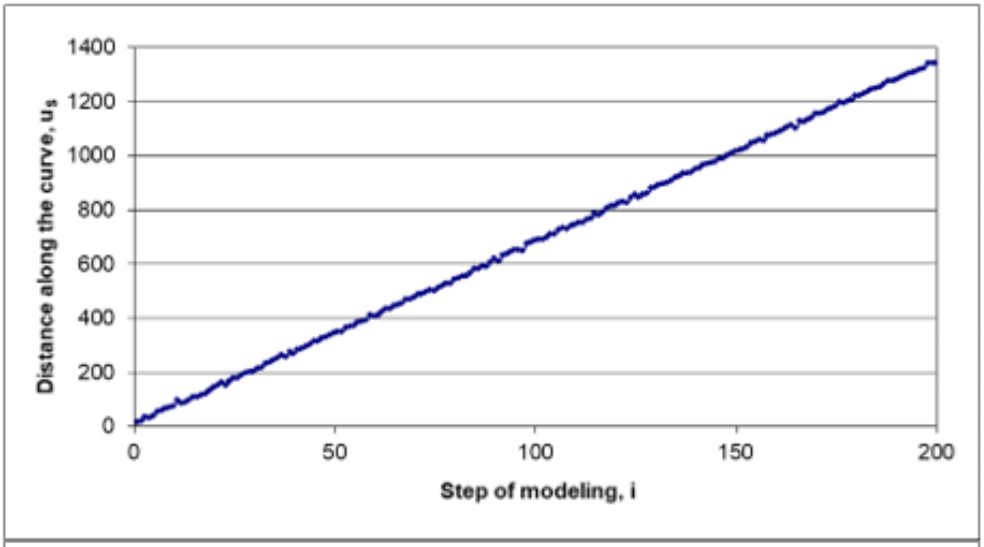


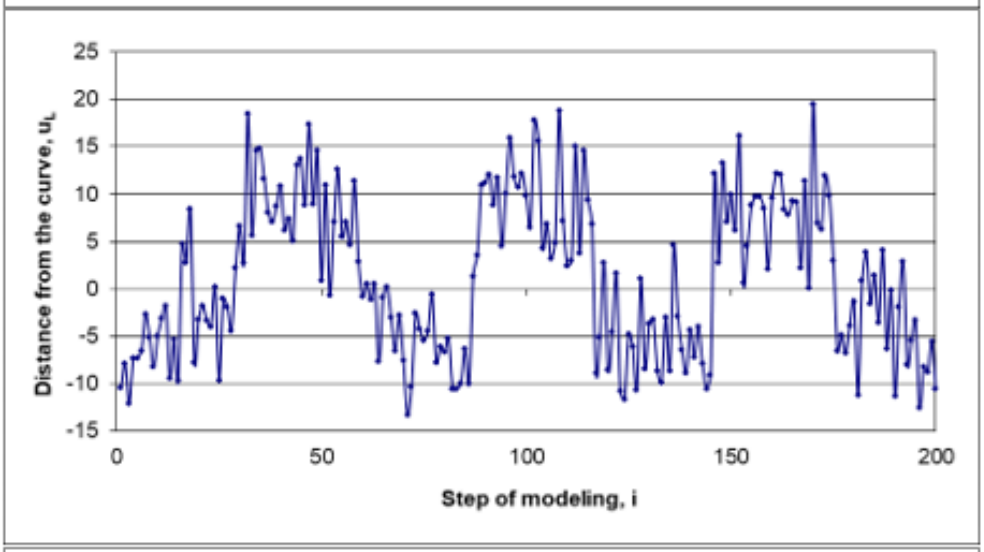
Fig. 2. Determination of the position and strength (radius of the circle) of the lightning flash

According to the constructed three time series constructor-assembler C_{VL} produced model video $C_{D,VL}(200,1,u_S(t),u_L(t),u_R(t))$. For comparison, Fig. 4 shows the lightning flashes of all frames of satellite (Fig. 4, a) and model (Fig. 4, b) lightning flashes.

Analysis of other satellite videos [18, 19] shows that the form of a thunderstorm cannot always be given by a single Bezier curve. Therefore, we in the constructor-

assembler C_{VL} provide the possibility of assigning several corresponding series of time series for them.





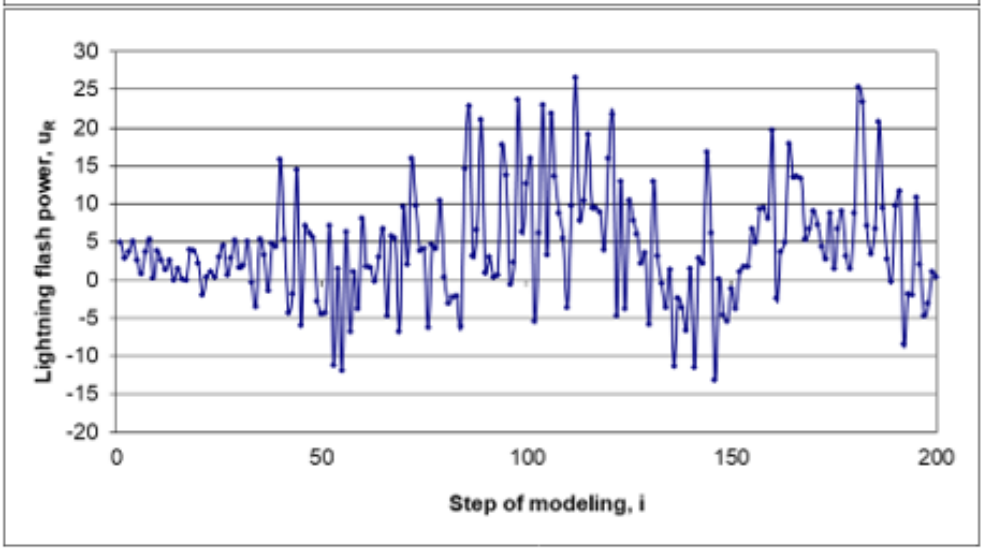


Fig. 3. Time series for distance along and from curve and lightning flash power

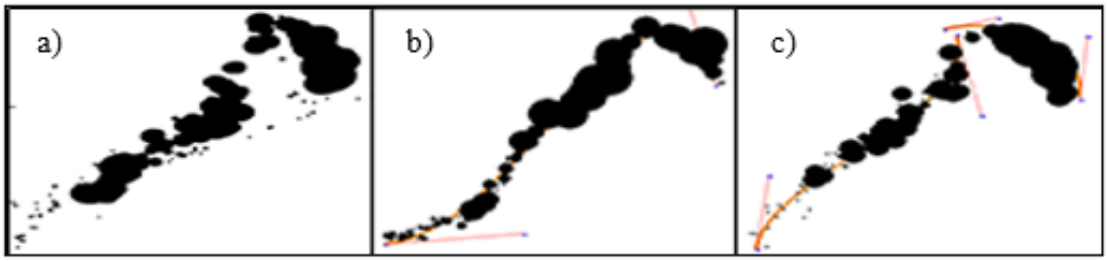


Fig. 4. Frames with all discharges of satellite and model lightning flashes

The model presented above was clarified as follows. We have two Bezier curves: long and short as an Fig. 5. Along the long curve we model two series of flashes of lightning (big and small):

— big flashes:

- $C_{A,MS}(f,\{f \to f+f-\},50)\ {}_R\mathrm{a}\ \ \Omega_4(C_{A,MS})$;
- $C_{B,TS}(\Omega_4(C_{A,MS}),0,12,150,10,50)\ {}_R\mathrm{a}\ \ \Omega_4(C_{B,TS}) = u_{S,1}(t)$;
- $C_{A,MS}(f,\{f \to f+f-f\},50)\ {}_R\mathrm{a}\ \ \Omega_5(C_{A,MS})$;
- $C_{B,TS}(\Omega_5(C_{A,MS}),8,12,150,10,50)\ {}_R\mathrm{a}\ \ \Omega_5(C_{B,TS}) == u_{L,1}(t)$;
- $C_{A,MS}(zzxzzzy,\{z \to zf, y \to yf-----f, x \to xf+++++f\},50)$ ${}_R\mathrm{a}\ \ \Omega_6(C_{A,MS})$;
- $C_{B,TS}(\Omega_6(C_{A,MS}),0,12,150,10,50)\ {}_R\mathrm{a}\ \ \Omega_6(C_{B,TS}) = u_{R,1}(t)$;

— small flashes:

- $C_{A,MS}(f,\{f \to f+f-\},50)\ {}_R\mathrm{a}\ \ \Omega_7(C_{A,MS})$;
- $C_{B,TS}(\Omega_7(C_{A,MS}),0,12,150,10,50)\ {}_R\mathrm{a}\ \ \Omega_7(C_{B,TS}) = u_{S,2}(t)$;
- $C_{A,MS}(f,\{f \to f+f-f\},50)\ {}_R\mathrm{a}\ \ \Omega_8(C_{A,MS})$;
- $C_{B,TS}(\Omega_8(C_{A,MS}),5,12,8,10,50)\ {}_R\mathrm{a}\ \ \Omega_8(C_{B,TS}) = u_{L,2}(t)$;
- $C_{A,MS}(f,\{f \to f+f-\},50)\ {}_R\mathrm{a}\ \ \Omega_9(C_{A,MS})$;
- $C_{B,TS}(\Omega_9(C_{A,MS}),1,12,0.1,10,50)\ {}_R\mathrm{a}\ \ \Omega_9(C_{B,TS}) = u_{R,2}(t)$.

Along the shot curve we model one series of flashes of lightning:

— $C_{A,MS}(f,\{f \to f+f-\},50)\ {}_R\mathrm{a}\ \ \Omega_{10}(C_{A,MS})$;
— $C_{B,TS}(\Omega_{10}(C_{A,MS}),15,12,30,10,50)\ {}_R\mathrm{a}\ \ \Omega_{10}(C_{B,TS}) = u_{S,3}(t)$;
— $C_{A,MS}(f,\{f \to f+f-\},50)\ {}_R\mathrm{a}\ \ \Omega_{11}(C_{A,MS})$;
— $C_{B,TS}(\Omega_{11}(C_{A,MS}),15,12,30,10,50)\ {}_R\mathrm{a}\ \ \Omega_{11}(C_{B,TS}) = u_{L,3}(t)$;
— $C_{A,MS}(f,\{f \to f+f-\},50)\ {}_R\mathrm{a}\ \ \Omega_{12}(C_{A,MS})$;

— $C_{B,TS}(\Omega_{12}(C_{A,MS}),15,12,30,10,50)$ $_R$a $\Omega_{12}(C_{B,TS}) = u_{R,3}(t)$.

According to the constructed time series constructor-assembler C_{VL} produced model video $C_{D,VL}(50,3,u_{S,1}(t),u_{L,1}(t),u_{R1}(t),u_{S,2}(t),u_{L,2}(t),u_{R,2}(t),u_{S,3}(t),u_{L,3}(t),$ $u_{R,3}(t))$. On Fig. 5 shows the video frames created by the constructor C_{VL}.

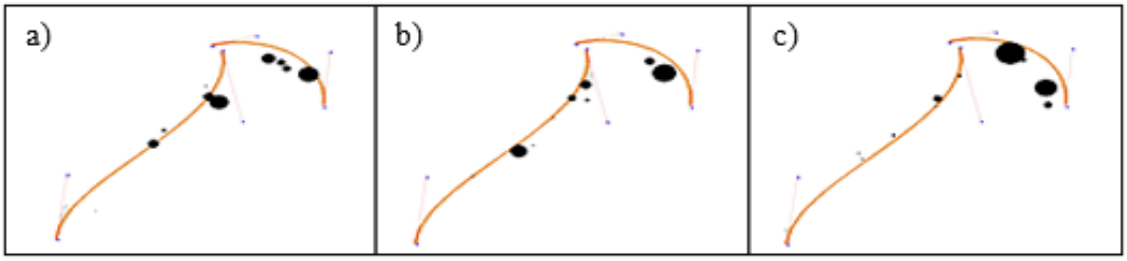


Fig. 5. Video frames of lightning flashes by improved model

These frames to some extent correspond to the frames from the satellite video (Fig. 1, column 3). Exact coincidence is not expected since the real and model processes are stochastic. However, all flashes of lightning from satellite video (Fig. 4, a) and from video of improved model (Fig. 4,c) correlate quite well.

4 Conclusions

Usage of modeling in the formation of lightning discharges based on the constructive-synthesizing approach allows obtaining the realistic description of the thunderstorm front lightning activity. This approach can be the basis for solving the dynamic problem on lightning protection of engineering constructions and civil objects, and development of strategy of aircraft behavior in order to mitigate the risks of lightning strokes in the conditions of movement in the thunderstorm front.

References

1. Rodger, C.J., Werner, S., Brundell, J.B., Lay, E.H.: Detection efficiency of the VLF World-Wide Lightning Location Network (WWLLN): initial case study. In: Ann. Geophys. (24), 3197-3214 (2006).
2. Kraaij, T., Cowling, R. M., van Wilgen, B. W.: Lightning and fire weather in eastern coastal fynbos shrublands: seasonality and long-term trends. In: International Journal of Wildland Fire, 22, 288–295 (2013). http://dx.doi.org/10.1071/WF11167
3. Pack, S., Piantini, A.: Lightning research and lightning protection technology. In: Electric Power Systems Research, 113, 1-2 (2014).
4. Williams, E.R:. Lightning and climate: A review. In: Atmospheric Research. 76 (1-4), 272-287 (2005). https://doi.org/10.1016/j.atmosres.2004.11.014
5. Yair, Y., Lynn, B., Price, C., Kotroni, V., Lagouvardos, K., Morin, E., Mugnai, A., Llasat, M. d. C.: Predicting the potential for lightning activity in Mediterranean storms based on

the Weather Research and Forecasting (WRF) model dynamic andmicrophysical fields. In: J. Geophys. Res., 115 (2010) D04205. https://doi.org/10.1029/2008JD010868.
6. Sátoria, G., Williams, E. Lempergeraams, I.: Variability of global lightning activity on the ENSO time scale. In: Atmospheric Research, 91(2–4), 500-507 (2009). https://doi.org/10.1016/j.atmosres.2008.06.014
7. Devendraa, S., RameshKumara, P., Kulkarnia, M.N., Singhb, R.P., Singhc, A.K. Lightning, convective rain and solar activity — Over the South/Southeast Asia. In: Atmospheric Research. 120–121, 99-111 (2013). https://doi.org/10.1016/j.atmosres.2012.07.026
8. Galanaki, E., Kotroni, V., Lagouvardos, K., Argiriou, A.: A ten-year analysis of cloud-to-ground lightning activity over the Eastern Mediterranean region. In: Atmospheric Research, 166, 213-222 (2015).
9. Galanakiab, E., Kotronia, V., Lagouvardosa, K., Argirioub, A.: A ten-year analysis of cloud-to-ground lightning activity over the Eastern Mediterranean region. In: Atmospheric Research, 166, 213-222 (2015).
10. Ahrens, M.: Lightning fires and lightning strikes. In: National Fire Protection Association, Quincy, 31p. (2013).
11. Fuchs, B.R, Bruning, E.C., Rutledge, S.A, Carey, L.D., Krehbiel, P.R., Rison, W.: Climatological analyses of LMA data with an open-source lightning flashclustering algorithm. In: Journal of Geophysical Research: Atmospheres, 121(14), 8625-8648 (2016).
12. Shynkarenko, V.I., Ilman, V.M: Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure. In: Cybernetics and Systems Analysis, 50(5), 665 – 662 (2014). https://doi.org/10.1007/ s10559-014-9655-z
13. Shynkarenko, V.I., Ilman, V.M.: Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part II. Refining Transformations. In: Cybernetics and Systems Analysis, 50(6), 829 – 841 (2014). https://doi.org/10.1007/s10559-014-9674-9
14. Shynkarenko, V.I, Ilman, V.M, Skalozub, V.V.: Structural models of algorithms in problems of applied programming. I. Formal algorithmic structures. In: Cybernetics and Systems Analysis, 45(3), 329-339 (2009). https://doi.org/10.1007/s10559-009-9118-0
15. Skalozub, V., Illman V., Shynkarenko V.: Development of ontological support of constructive-synthesizing modeling of information systems. In: Eastern-European Journal of Enterprise Technologies, 6/4(90), 58-69 (2017). https://doi.org/10.15587/1729-4061.2017 .119497
16. Lindenmayer, A.: Mathematical models for cellular interaction in development. In: Journal of Theoretical Biology, 18, 280 – 315 (1968).
17. First Images from GOES-16 Lightning Mapper. https://www.americaspace.com/2017/03/07/goes-16-satellite-returns-first-lightning-mapping-images-like-never-seen-before/. Accessed 11 July 2018.
18. Regional and Mesoscale Meteorology Branch. http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/20190116000000&number_of_images_to_display=120&loop_speed_ms=100. Accessed 11 July 2018.
19. Regional and Mesoscale Meteorology Branch. http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/20180815000000&number_of_images_to_display=100&loop_speed_ms=150. Accessed 11 July 2018.

Constructive Modeling of Lightning Activity in Thunderstorm Front

Viktor Shynkarenko
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
Dnipro, Ukraine
shinkarenko_vi@ua.fm

Kostiantyn Lytvynenko
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
Dnipro, Ukraine
kosta11111973@gmail.com

Robert Chyhir
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
robertchigir@ukr.net

Iryna Sansiieva
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
irinasansieva@gmail.com

***Abstract*—Using the tools of constructive-synthesizing modeling, constructors of fractal time series which determine the location, magnitude and rate of damping of lighting discharges are developed. Model video images of lightning in the thunderstorm front are formed according to constructors' implementation. The adequacy of the model is verified by comparison of the model video image with the same produced by NASA satellite.**

Keywords—L-system; constrictive-synthesizing modeling; fractal; lightning activity; thunderstorm front; time series

I. Introduction

The study of patterns of spatial distribution of thunderstorms is the relevant and practically important problem for solving both the essential tasks of atmospheric electricity and lightning protection of engineering constructions and thunderstorm fire risk of forest areas. One of the sources of data on the spatial distribution of thunderstorms is WWLLN (World Wide Lightning Location Network) [1].

Lightning monitoring was performed by satellites using detectors OTD (Optical Transient Detector) and LIS (Lightning Imaging Sensor). They are recording short bursts of infrared radiation, which arise from the lightning discharge and can be seen from space even in daytime under the clouds.

The main directions of modeling and studying of lightning activity are associated with the study of spreading of currents from clouds to the ground [2], impact of lightning on electrical systems [3], isolation of zones of the lightning activity in specific geographic areas [4], and their impact on breaking-out of fires [5]. Much lesser number of works deals with the problem of modeling of lightning in the thunderstorm front, which is primarily due to its complexity. Typically, such works are limited to isolation of compact zones (clusters) of lightning formation [6].

This paper refers to modeling of lightning activity in the thunderstorm front based on the generated fractal time series which determine the time, coordinates and duration of flashes, and comparison of the model video images to video images received from the satellite.

II. Constructive-Synthesizing Modeling of Fractal Time Series

The basis of constructive-synthesizing modeling is the concept of generalized constructive-synthesizing structure [7-9], or generalized constructor (GC):

$$C = \langle M, \Sigma, \Lambda \rangle, \tag{1}$$

where M – heterogeneous replenishable carrier, Σ – signature of relations and relevant operations, such as linking, substitution, and inference, over attributes, Λ – set of statements of the information support of construction (ISC) including: ontology, purpose, rules, restrictions, terms of starting and completion of construction.

Peculiarities of the constructive-synthesizing modeling are as follows [7-9]: attributiveness of elements and operations, replenishable carrier, model of performer in the form of its basic algorithms, relation of operations to the algorithms of their implementation.

Ontology of generalized constructor in its informal representation is given in [7, 8]; below we provide its part required for the subsequent presentation.

Signature Σ comprises sets of operations: Ξ – linking, Θ – substitution and inference, Φ – operations over attributes. The signature also contains the relations of substitution "$\rightarrow$".

Operations of linking of constructor elements combine the individual elements into constructions or parts thereof (intermediate forms).

Under the form ${}_{w_i}l$ with the set of attributes W_i we understand:

- ${}_{w_i}l = {}_{w_i}\otimes({}_{w_i}m_1, {}_{w_i}m_2, \text{K}, {}_{w_i}m_k)$ for $\forall {}_{w_i}m_i \in M$;
- ${}_{w_i}l = {}_{w_i}m_j$, if $l = {}_{w_0}\otimes(\varepsilon, \text{K}, \varepsilon, {}_{w_j}m_j, \varepsilon, \text{K}, \varepsilon)$;
- ${}_{w_i}l = {}_{w_i}\otimes({}_{w_1}l_1, {}_{w_2}l_2, \text{K}, {}_{w_k}l_k)$,

where ${}_{w_1}l_1, {}_{w_2}l_2, \text{K}, {}_{w_k}l_k$ – forms, ${}_{w_i}\otimes$ – any linking operation of Ξ with attribute w_o, ε – empty element.

The substitution relation is ${}_{w_i}l_i \rightarrow {}_{w_j}l_j$.

Let it be $s = \langle {}_{w_1}l_1 \rightarrow {}_{w_2}l_2, {}_{w_3}l_3 \rightarrow {}_{w_4}l_4, \text{K}, {}_{w_m}l_m \rightarrow {}_{w_{m+1}}l_{m+1} \rangle$ – sequence of substitution relations or $s = \varepsilon$, and $g = \langle \oplus_1(w_{1,1}, w_{2,1}, \text{K}, w_{k_1,1}), \oplus_2(w_{1,2}, w_{2,2}, \text{K}, w_{k_2,2}), \text{K}, \oplus_n(w_{1,n}, w_{2,n}, \text{K}, w_{k_n,n}) \rangle$ – sequence of operations over attributes. Substitution rule is $\psi : \langle s, g \rangle$. Here $\oplus$ is a any operation over attributes ($\oplus \in \Phi$).

A set of substitution rules is $\Psi = \{\psi_i : \langle s_i, g_i \rangle\}$.

Suppose the specified form ${}_{w_i}l = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \text{K}, {}_{w_h}l_h, \text{K}, {}_{w_k}l_k)$ and relation of substitution ${}_{w_h}l_h \rightarrow {}_{w_q}l_q$ is such that ${}_{w_h}l_h \text{ p } {}_{w_i}l$ (relation p – contains), then the result of ${}_{w_i}l^*$ trinary operation of substitution $\Rightarrow({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_i}l)$ will be the form ${}_{w_i}l^* = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \text{K}, {}_{w_q}l_q, \text{K}, {}_{w_k}l_k)$ where $\Rightarrow \in \Theta$.

Binary operation of partial inference ${}_{w_i}l^* = {}_{\psi_\rho}|\Rightarrow(\Psi, {}_{w_i}l)$ ($|\Rightarrow \in \Theta$) consists in:

- choice of one of available substitution rules $\psi_r : \langle s_r, g_r \rangle$ with the relations of substitution s_r;
- performance of substitution operations on its base;
- performance of operations over attributes g_r.

Operation of full inference ($||\Rightarrow \in \Theta$) consists in stepwise conversion of forms starting with the initial form and ending with the construction satisfying the condition of inference completion which involves the cyclic performance of partial inference operations. It is a binary operation ${}_{\Delta w_i}l^* = ||\Rightarrow(\Psi, {}_{w_i}l)$ where ${}_{w_i}l \in M$.

The resulting constructions of full inference operations belong to $\Omega(C)$.

With a view to forming the constructions, a number of clarifying transformations are carried out:

- specialization determines the subject area: semantic nature of the carrier, finite set of operations and their semantics, attributes of operations, order of their performance and limitations of substitution rules;
- interpretation binds of signature operations with their execution algorithms, thus connecting the information model of means of constructions' formation and performer model, which generate the constructive system;
- concratization of the constructor expands axiomatics with a set of substitution rules, assigning of specific sets of nonterminal and terminal characters with their attributes and, where appropriate, the attribute values;
- implementation, which essence is formation of a set of constructions using carrier elements.

Specialization of generalized constructor on the basis of constructive-synthesizing approach and L-systems [10] can be considered as

$$C = \langle M, \Sigma, \Lambda \rangle_{\text{S}}\text{a} \quad C_{\text{L}} = \langle M_{\text{L}}, \Sigma_{\text{L}}, \Lambda_{\text{L}} \rangle, \qquad (2)$$

where M_{L} includes the character terminals, as well as intermediate forms and multi-character constructions, Σ_{L} comprises a single operation, i.e. concatenation of characters and character strings (as a rule, the sign of operation between operands is omitted), Λ_{L} – information support includes the basics of constructive-synthesizing modeling and peculiar features of L-systems $\Lambda_{\text{L}} = \Lambda \cup \Lambda_{\text{l}}$.

Ontology of ISC Λ_{l} includes the above designations and their semantics, notions "character", "concatenation", "character string" and other well-known concepts of multi-character processing, as well as provisions given below.

Partial inference operation $|\Rightarrow(\Psi, {}_{w_i}l)$ is clarified: all permitted operations of substitution of Ψ applicable to terminals of the form ${}_{w_i}l$ are performed, with looking through from left to right, except the recursion.

Initial conditions are given as a character string (axiom).

A set of non-terminal is empty.

The constructive system allows to generate the certain set of constructions (possibly, one) or to perform the check of attribution of specified construction to the above set.

In some cases, it is necessary to form two or more distinct sets of constructions similarly, where the sets of constructions being formed are different, and the processes of their formation have little variability.

In such cases it is advisable to apply parametric constructors. Suppose the family of constructions is a set of constructions characterized by the limited number of

provisions of ISC. When determining the family we specify in parentheses the constructor parameters (variable of ISC elements within the family are listed).

Concretization of C_L to the level of the family of parametric multi-character constructors gives

$$C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle_K \text{a} \quad C_{MS}(\text{B,P,n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle \quad (3)$$

where B – initial character string (axiom), P – set of substitution rules, n – minimum number of terminals f in output string, $M_{MS} \supset \{f, x, y, p, m, d, u, +, -, /, \backslash\}$, $\Sigma_{MS} = \Xi_{MS} = \{\circ\}$, $\circ$ – concatenation operation, $\Lambda_{MS} = \Lambda_L \cup \Lambda_2$. Ontological component Λ_2 includes the above terms and their semantics, as well as provisions below:

- purpose of construction – formation of string of fractal structure;
- substitution rules are set by the parameter P ;
- limitations – there are no operations over attributes;
- initial conditions – the axiom is specified by B ;
- termination condition – number of terminals f in output string $\geq$ n .

As a result of interpretation, we form the constructive system as a set of two models: constructor and internal performer

$$\begin{gathered}\langle C_{MS}(\text{B,P,n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle, C_A = \langle M_A, \Sigma_A, \Lambda_A \rangle \rangle_I \text{a} \\ C_{A,MS}(\text{B,P,n}) = \langle M_{A,MS}, \Sigma_{A,MS}, \Lambda_{A,MS} \rangle,\end{gathered} \quad (4)$$

where C_A – model of the performer in the constructor form capable of executing the basic and constructed algorithms; M_A – a set of basic and constructed algorithms; $\Sigma_A = \{,, :\}$ includes operations of sequential and conditional algorithms execution; ISC Λ_A is given in [9]; $M_{A,MS} = \langle M_{MS}, M_A \rangle$, $\Sigma_{A,MS} = \langle \Sigma_{MS}, \Sigma_A \rangle$, $\Lambda_{A,MS} = \Lambda_{MS} \cup \Lambda_A \cup \{(A_1^0 |_{A_1, A_j}^{A_i} \dashv \cdot),$ $(A_2^0 |_{z,z,A}^{A_i} \dashv :), (A_3^0 |_{i_j, J_j}^{i_j \cdot J_j} \dashv \circ); (A_4 |_{h, J_0, d}^{J_j} \dashv \Rightarrow); (A_5 |_{i,\Psi}^{J_j} \dashv \models); (A_6 |_{i,\Psi}^{J_j} \dashv \| \Rightarrow)\}$.

The family of parametric constructors-converters from the character string to time series

$$C_{TS}(\Omega_i(C_{MS}), M_x, dM_x, \text{D}_x, d\text{D}_x, \text{m}) = \langle M_{TS}, \Sigma_{TS}, \Lambda_{TS} \rangle, \quad (5)$$

where $\Omega_i(C_{A,MS})$ – strings obtained as a result of implementation of the constructor $C_{A,MS}$; M_x – initial value of mathematical expectation of the time series value, dM_x – its increment (%), D_x – initial value of dispersion of the time series, $d\text{D}_x$ – its increment (%), m – number of time series points, M_{TS} includes a set of terminals $T = \{f, v, x, y, p, m, d, u, +, -, /, \backslash\}$ nonterminals $N = \{A\}$, $\Sigma_{TS} = \Xi_{TS} \cup \Phi_{TS}$, $\Xi_{TS} = \{q, f\}$, $\Phi_{TS} = \{\wedge, +, -, \times, :, /, \backslash\}$, $\Lambda_{TS} = \Lambda_L \cup \Lambda_3$.

Let's introduce the operations over attributes:

- addition, subtraction, multiplication and division, accordingly, $+(c,a,b)$, $-(c,a,b)$, $\times(c,a,b)$ and $:(c,a,b)$ with operands a, b and result c ;
- generation of the random normally distributed number $\wedge(c,a,b)$ with the mathematical expectation a and dispersion b.

ISC Λ_3 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series and the real numbers;
- purpose of construction is formation of the time series $v(t)$;
- rules of substitution:

$$\begin{gathered}\{\langle\langle A \to fA,\ A \to vA\rangle, \langle \wedge(v, \text{t}M_x, tD_x), +(t,t,dt)\rangle\rangle, \\ \langle\langle A \to +A\rangle, \langle q_1, +(\text{t}M_x, \text{t}M_x, qM)\rangle\rangle, \\ \langle\langle A \to -A\rangle, \langle q_1, -(\text{t}M_x, \text{t}M_x, qM)\rangle\rangle, \\ \langle\langle A \to /A\rangle, \langle q_2, +(\text{t}D_x, \text{t}D_x, qD)\rangle\rangle, \\ \langle\langle A \to \backslash A\rangle, \langle q_2, -(\text{t}D_x, \text{t}D_x, qD)\rangle\rangle, \\ \langle\langle A \to xA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to yA\rangle, \langle\varepsilon\rangle\rangle, \\ \langle\langle A \to pA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to mA\rangle, \langle\varepsilon\rangle\rangle, \\ \langle\langle A \to dA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to uA\rangle, \langle\varepsilon\rangle\rangle, \\ \langle\langle A \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle\},\end{gathered}$$

where $q_1 = \langle \times(\text{qM}, M_x, dM_x), : (\text{qM}, \text{qM}, 100)\rangle$,
$q_2 = \langle \times(\text{qD}, D_x, d\text{D}_x), : (\text{qD}, \text{qD}, 100)\rangle$.

- limitation – the rule $\langle A \to \varepsilon, \langle\varepsilon\rangle\rangle$ is observed if the other ones are not applicable;
- initial conditions – string $\Omega_i(C_{MS})$; initial time $t = 0$, its step $dt = 0.04$ seconds, current value $\text{t}M_x = M_x$, $\text{t}D_x = D_x$;
- termination condition – observance of the empty rule.

Let's define the constructive system by interpreting C_{TS} :

$$\begin{gathered}\langle C_{TS}(\Omega_i(C_{MS}), M_x, \text{d}M_x, \text{D}_x, \text{m}) = \langle M_{TS}, \Sigma_{TS}, \Lambda_{TS} \rangle, \\ C_A = \langle M_A, \Sigma_A, \Lambda_A \rangle \rangle_I \text{a} \\ C_{A,TS}(\Omega_i(C_{A,MS}), M_x, \text{d}M_x, \text{D}_x, \text{d}D_x, \text{m}) = \langle M_{A,TS}, \Sigma_{A,TS}, \Lambda_{A,TS} \rangle.\end{gathered}$$

III. Modeling of Lightning Activity in Thunderstorm Front

Based on the above constructors, the program modeling the lightning activity was developed.

Simulation results in comparison with the real thunderstorm, the video of which was obtained from the satellite are presented below. Fig.1 represents: frame of satellite video image (a) [11], the same with the removal of background (relief and cloudiness, b), with the flashes brought to the regular form (c), with all discharges (d).

A similar result was obtained as a result of modeling. The model of thunderstorm front is specified by Bezier curve. The pair of constructors $C_{AMS}(f,\{f \to f+f\},200)_R a\ \ \Omega_1(C_{AMS})$ and $C_{ATS}(\Omega_1(C_{AMS}),13,50,20,10,200)_R a\ \ \Omega_1(C_{ATS})$ forms the time series of the lightning discharge position along the curve $u_S(t)=\Omega_1(C_{ATS})$ (fig.1, e). The other pair $C_{AMS}(yxyxy,\{x \to +xf, y \to -yf\},200)_R a\ \ \Omega_2(C_{AMS})$ and $C_{ATS}(\Omega_2(C_{AMS}),9,5,20,10,200)_R a\ \ \Omega_2(C_{ATS})$ – distance from the curve $u_L(t)=\Omega_2(C_{ATS})$. The third pair $C_{AMS}(--\backslash\backslash\backslash\backslash\backslash\backslash ppmyuumxmxxmpmmxmpxmff,$ $\{y \to -yf, x \to +xf, f \to ff, p \to +p, m \to -m, d \to \backslash\backslash d,$ $u \to lu\},200)_R a\ \ \Omega_3(C_{AMS})$ and $C_{ATS}(\Omega_3(C_{AMS}),6,30,30,$ $15,200)_R a\ \ \Omega_3(C_{ATS})$ – value of the discharge $u_R(t)=\Omega_3(C_{ATS})$. According to the constructed three time series, a video is created, all discharges of which are shown in the (fig.1, f).

Conclusions

Usage of modeling in the formation of lightning discharges based on the constructive-synthesizing approach allows obtaining the realistic description of the thunderstorm front lightning activity. This approach can be the basis for solving the dynamic problem on lightning protection of engineering constructions and civil objects, and development of strategy of aircraft behavior in order to mitigate the risks of lightning strokes in the conditions of movement in the thunderstorm front.

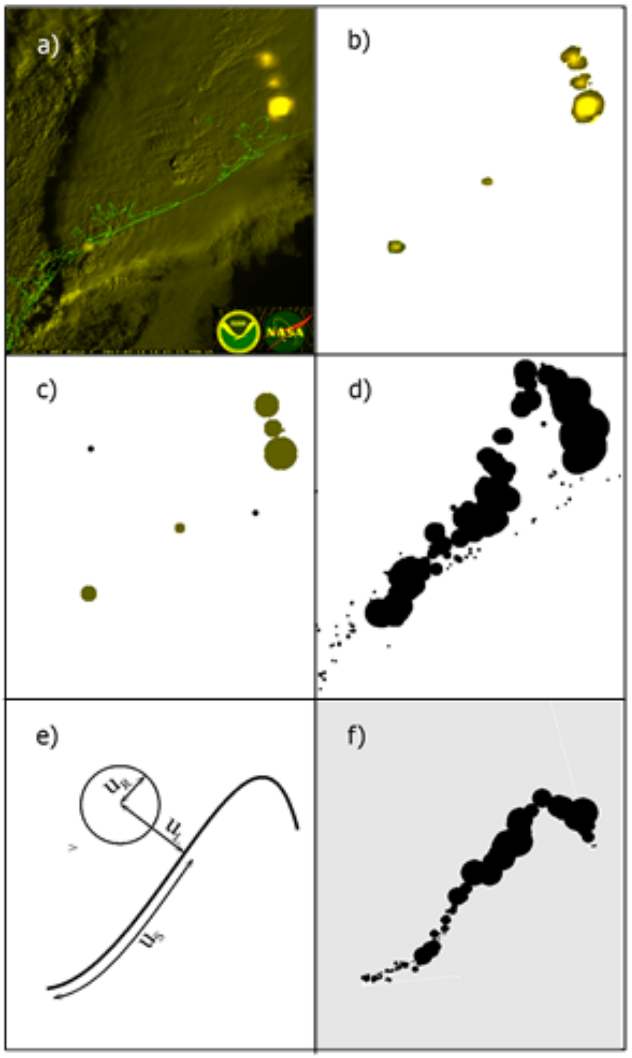


Fig. 1. Satellite and model images of thunderstorm front with lighning discharge

References

[1] C.J.Rodger, S.Werner, J.B.Brundell, E.H. Lay et al. "Detection efficiency of the VLF World-Wide Lightning Location Network (WWLLN): initial case study". Ann. Geophys, 2006, V. 24, pp. 3197-3214.

[2] T. Kraaij, Cowling R. M. and B. W. van Wilgen "Lightning and fire weather in eastern coastal fynbos shrublands: seasonality and long-term trends". International Journal of Wildland Fire. 2013, 22, pp. 288–295.

[3] S.Pack and Piantini A. "Lightning research and lightning protection technology". Electric Power Systems Research, vol. 113, 2014, pp. 1-2.

[4] E. Galanaki, V. Kotroni, K. Lagouvardos and A. Argiriou "A ten-year analysis of cloud-to-ground lightning activity over the Eastern Mediterranean region". Atmospheric Research, vol. 166, 2015, pp. 213-222.

[5] Marty Ahrens "Lightning fires and lightning strikes". National Fire Protection Association, Quincy, MA, 2013, 31p.

[6] B.R. Fuchs, E.C. Bruning, S.A. Rutledge, L.D. Carey, P.R. Krehbiel, and W. Rison "Climatological analyses of LMA data with an open - source lightning flashclustering algorithm". Journal of Geophysical Research: Atmospheres, vol. 121(14), 2016., pp. 8625-8648.

[7] V.I. Shynkarenko and V.M. Ilman "Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure". Cybernetics and Systems Analysis, vol. 50(5), 2014, pp. 665 – 662. doi: 10.1007/s10559-014-9655-z

[8] V.I. Shynkarenko and V.M. Ilman "Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part II. Refining Transformations". Cybernetics and Systems Analysis, vol. 50(6), 2014, pp. 829 – 841. doi: 10.1007/s10559-014-9674-9,

[9] V.I. Shynkarenko, V.M. Ilman and V.V. Skalozub "Structural models of algorithms in problems of applied programming. I. Formal algorithmic structures". Cybernetics and Systems Analysis, vol. 45(3), 2009, pp 329-339. doi: org/10.1007/s10559-009-9118-0

[10] A. Lindenmayer "Mathematical models for cellular interaction in development. Parts I and II.". Journal of Theoretical Biology, vol. 18, 1968, pp. 280 - 315.

[11] First Images from GOES-16 Lightning Mapper [Online]. Available: https://www.americaspace.com/2017/03/07/goes-16-satellite-returns-first-lightning-mapping-images-like-never-seen-before/

ITHEA

International Journal

INFORMATION TECHNOLOGIES & KNOWLEDGE

2019 Volume 13 Number 1

TABLE OF CONTENT

КОНСТРУКТИВНОЕ СООТВЕТСТВИЕ МУЛЬТИСИМВОЛЬНЫХ И ЛИНЕЙНЫХ ГЕОМЕТРИЧЕСКИХ ФРАКТАЛОВ

Виктор Шинкаренко, Константин Литвиненко, Роберт Чигирь

Аннотация: *Парадигма конструктивного представления окружающего мира основывается на положении, что весь мир состоит из конструкций и конструктивных процессов. Отдельные конструкции посредством конструктивных процессов преобразуются в другие. Между конструкциями различной природы может существовать явное или неявное соответствие. В данной работе конструктивно-продукционный подход к моделированию фракталов позволяет установить соответствие между мультисимвольными и линейными плоскими геометрическими фракталами, детерминированными и стохастическими. Это расширяет возможности конструирования последних с целью изучения их свойств и возможностей моделирования объектов реального мира. Средства конструктивно-продукционного моделирования получили развитие в виде семейства параметрических конструкторов, что позволяет варьировать возможности конструкторов.*

Ключевые слова: *конструктивно-продукционное моделирование, конструктор, фрактал, стохастические фракталы, L-система, специализация, интерпретация, конкретизация, реализация*

ITHEA Keywords: *F.4.2 Grammars and Other Rewriting Systems; I.1.1 Expressions and Their Representation; I.1.4 Applications; I.6.5 Model Development.*

Введение

Основные понятия фрактальной геометрии сформулированы в работе Б. Мандельброта «Фрактальная геометрия природы» [Mandelbrot, 1982] как обобщение и развитие идей А. Пуанкаре, П. Фату, Г. Кантора, Ф. Хаусдорфа. Указанная работа привела к появлению множества работ прикладного характера, в которых фрактальный подход начал применяться для решения практических задач из области хаоса и динамических систем [Федер,1991, Божокин, 2001, Peitgen, 2004], моделирования дендритов [Кроновер, 2000, Помулев, 2002, Безносюк, 2002], фрактальных свойств антенн [Слюсар, 2007], трафика видеосигналов, связи и интернета [Шелухин, 2008, Шелухин, 2011] и др.

В соответствии с определением Б. Мандельброта: фрактал – это структура, состоящая из частей, которые в каком-то смысле подобны целому. Поэтому фракталом является такой объект, который обладает свойством самоподобия, он более или менее единообразно устроен на широком диапазоне масштабов. В геометрическом случае самоподобие гарантирует инвариантность при любом изменении масштаба, однако это характерно только для регулярных, детерминированных фракталов. Если детерминированный процесс построения фрактальной структуры зашумлен случайными воздействиями, тогда формируются стохастические фракталы. Свойство самоподобия таких фракталов проявляется только при усреднении по всех статистически независимых реализациях объекта. Поэтому, часть фрактала при изменении масштаба не полностью инвариантна начальному фрагменту, однако их статистические характеристики совпадают.

Разработка эффективных алгоритмов и программных средств для моделирования двумерных фракталов и трехмерных фрактальных поверхностей продолжает оставаться в центре внимания специалистов, занимающихся решениями инженерных, медицинских, управленческих и других задач [Кравченко,2016, Kravchenko, 2017, Camps-Raga, 2010, Zhou, 1995, Chiu, 2006, Zhou, 2010].

Связанные работы

На базе фундаментального принципа общенаучных исследований – от частного к общему, а затем от общего к частному, в [Shynkarenko, 2014], выполнено обобщение возможностей и особенностей различных модификаций грамматик и грамматико-подобных систем с применением конструктивного подхода. Исходя из [Shynkarenko, 2014], конструкционно-продукционное моделирование может применяться для решения задач формирования, преобразования и анализа конструкций различной природы с применением операций связывания, подстановки, вывода и др. операций, а также правил подстановки. На основе такого подхода представляется возможным моделирование и формализация любых конструктивных процессов в области инженерии, биологии, информационных технологий, а также расширяется возможность учета свойства элементов, их формы и связи.

Отталкиваясь от общего подхода в указанных работах, удалось решить ряд частных задач:

- адаптации алгоритмов сжатия к архивируемым данным [Shynkarenko, 2015];
- совершенствования процесса ранжирования альтернатив методом анализа иерархий [Шинкаренко и Васецкая, 2016];
- адаптации структур данных в оперативной памяти [Шинкаренко и Забула, 2016];
- совершенствование структур хранения данных в задачах выявления плагиата [Шинкаренко и Куропятник, 2016].

В данной работе рассматриваются идеи конструктивно-продукционного подхода, как инструмента, позволяющего эффективно моделировать как регулярные, так и стохастические геометрические фракталы.

Цель и задача исследования

Цель исследования – применяя средства и методы конструктивно-продукционного моделирования к мультисимвольным и линейным геометрическим фракталам:

- формализовать процесс и результаты их формирования;
- разработать математический аппарат, устанавливающий соответствие между ними.

В соответствие с поставленными целями необходимо разработать:

- конструкторы мультисимвольных и линейных геометрических фракталов;
- программное обеспечение для их реализации.

Обобщенный конструктор. Основные положения

В основу конструктивно-продукционного моделирования положено понятие обобщенной конструктивно-продукционной структуры [Shynkarenko, 2014]. Как результат и продолжение указанных исследований, в работах [Shynkarenko, 2015, Шинкаренко и Васецкая, 2016, Шинкаренко и Забула, 2016, Шинкаренко и Куропятник, 2016], предложено рассматривать средство конструирования – обобщенный конструктор (ОК)

$$C = \langle M, \Sigma, \Lambda \rangle, \tag{1}$$

где M – неоднородный расширяемый носитель, Σ – сигнатура отношений и соответствующих операций: связывания, подстановки, вывода, над атрибутами, Λ – множество утверждений информационного обеспечения конструирования (ИОК), которое включает: онтологию, цель, правила, ограничения, условия начала и завершения конструирования.

В M можно выделить подмножества: T – терминалов, N – нетерминалов (вспомогательных, абстрактных элементов), со свойствами $T \cap N = \varnothing$, $\varepsilon \in T$, $\varepsilon \notin N$, где ε – пустой элемент.

Особенностями конструктивно-продукционного моделирования с применением являются [Shynkarenko, 2015, Шинкаренко и Васецкая, 2016, Шинкаренко и Забула, 2016, Шинкаренко и Куропятник, 2016]: атрибутивность элементов и операций, расширяемый носитель, модель исполнителя в виде его базовых алгоритмов, связь операций с алгоритмами их выполнения.

Онтология обобщенного конструктора в неформальном виде изложена в [Shynkarenko, 2014], ниже приведена ее часть необходимая для дальнейшего изложения.

Сигнатура Σ состоит из множества операций: Ξ – связывания, Θ – подстановки и вывода, Φ – операций над атрибутами. Сигнатура содержит также отношения подстановки «$\rightarrow$». Таким образом, формально сигнатура есть $\Sigma = \langle \Xi, \Theta, \Phi, \{\rightarrow\} \rangle$, со свойствами: $\Xi \bigcap \Theta = \varnothing$; $\Xi \bigcap \Phi = \varnothing$; $\Theta \bigcap \Phi = \varnothing$, $\varepsilon \in \Phi$. Сигнатура состоит из имен операций $\{\otimes_j\}$, обладающих набором атрибутов w_i, представляется как ${}_w\otimes \in \Sigma$.

Операции связывания элементов конструктора соединяют отдельные элементы в конструкции или их части (промежуточные формы).

В классических формальных грамматиках используется одна бинарная операция связывания (конкатенации) над элементами терминального и нетерминального алфавитов, однако для специализированных грамматик могут использоваться разнообразные операции связывания: по условию, многоместные, графических элементов и др.

Под формой ${}_{w_i}I$ с набором атрибутов w_i понимают:

- ${}_{w_i}I = {}_{w_0}\otimes({}_{w_1}m_1, {}_{w_2}m_2, \ldots, {}_{w_k}m_k)$ для $\forall {}_{w_j}m_j \in M$;
- ${}_{w_i}I = {}_{w_j}m_j$, если $I = {}_{w_0}\otimes(\varepsilon, \ldots, \varepsilon, {}_{w_j}m_j, \varepsilon, \ldots, \varepsilon)$;
- ${}_{w_i}I = {}_{w_0}\otimes({}_{w_1}I_1, {}_{w_2}I_2, \ldots, {}_{w_k}I_k)$, если ${}_{w_1}I_1, {}_{w_2}I_2, \ldots, {}_{w_k}I_k$ – формы.

Таким образом, операция связывания применяется как к элементам носителя, так и к формам, сконструированным с ее помощью на основе элементов носителя.

Отношение постановки – двуместное отношение с атрибутами ${}_{w_i}l_i \rightarrow {}_{w_j}l_j$.

Пусть $s = \left\langle {}_{w_1}l_1 \rightarrow {}_{w_2}l_2,\ {}_{w_3}l_3 \rightarrow {}_{w_4}l_4,\ \ldots,\ {}_{w_m}l_m \rightarrow {}_{w_{m+1}}l_{m+1} \right\rangle$ – последовательность отношений подстановки или $s = \varepsilon$, и $g = \left\langle \oplus_1(w_{1,1}, w_{2,1}, \ldots, w_{k_1,1}), \oplus_2(w_{1,2}, w_{2,2}, \ldots, w_{k_2,2}), \ldots, \oplus_n(w_{1,n}, w_{2,n}, \ldots, w_{k_n,n}) \right\rangle$ – последовательность операций над атрибутами. Назовем правилом продукции $p : \langle s, g \rangle$. Здесь $\oplus$ – произвольная операция над атрибутами ($\oplus \in \Phi$).

Множество правил продукций будем обозначать $\Psi = \{\psi_i : \langle s_i, g_i \rangle\}$.

Пусть задана форма ${}_{w_i}l = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_h}l_h, \ldots, {}_{w_k}l_k)$ и отношение подстановки ${}_{w_h}l_h \rightarrow {}_{w_q}l_q$ такое, что ${}_{w_h}l_h \prec {}_{w_i}l$ (отношение $\prec$ – содержит), тогда результатом ${}_{w_i}l^*$ трехместной операции подстановки $\Rightarrow({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_i}l)$ будет форма ${}_{w_i}l^* = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_q}l_q, \ldots, {}_{w_k}l_k)$, где $\Rightarrow \in \Theta$.

Двухместная операция частичного вывода ${}_{w_j}l^* = {}_{\nu_p}|\!\Rightarrow(\Psi, {}_{w_i}l)$ $(|\!\Rightarrow \in \Theta)$ заключается в:

- выборе одного из доступных правил подстановки $p_r : \langle s_r, g_r \rangle$ с отношениями подстановки s_r;
- выполнении на его основе операций подстановки;
- выполнении операций над атрибутами g_r в соответствующей последовательности.

Операция полного вывода или просто вывода $(\|\!\Rightarrow \in \Theta)$ заключается в пошаговом преобразовании форм, начиная с начального нетерминала и заканчивая конструкцией, удовлетворяющей условию окончания вывода, что подразумевает циклическое выполнение операций частичного вывода. Операция двухместная ${}_{\Delta,w_j}l^* = \|\!\Rightarrow(\Psi, {}_{w_i}l)$, где ${}_{w_i}l \in U$.

Результирующие конструкции операций полного вывода принадлежат $\Omega(C_L)$.

Для формирования конструкций выполняется ряд уточняющих преобразований:

- специализация определяет предметную область: семантическую природу носителя, конечное множество операций и их семантику, атрибутику операций, порядок их выполнения и ограничения на правила подстановки $C \ {}_{S}\mapsto {}_{S}C$;
- интерпретация заключается в связывании операций сигнатуры C_A с алгоритмами выполнения некоторой алгоритмической структуры, что связывает информационную модель средств формирования конструкций и модель исполнителя ${}_{S}C,C_{AI} \mapsto <{}_{S,I}C,C_A>$, образуя конструктивную систему;
- конкретизация конструктора заключается в расширении аксиоматики множеством правил продукций, задании конкретных множеств нетерминальных и терминальных символов с их атрибутами и, при необходимости, значений атрибутов ${}_{S,I}C_K \mapsto {}_{S,I,K}C$;
- реализация конструктора заключается в формировании множества конструкции из элементов носителя конструктора путем выполнения алгоритмов, связанных с операциями сигнатуры ${}_{S,I,K}C_R \mapsto \Omega$.

В работах [Shynkarenko, 2015, Шинкаренко и Васецкая, 2016, Шинкаренко и Забула, 2016, Шинкаренко и Куропятник, 2016], уточняющие преобразования выполняются в такой последовательности

$$C \ {}_{S}\mapsto {}_{S}C \mapsto {}_{S}C,C_A \ {}_{I}\mapsto <{}_{S,I}C,C_A> \ {}_{K}\mapsto <{}_{S,I,K}C,C_A> \ {}_{R}\mapsto \Omega. \qquad (2)$$

Однако как оказалось, такой подход не является единственно возможным и необходимым. Вариативность порядка применения уточняющих преобразований приводит к достаточно интересным и полезным результатам. Для этого, в частности, в рамках конструктивно-продукционного моделирования используем идеи L-систем.

Семейство параметрических конструкторов мультисимвольных фракталов

Как известно, продукционные L-системы (Lindenmayer system) [Lindenmayer, 1968] широко используются для моделирования различных систем и процессов, компьютерной графики, биологии, музыки и др.

Основная отличительная особенности L-систем относительно других классических грамматик состоит в отсутствии нетерминалов, атрибутивности терминалов, выполнении «параллельной» подстановки, порядка формирования множества выводимых конструкций и аксиомы в виде начальной конструкции.

Специализацию обобщенного конструктора на основе конструктивно-продукционного подхода и L-систем можно рассматривать как

$$C = \langle M, \Sigma, \Lambda \rangle \ {}_{S}\mapsto \ C_{L} = \langle M_{L}, \Sigma_{L}, \Lambda_{L} \rangle, \qquad (3)$$

где M_{L} включает символьные терминалы, а также промежуточные формы и мультисимвольные конструкции, Σ_{L} состоит из единственной операции – конкатенации символов и символьных цепочек (знак операции между операндами, как правило, опускается, Λ_{L} – информационное обеспечение включает основы конструктивно-продукционного моделирования и особенности L-систем $\Lambda_{L} = \Lambda \bigcup \Lambda_{1}$.

Онтология ИОК Λ_{1} включает приведенные выше обозначения и их семантику, понятия «символ», «конкатенация», «цепочка символов» и другие известные понятия мультисимвольной обработки, а также приведенными ниже положениями.

Уточняется операция частичного вывода $\models (\Psi, {}_{w_i} I)$: выполняются все допустимые операции подстановки из Ψ, применимые к терминалам из формы ${}_{w_i} I$, просматривая её слева направо за исключением рекурсии.

Начальные условия задаются в виде цепочки символов (аксиомы).

Множество нетеминалов пусто.

Конструктивная система позволяет конструировать некоторое множество конструкций (возможно и одну) либо выполнять проверку принадлежности заданной конструкции этому множеству.

В ряде случаев возникает необходимость схожим образом формировать два или более отличных друг от друга множества конструкций. Другими словами, множества формируемых конструкций различны, а процессы их формирования имеют незначительную вариативность.

В таких случаях целесообразно применять параметрические конструкторы. Назовем семейством конструкторов множество конструкторов, отличающихся ограниченным количеством положений информационного обеспечения. При определении семейства в круглых скобках задаются параметры конструкторов (перечисляются вариативные в рамках семейства элементы ИО). Определим семейство параметрических конструкторов следующим образом:

$$C(a_1, a_2 \ldots a_n) = \langle M, \Sigma, \Lambda \rangle \qquad (4)$$

где $a_i \in \Lambda$ – идентификаторы положений информационного обеспечения. Их значения задаются внешним исполнителем путем дополнительной конкретизации.

Конкретизируем C_L до уровня семейства параметрических мультисимвольных конструкторов

$$C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle_K \mapsto C_{MS}(\mathrm{B},\mathrm{P},\mathrm{n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle \qquad (5)$$

где B – начальная цепочка символов (аксиома), P – множество правил подстановки, n – количество операций частичного вывода, $M_{MS} = M_L \cup \{f, z, y, +, -\} \cup M_1$, M_1 включает цепочки из символов $\{f, z, y, +, -\}$, $\Sigma_{MS} = \Xi_{MS} = \{\circ\}$, $\circ$ – операция конкатенации, $\Lambda_{MS} = \Lambda_L \cup \Lambda_2$, Онтологическая

составляющая Λ_2 включает приведенные выше обозначения и их семантику, а также следующие положения:

- *цель конструирования* – формирование мультисимвольной цепочки фрактальной структуры;
- *правила подстановки* задаются параметром P ;
- *ограничения* – операции над атрибутами отсутствуют, правила подстановки содержат единственное отношение подстановки;
- *начальные условия* – аксиома задается параметром B ;
- *условие завершения* – выполнение n операций частичного вывода.

В результате **интерпретации** формируем конструктивную систему как совокупность двух моделей: конструктора и внутреннего исполнителя (последняя в виде конструктора алгоритмов, которые способен выполнить исполнитель)

$$\begin{aligned}&\left\langle C_{MS}(\mathrm{B},\mathrm{P},\mathrm{n})=\left\langle M_{MS},\Sigma_{MS},\Lambda_{MS}\right\rangle, C_A=\left\langle M_A,\Sigma_A,\Lambda_A\right\rangle\right\rangle_I\mapsto\\&C_{A,MS}(\mathrm{B},\mathrm{P},\mathrm{n})=\left\langle M_{A,MS},\Sigma_{A,MS},\Lambda_{A,MS}\right\rangle,\end{aligned}\qquad(6)$$

где C_A – модель исполнителя в виде конструктора, который способен выполнять базовые и сконструированные алгоритмы; M_A – множество базовых $\{A_1^0\,|_{A_i,A_j}^{A_i\cdot A_j},\ A_2^0\,|_{Z_1,Z_2,A_i}^{:A_i},\ A_3^0\,|_{I_i,J_j}^{I_i\to J_j}\}\subset \mathrm{M}_A$ и сконструированных $\{A_4\,|_{I_0,J_q,I_i}^{I_j},$ $A_5\,|_{I_j,\Psi}^{I_j},\ A_6\,|_{I_j,\Psi}^{I_j}\}\subset\Omega(C_A)$ алгоритмов; $\Sigma_A=\{\cdot,:\}$ включает сигнатуру операций последовательного и условного выполнения алгоритмов; ИО Λ_A приведено в [Shynkarenko, 2009], $M_{A,MS}=\left\langle M_{MS},M_A\right\rangle$, $\Sigma_{A,MS}=\left\langle\Sigma_{MS},\Sigma_A\right\rangle$, $\Lambda_{A,MS}=\Lambda_{MS}\cup\Lambda_A\cup\Lambda_3$.

Алгоритмы M_A:

- выполнения операции композиции алгоритмов $A_1^0\,|_{A_i,A_j}^{A_i\cdot A_j}$ ($A\,|_X^Y$ – алгоритм над данными из входного множества X со значениями из

множества Y, A^0 – образующий алгоритм), $A_i, A_j \in \Omega(C_{A,MS})$, $A_i \cdot A_j$ – последовательное выполнение алгоритма A_j после алгоритма A_i;

- условного выполнения $A_2^0 \left[{}^{A}_{Z_1,Z_2,A_i}\right.$, который заключается в выполнении алгоритма A_i при условии $Z_1 \supseteq Z_2$;
- конкатенации цепочек символов $A_3^0 \left|{}^{I_i \cdot I_j}_{I_i,I_j}\right.$, $I_i, I_j \in M_{MS}$,;
- выполнения операции подстановки $\{A_4 \left|{}^{I_j}_{I_h,I_g,I_i}\right.$, , $I_i, I_j, I_h, I_s \in M_{MS}$, I_i, I_j – текущая форма, в которой выполняется операция подстановки до и после ее выполнения, I_h, I_s – цепочки в левой и правой части отношения подстановки, согласно которому выполняется;
- выполнения операций частичного и полного вывода $A_5 \left|{}^{I_j}_{I_i,\Psi}\right.$, $A_6 \left|{}^{I_j}_{I_i,\Psi}\right.$, $\Psi \subset \Lambda_{MS}$ – множество правил подстановки.

ИОК Λ_A включает приведенные выше определения, обозначения и их семантику

$$\Lambda_3 = \{(A_1^0 \left|{}^{A_i \cdot A_j}_{A_i,A_j}\right. \lrcorner \cdot), (A_2^0 \left[{}^{A}_{Z_1,Z_2,A_i}\right. \lrcorner :), (A_3^0 \left|{}^{I_i \cdot I_j}_{I_i,I_j}\right. \lrcorner \circ); (A_4 \left|{}^{I_j}_{I_h,I_g,I_i}\right. \lrcorner \Rightarrow);$$
$$(A_5 \left|{}^{I_j}_{I_i,\Psi}\right. \lrcorner \models); (A_6 \left|{}^{I_j}_{I_i,\Psi}\right. \lrcorner \|\Rightarrow)\}. \qquad (7)$$

Выполним завершающую конкретизацию и реализацию двух конструкторов семейства: мультисимвольных дракона Хартера – Хейтуэя и снежинки Коха [Кроновер, 2000].

Реализация мультисимвольного дракона Хартера – Хейтуэя

$$\langle C_{MS}(fz, \{y \to -fz - y;\ z \to z + y\ f +\}, 7), C_A \rangle \ {}_R\!\mapsto \Omega_1(C_{MS}) \qquad (8)$$

заключается в параллельном выполнении подстановок:

- начальное состояние $(n = 0)$ текущей формы

$$fz;$$

– в результате первой операции частичного вывода $(n=1)$ получаем

$$fz+yf+;$$

– при $n=2$ имеем

$$fz+yf++-fz-yf+;$$

– при $n=3$ имеем

$$fz+yf++-fz-yf++-fz+yf+--fz-yf+$$ и так далее.

В результате реализации получаем мультисимвольную конструкцию $\Omega_1(C_{MS})$, которая обладает свойством самоподобия, что наглядно представлено процедурой её формирования.

Реализацию мультисимвольной снежинки Коха выполним таким же образом

$$\langle C_{MS}(f++f++f,\{f\to f-f++f-f\},4),C_A\rangle_R \mapsto \Omega_2(C_{MS}). \quad (9)$$

Семейство параметрических конструкторов-преобразователей

Семейство параметрических конструкторов-преобразователей из конструкции в виде цепочки символов в конструкцию в виде изображения геометрического фрактала

$$C_L=\langle M_L,\Sigma_L,\Lambda_L\rangle_K \mapsto C_{GF}(\Omega_i(C_{MS}),M_x,\mathrm{d}M_x,\mathrm{D}_x,\alpha,\mathrm{m})=\langle M_{GF},\Sigma_{GF},\Lambda_{GF}\rangle, \quad (10)$$

где $\Omega_i(C_{MS})$ – цепочки символов, полученные в результате реализации конструктора C_{MS}; M_x – начальная длина отрезка, $\mathrm{d}M_x$ – приращение длины отрезка (%), D_x – дисперсия длины отрезка, α – угол, m – количество формируемых геометрических фигур, M_{GF} – включает множество терминалов T (всех возможных ломаных на плоскости и символов $\{f,z,y,+,-\}$), нетерминалов $N=\{A\}$, правил подстановки, $\Sigma_{GF}=\Xi_{GF}\cup\Phi_{GF}$, $\Xi_{GF}=\{\circ,f\}$, $\Phi_{GF}=\{*,\wedge,+,-,\times,:\}$, $\Lambda_{GF}=\Lambda_L\cup\Lambda_4$.

Обозначим отрезок ломаной ν с атрибутами $i \lrcorner \nu$ – порядковый номер при формировании ломаной, $X_i \lrcorner \nu, Y_i \lrcorner \nu$ – координаты начала, $l \lrcorner \nu$ – длина, $\beta \lrcorner \nu$ угол наклона.

Введем операции над атрибутами:

- сложения, вычитания, умножения и деления соответственно $+(c,a,b)$, $-(c,a,b)$, $\times(c,a,b)$ и $:(c,a,b)$ с операндами a, b и результатом c;
- вычисления конца текущего отрезка (и начала следующего) $^*(\mathrm{t}M_x, \beta, X_i, Y_i, X_{i+1}, Y_{i+1})$ с начальными координатами X_i, Y_i, длине $\mathrm{t}M_x$ и углом наклона β;
- генерация случайного, нормально распределенного числа $\wedge(c,a,b)$ с математическим ожиданием a и дисперсией b.

ИОК Λ_4 включает приведенные выше определения, обозначения и их семантику, а также следующие положения:

- *онтология* дополняется известными понятиями «плоскость», «отрезок», «вещественное число», «координаты», «угол», и другими, позволяющими оперировать как с линейными геометрическими фигурами, так и с вещественными числами;
- *цель конструирования* – формирование линейного геометрического фрактала на плоскости;
- *правила подстановки*:

$$\{\langle\langle A \to fA,\ A \to \nu A\rangle, \langle {}^*(\mathrm{t}M_x, \beta, X_i, Y_i, X_{i+1}, Y_{i+1}), = (i,i,1)\rangle\rangle,$$
$$\langle\langle A \to zA\rangle, \langle\varepsilon\rangle\rangle,$$
$$\langle\langle A \to yA\rangle, \langle\varepsilon\rangle\rangle,$$
$$\langle\langle A \to +A\rangle, \langle \times(\mathrm{qM}, M_x, dM_x), : (\mathrm{qM}, \mathrm{qM}, 100), +(\mathrm{t}M_x, \mathrm{t}M_x, qM), \wedge(l, \mathrm{t}M_x, D_x), +(\beta,\beta,\alpha)\rangle\rangle,$$
$$\langle\langle A \to -A\rangle, \langle \times(\mathrm{qM}, M_x, dM_x), : (\mathrm{qM}, \mathrm{qM}, 100), -(\mathrm{t}M_x, \mathrm{t}M_x, qM), \wedge(l, \mathrm{t}M_x, D_x), -(\beta,\beta,\alpha)\rangle\rangle,$$
$$\langle\langle A \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle\};$$

- *ограничения* – правило $\langle A \to \varepsilon, \langle \varepsilon \rangle \rangle$ выполняется, если неприменимы все остальные;
- *начальные условия* – цепочка $\Omega_i(C_{MS})$; начальная точка $X_0 = 0, Y_0 = 0$; начальный угол $\beta = 0$, начальный номер точки $i = 0$, текущая длина отрезка $\mathrm{t}M_x = M_x$;
- *условие завершения* – выполнение правила $\langle A \to \varepsilon, \langle \varepsilon \rangle \rangle$.

Определим конструктивную систему, интерпретировав C_{GF}:

$$\begin{aligned} &\langle C_{GF}(\Omega_i(\mathrm{C}_{MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \alpha, \mathrm{m}) = \langle M_{GF}, \Sigma_{GF}, \Lambda_{GF} \rangle, C_B = \langle M_B, \Sigma_B, \Lambda_B \rangle \rangle_I \mapsto \\ &C_{B,GF}(\Omega_i(\mathrm{C}_{MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \alpha, \mathrm{m}) = \langle M_{B,GF}, \Sigma_{B,GF}, \Lambda_{B,GF} \rangle, \end{aligned} \tag{11}$$

где C_B – конструктор, расширяющий возможности C_A наличием сформированных алгоритмов $\{A_7 \|^{c}_{a,b}, A_8 \|^{c}_{a,b}\ A_9 \|^{X_{i+1},Y_{i+1}}_{\mathrm{t}M_x,\beta,X_i,Y_i}, A_{10} \|^{qM}_{dM_x,D_x}\} \subset \Omega(C_B)$, $M_B \supset M_A$, $M_{B,GF} = \langle M_{GF}, M_B \rangle$, $\Sigma_B = \Sigma_A$, $\Sigma_{B,GF} = \langle \Sigma_{GF}, \Sigma_B \rangle$, $\Lambda_B = \Lambda_A$, $\Lambda_{B,GF} = \Lambda_{GF} \cup \Lambda_B \cup \Lambda_5$

ИОК Λ_5 включает приведенные выше обозначения и их семантику, а также определение атрибутики операций (алгоритмов их реализующих):

$$\begin{aligned} \Lambda_5 = \{&(A_7 \|^{c}_{a,b} \lrcorner +), (A_8 \|^{c}_{a,b} \lrcorner -), (A_9 \|^{c}_{a,b} \lrcorner \times), (A_{10} \|^{c}_{a,b} \lrcorner :), \\ &(A_{11} \|^{X_{i+1},Y_{i+1}}_{t,\beta,X_i,Y_i} \lrcorner ^*), (A_{12} \|^{qM}_{dM_x,D_x} \lrcorner \wedge), (A_{13} \|^{r}_{t,\beta,X_i,Y_i} \lrcorner f)\}. \end{aligned} \tag{12}$$

В дальнейшем выполняется конкретизация путем задания значений параметров в конструктивной системе и соответствующая реализация.

Рассмотрим их на примерах.

Детерминированный фрактал «снежинка Коха», представленный на рис. 1, является реализацией одного из семейства параметрических конструкторов с параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 60^\circ,\ m = 1$ $\langle C_{GF}(\Omega_2, 1, 0, 0, 60, 1), C_B \rangle_R \mapsto \Omega_3$.

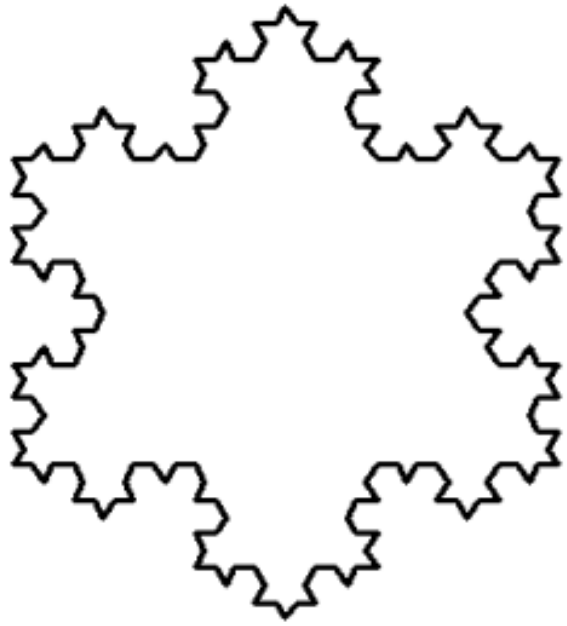


Рис. 1. Реализация конструктора Ω_3 детерминированного фрактала «снежинка Коха»

Один из 20 стохастических вариантов этого фрактала, реализованных конструктором $\langle C_{GF}(\Omega_2,1,50,50,60,20),C_B\rangle_R \mapsto \Omega_4$, представлен на рис. 2.

Рис. 2. Реализация конструктора Ω_4 стохастического фрактала «снежинка Коха»

Детерминированный фрактал «дракон Хартера – Хейтуэя», представленный на рис. 3, является реализацией конструктора $\langle C_{GF}(\Omega_1,1,0,0,90,1),C_B\rangle_R \mapsto \Omega_5$.

Один из 20 стохастических вариантов этого фрактала, реализованных конструктором $\langle C_{GF}(\Omega_1,1,50,50,90,20),C_B\rangle_R \mapsto \Omega_6$, представлен на рис. 4.

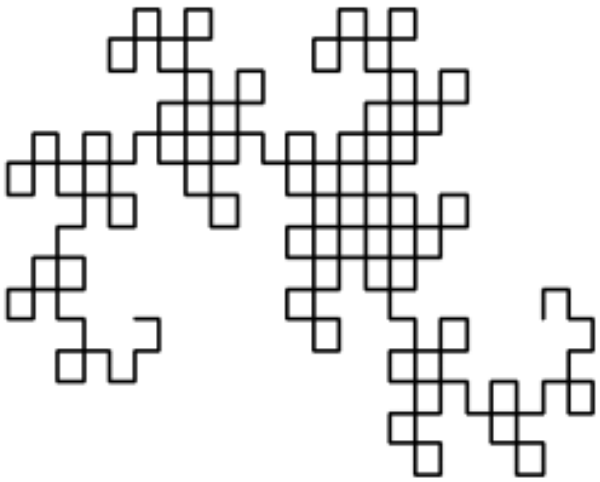


Рис. 3. Реализация конструктора Ω_5 детерминированного фрактала «дракон Хартера – Хейтуэя»

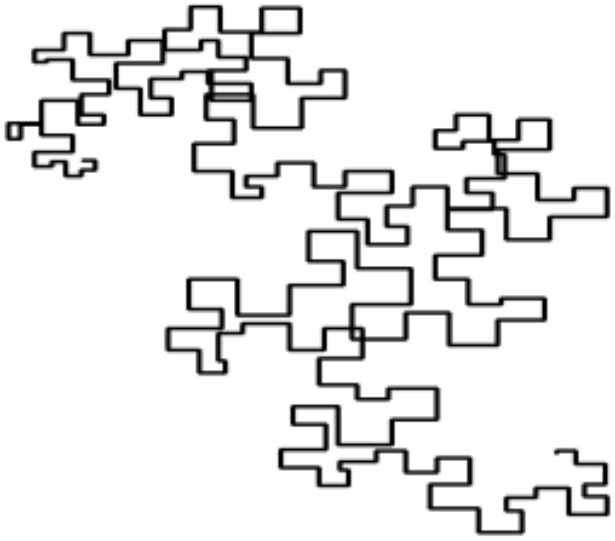


Рис. 4. Реализация конструктора Ω_4 стохастического фрактала «дракон Хартера – Хейтуэя»

Таким образом, конструктивно-продукционный подход к задачам моделирования позволяет формировать линейные геометрические фракталы путем преобразований двух параметрических семейств конструкторов:

$$C_{S} \mapsto C_{L_K} \mapsto \begin{cases} C_{MS}(\mathrm{B},\mathrm{P},\mathrm{n}),\ C_{A_I} \mapsto C_{A,MS_R} \mapsto \Omega_i(C_{MS}) \\ C_{GF}(\Omega_i(C_{MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \alpha, \mathrm{m})_I \mapsto C_{B,GF_R} \mapsto \Omega_i(C_{GF}). \end{cases} \quad (13)$$

Компьютерная программа

Разработанный программный инструментарий на языке программирования Python 2.7 в полной мере реализует возможности представленных моделей. На рис. 5. представлен скриншот главного окна программы.

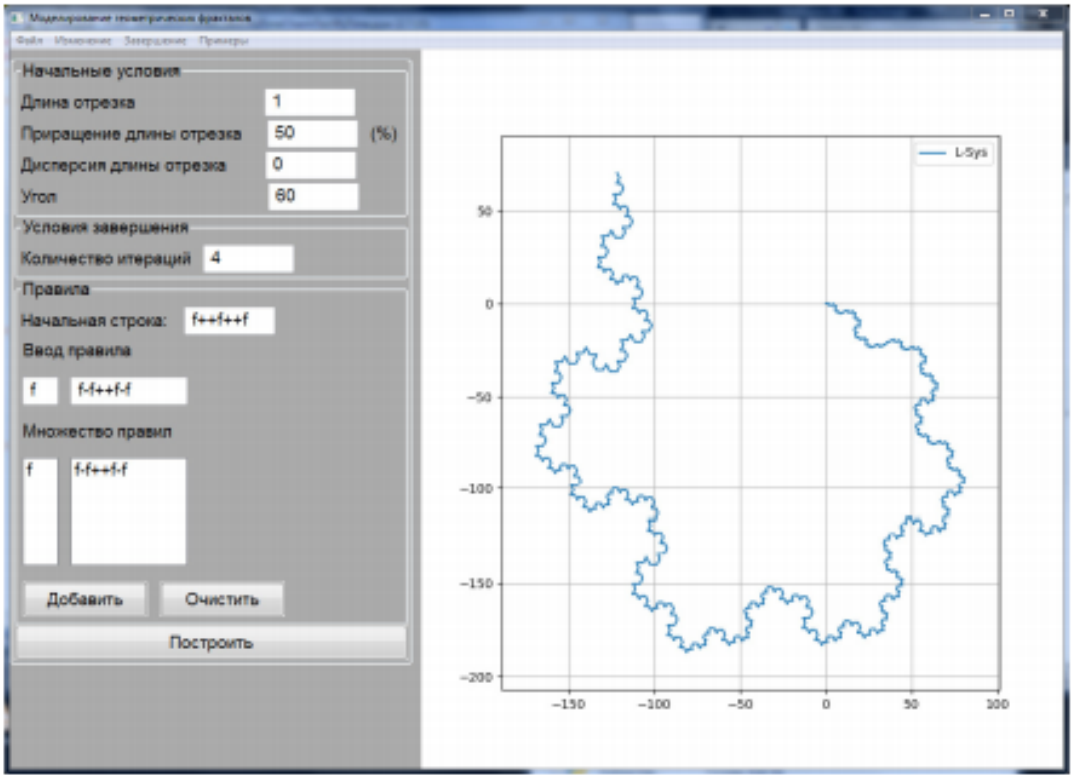


Рис .5. Скриншот главного окна программы моделирования линейных геометрических фракталов

Варьируя количеством и содержанием правил подстановки, количеством итераций (параметрами системы конструкторов $C_{A,MS}$), значениями и изменениями длины отрезков, углом поворота (параметрами системы конструкторов $C_{B,GF}$), исследователь получает широкие возможности моделирования линейных плоских геометрических фракталов.

Предусмотрена возможность формирования известных фракталов: «Снежинка Коха», «Кривая Коха», «Кривая Серпинского», «Дракон Хартера – Хейтвея», «Ледяной узор», «Фрактальные треугольники», воспользовавшись меню «Примеры».

Результаты

Разработано семейство параметрических конструкторов формирования мультисимвольных фракталов и связанное с ним семейство параметрических конструкторов-преобразователей мультисимвольных в линейные плоские геометрические фракталы. Оба семейства базируются на идеях L-систем.

Процесс формирования фракталов с различной элементной базой построен таким образом, что на основе разбора (анализа) одних из них формируются другие. Явным образом устанавливается связь между ними.

Авторами разработана программа, реализующая представленные в данной работе семейства параметрических конструкторов.

Заключение

В работе получил дальнейшее развитие конструктивно-продукционный подход формализации конструкций и конструктивных процессов, допускающих их определение и моделирование. Представленный в статье подход к моделированию процессов на основе обобщенного конструктора позволяет эффективно и просто строить детерминированные и

стохастические линейные геометрические фракталы на базе продукционных L – систем.

Предложенная вариативность на основе семейства параметрических конструкторов позволяет формализовать взаимнооднозначное соответствие между конструкциями и различной природы, что открывает новые возможности их изучения.

Разработанный формализм, в совокупности с другими работами по конструктивно-продукционному моделированию, позволяет создавать более универсальные программные средства конструирования и оптимизации конструкций и конструктивных процессов различной природы.

Дальнейшая работа

Ведется работа по установлению соответствия между мультисимвольными, геометрическими фракталами и фрактальными временными рядами и расширением возможностей программных средств.

Благодарности

Статья публикуется с частичной поддержкой ITHEA ISS (www.ithea.org) и ADUIS (www.aduis.com.ua).

The paper is published with partial support by the ITHEA ISS (www.ithea.org) and the ADUIS (www.aduis.com.ua).

Литература

[Camps-Raga, 2010] Camps-Raga B., Islam N. E.Optimized simulation algorithms for fractal simulation and analysis. Progress In Electromagnetics Research M. Vol. 11, 2010. - P. 225 -240.

[Chiu, 2006] Chiu W.K., Yeung, Y.C., Yu, K.M. Toolpath generation for layer manufacturing of fractal objects. Rapid Prototyping Journal. Vol. 12, № 4, 2006. P. 214 – 221.

[Lindenmayer, 1968] Lindenmayer A. Mathematical models for cellular interaction in development. Parts I and II. Journal of Theoretical Biology. V. 18, 1968. P. 280 - 315.

[Kravchenko, 2017] Kravchenko G. Modeling the External Structure of a Fractals. IOP Conf. Series: Earth and Environmental Science, 2017, doi.10.1088/1755-1315/90/1/012100.

[Mandelbrot, 1982] Mandelbrot B.B. The Fractal Geometry of Nature. San Francisco, 1982. 462 p.

[Peitgen, 2004] Peitgen H.- O., Jurgens H., Saupe D. Chaos and Fractals. N.Y.: Springer, 2004, 864 p.

[Shynkarenko, 2009] Shynkarenko V.I., Ilman V.M., Skalozub V.V. Structural models of algorithms in problems of applied programming. I. Formal algorithmic structures. Cybernetics and Systems Analysis, Vol. 45, No 3. Springer, 2009. pp 329-339. ISSN: 1060-0396 (Print) 1573-8337 (Online), doi: org/10.1007/s10559-009-9118-0 https://link.springer.com/article/10.1007/s10559-009-9118-0

[Shynkarenko, 2014] Shynkarenko V.I., Ilman V.M. Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure. Cybernetics and Systems Analysis, Vol. 50, No 5. Springer, 2014. P. 665 – 662. Part II. Refining Transformations. – Vol. 50, No 6, 2014. P. 829 – 841. ISSN: 1060-0396 (Print) 1573-8337 (Online), doi: 10.1007/s10559-014-9655-z,

https://link.springer.com/article/10.1007/s10559-014-9655-z, doi: 10.1007/s10559-014-9674-9, https://link.springer.com/article/10.1007/s10559-014-9674-9

[Shynkarenko, 2015] Shynkarenko V.I., VasetskaT.M. Modeling the Adaptation of Compression Algorithms by Means of Constructive-Synthesizing Structures. Cybernetics and Systems Analysis, Vol. 51, No 6. Springer, 2015. P. 849-861. doi: 10.1007/s10559-015-9778-x https://link.springer.com/article/10.1007/s10559-015-9778-x

[Zhou, 1995] Zhou Jack G.; Leu M. C.; Blackmore D.: Fractal Geometry Modeling with Applications in Surface Characterization and Wear Prediction. / International Journal of Machine Tools & Manufacture. Vol. 35, No. 2, 1995. P. 203-209.

[Zhou, 2010] Zhou J., Vas A., Blackmore D. Fractal geometry surface modeling and measurement for musical cymbal surface texture design and rapid manufacturing. – Режим доступа: https://www.researchgate.net/publication/250330003.

[Безносюк, 2002] Безносюк С.А., Лерх Я.В., Жуковская Т.М. Компьютерное моделирование самоорганизации фрактальных кластерных нанодендритов. Ползуновский вестник, 2002. С. 160 - 166.

[Божокин, 2001] Божокин С.В., Паршин Д.А. Фракталы и мультифракталы. М., Ижевск: НИЦ Регулярная и хаотическая динамика, 2001. 128 с.

[Кравченко,2016] Кравченко Г.М., Васильев С.Э., Пуданова Л.И. Моделирование фракталов. Инженерный вестник Дона. №4, 2016. - Режим доступа: https://www ivdon.ru/ru/magazine/archive/n4y2016/3930.

[Кроновер, 2000] Кроновер Р. М. Фракталы и хаос в динамических системах. Основы теории. М.: Постмаркет, 2000. 352 с.

[Помулев, 2002] Помулев В.В., Михалев А. И., Бондаренко Я. С., Деревянко А. И Моделирование и фрактальная параметризация дендритов нейронов. Адаптивные системы автоматического управления, №5(25), 2002. С. 160 - 166. ISSN 1560-8956

[Слюсар, 2007] Слюсар В. Фрактальные антены – принципиально новый тип ломаных антенн. Электроника: Наука, Технология, Бизнес. № 5, 2007. С.78 - 83. - Режим доступа: https://www.elektronics.ru/files/article_pdf/0/article_611_312.pdf.

[Федер,1991] Федер Е. Фракталы. М.: Мир, 1991. 262 с.

[Шелухин, 2008] Шелухин О.И., Осин А.В., Смольский С.М. Самоподобие и фракталы. Телекомуникационные приложения. М. Физматлит, 2008. 365 с.

[Шелухин, 2011] Шелухин О.И. Мультфракталы. Инфокоммуникационные приложения. М. Горячая линия – Телеком, 2011. 576 с.

[Шинкаренко и Васецкая, 2016] Шинкаренко В. И., Васецкая Т. Н. Моделирование процесса ранжирования альтернатив методом анализа иерархий средствами конструктивно-продукционных структур. Математические машины и системы. № 1, 2016. С. 39-47. ISSN 1028-9763

[Шинкаренко и Забула, 2016] Шинкаренко В. И., Забула Г. В. Конструктивная модель адаптации структур данных в оперативной памяти. ЧАСТЬ I. Конструирование текстов программ. Наука и прогресс транспорта. № 1 (61), 2016. С. 109-121.; ЧАСТЬ II. Конструкторы сценариев и процессов адаптации. № 2 (62), 2016. С. 88-97. ISSN 2307-6666

[Шинкаренко и Куропятник, 2016] Шинкаренко В. И., Куропятник Е. С. Конструктивно-продукционная модель графового представления текста. Проблемы программирования. № 2-3, 2016. С. 63-72. ISSN 1727- 4907, http://dspace.nbuv.gov.ua/bitstream/handle/123456789/126391/07-Shinkarenko.pdf?sequence=1

Информация об авторах

Виктор Шинкаренко *– д.т.н., профессор, зав. кафедрой «Компьютерные информационные технологии» Днепровского национального университета железнодорожного транспорта имени академика В. Лазаряна; ул. Лазаряна, 2, 49010, Днепр, Украина;*

e-mail*: shinkarenko_vi@ua.fm*

Основные области научных исследований*: конструктивно-продукционное моделирование, качество программного обеспечения, искусственный интеллект.*

Константин Литвиненко *– к.т.н., доцент кафедры «Компьютерные информационные технологии» Днепровского национального университета железнодорожного транспорта имени академика В. Лазаряна; ул. Лазаряна, 2, 49010, Днепр, Украина;*

e-mail*: kosta11111973@gmail.com*

Основные области научных исследований*: конструктивно-продукционное моделирование, симметрия в задачах оптимизации, моделирование рисков сложных систем*

Роберт Чигирь *– студент Днепровского национального университета железнодорожного транспорта имени академика В. Лазаряна; ул. Лазаряна, 2, 49010, Днепр, Украина;*

e-mail*: kosta11111973@gmail.com*

Основные области научных исследований*: геометрические фракталы, фрактальные временные ряды*

Constructive compliance of multicharacter and linear geometric fractals

Viktor Shynkarenko, Kostiantyn Lytvynenko, Robert Chyhir

Abstract: *The paradigm of a constructive conception of the surrounding world is based on the thesis that the whole world consists of constructions and constructive processes. Certain constructions are transformed into others by means of constructive processes. There may be an explicit or implicit compliance between constructions of different nature. In this paper, the constructive-synthesizing approach to fractal modeling allows us to set up a correspondence between multicharacter and linear flat geometric determined and stochastic fractals. This expands the possibilities of constructing the latter in order to study their properties, as well as our ability to model real-world objects. Means of constructive-synthesizing modeling developed into the form of a family of parametric designers that allows to vary the capabilities of designers solidly.*

Keywords: *constructive-synthesizing modelling, designer, fractal, stochastic fractals, L-system, specialization, interpretation, specification, implementation.*

XIV Міжнародна науково-практична конференція
XIV Международная научно-практическая конференция
XIV International Scientific & Applied Conference

ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ ПОБУДОВИ ПРОГРАМНИХ СИСТЕМ

ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ ПОБУДОВИ ПРОГРАМНИХ СИСТЕМ

ТЕОРЕТИЧЕСКИЕ И ПРАКТИЧЕСКИЕ АСПЕКТЫ ПОСТРОЕНИЯ ПРОГРАММНЫХ СИСТЕМ

TAAPSD'2017

Праці конференції
Труды конференции
Proceeding

4–8 грудня 2017 року
Київ ○ Киев ○ Kyiv

В. И. Шинкаренко, К. В. Литвиненко, Р. Р. Чигирь,
А. А. Жадан

Вариативность уточняющих преобразований конструктивно-продукционного моделирования

Днепропетровский национальный университет железнодорожного транспорта имени академика В.Лазаряна, Днепр, Украина.

Применяя один из базовых принципов научных исследований – от частного к общему, а затем от общего к частному, в [1],[2] выполнено обобщение возможностей и особенностей различных модификаций грамматик и грамматико-подобных систем. В результате имеем обобщенный конструктор (ОК) – средство конструирования:

$$C_G =< M, \Sigma, \Lambda >,$$

где M – неоднородный расширяемый носитель структуры, Σ – сигнатура, состоящая из множеств операций связывания, подстановки и вывода, операций над атрибутами и отношения подстановки, Λ – информационное обеспечение конструирования, которое включает онтологию, цели, правила и ограничения конструирования. Онтология ОК в неформальном виде представлена в [2].

Цели конструкционно-продукционного моделирования могут заключаться в формировании, преобразовании и анализе конструкций с помощью операций связывания, подстановки, вывода и др. операций, задаваемых правилами.

Особенностями конструктивно-продукционного моделирования являются: атрибутивность элементов и операций, расширяемый носитель, модель исполнителя в виде его базовых алгоритмов, связь операций с алгоритмами их выполнения.

Отталкиваясь от общего удалось решить ряд честных задач:

- адаптации алгоритмов сжатия к архивируемым данным [3];
- совершенствования процесса ранжирования альтернатив методом анализа иерархий [4];
- адаптации структур данных в оперативной памяти [5];
- совершенствование структур хранения данных в задачах выявления плагиата [6];
- рационального распределения энергии рекуперации тяги постоянного тока [7].

Для формирования конструкций выполняется ряд уточняющих преобразований:

- специализация определяет предметную область: семантическую природу носителя, конечное множество операций и их семантику, атрибутику операций, порядок их выполнения и ограничения на правила подстановки $C_G \ {}_S \mapsto \ {}_SC$;
- интерпретация заключается в связывании операций сигнатуры с алгоритмами выполнения некоторой алгоритмической структуры C_A [8], что связывает информационную модель способа построения конструкций с модели исполнителя
  $< {}_SC, C_A > \ {}_I \mapsto \ {}_{S,I}C$;
- конкретизация конструктора заключается в расширении аксиоматики множеством правил продукций, задании конкретных множеств нетерминальных и терминальных символов с их атрибутами и, при необходимости, значений атрибутов
  ${}_{S,I}C \ {}_K \mapsto \ {}_{S,I,K}C$;
- реализация конструктора заключается в формировании множества конструкции Ω из элементов носителя конструктора путем выполнения алгоритмов, связанных с операциями сигнатуры ${}_{S,I,K}C \ {}_R \mapsto \ {}_{S,I,K,R}C \mapsto \Omega$.

В работах [3], [4], [5], [6], [7] уточняющие преобразования выполняются именно в такой последовательности:

$$C_G \ {}_S \mapsto \ {}_SC \ \mapsto < {}_SC, C_A > \ {}_I \mapsto \ {}_{S,I}C \ {}_K \mapsto \ {}_{S,I,K}C \ {}_R \mapsto \ {}_{S,I,K,R}C$$

Как оказалось такой подход не является единственно возможным и необходимым. Вариативность порядка применения уточняющих преобразований приводит к достаточно интересным результатам.

Рассмотрим случай с множественностью интерпретаций, демонстрирующий связь между конструкциями и конструктивными процессами, обладающими свойством фрактальности [9] – самоподобия:

$$C_G =< M_G, \Sigma_G, \Lambda_G > \ {}_K \mapsto \ {}_KC =< M_K, \Sigma_K, \Lambda_K > \ {}_S \mapsto$$

$$S \mapsto \begin{cases} {}_{K,S_1}C =< M_{S_1}, \Sigma_{S_1}, \Lambda_{S_1} > \mapsto < {}_{K,S_1}C, C_A >_I \mapsto \\ \qquad {}_{K,S_1,I_1}C =< M_{I_1}, \Sigma_{I_1}, \Lambda_{I_1} >_R \mapsto {}_{K,S_1,I_1,R_1}C \mapsto \Omega_1; \\ {}_{K,S_2}C =< M_{S_2}, \Sigma_{S_2}, \Lambda_{S_2} > \mapsto < {}_{K,S_2}C, C_A >_I \mapsto \\ \qquad {}_{K,S_2,I_2}C =< M_{I_2}, \Sigma_{I_2}, \Lambda_{I_2} >_R \mapsto {}_{K,S_2,I_2,R_2}C \mapsto \Omega_2; \\ {}_{K,S_3}C =< M_{S_3}, \Sigma_{S_3}, \Lambda_{S_3} > \mapsto < {}_{K,S_3}C, C_A >_I \mapsto \\ \qquad {}_{K,S_3,I_3}C =< M_{I_3}, \Sigma_{I_3}, \Lambda_{I_3} >_R \mapsto {}_{K,S_3,I_3,R_3}C \mapsto \Omega_3. \end{cases}$$

Пусть носитель M_K конструктора ${}_KC$ содержит терминалы $\{f, z, y, x\} \in M_K$ и атрибуты конструирования $\{M_x, dM_x, D_x, \alpha, \alpha t\}$; сигнатура – операций $\{+, -\} \in \Sigma_K$; информационное обеспечение Λ_K включает:

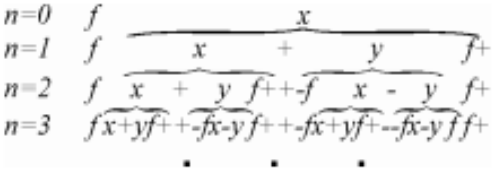
- онтологию из Λ_G, дополненную онтологией L-систем [2], [9];
- правила конструирования: аксиому fz, правила подстановки $y \to fz - y$ и $z \to z + yf+$;
- ограничения: $n = 8, \alpha = 90, \alpha t = 90, M_x = 1, dM_x = 0, D_x = 0$;
- цель: формирование конструкций и конструктивных процессов.

Рассмотрим первый вариант специализации, интерпретации и реализации ${}_{K,S_1}C =< M_{S_1}, \Sigma_{S_1}, \Lambda_{S_1} > \mapsto < {}_{K,S_1}C, C_{A_1} >_I \mapsto$ $_I \mapsto {}_{K,S_1,I_1}C =< M_{I_1}, \Sigma_{I_1}, \Lambda_{I_1} >_R \mapsto {}_{K,S_1,I_1,R_1}C \mapsto \Omega_1$ – конструктор мультисимвольных цепочек.

Специализация: онтологию в Λ_{S_1} дополним семантикой элементов носителя и операций. Их значениями являются изображения соответствующих символов $\{f, z, y, x, +, -\}$.

Интерпретация дополняет онтологию связями операций и терминалов с соответствующими алгоритмами их реализации (нотация [8]):
$(A_j|^{\widetilde{+}}_{+} \hookleftarrow +), (A_j|^{\widetilde{-}}_{-} \hookleftarrow -), (A_j|^{\widetilde{f}}_{f} \hookleftarrow f), (A_j|^{\widetilde{y}}_{y} \hookleftarrow y), (A_j|^{\widetilde{z}}_{z} \hookleftarrow z)$.
Будем считать, что алгоритмическая структура C_A имеет все необходимые базовые алгоритмы и алгоритм A_i, который строит изображение входного символа.

Реализация заключается в параллельном выполнении подстановок:



В результате реализации получаем мультисимвольную конструкцию из 1022 символов: $(fz+yf++-fz-yf++\ \cdots\ -fz-yf+)\in\Omega_1$, которая обладает свойством самоподобия, что наглядно представлено процедурой её формирования.

Второй вариант уточняющих преобразований позволяет моделировать фрактальные временные ряды с постоянным шагом.

Специализация: f –указатель необходимости вычисления значения временного ряда, z,y – не имеют значения, операции $+,-$ изменяют значение M_x на величину dM_x.

Интерпретация: $(A_k|^{x(t_i),i}_{M_x,D_x,i}\ \leftarrow\!\!\lfloor\ f)$, $(A_{k+1}|^{\oslash}_{z}\ \leftarrow\!\!\lfloor\ z)$, $(A_{k+2}|^{\oslash}_{y}\ \leftarrow\!\!\lfloor\ y)$, $(A_{k+3}|^{M_x}_{M_x,dM_x}\ \leftarrow\!\!\lfloor\ +)$, $(A_{k+4}|^{M_x}_{M_x,dM_x}\ \leftarrow\!\!\lfloor\ -)$, где A_k – выдает нормально распределенное случайное значение временного ряда $x(t_i)$ с матожиданием M_x и дисперсией D_x и увеличивает счетчик i на еденицу, A_{k+1},A_{k+2} – пустые алгоритмы, A_{k+3},A_{k+4} – увеличивают и уменьшают (соответственно) M_x на величину dM_x.

Реализацией является временной ряд представленный в табл.1 (строка 1).

Третий вариант уточняющих преобразований позволяет моделировать известный фрактал – дракон Хартера-Хейтуэя [10] со стохастическими шумами.

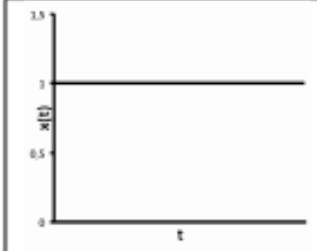
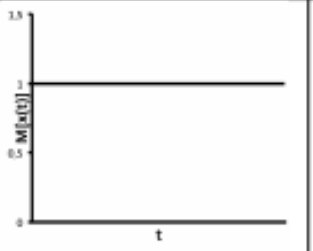
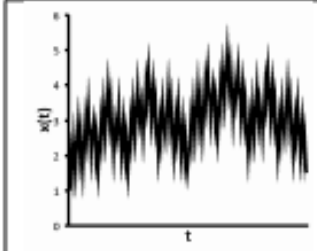
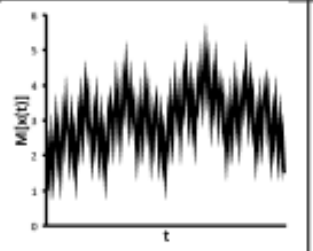
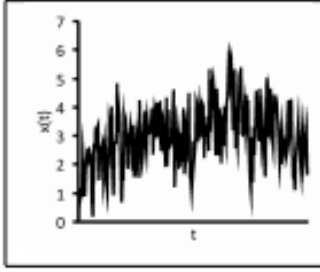
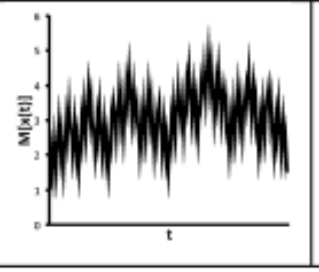
Специализация: f – указатель необходимости отображения отрезка, z,y – не имеют значения, операции $+,-$ изменяют значение M_x на величину dM_x и α на αt.

Интерпретация: $(A_l|^{\widetilde{f}}_{M_x,D_x,\alpha}\ \leftarrow\!\!\lfloor\ f)$, $(A_{l+1}|^{\oslash}_{z}\ \leftarrow\!\!\lfloor\ z)$, $(A_{l+2}|^{\oslash}_{y}\ \leftarrow\!\!\lfloor\ y)$, $(A_{l+3}|^{M_x,\alpha}_{M_x,dM_x,\alpha,\alpha t}\ \leftarrow\!\!\lfloor\ +)$, $(A_{l+4}|^{M_x,\alpha}_{M_x,dM_x,\alpha,\alpha t}\ \leftarrow\!\!\lfloor\ -)$, где A_l – вырисовывает отрезок начиная с текущей точки, длина которого – случайная, нормально распределенная величина с матожиданием M_x и дисперсией D_x под углом αt, A_{l+1},A_{l+2} – пустые алгоритмы, A_{l+3},A_{l+4} – увеличивают и уменьшают (соответственно) M_x на величину dM_x и αt на величину α.

Реализацией является детерминированное изображение дракона в табл.1 (строка 1).

Первая строка таблицы соответствует ограничениям специализированного конструктора $M_x=1$, $dM_x=0$, $D_x=0$, вторая – $M_x=1$, $dM_x=0,5$, $D_x=0$ и третья – $M_x=1$, $dM_x=0,5$, $D_x=0,5$. Первый и второй случай соответствует детерминированному временному ряду и дракону, третей – стохастическим. Все приведенные реализации обладают свойством фрактальности.

Таблица 3: Реализации второго и третьего вариантов уточняющих преобразований

Временной ряд $x(t_i)$	Матожидание $M[x(t_i)]$	Фрактал – дракон
		
		
		

Выводы. Предложенная вариативность уточняющих преобразований конструктора позволяет формализовать детерминированное и стохастическое взаимооднозначное соответствие между конструкциями и/или конструктивными процессами различной природы. Что очень важно в области программирования для установления связи между программой и процессом ее выполнения.

[1] *Шинкаренко В. И.* Конструктивно-продукционные структуры и их грамматические интерпретации. I. Обобщенная формальная конструктивно-продукционная структура. / В. И. Шинкаренко, В. М. Ильман // Кибернетика и системный анализ. — 2014. — №5. — С.8-16.

[2] *Шинкаренко В. И.* Конструктивно-продукционные структуры и их

грамматические интерпретации. II. Уточняющие преобразования / В. И. Шинкаренко, В. М. Ильман // Кибернетика и системный анализ. — 2014. — № 6. — С. 15-28.

[3] *Шинкаренко В. И.* Моделирование процесса адаптации алгоритмов сжатия средствами конструктивно-продукционных структур / В. И. Шинкаренко, Т. Н. Васецкая // Кибернетика и системный анализ. — 2015. — № 6. — С. 19-34.

[4] *Шинкаренко В. И.* Моделирование процесса ранжирования альтернатив методом анализа иерархий средствами конструктивно-продукционных структур / В. И. Шинкаренко, Т. Н. Васецкая // Математичні машини та системи. — 2016. — № 1. — С. 39-47.

[5] *Шинкаренко В. И.* Конструктивная модель адаптации структур данных в оперативной памяти. ЧАСТЬ I. Конструирование текстов программ / В. И. Шинкаренко, Г. В. Забула // Наука та прогрес транспорту. Вісн. Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна. — 2016. — № 1 (61) — С. 109-121.; ЧАСТЬ II. Конструкторы сценариев и процессов адаптации. — 2016. — № 2 (62) — С. 88-97.

[6] *Шинкаренко В. И.* Конструктивно-продукционная модель графового представления текста / В. И. Шинкаренко, Е. С. Куропятник // Проблеми програмування. — 2016. — № 2-3. — С. 63-72.

[7] *Шинкаренко В. И.* Конструктивное моделирование зоны распределения энергии рекуперации тяги постоянного тока / В. И. Шинкаренко, О.И. Саблин, А.П. Иванов // Наука та прогрес транспорту. Вісн. Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна. — 2016. — № 5 (65) — С. 125-135.

[8] *Шинкаренко В. И., В. М. Ильман, В. В. Скалозуб.* Структурные модели алгоритмов в задачах прикладного программирования Часть I. Формальные алгоритмические структуры // Кибернетика и системный анализ. — 2009. — № 3. — С. 3-14.

[9] *Lindenmayer A.* Mathematical models for cellular interaction in development. Parts I and II. / A. Lindenmayer// Journal of Theoretical Biology, — 1968. — V. 18. — С. 280 — 315.

[10] *Божокин С.В.* Фракталы и мультифракталы / С.В. Божокин, Д.А. Паршин. — М., Ижевск: Регулярная и хаотическая динамика — 2001. — 128с.

E-mail: ✉ *author_shinkarenko_vi@ua.fm.*

Матеріали міжнародної наукової конференції
Материалы международной научной конференции
Conference proceedings

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ ТА ПРОБЛЕМИ ОБЧИСЛЮВАЛЬНОГО ІНТЕЛЕКТУ

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ПРИНЯТИЯ РЕШЕНИЙ И ПРОБЛЕМЫ ВЫЧИСЛИТЕЛЬНОГО ИНТЕЛЛЕКТА

INTELLECTUAL SYSTEMS FOR DECISION MAKING AND PROBLEMS OF COMPUTATIONAL INTELLIGENCE

ISDMCI 2018

Міжнародний Центр НАН та МОН України

May 21-27 2018
Zaliznyi Port, Ukraine

21-27 мая 2018
Железный Порт, Украина

21-27 травня 2018
Залізний Порт, Україна

КОНСТРУКТИВНО-ПРОДУКЦИОННОЕ МОДЕЛИРОВАНИЕ ФРАКТАЛОВ

Шинкаренко В.И., Литвиненко К.В., Чигирь Р.Р., Жадан А.А.
Днепропетровский национальный университет железнодорожного транспорта имени академика В. Лазаряна, г. Днепр, ул. В. Лазаряна 2, e-mail: Shinkarenko_vi@ua.fm

Как известно, фракталом является объект, который обладает свойством самоподобия, он более или менее единообразно устроен на широком диапазоне масштабов. В идеальном случае, самоподобие гарантирует инвариантность при любом изменении масштаба, однако это характерно только для регулярных, детерминированных фракталов. Если к детерминированному процессу построения включен алгоритм порождения случайности, тогда возникают стохастические, случайные фракталы. Свойство самоподобия стохастических фракталов проявляется только при усреднении по всех статистически независимых реализациях объекта. Поэтому, часть фрактала при изменении масштаба не полностью инвариантна начальному фрагменту, однако их статистические характеристики совпадают.

В работах [1], [2] обобщены возможности различных модификаций грамматик и грамматико-подобных систем. В результате предложено рассматривать обобщенный конструктор (ОК) как средство конструирования систем

$$C_G < M, \sum, \Lambda >,$$

где M неоднородный расширяемый носитель структуры, $\sum$ сигнатура, состоящая из множеств операций связывания, подстановки и вывода, операций над атрибутами и отношения подстановки, Λ информационное обеспечение конструирования, которое включает онтологию [2], цели, правила и ограничения конструирования.

В результате функционирования ОК происходит формирование и преобразование конструкций с использованием операций, задаваемых правилами: связывания, подстановки, вывода и др.

Продукционные L-системы (Lindenmayer system) [3] широко используются для моделирования различных биологических систем, процессов компьютерной графики, музыки. Отличительные особенности L-систем (относительно классических грамматик) является отсутствие нетерминалов, атрибутивность терминалов, «параллельная» подстановка, порядок формирования множества выводимых конструкций. Для L-систем аксиомой выступает начальная конструкция.

Синтез идей ОК и L-систем позволяет построить эффективные процедуры моделирования фрактально-подобных детерминированных и стохастических систем.

Пусть носитель M конструктора C содержит терминалы $\{f, z, y, x\} \in M$ и атрибуты конструирования $M_x, dM_x, D_x, \alpha, \alpha t$; сигнатура операций $\{+, -\} \in \Sigma$; информационное обеспечение состоит из онтологии Λ, дополненную онтологией L - систем, правила конструирования начальную аксиому f_Z и правила подстановки $y \to f_Z - y, z \to z + yf +$.

Соответствие символов для ОК операциям L-систем в процессе моделирования представлено в табл. 1.

Таблица 1

Соответствие множества символов мультисимвольной цепочки операциям L-систем

Множество символов	**L-система**	**Множество символов**	**L-система**
Символ	*Операция*	*Символ*	*Операция*
*	Увеличить математическое ожидание длины отрезка	-	Уменьшить дисперсию угла
:	Уменьшить математическое ожидание длины отрезка	>	Увеличить положительный угол
\	Увеличить дисперсию длины отрезка	<	Уменьшить положительный угол
/	Уменьшить дисперсию длины отрезка	!+	Увеличить отрицательный угол
+	Поворот на положительный угол	!-	Уменьшить отрицательный угол

Результатом последовательного выполнения преобразований конструктора являются мультисимвольная цепочка со свойством самоподобия и изображение фрактала, как реализация алгоритма графического изображения символов построенной мультисимвольной последовательности.

Ниже в табл. 2 и табл. 3 представлены параметры, а на рис. 1 и рис. 2 результаты моделирования в рамках ОК регулярных и стохастических известных фракталов.

Таблица 2

Параметры моделирования в рамках ОК фрактала «снежинка Коха»

	Вариант 1	Вариант 2	Вариант 3
Начальное мат. ожидание величины и изменение	1; 0	1; 0,5	1; 0,5
Начальная дисперсия величины и изменение	0; 50	0,5; 0,25	0; 0
Начальное мат. ожидание периода и изменение периода	60; 50	60, 30	60, 30
Аксиома	f++f++f		

Правила подстановки	f:\<!--f**//>>!+!+++f:\<!--f
Мультисимвольная цепочка, количество итераций - 3	f:\<!--f**//>>!+!+++f:\<!--f:\<!--f:\<!--f**//>>!+!+++f:\<!-- f**//>>!+!+++f:\<!--f**//>>!+!+++f:\<!--f:\<!--f:\<!-- f**//>>!+!+++f:\<!--f:\<!--f:\<!--f**//>>!+!+++...

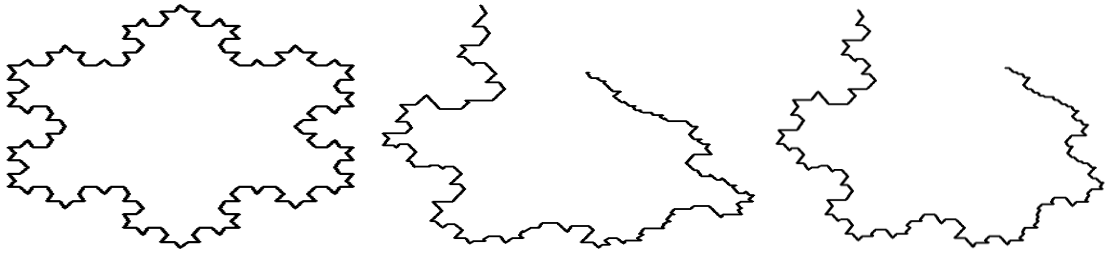


Рис.1 Изображение результатов моделирования табл.2 для вариантов 1,2,3 слева направо соответственно

Таблица 3

Параметры моделирования в рамках ОК фрактала «Дракон Хартера-Хейтуэя»

	Вариант 1	Вариант 2	Вариант 3
Начальное мат. ожидание величины и изменение	1; 0	1; 0,5	1; 0,5
Начальная дисперсия величины и изменение	0; 0	0,5; 0,25	0; 0
Начальное мат. ожидание периода и изменение периода	90; 45	90, 45	90, 45
Аксиома	fx		
Правила подстановки	y→x*\>!++yf*\>!++; x→:/<!--fx:/<!--y		
Мультисимвольная цепочка, количество итераций - 8	fx*\>!++yf*\>!++*\>!++:/<!--fx:/<!--yf*\>!++*\>!++:/<!-- fx*\>!++yf*\>!++:/<!--:/<!--fx:/<!--yf*\>!++*\>!++:/<!--		

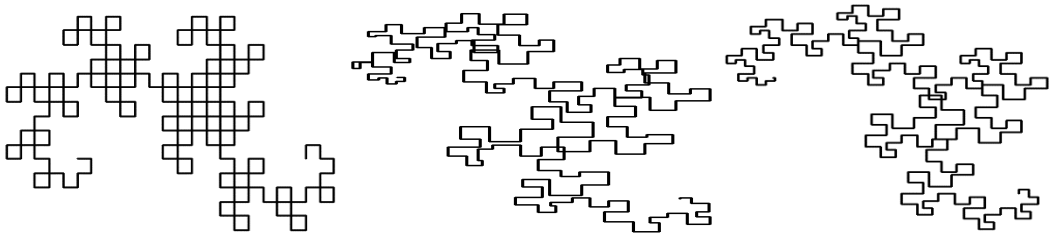


Рис.2 Изображение результатов моделирования табл.3 для вариантов 1,2,3 слева направо соответственно

Представленный подход к моделированию на основе объединенного конструктора позволяет эффективно и просто строить регулярные и стохастические фракталы на основе продукционных L-систем.

Литература.

1. Шинкаренко В. И. Конструктивно-продукционные структуры и их грамматические интерпретации. I. Обобщенная формальная конструктивно-продукционная структура. / В.И. Шинкаренко, В.М. Ильман // Кибернетика и системный анализ. 2014. №5. С.8-16.

2. Шинкаренко В. И. Конструктивно-продукционные структуры и их грамматические интерпретации. II. Уточняющие преобразования / В.И. Шинкаренко, В.М. Ильман // Кибернетика и системный анализ. - 2014. № 6. С. 15-28.

3. Lindenmayer A. Mathematical models for cellular interaction in development. Parts I and II. / A. Lindenmayer // Journal of Theoretical Biology. 1968. V. 18. P. 280 - 315.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Дніпровський державний технічний університет

ПРОБЛЕМИ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ

Матеріали Всеукраїнської науково-методичної конференції
23-25 травня 2018 року

м. Кам'янське
ДДТУ
2018

В.И. Шинкаренко, д. техн. н., **К.В. Литвиненко** к. техн. н., **Чигирь Р.Р., Жадан А.А.**

Днепропетровский национальный университет железнодорожного транспорта имени В. Лазаряна

КОНСТУКТИВНО-ПРОДУКЦИОННОЕ МОДЕЛИРОВАНИЕ ФРАКТАЛЬНЫХ ВРЕМЕННЫХ РЯДОВ НА ОСНОВЕ L-СИСТЕМ

Фрактальность, как самоподобие для временных рядов, впервые была описана Херстом и настоящее время фрактальные временные ряды начали активно изучаться в различных прикладных науках, однако вопросы формального моделирования фрактальных временных рядов не достаточно исследованы.

В [1, 2] заложены основы математико-алгоритмического конструктивизма. На этой основе представляется возможным моделирование и формализация любых конструктивных процессов с использованием обобщенного конструктора (ОК). Для формирования конструкций в рамках ОК необходимо выполнять ряд уточняющих преобразований в следующей последовательности: специализация, интерпретация, конкретизация, реализация.

Продукционные L-системы (Lindenmayer system) [3] широко используются для моделирования различных биологических систем, процессов компьютерной графики, музыки.

Обобщенным конструктором называется тройка

$$U_3 = \{\sigma\},$$

где M – неоднородный носитель структуры, Σ – сигнатура, состоящая из множеств операций связывания, подстановки и вывода, операций над атрибутами и отношения подстановки, Λ – информационное обеспечение.

Назовем L-структурой следующую специализацию ОК:

$$C = \langle M, \Sigma, \Lambda \rangle_S \mapsto C_{LS} = \langle M_{LS}, \Sigma_{LS}, \Lambda_{LS} \rangle,$$

Носитель M_{LS} структуры C_{LS} состоит из конструктивных элементов с набором атрибутов, которые связаны со статическими, динамическими, или составными свойствами элементов. Сигнатура Σ_{LS} состоит из имен операций, обладающих набором атрибутов, и состоит из множества операций: связывания, подстановки и вывода,

операций над атрибутами.

Рассмотрим возможности применения ОК для моделирования фрактальных временных рядов на основе L-систем.

Пусть носитель M_K обобщенного конструктора ${}_KC$ содержит терминалы $\{f\} \in M_K$ и атрибуты конструирования $M_x, dM_x, D_x, \alpha, \alpha t$; сигнатура операций $\{+,-\} \in \sum K$; информационное обеспечение Λ_K включает: онтологию из Λ_G [2], дополненную онтологией L-систем, правила конструирования: аксиому и правила подстановки, ограничения: $n = 8, M_x = 1, dM_x = 0, D_x = 0, \alpha = 90, \alpha t = 90$.

На первом шаге ОК формирует мультисимвольные цепочки, обладающие самоподобными свойствами. Рассмотрим уточняющие преобразования ОК:

$$ {}_{K,S}C =< M_{S_1}, \sum S_1, \Lambda_{S_1} > \mapsto <_{K,S_1} C, C_A >_I \mapsto_{K,S_1,I_1} C =< M_{I_1}, \sum I_1, \Lambda_{I_1} >_R \mapsto {}_{K,S_1,I_1,R_1}C.$$

Специализация ОК: онтологию в Λ_{S_1} дополним семантикой элементов носителя и операций. Их значениями являются изображения соответствующих символов $\{f, +, -\}$.

Специализация ОК: f – указатель необходимости вычисления значения временного ряда, операции $+, -$ изменяют значение M_x на величину dM_x.

Интерпретация ОК: $(A_k|^{x(t_i),i}_{M_x,dM_x,i} \hookleftarrow f)$; $(A_{k+1}|^{M_x}_{M_x,dM_x} \hookleftarrow +)$; $(A_{k+2}|^{M_x}_{M_x,dM_x} \hookleftarrow -)$,

где A_k – выдает нормально распределенное случайное значение временного ряда $x(t_i)$ с математическим ожиданием M_x и дисперсией dM_x, а так же увеличивает счетчик i на единицу, A_{k+1}, A_{k+2} – увеличивают и уменьшают (соответственно) M_x на величину dM_x.

Рассмотрим конкретный пример ОК. Если начальная аксиома имеет вид *f++f++f* , правила подстановки *f→f-f++f--f*, тогда ОК строит строку *f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-*.

Для значений атрибутов конструирования: начальное математическое ожидание величины и изменение математического ожидания соответственно $M_x = 1, dM_x = 0{,}5$, начальная дисперсия величины и изменение величины дисперсии соответственно $D_x = 0, dD_x = 0$, начальное математическое ожидание периода и изменение периода $\alpha = 60, \alpha t = 30$. Реализацией является временной ряд, представленный на рис.1. На рис.2

представлен фрактальный временной ряд со стохастическими шумами$D_x = 0{,}5, dD_x = 0{,}25$.

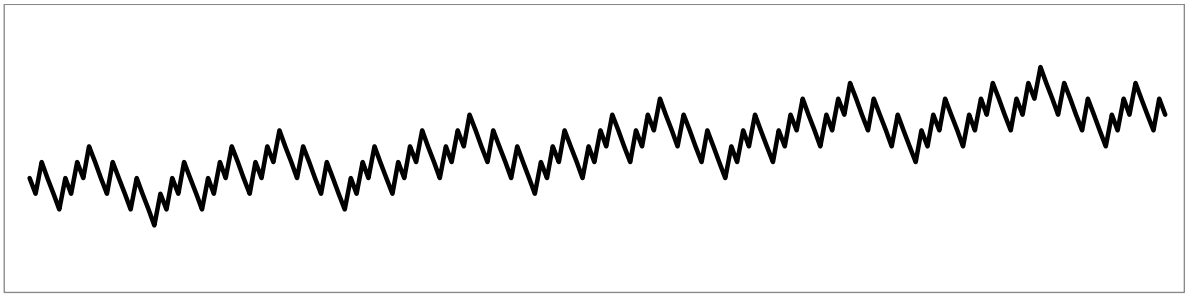


Рис.1. Временной ряд с нулевой дисперсией

Рис.2. Временной ряд со стохастическими шумами

Представлен подход позволяет выполнять построения фрактальных временных рядов, порождаемых использованием правил L-систем.

Литература

1. Шинкаренко В. И. Конструктивно-продукционные структуры и их грамматические интерпретации. I. Обобщенная формальная конструктивно-продукционная структура. / В. И. Шинкаренко, В. М. Ильман // Кибернетика и системный анализ. – 2014. – №5.– С.8-16.

2. Шинкаренко В. И. Конструктивно-продукционные структуры и их грамматические интерпретации. II. Уточняющие преобразования / В. И. Шинкаренко, В. М. Ильман // Кибернетика и системный анализ. – 2014. – № 6. – С. 15-28.

3. Aristid Lindenmayer, «Mathematical models for cellular interaction in development» *J. Theoret. Biology*, 18:280—315, 1968.

УДК 681.5; 656.2

№ держреєстрації 0117U004392

Інв. №_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Дніпропетровський національний університет залізничного транспорту
імені академіка В. Лазаряна

49010, м. Дніпро, вул. Лазаряна,2
тел. +38(056)776-84-98

ЗАТВЕРДЖУЮ
Ректор університету
д.т.н., професор

_____ О.М. Пшінько

ЗВІТ

ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ

«ПІДВИЩЕННЯ КОНКУРЕНТОСПРОМОЖНОСТІ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ НА ОСНОВІ УНІФІКОВАНИХ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ПРОЦЕСІВ ПЕРЕВЕЗЕНЬ ТА ЕКСПЛУАТАЦІЇ ПАРКІВ ТЕХНІЧНИХ СИСТЕМ»

№95.03.17.18
(заключний)

Керівник НДР №95.03.17.18

Зав. кафедрою «Прикладна математика»
професор, д.ф-м.н. С.О. Пічугов

Рукопис закінчено «21» грудня 2018 р.

Результати цієї роботи розглянуто Вченою радою ДНУЗТ
протокол №6 від «26» грудня 2018 р.

м. Дніпро
2018 рік

СПИСОК АВТОРІВ

1.	Керівник НДР, зав. кафедрою ПМ, професор, д.ф-м.н.		Пічугов С.О. (Реферат, вступ, висновки, розділ 6 пункт 6.6)
2.	Головний науковий співробітник, зав. кафедрою ЕОМ, професор, д.т.н.		Жуковицький І.В. (Реферат, вступ, висновки, розділ 1, 2 пункт 2.1-2.3)
3.	Головний науковий співробітник, декан фак-ту ТК, професор, д.т.н.		Скалозуб В.В. (розділ 1, розділ 2 пункт 2.1, розділ 4 пункт 4.1-4.4, розділ 6 пункт 6.1- 6.3, 6.7)
4.	Головний науковий співробітник, зав. кафедрою КІТ, професор, д.т.н.		Шинкаренко В.І. Розділ 5 пункт 5.2.,5.4
5.	Головний науковий співробітник, професор кафедри ЕОМ, д.т.н.		Косолапов А.А. (розділ 7 пункт 7.1-7.6)
6.	Провідний науковий співробітник, доцент кафедри ЕОМ, к.т.н.		Пахомова В.М. (розділ 8 пункт 8.1-8.5)
7.	Провідний науковий співробітник, доцент кафедри КІТ, к.т.н.		Ільман В.М. (розділ 5 пункт 5.1,5.3)
8.	Фахівець 3 категорії, ст. викладач кафедри ЕОМ		Івін П.В. (розділ 7 пункт 7.5,7.6)
9.	Фахівець 3 категорії, ст. викладач кафедри ПМ		Максименкова Ю.А. (розділ 6 пункт 6.6)
10.	Ст. викладач кафедри ЕОМ		Дзюба В.В. (розділ 2 пункт 2.2)
11.	Ст. викладач кафедри КІТ		Панік Л.О. (розділ 4 пункт 4.1-4.4)
12.	Асистент кафедри ЕОМ		Клюшник І.А. (розділ 6 пункт 6.5)
13.	Асистент кафедри ЕОМ		Заєць О.П. Розділ 3 пункт 3.1-3.5
14.	Асистент кафедри ЕОМ		Лобода Д.Г. (розділ 7 пункт 7.6)
15.	Асистент кафедри КІТ		Куроп'ятник О.С. (розділ 5 пункт 5.4)
16.	Асистент кафедри КІТ		Забула Г.В. (розділ 5 пункт 5.4)
17.	Асистент кафедри КІТ		Білий Борис Борисович (розділ 6 пункт 6.4)

18.	Асистент кафедри КІТ		Клименко Іван Вікторович (розділ 6 пункт 6.1, 6.2)
19	Магістрант кафедри ЕОМ		Мотиленко В.А. (розділ 8 пункт 8.4)
20.	Студент		Чигір Р.Р. (розділ 5 пункт 5.4)
21.	Студент		Жадан А.А. (розділ 5 пункт 5.4)

УДК 510.25+004.9+519.8
№ держреєстрації 0118U004215

Міністерство освіти і науки України
Дніпровський національний університет
залізничного транспорту імені академіка В. Лазаряна
49010, м. Дніпро, вул. Лазаряна, 2;
тел. +38(056) – 776 – 59 – 47, факс . +38(056) – 562 – 47 – 18 – 66

ЗАТВЕРДЖУЮ
Ректор університету
д.т.н, проф. Пшінько О.М.

ЗВІТ З НАУКОВО-ДОСЛІДНОЇ РОБОТИ «КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ ФРАКТАЛІВ»

Керівник: доцент кафедри «Комп'ютерні інформаційні технології», к. т. н. К.В. Литвиненко

Керівник науково-дослідної роботи
«Конструктивно-продукційне моделювання фракталів»
доцент кафедри «Комп'ютерні інформаційні технології» (КІТ),
к.т.н., доц. каф. КІТ Литвиненко К. В.

Відповідальний виконавець
д.т.н., проф., зав. каф. КІТ Шинкаренко В.І.
Виконавці: ас. каф. КІТ Васецька Т. М., студенти Чигирь Р.Р., Жадан А.А.,
Сансієва І.М.
Виконання розділів:

№	Розділ	Виконавці
1	Конструктивні фрактали, та методи їх моделювання.	к.т.н. Литвиненко К.В. д.т.н. Шинкаренко В.І. ас. каф. Васецька Т. М.
2	Моделювання геометричних фракталів на основі L-систем.	д.т.н. Шинкаренко В.І. к.т.н. Литвиненко К.В. ст. Чигирь Р.Р., ст. Жадан А.А. ст. Сансієва І.М.
3	Моделювання часових рядів на основі L-систем. Множинна інтерпретація для конструктивно-продукційного моделювання фракталів. Застосування конструктивно-продукційного підходу для моделювання активності блискавок грозового фронту.	д.т.н. Шинкаренко В.І. к.т.н. Литвиненко К.В. ас. каф. Васецька Т. М. ст. Чигирь Р.Р., ст. Жадан А.А. ст. Сансієва І.М.