Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

«Хмарна система MathLearning: : робочий зошит та функціональне меню» 122

Керівник курсової роботи
професор Малашонок Г. І.
_____
(підпис)
“____”_____2024 р.

Виконав студент 3 р. н.
Іванічок О. Р.
“____”_____2024 р.

Київ 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. Кафедри мережних технологій,
професор, д.ф-м.н
_____Малашонок Г. І.
(підпис)
„_____”_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Іванічку Олексію Руслановичу

3-го курсу факультету інформатики

ТЕМА: Хмарна система MathLearning: :

робочий зошит та функціональне меню

Вихідні дані:

Зміст ТЧ до курсової роботи

Вступ

Анотація

1. Аналіз предметної області
2. Вибір технологій та засобів для розробки
3. Концепція ігрового штучного інтелекту в шахах
4. Реалізація застосунку

Висновки

Список використаних джерел

Дата видачі „___”_____2024 р. Керівник _____

Завдання отримав _____

Календарний план виконання курсової роботи

Тема: Хмарна система MathLearning: робочий зошит та функціональне меню

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	Жовтень 2023 р.	
2.	Аналіз предметної області	Листопад 2023 р.	
3.	Вибір концепції додатку	Січень-лютий 2024 р.	
4.	Розробка додатку	Березень 2024 р.	
5.	Написання текстової частини курсової	квітень 2024 р.	
6.	Оформлення презентації для захисту	травень 2024 р.	
7.	Здача роботи для перевірки на плагіат	14 травня 2024 р.	
8.	Захист курсової роботи	20 травня 2024 р.	

Студент: Іванічок О.Р.

Керівник: Малашонок Г. І.

“_____” _____

Зміст

Вступ

У сучасному світі, де роль точних наук зростає з кожним днем, значно збільшується потреба в ефективних інструментах для вирішення складних математичних задач. Розробка програмного забезпечення, яке може спростити та автоматизувати процес обчислення великої кількості складних математичних виразів, є важливим кроком у напрямку підтримки наукових, освітніх та інженерних діяльностей.

Ця курсова робота присвячена розробці програмного рішення для Хмарної системи MathLearning: : робочий зошит та функціональне меню, дальше у тексті буде згадувати під назвою "Mathpar" - калькулятора складних математичних виразів, призначеного для студентів, викладачів, інженерів та науковців.

Метою роботи є створення потужного інструмента з великим функціоналом і зручним інтерфейсом, який здатен ефективно розв'язувати математичні задачі.

У першому розділі курсової роботи представлено огляд сучасних технологій і існуючих рішень у цій області, аналіз їхніх переваг та недоліків та вибір технологій для реалізації застосунку.

Другий розділ описує технічну реалізацію "Mathpar", включаючи архітектурні особливості, використовувані технології та підходи до розробки.

У третьому розділі описано функціонал «Mathpar».

Анотація

Курсова робота присвячена розробці фронтенд веб-застосунку калькулятора Mathpar для розв’язання математичних задач за допомогою TypeScript і фреймворку Angular.

Додаток надає зручний інтерфейс вводу і відображення математичних виразів за допомогою поля для вводу і бічної панелі інструментів на якій розміщені всі важливі функції.

1. Вибір технологій та засобів для розробки

1.1. Мова програмування Typescript

Єдиною мовою програмування, що запускається в браузері є JavaScript. Але за останній час дуже великою популярності набрав TypeScript, який використовується при розробці, і компілюється в JS. Його основні переваги:

1. Строга типізація. TypeScript додає строгу типізацію до JavaScript, тепер ми маємо визначити тип кожної змінною, що дозволяє виявляти помилки ще до виконання коду.
2. Підтримка ООП. Typescript додає ООП до Javascript. Хоч і в стандарті EcmaScript 6 добавили можливість писати ООП код на JS, але таких функцій як: модифікатори доступу, інтерфейси там не вистачає.

1.2. Фреймворк Angular

На момент 2023 року фреймворки для розробки великих фронтенд застосунків стали звичайною практикою. Angular став одиним з найбільш надійних, зручних рішень. Його переваги:

1. Структурований код. Angular дозволяє створювати додатки, які є добре структурованими та легко масштабованими завдяки своїй компонентній архітектурі. Це особливо корисно для складних проектів, які можуть рости і розвиватися з часом.
2. Typescript. Angular з коробки використовує Typescript, тому що сам написаний на ньому.
3. Розширені можливості. Angular надає широкий набір функцій, таких як двосторонній байндінг, директиви, сервіси і впровадження залежностой (dependency injection), які можуть спростити розробку і управління станом додатка.
4. Angular CLI. Angular має власний інтерфейс командного рядка - Angular CLI, який спрощує створення компонентів, сервісів та інших

функціональних елементів.

5. Реактивність. Angular написаний з використання бібліотеки для реактивного програмування RxJS, що значно полегшує написання асинхронних операцій завдяки реалізованому механізму Observable.
6. ООП підхід. В Angular використовується ООП підхід до написання програм.
7. Розроблений Google. Angular створено компанією Google, що означає якісний функціонал і довго підтримку продукту.

1.3. Бібліотека компонентів Angular Material

1. Консистентність інтерфейсу. Angular Material пропонує набір готових до використання компонентів UI, які дотримуються принципів Material Design від Google. Це дозволяє швидко створювати візуально привабливий і консистентний інтерфейс користувача.
2. Інтеграція з Angular. Як продукт, що розробляється командою Angular, Angular Material легко інтегрується з Angular-додатками, забезпечуючи плавність використання та велику сумісність із різними версіями Angular.
3. Широкий вибір компонентів. Angular Material включає широкий спектр компонентів, таких як кнопки, карточки, форми, списки, меню та багато іншого, що спрощує процес розробки та знижує потребу в створенні власних компонентів з нуля.

1.4. Mathjax-angular

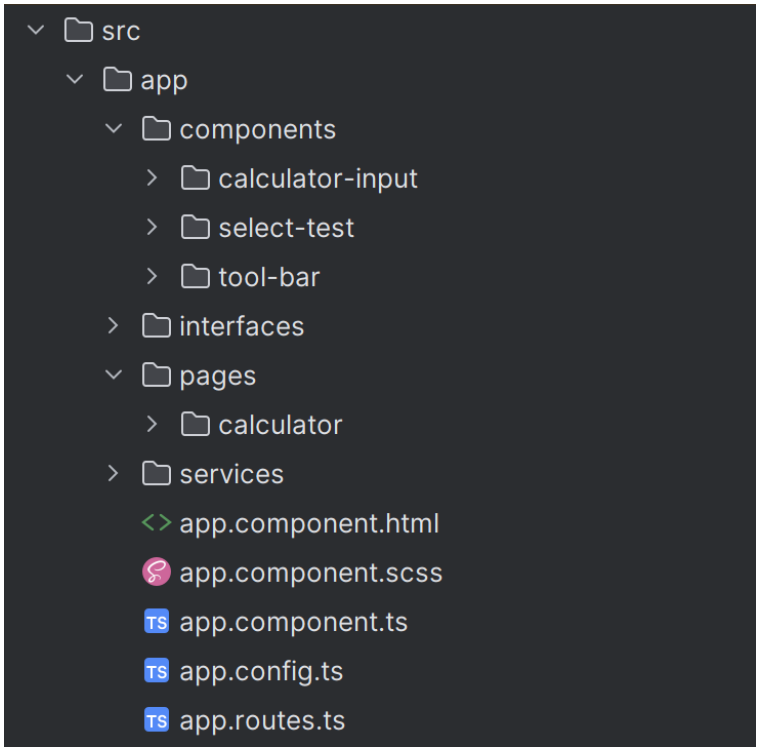
1. Підтримка складних математичних формул: MathJax є однією з найпопулярніших бібліотек для відображення складних математичних формул в браузері. MathJax-angular дозволяє легко використовувати mathjax в середині компонетів Angular.
2. Сумісність з LaTeX. MathJax добре підтримує LaTeX, що є стандартом для написання складних математичних формул.
3. Кросплатформеність. MathJax працює в усіх сучасних веб-

браузерах без необхідності встановлення додаткових плагінів або програмного забезпечення.

2. Реалізація застосунку

2.1 Структура проєкту

Даний проект реалізований за принципами які надає фреймворк Angular.



Проект розбивається на компоненти, у яких записна логіка для відображення інтерфейсу. Більш складна логіка для обміну даними виноситься в сервіси.

2.2 Підхід для розробки перевикористовуваних компонентів.

Якщо є якийсь фрагмент коду, що відповідає за відображення і може бути використаний більше одного разу його потрібно винести в окремий компонент. Компонент в Angular складається з трьох файлів: component.html – темплейт для відображення, component.ts логіка компоненту, component.scss – стилі компоненту. Передаємо всі необхідні вхідні дані в дочірний компонент в темплейті батьківського компоненту використовуючи декоратор @Input(). А якщо потрібно з дочірнього компоненту передати дані в батьківський використовуємо декоратор @Output(). Наприклад мій компонент для поля вводу калькулятора. Використання його

```
<app-calculator-input [expressionData]="input.expressionData" [isActive]="input.isActive"
                      (setActive)="setActive(input.id)" (deleteInput)="deleteInput(input.id)"
                      (addInput)="addInput(input.id)"
/>
```

Сам компонент

```
export class CalculatorInputComponent implements OnChanges {
  @ViewChild('input') inputRef!: ElementRef<HTMLTextAreaElement>;
  @Input() expressionData!: ExpressionData;
  @Input() isActive!: boolean;
  @Output() setActive : EventEmitter<void>  = new EventEmitter<void>();
  @Output() deleteInput : EventEmitter<void>  = new EventEmitter<void>();
  @Output() addInput : EventEmitter<void>  = new EventEmitter<void>;
```

Таким чином ми один раз створюємо компонент і потім коли нам потрібно його використати передаємо необхідні дані всередину для його використання.

2.3. Обмін даними між компонентами

Якщо компоненти знаходяться на рівні батько – дитина, обмін даними між ними реалізовується через механізм описаний вище. Але що робити якщо компоненти знаходяться на різних рівнях один від одного, але потрібно передати дані. В Angular ця задача вирішується за допомогою сервісів. Це спеціальний клас який створюється за допомогою декоратора @Injectable за паттерном Singleton, якщо передати в декоратор параметр providedIn зі значенням ‘root’, то цей клас буде єдиний для цілої програми. Після чого за допомогою механізму Dependency Injection, ми можемо впровадити цей клас у конструкторі будь якого компоненту. Наприклад сервіс для передачі даних при кліку на кнопку на нашій панелі інструментів у поле для вводу, в якому ми створюємо Subject expression$, який також є Observable.

```
1+ usages  Oleksii *
@Injectable({
  providedIn: 'root'
})
export class InputEmitterService {
  expression$: Subject<ExpressionData> = new Subject();
}
```

Це означає що на нього можна підписатися, і кожен коли ми передаватимемо в нього нове значення, буде виконуватися потрібна нам дія.

```
constructor(private inputEmitterService: InputEmitterService) {
}
```

```
ngOnInit(): void {
  this.inputEmitterService.expression$.pipe(
    takeUntil(this.destroy$)
  ).subscribe( observerOrNext: value : ExpressionData  => {
    this.calculatorInputs[this.lastActiveIndex] ? this.calculatorInputs[this.lastActive
  })
}
```

2.4. Графічний інтерфейс

При запуску користувач бачить перед собою поле для вводу математичних виразів з кнопками управління зверху.

Перша кнопка «Run» відповідає за запуск обчислення.

Друга «Switch» за переключення між типом перегляду звичайного тексту і Mathjax.

Третя кнопка «Add» для створення ще одного поля для вводу.

Четверта «Close» для видалення поля для вводу.

Шоста кнопка «Clear» для очищення змінних з середовища виконання.

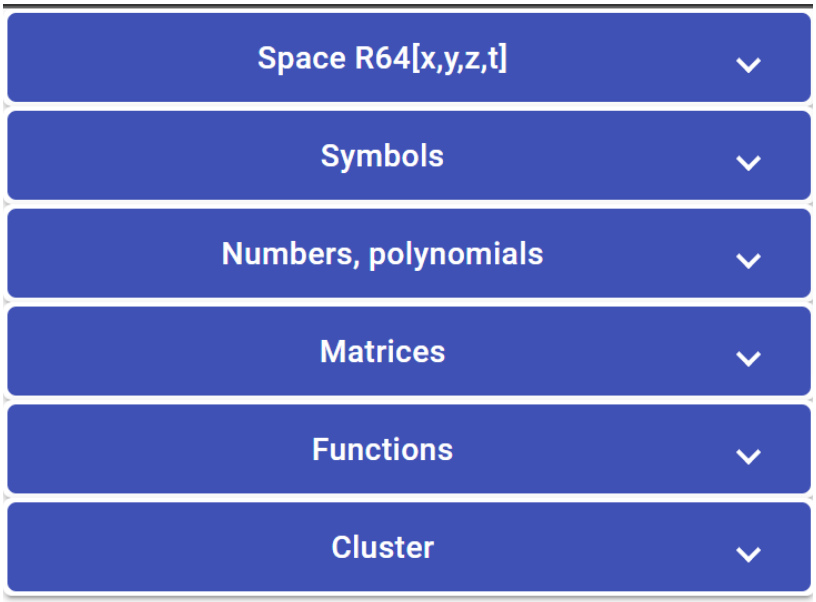
Шоста кнопка «Clean» для очищення поля для вводу.

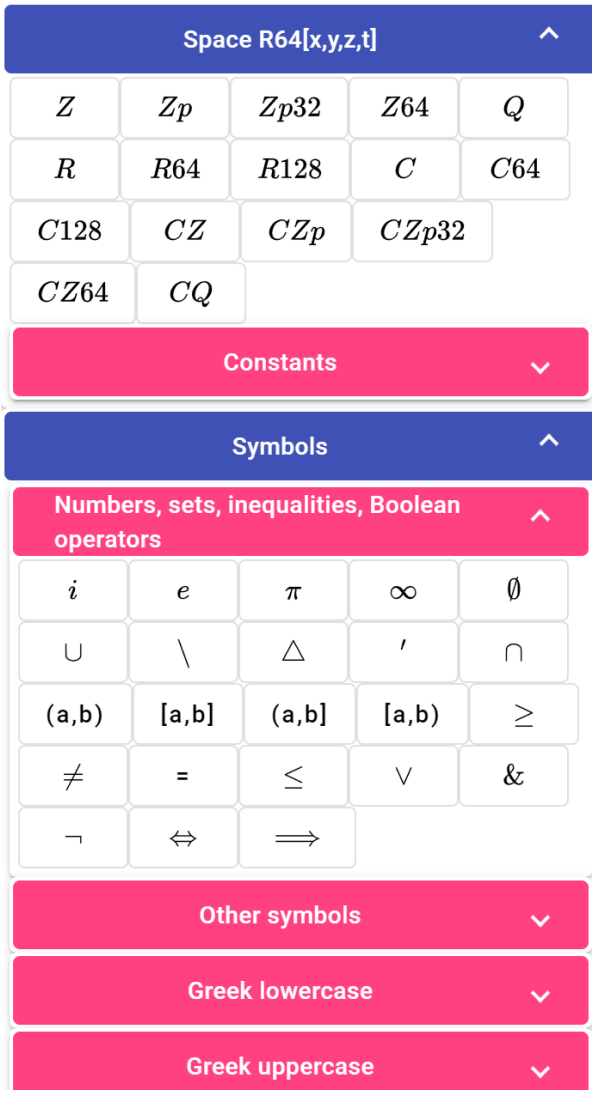
Наприклад запишемо рівняння

Ось в такому вигляді нам показується результат

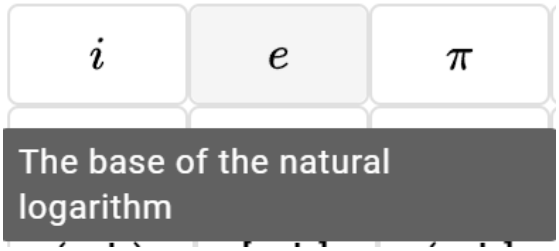
За допомогою кнопки «Switch» можемо повернутися до нашого запису

Зліва находиться панель з інструментами, яка зроблена у вигляді розділів з розгортаючими списками, які містять всередині кнопки, при натиснення на які буде вставленно відповідний вираз в поле для вводу.





При наведенні на кнопку буде відображенно підказку що робить ця функція



3. Функціонал Mathpar

Матеріал для цього розділу взятий з першої версії Mathpar https://mathpar.ukma.edu.ua/en/help/index.html, і перекладений українською мовою.

Mathpar допоможе Вам робити прості числові чи алгебраїчні операції.

Він допоможе Вам вирішувати завдання різних розділів математичного аналізу, алгебри, геометрії, завдання з фізики, хімії та інші.

Якщо ж Ви професійно застосовуєте математику, то він допоможе Вам позбавитися рутинних обчислень і оперувати з дуже великими математичними об'єктами. Mathpar дозволяє оперувати з функціями та функціональними матрицями, отримувати як точні чисельно-аналітичні рішення, так і рішення, в яких числові коефіцієнти виходять з необхідного рівня точності.

В основі мови Mathpar лежить широко використовується математиками та фізиками мова TeX, яку зазвичай використовують для набору математичних текстів.

3.1. Основи

Мова Mathpar, що є обєктом курсової роботи, може розглядатися як деякий розвиток мови TeX. Мова TeX призначена для запису математичних текстів та підготовки їх до публікації. Його можна вважати пасивним у порівнянні з мовою Mathpar, яка дозволяє робити обчислення, тобто є активною мовою математики. Як формулювання завдання, і результати обчислень, записуються мовою Mathpar.

Наприклад, матриця A, розміру 2×2, в Mathpar буде записано так:

```
A = [[a, b], [c, d]];
```

у TeX вона виглядає так:

```
A= \left(\begin{array}{cc}a&b\\c&d\end{array}\right).
```

У MathML це ще більш громіздкий вираз. Отриманий текст на мові TeX або MathML можна скопіювати та помістити в TeX- або html-файл та

використовувати для публікації. Крім того, можна отримати звичайне зображення та розмістити його у будь-якому документі. Це необхідно, наприклад, коли потрібно зберегти графік функції або розв'язання задачі.

3.1.1.Математичні функції

Прийнято такі позначення для елементарних функцій та констант.

Константи:

\i - уявна одиниця,

\e - основа натурального логарифму,

\pi - число π, тобто відношення довжини кола до діаметру,

\infty – знак нескінченності.

Функції одного аргументу:

\ln - натуральний логарифм,

\lg - десятковий логарифм,

\sin - синус,

\cos - косинус,

\tg - тангенс,

\ctg - котангенс,

\arcsin - арксинус,

\arccos - арккосинус,

\arctg - арктангенс,

\arcctg - арккотангенс,

\sh - синус гіперболічний,

\ch - косинус гіперболічний,

\th - тангенс гіперболічний,

\cth - котангенс гіперболічний,

\arcsh - арксинус гіперболічний,

\arcch - арккосинус гіперболічний,

\arcth - арктангенс гіперболічний,

\arccth - арккотангенс гіперболічний,

\exp - експонента,

\sqrt - корінь квадратний,

\abs - абсолютне значення для дійсних чисел, модуль для комплексного числа,

\sign – знак числа. Повертає 1, 0, -1, коли число позитивне, нуль або негативне, відповідно,

\unitStep(x) - це функція, яка при $x \geqslant 0$ набуває значення 1, а при $x<0$ набуває значення 0;

\fact - факторіал. Визначено для цілих позитивних чисел. Рівносильний запис - <<n!>>.

Функції двох аргументів

^ - Ступінь,

\log - логарифм від функції за вказаною основою,

\rootOf(x, n) - корінь ступеня n з x,

\Gamma - функція Гамма,

\Gamma2 - функція Гамма 2,

\binomial - Число поєднань.

3.1.2. Дії з функціями

Для перелічених вище функцій та його композицій можна обчислити значення функції у точці, підставити висловлювання функцію замість аргументів, обчислити межу функції, її похідну. Для цього визначено такі команди.

Для обчислення значення функції у точці необхідно виконати команду value(f, [var1, var2,…, varn]), де f - функція, а var1, var2, ..., varn - Значення відповідних змінних кільця. Для встановлення виразів у функцію необхідно виконати команду value(f, [func1, func2, …, funcn]), де f - це функція, func1, func2, ..., funcn - Функції, які підставляються замість відповідних змінних.

Для обчислення межі функції у точці необхідно виконати команду lim(f,

var), де f - це функція, а var - Точка, в якій потрібно знайти межу.

Для обчислення похідної функції f по змінній y з кільця Z[x,y,z] необхідно виконати команду D(f, y). Для знаходження змішаної похідної першого порядку функції f існує команда D(f, [x, y]), для знаходження похідної вищих порядків потрібно використовувати команду \D(f,[xˆk,zˆm,yˆn]) де k,m,n вказують, якого порядку відповідною змінною обчислюється похідна.

3.1.3. Розв'язання алгебраїчних рівнянь

Для вирішення рівнянь алгебри потрібно виконати команду solve. Нижче використовується команда налаштування оточення <<FLOATPOS=N>>. Вона встановлює число десяткових знаків після коми (N), які мають з'явитися під час виведення числового результату наближених обчислень. Вона пов'язані з процесом обчислень, лише з висновком. За замовчуванням FLOATPOS=2.

$SPACE = R64[x];$
$b = \mathbf{solve}(x^2 - 5x + 6 = 0);$
$out:$

$[3, 2]$

3.1.4. Вирішення систем алгебраїчних нерівностей

Для розв'язання систем алгебраїчних нерівностей необхідно виконати команду solve[In1, In2, ..., Ink], де [In1,In2,...,Ink] - Вектор нерівностей. Система може містити суворі та не суворі алгебраїчні нерівності. Відкритий інтервал позначається круглими дужками ( ), а закритий інтервал — квадратними дужками [ ], множина позначається фігурними дужками { }.

3.1.5. Операції на підмножинах дійсних чисел

Підмножина, що містить декілька інтервалів, можна задати так set((a,b),(c,d]) , де a,b,c,d - Числа. Тут інтервал позначається круглими дужками ( ), напіввідкритий інтервал - однією круглою і однією

квадратною дужкою [ ] або ( ), а відрізок - квадратними дужками [ ].

Прості підмножини позначаються такими ж дужками, але перед кожною дужкою необхідно додавати backslash (\). Наприклад \(3,4.5)\] або \[7,7\]. Оператор \set не вимагається.

$$SPACE = R64[x];$$
$$a = \mathbf{set}((-2,1),[2,5),(5.75,6],8);$$
$$out:$$

$$((-2),1) \cup [2,5) \cup (5.75,6] \cup \{8\}$$

3.1.6. Вектори та матриці

Для завдання вектора слід перерахувати його елементи у квадратних дужках. Так задаються векторні рядки. Для завдання матриці потрібно укласти квадратні дужки її вектор-рядки, розділені комами, наприклад, A=[[1,2],[3,4]] .

Підматрицю розміру Nr×Nc матриці A визначає команда $ \ backslash submatrix (A, r1, Nr, c1, Nc). r1,c1$ - це позиція верхнього лівого елемента.

Елемент матриці можна отримати, вказавши номер рядка і стовпця в нижніх індексах елемента матриці, а елемент вектора можна отримати вказавши один індекс. Наприклад, можна визначити елементи матриці так: a=\elementOf(A) , а потім звертатися до окремих елементів: a _{i,j }. Або визначити елементи вектора B так: b=\elementOf(B) , потім звертатися до них b _{i }.

Можна отримати рядок і матриці у вигляді векторного рядка: a _{i,? } або стовпець матриці j у вигляді вектор-стовпця: a _{?,j }. Імена некомутативних об'єктів, наприклад матриць та векторів, належить писати з першим символом backslash (\ ) та великої латинської літери, якщо передбачається їх використовувати у таких висловлюваннях, у яких не можна допускати перестановок. Наприклад \A∗\B−\B∗\A не призведе автоматично до нуля, на відміну від A∗B−B∗A що відразу

спроститься в 0.

Для позначення нульової та одиничної матриці використовуються великі літери \O та \I , у яких зазначено два індекси, що позначають число рядків та стовпців. За допомогою символу \I можна створювати прямокутні матриці будь-якого розміру, які мають елементи на головній діагоналі рівні 1 а інші елементи нульові. Наприклад, \I _{2,3 } та \O _{2,2 } позначають матриці (100010) та (0000) . Можна задавати нульові вектори, вказуючи в індексі кількість елементів: \O _{3 } позначає вектор [0,0,0] , а I _{3 } позначає вектор [1,0,0] .

Зазначимо, що як одновимірні і двовимірні масиви в мові Mathpar використовуються вектори і матриці, наприклад, On , On, m .

Вектор-стовпець може бути утворений транспонуванням вектор-рядки, наприклад, D=[7,2,3]T — це вектор-стовпець із трьох елементів. Крім звичайних арифметичних операцій (+,-,*) можна обчислювати функції векторів поелементно.

$$SPACE = Z[x];$$
$$A = \begin{pmatrix} x & 4 \\ y & 5 \end{pmatrix};$$
$$V = [x, y, 1, 2, x^6];$$
$$\mathbf{print}(A, V);$$
$$out:$$

$$A = \begin{pmatrix} x & 4 \\ y & 5 \end{pmatrix}$$

$$V = [x, y, 1, 2, x^6]$$

3.1.7. Створення випадкових елементів

Mathpar може створювати випадкові числа, поліноми та матриці. Це зручно, коли Вам потрібно створити якийсь довільний складний об'єкт або потрібно отримати багато випадкових об'єктів.

Створення випадкових чисел

Щоб отримати випадкове число, необхідно виконати команду randomNumber(k), де в аргументі k вказується кількість двійкових

розрядів запису випадкового числа. Це відповідає приблизно 0.3k десятковим цифрам.

Створення випадкових поліномів

Для того щоб створити випадковий поліном від s змінних, необхідно виконати команду randomPolynom(d1, d2, ..., ds, dens, bits), де dens - щільність полінома, а bits - кількість двійкових розрядів у записі випадкового числа, d1, d2, ..., ds означають старші ступені змінних. Якщо dens=100 , то буде отримано поліном, у якого всі коефіцієнти відмінні від нуля, всього (d1+1)(d2+1)..(ds+1) членів. Якщо dens<100 , то dens% будуть ненульові, а (100−dens)% нульових.

Створення випадкових матриць

Щоб отримати випадкову числову матрицю, необхідно виконати команду randomMatrix(m, n, dens, bits), де m - Кількість рядків у матриці, n - кількість стовпців матриці, dens - це щільність матриці у відсотках, bits - Число двійкових розрядів у записі числових коефіцієнтів.

Щоб отримати випадкову поліноміальну матрицю, необхідно виконати команду randomMatrix(m, n, dens, d1, d2, …, ds, polDens, polBits)), де m - Кількість рядків у матриці, n - кількість стовпців матриці, dens - Це щільність матриці, d1, d2, ..., ds - Найбільші ступені змінних поліномів, polDens - щільність поліномів, polBits - Кількість двійкових розрядів у записі коефіцієнтів поліномів.

3.2. Вибір оточення для математичних об'єктів

3.2.1. Оточення

Перш ніж буде заданий будь-який математичний об'єкт, число, функція або символ, має бути чітко визначено «оточення» - простір, в якому будуть визначатися об'єкти. У цьому розділі описуються методи завдання оточення. Переміщення з деякого оточення в поточне, як правило, повинно виконуватися явно за допомогою функції NewRing. У деяких випадках таке перетворення до поточного оточення відбувається автоматично.

Для вибору оточення визначається алгебраїчне простір змінних. Воно

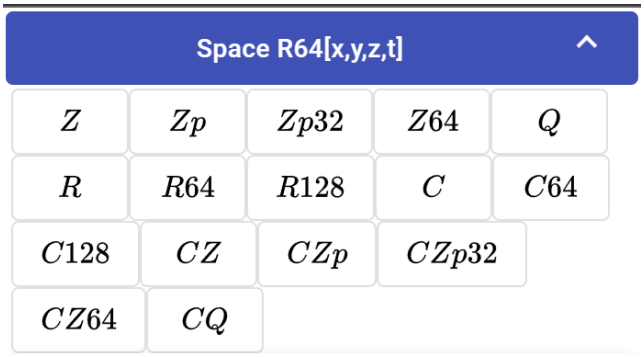
визначається іменами змінних та числовими просторами, в яких ці змінні набувають значення. Порядок змінних у списку змінних визначає лінійний порядок цих змінних. Зліва направо розташовуються змінні, упорядковані за старшинством від молодших до старших.

За умовчанням визначено простір R64 [x, y, z, t] чотирьох змінних, наймолодша - x, найстарша - t.

У будь-який момент користувач може змінити оточення, задавши новий простір алгебри змінних за допомогою команди установки <<SPACE=>>. Наприклад, для завдань обчислювальної математики може бути достатньо простору типу R64[x] або Q[x]. Команда установки: <<SPACE=R64[x];>> або <<SPACE=Q[x];>>, відповідно.

Якщо ім'я змінної починається із символу \ і великої літери (верхній регістр), то така змінна позначає елемент алгебри, у якої операція множення некомутативна, для решти змінних операція множення коммутативна.

3.2.2.Числові множини



Визначено такі числові множини:

Z - безліч цілих чисел Z,

Zp - кінцеве поле з p=MOD елементів Z/pZ, MOD - постійна,

Zp32 - кінцеве поле з p=MOD32 елементів Z/pZ, MOD32 менше 2^31,

Z64 - кільце цілих чисел z таких, що $-2^{63} \leqslant z < 2^{63}$,

Q - безліч раціональних чисел, R - безліч чисел з плаваючою точкою для зберігання наближених дійсних чисел з довільною мантисою,

R64 - безліч чисел з плаваючою точкою для зберігання наближених

дійсних чисел з подвійною точністю (зі стандартною 52-розрядною мантисою та окремим 11-розрядним полем для зберігання порядку), R128 - стандартні 64-бітні числа з плаваючою точкою для зберігання наближених дійсних чисел 52-розрядною мантисою та окремим 64-розрядним полем для зберігання порядку,

C - комплексний клас, утворений з класу R,

C64 - комплексний клас, утворений з класу R64,

C128 - комплексний клас, утворений з класу R128,

CZ - комплексний клас, утворений із класу Z,

CZp - комплексний клас, утворений із класу Zp,

CZp32 - комплексний клас, утворений з класу Zp32,

CZ64 - комплексний клас, утворений з класу Z64,

CQ - комплексний клас, утворений із класу Q.

Приклади простих поліноміальних кілець:

SPACE = Z[x, y, z];

SPACE = R64[u, v];

SPACE = C[x].

3.2.3.Визначення кількох числових множин

Дозволяється встановлювати алгебраїчні простори з декількох числових множин, наприклад, простір <<C[z]R[x, y]Z[n, m]>> дозволяє працювати з п'ятьма іменами змінних, визначених у множинах C, R та Z відповідно. Перша множина вважається основною і до неї будуть наводиться, при необхідності, решта змінних. В даному випадку це C.

Його можна розглядати як кільце поліномів п'яти змінних над C, при цьому воно має додаткові властивості. Якщо поліном не містить змінної z, то це поліном з коефіцієнтами R. Якщо поліном не містить змінних z, x, y, то це поліном з коефіцієнтами Z.

Приклади:

SPACE=Z[x, y]Z[u];

SPACE = R64[u, v]Z[a, b];

SPACE=C[x]R[y, z];

Кільце <<Z[x, y, z]Z[u, v, w]>>, в якому шість змінних розділені на дві групи, можна використовувати для завдань, у яких будуються поліноми, у яких коефіцієнти є поліномами або функціями інших змінних. Наприклад, характеристичний поліном для матриці над кільцем Z[x,y,z] буде отриманий як поліном з невідомою u коефіцієнти якого лежать в кільці Z [x, y, z].

3.2.4.Групові алгебри

%Групову алгебру означає символ G. Після нього стоїть список утворюючих, а перед ним %% простір, в якому діє група. Утворюючі групи некомутативні.

%Приклади вільних групових алгебр:

%SPACE=Z[x, y]G[U, V]; (утворюють U, V),

%SPACE=R64[u, v]G[A, B]; (утворюють A, B),

%SPACE=C[]G[X, Y, Z, T]; (Утворюють X, Y, Z, T).

%Кожен елемент алгебри є сумою термів з коефіцієнтами, що є функціями. %Наприклад, <<R64[x, y]G[X, Y, Z]>> - це вільна групова алгебра з трьома некомутативними %утворюючими X, Y, Z над функціями %в R64[x,y]. Тоді, наприклад, % A=(t2+1)X+sin(t)Y+3X2y3+(t2+1)XY3X2Y−2x2 - Елемент такої алгебри.

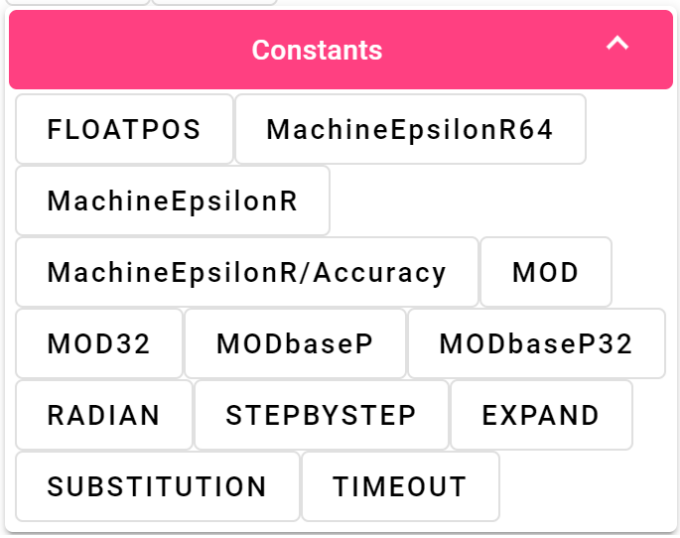
3.2.5.Ідемопотентні алгебри. Тропічна математика

Крім класичних числових алгебр з операціями <<+,~-,~*>> та операцією <</>> для полів, будуть доступні користувачеві та ідемпотентні алгебри. Для числової множини R64, можна буде використовувати алгебри R64MaxPlus, R64MinPlus, R64MaxMin, R64MinMax, R64MaxMult, R64MinMult. Для числової множини R, можна буде використовувати алгебри RMaxPlus, RMinPlus, RMaxMin, R64MinMax, RMaxMult, RMinMult. Для числової множини Z, можна буде використовувати алгебри ZMaxPlus, ZMinPlus, ZMaxMin, ZMinMax, ZMaxMult, ZMinMult.

$SPACE = ZMaxPlus[x, y];$
$a = 2; b = 9; c = a + b; d = a \cdot b;$ $\mathbf{print}(c, d)$
$out:$

$c = 9$
$d = 11$

3.2.6. Константи



Можна встановити або замінити такі постійні.

FLOATPOS – число десяткових знаків після коми, які виводяться на друк. За промовчанням приймається значення 2.

MachineEpsilonR — машинний епсілон для чисел типу R. За замовчуванням приймається значення 10-29. Число, модуль якого менше 10-29, вважається машинним нулем. Для встановлення нового значення 10-30 потрібно ввести команду <<MachineEpsilonR64=30>>.

MachineEpsilonR64 - машинний епсілон для чисел типу R64. За замовчуванням приймається значення 2-36. Число, модуль якого менше 2-36, вважається машинним банкрутом. Зазначимо, що числа R64 мають 52 розряди в мантисі. Для встановлення нового значення 2-48 потрібно ввести команду <<MachineEpsilonR64=48>>.

Постійна MachineEpsilonR (і MachineEpsilonR64) використовується при факторизації поліномів з коефіцієнтами типу R (або R64). Кожен коефіцієнт такого полінома попередньо ділиться на число

MachineEpsilonR (або MachineEpsilonR64) і округляється до цілого значення.

ACCURACY визначає кількість точних десяткових позицій після коми для чисел типу R та C в операціях множення та поділу. За промовчанням ACCURACY має значення MachineEpsilonR∗10−5. Якщо n<m, то команда <<MachineEpsilonR=n/m>> встановить одночасно MachineEpsilonR=10−n та ACCURACY=10−m.

MOD32 — модуль для простого поля, що не перевищує 231 (за промовчанням приймається значення 268435399). Просте число MOD32 – характеристика кінцевого поля. Константа MOD32 використовується в тому випадку, коли обчислення відбуваються в кінцевому полі Zp32 і вона повинна бути меншою від числа 231.

MOD – модуль типу Z для простого поля (за замовчуванням приймається значення 268435399). Просте число MOD - це характеристика кінцевого поля, але, на відміну від MOD32, у нього немає обмеження на абсолютне значення. Константа MOD використовується у разі, коли обчислення відбуваються у кінцевому полі Zp.

RADIAN може набувати значень 1 або 0. Якщо RADIAN = 1, то кути вимірюються в радіанах, інакше — у градусах. RADIAN=1 за промовчанням.

STEPBYSTEP може приймати значення 1 або 0. Якщо STEPBYSTEP = 1, виводитимуться проміжні результати обчислень. Типово STEPBYSTEP = 0.

EXPAND може набувати значення 1 або 0. Якщо EXPAND = 1, то у вхідному виразі будуть розкриватися всі дужки. За промовчанням EXPAND = 1.

SUBSTITUTION може набувати значення 1 або 0. Якщо SUBSTITUTION = 1, то у вхідному виразі будуть підставлятися замість імен виразів їх значення, якщо вони були визначені раніше. Типово SUBSTITUTION = 1.

3.3. Ряди

Ряд задається як f=\sum _{i=k} ˆ { \ infty} F(i, x, y, ..., z) , де i - Індекс підсумовування, k - Початкове значення i , F(i,x,y,…,z) - Функція багатьох змінних, тобто. F може залежати і від i .

Над рядами визначені такі арифметичні операції як додавання, віднімання та множення.

Нехай f та g - Ряди.

Для складання двох рядів необхідно виконати команду seriesAdd(f, g).

Для різниці двох рядів необхідно виконати команду seriesSubtract(f, g).

Для множення двох рядів необхідно виконати команду seriesMultiply(f, g).

Для розкладання функції до ряду Тейлора з певною кількістю членів ряду необхідно виконати команду teilor(f, point, num), де f - функція, point - Точка, num - Загальна кількість членів ряду.

$SPACE = R[x];$
$FLOATPOS = 15;$
$a = \mathbf{teilor}(\mathbf{sin}(x), 0, 7);$
$c = \mathbf{value}(a);$
$\mathbf{print}(a, c);$
$out:$

$$a = (1/1! \cdot x + 0/2! \cdot x^2 + (-1)/3! \cdot x^3 + 0/4! \cdot x^4 + 1/5! \cdot x^5 + 0/6! \cdot x^6 + (-1)/7! \cdot x^7)$$
$$c = -0.000198412698413x^7 + 0.008333333333333x^5 - 0.166666666666667x^3 + x$$

3.4. Функції однієї та кількох змінних

3.4.1. Обчислення значень функції у точці

Для обчислення значення функції у точці необхідно виконати команду value(f, [var1, var2,…, varn]), де f - функція, а var1, var2, ..., varn - Значення відповідних змінних.

Для тригонометричних функцій мірою кута вважається радіан чи градус. Вказівка міри кута визначається константою RADIAN. Якщо не вказувати кутову міру, то кутовим заходом вибирається радіан. Щоб змінити кутову міру з радіан на градуси, необхідно виконати команду <<RADIAN=0;>>. Якщо потрібно змінити кутову міру з градусів на радіани, потрібно виконати команду <<RADIAN=1;>>.

Якщо аргументами є цілі числа 15k та 18k градусів або πk/12 та πk/10

радіан ($k \in Z$) , то значення тригонометричних функцій є алгебраїчні числа.

3.4.2. Підстановка виразів у функції

Для обчислення композиції функцій потрібно «підставляти» у функцію замість її аргументів інші функції. Для цього необхідно виконати команду value(f, [func1, func2,…, funcn]), де f

- дана функція, func1, func2, ..., funcn
- Функції, які підставляються замість відповідних змінних.

3.4.3. Обчислення межі функції у точці

Для обчислення межі функції у точці необхідно виконати команду lim(f, var), де f - це функція, а var — точка, можливо нескінченна, у якій потрібно знайти межу, кінцеву чи нескінченну.

3.4.3.Диференціювання функцій

Для обчислення похідної функції f змінної y з кільця Z[x,y,z] необхідно виконати команду D(f, y). Обчислення третьої похідної з y можна виконати \D(f,[yˆ3]) . Якщо потрібно знайти похідну функції f один раз по першій змінній з поточного кільця (в даному випадку x) можна записати D(f) або D(f,x).

Для знаходження змішаної похідної першого порядку функції f існує команда D(f, [x, y]), для знаходження похідної вищих порядків потрібно використовувати команду \D(f,[xˆk,zˆm,yˆn]) де k,m,n вказують, якого порядку відповідною змінною обчислюється похідна.

3.5. Розв'язання диференціальних рівнянь та їх систем

3.5.1. Вирішення диференціальних рівнянь першого порядку

1. Встановити простір змінних (SPACE).
2. Задати рівняння та отримати рішення у функції (solveDE).

Рівняння з змінними, що розділяються:

$$SPACE = Q[x, y];$$
$$\mathbf{solveDE}(\partial_x(y) = (2/y)\mathbf{sin}(x)\mathbf{cos}(x));$$
$$out:$$

$$\{ y = \sqrt{(2 \cdot\ C - \ \mathbf{cos}(2x))}$$

Лінійне однорідне рівняння першого порядку:

$$SPACE = Q[x, y];$$
$$\mathbf{solveDE}(x\partial_x(y) = y - xe^{y/x});$$
$$out:$$

$$\{ y = (-x) \cdot\ \ \mathbf{ln}(\mathbf{ln}(\mathbf{abs}(x)) - C)$$

Рівняння у повних диференціалах:

$$SPACE = Q[x, y];$$
$$\mathbf{solveDE}((3x^2 - 3y^2 + 4x)\mathbf{d}(x) - (6xy + 4y)\mathbf{d}(y) = 0);$$
$$out:$$

$$\{ (((x^3 + 2x^2) + C) - (3y^2x + 2y^2)) = 0$$

3.5.2. Розв'язання диференціальних рівнянь

Для вирішення диференціального рівняння із постійними коефіцієнтами необхідно виконати такі кроки.

1. Встановити простір змінних (SPACE).

2. Встановити рівняння (systLDE).

3. Встановити початкові умови (initCond).

4. Отримати рішення (solveLDE).

$$SPACE = R64[t];$$
$$g = \{ \mathbf{d}(y, t, 3) + 3\mathbf{d}(y, t, 2) + 3\partial_t(y) + y = 1 \ \ ;$$
$$f = \begin{cases} \mathbf{d}(y, t, 0, 0) = 0 \\ \mathbf{d}(y, t, 0, 1) = 0 \\ \mathbf{d}(y, t, 0, 2) = 0 \end{cases} ;$$
$$h = \mathbf{solveLDE}(g, f);$$
$$\mathbf{print}(h);$$
$$out:$$

$$h = [(1 + (-1) \cdot\ e^{-t} + (-1) \cdot\ e^{-t} \cdot\ t + (-1) \cdot\ e^{-t} \cdot\ t^2/2)]$$

3.6. Поліноміальні обчислення

3.6.1. Обчислення значення полінома у точці

Для обчислення значення функції у точці необхідно виконати команду value(f, [var1, var2,…, varn]), де f — це поліном, який на позиції змінних кільця підставляємо відповідні значення var1,var2,…,varn.

$$SPACE = R[x, y];$$
$$f = x^2 + 5x(y^3 + x);$$
$$g = \mathbf{value}(f, [1, 2]);$$
$$\mathbf{print}(g);$$
$$out:$$

$$g = 46$$

3.6.2. Приведення поліномів до стандартного вигляду та розкладання поліномів на множники

Для приведення полінома до стандартного вигляду необхідно виконати команду expand(f), де f - це поліном.

Для розкладання полінома на множники необхідно виконати команду factor(f), де f - це поліном.

$$SPACE = Q[x, y];$$
$$f = (y^3 + x)^2(x + 1)^3;$$
$$g = \mathbf{expand}(f);$$
$$h = \mathbf{factor}(g);$$
$$\mathbf{print}(g, h);$$
$$out:$$

$$g = y^6x^3 + 3y^6x^2 + 3y^6x + y^6 + 2y^3x^4 + 6y^3x^3 + 6y^3x^2 + 2y^3x + x^5 + 3x^4 + 3x^3 + x^2$$
$$h = (x + 1)^3(y^3 + x)^2$$

3.6.3. Підсумовування полінома за змінними. Геометричні прогресії.

Для підсумовування полінома за змінними необхідно виконати команду SumOfPol(f, [x, y], [x1, x2, y1, y2]), де f - поліном, x, y - Змінні за якими ведеться підсумовування, x1, x2 - інтервал підсумовування x , y1,y2 - інтервал підсумовування по y.

Якщо інтервали підсумовування всім змінних збігаються, можна записати SumOfPol(f, [x, y], [x1, x2]), де x1,x2 - інтервал підсумовування x та y.

$$SPACE = R[x, y, z];$$
$$f = x^2z + xy + y^3xz;$$
$$res = \mathbf{SumOfPol}(f, [x, y], [2, 4, -2, 3]);$$
$$\mathbf{print}(res);$$
$$out:$$

$$res = 81$$

Для перетворення полінома за допомогою формули суми геометричної прогресії необхідно виконати команду SearchOfProgression(f). Ця команда шукає геометричну прогресію з найбільшим числом членів серед мономів полінома, потім робить це ще раз для членів, що залишилися, і так далі. Знайдені прогресії записуються як Sn=b1(qn−1)/(q−1)

де Sn

- Сума перших n членів, b1
- Перший член геометричної прогресії, q
- Знаменник прогресії.

$$SPACE = R[x, y, z];$$
$$f = x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{12} + x^{13};$$
$$g = x + x^5 + x^9 + x^{13} + xyz + 7x^2y^2z^2 + 7x^3y^3z^3 + 100xy + x + x^2 + x^3 + x^4;$$
$$f1 = \mathbf{SearchOfProgression}(f);$$
$$g1 = \mathbf{SearchOfProgression}(g);$$
$$\mathbf{print}(f1, g1);$$
$$out:$$

$$f1 = ((x^{14} - x^3)/(x - 1))$$
$$g1 = (((6z^3y^3x^3 + 6z^2y^2x^2 + 100yx + x^{13} + x^9 + x) + ((x^6 - x)/(x - 1) + (z^4y^4x^4 - zyx)/(zyx - 1))))$$

3.7. Матричні функції

Для обчислення транспонованої матриці для матриці A необхідно виконати команду transpose(A) або Aˆ T.

Для обчислення зворотної матриці для матриці A необхідно виконати команду inverse(A) або A(-1).

Для обчислення приєднаної матриці для заданої матриці A необхідно

виконати команду adjoint(A) або A^\star.

Для обчислення рангу матриці A необхідно виконати команду rank(A), а обчислення визначника матриці A — команду det(A).

Для обчислення сполученої матриці необхідно виконати команду conjugate(A) або A^\ast.

Для обчислення SVD-розкладання матриці необхідно виконати команду SVD(A). В результаті будуть обчислені три матриці [U, D, V]. Матриці U,V – унітарні, матриця D – діагональна: A=UDV.

Для обчислення узагальненої зворотної матриці Мурра-Пенроуза необхідно виконати команду genInverse(A) або A+.

Для обчислення ешелонної форми матриці A необхідно виконати команду toEchelonForm(A).

Для обчислення ядра оператора матриці A потрібно виконати команду kernel(A).

Для обчислення характеристичного полінома матриці A, елементи якої з R[x1,…,xm], необхідно задати кільце поліномів R[x1,…,xm]R[t] або R[t,x1,…,xm], в якому змінна t - це змінна, за якою будується поліном, і виконати команду charPolynom(A). Наприклад, якщо елементи вихідної матриці з кільця Z [x, y], можна вказати Z [x, y] Z [t] або Z [t, x, y].

3.8. Функції теорії ймовірностей та математичної статистики

Функції безперервних випадкових величин

Безперервна випадкова величина визначається за допомогою інтервалу (a,b) і щільності розподілу ймовірностей f(x). Наприклад, a=0;b=2;f=1/4∗x2.

Для безперервних випадкових величин визначено такі функції:

mathExpectation(a, b, f(x)) обчислює математичне очікування безперервної випадкової величини.

dispersion(a, b, f(x)) обчислює дисперсію безперервної випадкової величини.

meanSquareDeviation(a, b, f(x)) обчислює середнє квадратичне відхилення

безперервної випадкової величини.

plotDistributionFunction(a, b, F(x)) будує функцію розподілу безперервної випадкової величини, де F(x) — функція розподілу на (a,b).

Функції дискретних випадкових величин

Дискретна випадкова величина визначається як матриця, що має два рядки. У першому рядку записані значення випадкової величини, у другому відповідні їм ймовірності. Тобто, кожен елемент другого рядка є числом на відрізку [0 , 1], при цьому і сума всіх елементів другого рядка повинна дорівнювати 1 . Наприклад, DRQ=([1,2,3,4,5],[0.4,0.1,0.1,0.2,0.2]). Для дискретних випадкових величин визначено такі функції.

mathExpectation(DRQ) обчислює математичне очікування дискретної випадкової величини DRQ .

dispersion(DRQ) обчислює дисперсію дискретної випадкової величини DRQ .

meanSquareDeviation(DRQ) обчислює середнє квадратичне відхилення дискретної випадкової величини DRQ .

addQU(DRQ1, DRQ2) складає дві дискретні випадкові величини DRQ1 та DRQ2 .

multiplyQU(DRQ1, DRQ2) множить дві дискретні випадкові величини DRQ1 та DRQ2 .

covariance(DRQ1, DRQ2) обчислює коефіцієнт коваріації двох дискретних випадкових величин DRQ1 та DRQ2 .

correlation(DRQ1, DRQ2) обчислює коефіцієнт кореляції двох дискретних випадкових величин DRQ1 та DRQ2 .

plotPolygonDistribution(DRQ) будує багатокутник розподілу дискретної випадкової величини DRQ .

plotDistributionFunction(DRQ) будує функцію розподілу дискретної випадкової величини DRQ .

simplifyQU(DRQ) полегшує дискретну випадкову величину DRQ .

3.9. Оператори керування. Процедурне програмування

Система Mathpar дозволяє створювати свої процедури та функції. Для цього використовується команда procedure. Після команди вказується ім'я процедури та у фігурних дужках описується сама процедура.

```
\procedure myProc2() {
  d = 4;
  \print(d);
}
\procedure myProc(c,  d) {
  if (c < d) {
    \return d;
  } else {
    \return d+5;
  }
}
\myProc2();
a = 10;
c = \myProc(5 + a, a);
\print(a, c);
```

```
d = 4
a = 10
c = 15
```

Оператори розгалуження та циклів

Система Mathpar дає можливість використовувати оператори розгалуження та циклів.

if () {} else {} - Оператор розгалуження;

while () { } - Оператор циклу з передумовою;

for (; ;) { } - Оператор циклу з лічильником.

```
a = 5; b = 1;
if (b < a) {
  b = b + a;
} else {
  \print(a, b);
}
if (b < a) {
  b = b + a;
} else {
  \print(a, b);
}
```

```
a = 5
b = 6
```

Висновки

Результатом виконання цієї роботи став калькулятор Mathpar, за допомогою якого можна розв’язувати математичні задачі. Він завжди може стати вашим помічником, коли вам потрібно скористатися математикою: чи це вирішення завдання в школі, так і в університеті, виконання наукових розрахунків або рішення виробничого завдання.

В ході розробки було досліджено мову програмування TypeScript, фреймворк для фронтенд застосунків Angular, бібліотеку для реактивного програмування RxJS, бібліотеку компонентів Material.

Перспектив у цього додатку достатньо. Математика не обмежується наявним функціоналом калькулятора. Ще один з можливих варіантів розвитку – це інтеграція Mathpar в середину навчальної платформи.

Список використаної літератури

1) Перша версія Mathpar
   https://mathpar.ukma.edu.ua/
2) Посібник з мови Mathpar
   https://mathpar.com/downloads/MathparHandbook_en.pdf
3) Документація Angular
   https://angular.io/docs
4) Документація Material
   https://material.angular.io/
5) Документація MDN
   https://developer.mozilla.org/en-US/
6) Документація Mathjax
   https://docs.mathjax.org/en/latest/