Дрогобицький державний педагогічний університет імені Івана Франка

ІГОР ЛАЗУРЧАК, ТАРАС КОБИЛЬНИК

СИСТЕМА КОМП'ЮТЕРНОЇ МАТЕМАТИКИ MATHEMATICA

Навчальний посібник

для студентів спеціальності «Інформатика»

Дрогобич – 2009

УДК 51-37(075.8)
ББК 22.18я73
Л

Лазурчак І.І., Кобильник Т.П. Система комп'ютерної математики Mathematica. – Дрогобич:, 2009.– 60 с.

Посібник написано відповідно до програми навчальної дисципліни „Системи комп'ютерної математики", затвердженої для підготовки студентів освітньо-кваліфікаційного рівня „Бакалавр" спеціальності „Інформатика". У посібнику описані найважливіші команди, процедури та функції, а також математичні пакети, призначені для виконання символьних перетворень виразів, розв'язування різноманітних задач з математичного аналізу, лінійної алгебри, теорії диференціальних рівнянь, лінійного програмування. Описуються засоби 2D- та 3D- графіки, Особлива увага звертається на функціональне та процедурне програмування в системі Mathematica.

Бібліографія 12 назв

Рекомендовано до друку рішенням Вченої ради Дрогобицького державного педагогічного університету імені Івана Франка
Протокол № 10 від 16 жовтня 2008 року

Рецензенти: **Жалдак Мирослав Іванович**, заслужений діяч науки і техніки України, академік АПН України, доктор педагогічних наук, завідувач кафедри інформатики Національного педагогічного університету імені М.П.Драгоманова, професор;

Одрехівський Микола Васильович, завідувач кафедри економічної кібернетики та інноватики Дрогобицького державного педагогічного університету імені Івана Франка, кандидат фізико-математичних наук, доцент.

ВСТУП

Під комп'ютерною математикою розуміються апаратні та програмні засоби для автоматизованого виконання широкого класу математичних обчислень та математичного моделювання з допомогою комп'ютерної техніки. Комп'ютерна математика, як новий науковий напрям, зародилася на стику класичної математики та інформатики в середині 80-років XX сторіччя.

Термін „комп'ютерна математика" є узагальненням кількох раніше введених термінів, таких як символьна математика, комп'ютерна алгебра, обчислювальна математика, конкретна математика, математичне моделювання та комп'ютерне моделювання. По суті, мова йде про автоматизацію розв'язування математичних задач (включаючи моделювання) з допомогою персонального комп'ютера (ПК), тобто про комп'ютерну математику. При цьому використовується цілий ряд нових специфічних алгоритмів та методів розв'язування таких задач, які були породжені саме можливістю застосування сучасних комп'ютерів.

Системи комп'ютерної математики (СКМ) значно полегшують розв'язування типових математичних задач, таких як обчислення значень функцій та побудова їх графіків, розв'язування рівнянь, нерівностей та їх систем, обчислення інтегралів, знаходження похідних функції тощо. При цьому у користувача немає потреби у різноманітних довідниках та математичних таблицях, і разом з тим з'являються можливості у короткі терміни розв'язувати значну кількість математичних задач, готувати електронні книги. Використання СКМ стимулює інтерес студентів одночасно і до математики, і до новітніх інформаційних технологій та програмування.

Працюючи з СКМ, студент не тільки засвоює та застосовує методи математики, але й закріплює та суттєво розширює свої уміння в роботі з прикладними програмами. Паралельно він засвоює й основи програмування. Така унікальна інтеграція пізнавальних можливостей виключно корисна в умовах скорочення часу на вивчення математичних дисциплін у педагогічному ВНЗ.

Застосування СКМ в освіті позбавляє студентів від виконання рутинних обчислень, вивільняє час на обмірковування алгоритмів

розв’язування задач і побудови відповідних математичних моделей, подання результатів у найбільш зручній формі. Вивільнений час можна використати для більш глибокого вивчення математичної сутності задач і методів їх розв’язання. При цьому відкриваються нові можливості щодо гуманізації навчального процесу та гуманітаризації освіти, диференціації навчання відповідно до запитів, нахилів і здібностей студентів. Використання СКМ не тільки не позбавляє студентів вмінь розв’язувати математичні задачі, а навпаки, здатне суттєво їх поглибити.

Разом з тим не можна розв’язування задач повністю перекласти на ПК і обмежитись тільки аналізом отриманих результатів; використання ПК може допомогти у побудові графіків функцій, виконанні громіздких обчислень і перетворень; комп’ютер – це один із засобів, за допомогою якого може бути знайдений шлях до розв’язання, його використання не позбавляє від необхідності володіння певними знаннями стосовно методів розв’язування задач та уміннями їх застосовувати. Для ефективного застосування СКМ необхідні:

- знання математичної термінології;
- знання методів і засобів розв’язування задач;
- вміння правильно сформулювати задачу, яку повинен виконати комп’ютер;
- здатність передбачити результат;
- вміння контролювати правильність розв’язування задачі на проміжних етапах;
- вміння аналізувати і досліджувати отриманий результат.

Нинішній етап розвитку комп’ютерної математики характеризується підтримкою та просуванням нових напрямків в математиці, таких як методи розв’язування некоректних задач, нечітка логіка, нейронні мережі тощо. Широке розповсюдження отримали системи математичного моделювання природних та суспільних явищ, систем та пристроїв. Поступово у сферу СКМ проникають засоби реалізації віртуальної реальності, штучного інтелекту.

Сучасні СКМ – це перш за все потужні електронні довідники та бази даних зі всіх сучасних напрямів математики, ефективні засоби розв’язування

більшості математичних задач та засоби підготовки високоякісних електронних уроків, статей, книг.

Умовно СКМ можна поділити на такі:

1) системи для чисельних розрахунків;
2) табличні процесори;
3) матричні системи;
4) системи для статистичних розрахунків;
5) системи для спеціальних розрахунків; графічні та аудіо- засоби;
6) системи для аналітичних розрахунків (комп'ютерної алгебри);
7) універсальні системи.

Кожна з СКМ має певні особливості, які потрібно враховувати при розв'язуванні конкретних математичних задач. Завдяки реалізації СКМ на ПК вони стають все більш доступними для використання.

Кожна СКМ може мати нюанси у своїй архітектурі. Тим не менше, сучасні універсальні СКМ мають наступну типову структуру:

Центральне місце займає ядро системи. Воно являє собою безліч вбудованих функцій та процедур, які представлені у машинних кодах та забезпечують досить широкий набір вбудованих швидких команд, процедур, функцій та операторів. Роль ядра особливо велика у системах символьної математики, де в ядрі зберігаються надзвичайно багато правил аналітичних перетворень математичних виразів.

Інтерфейс дає можливість користувачеві звертатися до ядра зі своїми запитами і отримувати результат розв'язування на екрані монітора. Інтерфейс сучасних СКМ базується на ідеях операційних систем. Часто інтерфейс СКМ забезпечує можливість задавати та редагувати бібліотечні модулі та пакети розширень систем.

Функції та процедури, розміщені в ядрі, виконуються надзвичайно швидко. З цієї точки зору в ядро було б вигідно включати якнайбільше

обчислювальних засобів, проте це мимоволі призводить до уповільнення пошуку потрібних засобів у зв'язку із зростанням їх кількості, збільшенню часу завантаження ядра, інших небажаних наслідків. Тому обсяг ядра обмежують, але до нього додають бібліотеки процедур та функцій, які використовуються рідше. До них звертається користувач, якщо в ядрі не виявлена потрібна процедура чи функція. В деяких системах передбачено можливість модернізації бібліотек силами користувача.

Кардинальне розширення застосувань СКМ та їх адаптація до розв'язування задач користувачами досягається за рахунок пакетів розширень. Ці пакети, як правило, описані вбудованою мовою програмування тієї чи іншої системи. Пакети розширень можуть надаватися й у вигляді електронних книг, присвячених тим чи іншим напрямам прикладного застосування математики.

Довідкова система забезпечує отримання оперативних довідок з будь-яких питань при роботі з СКМ, з прикладами такої роботи. Вона містить і численний гіпертекстовий матеріал – математичні та фізичні таблиці, формули для знаходження похідних та інтегралів, алгебраїчних перетворень тощо. Ядро, бібліотеки, пакети розширень та довідкова система сучасних СКМ акумулюють знання в галузі математики, накопичені за тисячоліття її розвитку, і забезпечують їх швидке представлення користувачеві. Це відповідає їх призначенню в ролі експертних систем.

У всьому світі створені десятки СКМ, проте популярність отримали тільки деякі з них.

MuPAD (фірма-розробник SciFace Software GmbH & Co [11]) – СКМ початкового рівня. Вони орієнтовані на шкільну та вищу освіту за спеціальностями, що не вимагають розширеної математичної підготовки. В них недостатньо розвинені засоби графічної візуалізації отриманих результатів, хоча вони постійно покращуються.

Mathcad [4] (фірма-розробник Mathsoft Engineering & Education. Inc (раніше Mathsoft) [9]) – система, що орієнтована на вищу освіту, виконання складних чисельних та аналітичних розрахунків з максимальним використанням звичайної математичної мови представлення обчислень. Має

чудовий інтерфейс і потужні засоби графічної візуалізації обчислень, є найбільш масовою СКМ.

Слід зазначити, що однією з перших СКМ була **Macsyma** (потім **Maxima** [7]). Maxima серед СКМ відзначається досить широкими можливостями її використання при виконанні символьних обчислень. Це, по суті, єдина з вільно поширюваних відкритих систем, яка не поступається комерційним СКМ Mathematica та Maple. Сильні сторони Maxima – розвинений апарат лінійної алгебри та диференціальних рівнянь. Система орієнтована на прикладні розрахунки і не призначена для теоретичних досліджень в галузі математики. У зв'язку з цим у програмі відсутні або скорочені розділи, присвячені теоретичним методам (теорія чисел, теорія груп, математична логіка і т.п.). Головною перевагою Maxima перед іншими універсальними системами є те, що користувач має змогу аналітично та чисельно розв'язувати велику кількість різних типів рівнянь у частинних похідних.

Maple [2; 6] (фірма-розробник Maple Waterloo Inc [8]) – потужна обчислювальна система, призначена для виконання складних обчислень аналітичними та чисельними методами з власною мовою програмування. Система Maple містить (як і більшість СКМ) пакети розширень для розв'язування задач лінійної та тензорної алгебри, аналітичної геометрії, теорії чисел, теорій ймовірностей та математичної статистики, комбінаторики, інтегральних перетворень, лінійного та нелінійного програмування, теорії графів тощо. Особливе місце займає так званий „центр додатків Maple" (Maple Application Center, http://www.mapleapps.com), що містить багато програм для розв'язування задач з різноманітних галузей засобами системи Maple.

Mathematica [1; 3] (фірма-розробник Wolfram Research Inc [12]) – універсальна система, орієнтована на виконання аналітичних обчислень на будь-якому рівні. Містить власну мову програмування та пакети розширень, що значно розширюють можливості її використання для розв'язування спеціальних задач. Фірма-розробник підтримує електронний довідник

(http://mathworld.wolfram.com), у якому міститься велика кількість наукових та навчальних програмних продуктів..

MatLab [2; 5] (фірма-розробник MathWorks Inc [10]) – потужні та об'ємні системи, орієнтовані на матрично-чисельні методи, реалізацію чисельних обчислень підвищеної складності, математичне моделювання систем та пристроїв. Включають десятки пакетів розширень з різних галузей математики і багатьох сфер її застосувань.

Можна визначити наступні напрямки розвитку сучасних СКМ:

- перетворення СКМ в інтелектуальні системи подання знань та їх експертної оцінки;
- інтеграція систем одна з одною та деякими офісними та графічними програмами;
- розширені можливості обчислень, що охоплюють практично всі галузі застосувань математики;
- розширені засоби візуалізації обчислень;
- перетворення СКМ в універсальні системи;
- впровадження нових функцій, наприклад для реалізації нечіткої логіки, нейронних мереж тощо;
- впровадження в СКМ засобів, що дозволяють на їх основі створювати електронні підручники у різних форматах;
- можливість створення документів з текстами, формульними виразами, рисунками та графіками найвищої поліграфічної якості.

Перераховані можливості використання СКМ роблять їх однаково привабливими для наукових співробітників та інженерів, викладачів та студентів навчальних закладів і навіть для тих, хто просто захоплюється математикою.

Навіть невелика дослідницька проблема поділяється на кілька етапів, для виконання яких доводиться використовувати ПК. Це – постановка задачі та її уточнення, аналіз найпростіших моделей та ключових факторів, пробне дослідження, побудова чисельної моделі, опрацювання результатів. Перераховані етапи якраз і входять в так званий термін "обчислювальний експеримент", введений акад.А.А.Самарським.

При математичному моделюванні потрібно бути готовим до того, що для проведення запланованих робіт може виявитися недостатньо наявних знань та умінь і виникне потреба освоїти щось нове. Наперед невідомо, які труднощі виникнуть при розв'язуванні задачі і які засоби можуть допомогти розв'язати проблему.

Цікавим є дослідження, яке проводить С. Стейнхаус [13]. Деякі з результатів тестування СКМ (станом на 15 квітня 2008 року), наведених у роботі [13], подано у табл. 1, з якої видно, що кращими в середньому за всіма категоріями порівняння є системи Mathematica 6.0 (71,05%) та Matlab 2008a (69,58%), далі GAUSS 8.0 (52,11%) та Maple V11 (51,13%). При цьому за математичними характеристиками кращою є система Mathematica 6.0 (76,04%), значно відстають системи GAUSS (69,56%) та Matlab (68,79%).

Таблиця 1

Результати тестування СКМ

Програма (версія)	**GAUSS 8.0**	**Maple V11**	**Mathematica 6.0**	**Matlab 2008a**	**O-Matrix 5.2**	**Ox Prof. 3.2**	**Scilab 4.12**
Категорія порівняння	**%**	**%**	**%**	**%**	**%**	**%**	**%**
Інсталяція (15%)	35,41	87,54	96,27	76,52	41,18	34,36	39,69
Математичні операції (35%)	69,59	55,10	76,04	68,79	36,58	54,38	43,88
Графічні операції (10%)	60,86	60,88	84,63	88,49	47,14	43,96	51,32
Засоби програмування (11%)	62,70	50,81	64,86	72,43	41,62	72,43	62,16
Управління даними (5%)	67,43	64,06	76,03	72,77	49,89	55,43	53,71
Доступні операційні платформи (2%)	62,43	69,23	100,00	76,77	15,38	92,31	46,15
Швидкість обчислень (22%)	21,85	11,16	39,07	54,68	83,42	42,36	24,51
Загальний результат	**52,11**	**51,13**	**71,05**	**69,58**	**49,43**	**50,49**	**42,28**

Саме системі Mathematica – найкращій системі за всіма категоріями порівняння і присвячений навчальний посібник.

§ 1. СТАНДАРТНІ МАТЕМАТИЧНІ ФУНКЦІЇ ТА ОПЕРАЦІЇ

1.1 ОСНОВИ СИНТАКСИСУ ДЛЯ ЗАПИСІВ ВИРАЗІВ ТА КОМАНД

У системі Mathematica вказівки введення мають ідентифікатор **In[]**, а виведення – **Out[]** з відповідними номерами, які подаються у квадратних дужках. Імена комірок можна використовувати для побудови виразів. Вміст будь-якої вхідної комірки можна відредагувати, замінити частину або весь вхідний вираз.

Для подання виразів у системі Mathematica використовуються круглі, квадратні та фігурні дужки. Призначення круглих дужок – задавати порядок дій у математичних виразах. Квадратні дужки потрібні для запису аргументів функцій та команд. Фігурні дужки використовуються для формування списків (набору будь-яких об'єктів системи Mathematica). Знаком процента (**%**) позначається останній результат, два знаки процента (**%%**) – передостанній і т.д.

Використовуючи дужки, ідентифікатори змінних, константи, знаки арифметичних та інших операцій, складають вирази – основний об'єкт для багатьох команд.

Пріоритети виконання арифметичних операцій загальноприйняті. Для позначення операцій відношення використовуються знаки **>**, **<**, **>=**, **<=**, **==**, **!=** (не дорівнює), а для побудови логічних булевих виразів використовуються операції **Not, Or, And**. Логічними прийнято називати операції відображення логічних співвідношень між даними. В інформатиці логічні вирази (твердженнями) набувають двох значень – **True** (Істинно, або Так) та **False** (Хибно, або Ні). Слова **True** та **False** є символьними константами, через які визичаються результати логічних операцій в системі Mathematica.

Для виділення коментарів використовується структура **(*коментар*)**.

Знак рівності (**=**) використовується для надання значень змінним, а знак **:=** – відкладене надання значень (необчислене значення). Для відміни раніше наданого значення змінній **var** використовується запис **var=**. або команду **Clear[var]**. Для виконання локальних підстановок використовуються наступні конструкції:

expr/.var->value – у вираз **expr** замість змінної **var** підставляється значення **value**;

expr/.{var1->value1, var2->value2,...} – у вираз **expr** замість змінних **var1**, **var2...** підставляються значення **value1**, **value2**.... Для виконання глобальних підстановок використовується конструкція **var=a**; **expr** – у виразі expr змінна **var** замінюється на значення **a**.

Введення команди чи функції може закінчуватися крапкою з комою (;) або ніяким розділовим знаком. У першому випадку значення виразу обчислюється, проте не виводиться в полі виведення, у другому – значення виразу обчислюється і результат виводиться в полі виведення.

1.2 ДОВІДКОВА СИСТЕМА

Звернення до довідкової система дозволяє уточнити призначення будь-якої функції, оператора чи службового слова і поступово ознайомитися з можливостями використання пакету Mathenatica.

Довідкова база даних системи Mathematica орієнтована перш за все на отримання довідок про ту чи іншу функцію або команду чи деякий елемент інтерфейсу.

Для виклику довідкової системи призначений пункт *Help* головного меню. У вікні довідкової системи (див. рис.2.1) можна вибрати наступні розділи:

Build-in Functions – вбудовані функції;

Add-ons – пакети розширень;

The Mathematica Book – математичний довідник;

Getting Started/Demos – початок роботи та демонстрації;

Other Information – інші відомості;

Master Index – довідка за індексом (алфавітний каталог).

На рис.1 наочно показаний пошук відомостей про вбудовану функцію **Cos**, за якою обчислюється значення косинуса кута. При цьому можливий як прямий пошук за іменем (в даному випадку **Cos**), так і пошук за контекстом. Практично до кожної функції наведено кілька прикладів, які можна відкрити при зверненні до гіперпосилання у вигляді трикутника з надписом **Further**

Examples (спочатку приклади сховані). Приклади є „живими” в тому розумінні, що, не виходячи з системи допомоги, можна модифікувати вміст будь-якої комірки введення і тут же отримати новий результат. Також можна, виділивши комірки прикладів, скопіювати

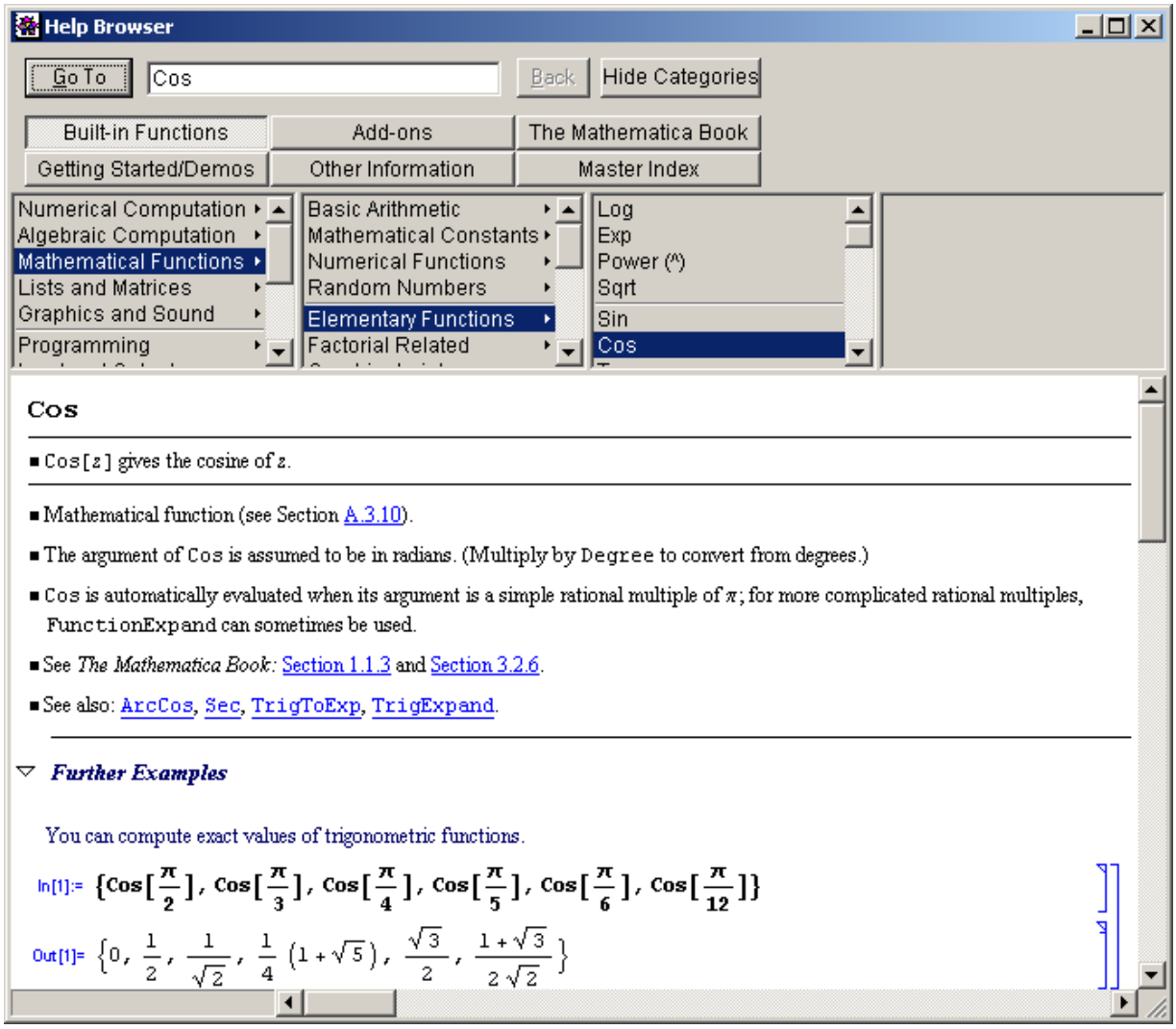


Рис.1

їх вміст в поточний документ, використовуючи буфер обміну. Такий приклад можна редагувати та використовувати для розв’язування своїх задач.

Система Mathematica містить більше десяти пакетів розширень **Add-ons**. Вони служать для розширення функціональних характеристик системи в таких галузях як алгебра, геометрія, наближені обчислення, дискретна математика, теорія чисел, математична статистика та ін. Доступ до команд пакетів розширень здійснюється оголошенням відповідного пакету:

<<NamePackage`

де **NamePackage** – ім’я пакету розширень.

Для доступу до довідки з пакетів розширень призначений розділ **Add-ons** довідкової системи.

The Mathematica Book – це електронний варіант книги **S.Wolfram** з відповідної версії системи. У книзі міститься багато суто довідкових відомостей (формули, графіки, приклади обчислень та ін.), вона є систематизованою інструкцією щодо застосування системи, в ній використовуються як вбудовані функції, так і функції з пакетів розширень.

Для демонстрації можливостей використання системи Mathematica служить електронна книга **Getting Started/Demos**, яка містить багато корисних прикладів застосувань системи Mathematica. Розділ **Other Information** присвячений, як правило, різноманітним відомостям про інтерфейс системи Mathematica.

Оперативну довідку під час роботи в системі про призначення деякої функції можна отримати, використовуючи наступні звернення:

?Name – довідка за заданим словом **Name**;

??Name – розширена довідка за заданим словом **Name**;

?Abc* – перелік всіх слів, які починаються з **Abc**;

Options[Name] – отримання відомостей про опції, пов'язані з об'єктом **Name**.

Наприклад, отримаємо відомості про функцію **Cos**:

```
?Cos
```

```
Cos[z] gives the cosine of z. More...
```

```
??Cos
```

```
Cos[z] gives the cosine of z. More...

Attributes[Cos] = {Listable, NumericFunction, Protected}
```

та про всі функції системи Mathematica, які у своїй назві містять буквосполучення **Cos:**

```
?Cos*
```

```
System`
Cos  Cosh  CoshIntegral  CosIntegral
```

Слід зазначити, що є два рівні коментарів оперативної довідки. Так, за допомогою звернення **?Cos** отримується тільки повідомлення про призначення функції. За допомогою звернення **??Cos** отримуються додаткові

відомості про ознаки функції. Вбудовані функції захищені від модифікації атрибутом **Protected**.

1.3 ТИПИ ДАНИХ

У системі Mathematica містяться кілька математичних констант (основні з них наведено у табл.2.1):

Таблиця 1. Математичні константи

Ім'я	Опис
E	Число e
Pi	Число π
I	Уявна одиниця ($\sqrt{-1}$)
Infinity	Нескінченість (∞)
Degree	Кількість радіан в одному градусі
GoldenRatio	$\left(1+\sqrt{5}\right)/2$
True, False	Булеві константи (істинно, хибно)

У системі Mathematica розрізняються наступні типи чисел: **Integer** (цілий), **Rational** (раціональний), **Real** (дійсний), **Complex** (комплексний). До типу **Integer** належать всі цілі числа (наприклад, -5, 6, 13, -66). До типу **Rational** належать всі числа виду $\frac{a}{b}$, де a і b – цілі числа ($b \neq 0$) (наприклад: $\frac{6}{7}, -\frac{9}{4}, \frac{-13}{66}$). До типу **Real** числа, в записі яких міститься мантиса (наприклад, 2.65, 6.9, -6.13, 52., 159.132). Будь-яка змінна (наприклад, **a**, **b**), якій не надано ніякого значення, належить типу **Symbol**. До цього ж типу належать і константи системи Mathematica (наприклад, **Pi**, **E**). Символи, що містяться в подвійних лапках, належать до типу **String** (наприклад, **"2"**, **"math"**). Набір символів, що містяться у фігурних дужках, відносяться до типу **List** (наприклад, **{1,2,5,a}**). У системі Mathematica не переводяться безпосередньо дані з одного типу до іншого. Для цього треба застосовувати відповідні вказівки.

За вказівкою **N[value,n]** число value подається як дійсне, що містить з **n** цифр. Наприклад, для обчислення значення виразу $\left(1+\sqrt{5}\right)/2$, запис якого містить 20 цифр, слід подати вказівку:

```
N[(1+Sqrt[5])/2,20]
```

```
1.6180339887498948482
```

За вказівкою **Rationalize[value,dn]** число **value** подається у вигляді дробу з заданою точністю **dn** . Наприклад, для подання числа π у вигляді дробу з точністю 0.001, слід подати вказівку:

```
Rationalize[Pi,0.001]
```

$$\frac{201}{64}$$

Для перетворення дійсних чисел в цілі використовуються наступні вказівки:

Ceiling[x] – повертається значення найменшого цілого числа, яке більше або рівне **x**;

Floor[x] – повертається значення найбільшого цілого числа, яке не перевищує **x**;

Round[x] – заокруглюється число **x** до найближчого цілого;

Слід зазначити, що аргументами функцій можуть бути списки чисел. Наприклад:

```
Ceiling[{-2.5,3.14,-13.15}]
```

```
{-2, 4, -13}
```

Для реалізації статистичних методів моделювання використовуються випадкові числа. Система Mathematica містить генератор псевдовипадкових чисел, доступ до яких забезпечується через наступні вказівки:

Random[] – повертається псевдовипадкове число типу **Real** з інтервалу [0,1];

Random[type,range] – повертається псевдовипадкове число вказаного типу **type** з заданого інтервалу **range**. До допустимих типів належать: **Real**, **Integer** та **Complex**. За замовчуванням приймається інтервал від **0** до **1**. Інтервал задається як список **{min, max}**, де **min** – початкове значення інтервалу, **max** – кінцеве значення інтервалу.

1.4 ФУНКЦІЇ ДЛЯ РОБОТИ З ЦІЛИМИ ЧИСЛАМИ

За допомогою кількох функцій забезпечується знаходження дільників цілих чисел та найменше спільне кратне:

Divisors[n] – повертається список натуральних дільників числа **n**;

GCD[n1,n2,...] – обчислюється найбільший спільний дільник цілих чисел **n1,n2,...** ;

LCM[n1,n2,...] – обчислюється найменше спільне кратне цілих чисел **n1,n2,....**

До цілочисельних функцій можна віднести функції обчислення факторіала та подвійного факторіалу:

Factorial[n] або **n!** – повертається значення факторіала числа (**n!=n*(n-1)*(n-2)*...*2*1**)

Factorial2[n] або **n!!** – повертається значення подвійного факторіала числа (**n!!=n*(n-2)*(n-4)***…)

Наступні функції використовуються для отримання простих чисел та деяких їх характеристик:

Prime[n] – обчислюється **n**-е просте число. Наприклад, за вказівкою **Prime[13]** обчислюється тринадцяте просте число **41**.

PrimePi[x] – відшукується кількість простих чисел, які не більші за **x**. Наприклад, за вказівкою **PrimePi[13]** обчислюється кількість простих чисел, які не більші за 13 (відповідь: 6).

За вказівкою **Mod[m,n]** відшукується остача від ділення націло числа **m** на число **n**. При цьому знак значення результату такий самий, як знак числа **n**.

1.5 ФУНКЦІЇ КОМПЛЕКСНОГО АРГУМЕНТУ

Аргументами елементарних функцій в системі Mathematica можуть бути як дійсне число **x**, так і комплексне **z**. Аргументи вказуються як параметри функцій у квадратних дужках.

Розглянемо функції для роботи з комплексними числами:

Abs[z] – повертається модуль комплексного числа **z**;

Arg[z] – повертається аргумент комплексного числа **z**;

Conjugate[z] – повертається комплексно спряжене число до числа **z**;

Im[z] – повертається уявна частина комплексного числа **z**;

Re[z] – повертається дійсна частина комплексного числа **z**;

Наведемо основні елементарні функції, вбудовані в ядро системи Mathematica:

ArcCos[z] – повертається значення арккосинуса комплексного аргумента **z**;

ArcCot[z] – повертається значення арккотангенса комплексного аргумента **z**;

ArcCoth[z] – повертається значення оберненого гіперболічного котангенса комплексного аргумента **z**;

ArcSin[z] – повертається значення арксинуса комплексного аргумента **z**;

ArcSinh[z] – повертається значення оберненого гіперболічного синуса комплексного аргумента **z**;

ArcTan[z] – повертається значення арктангенса комплексного аргумента **z**;

Cos[z] – повертається значення косинуса комплексного аргумента **z**;

Cosh[z] – повертається значення гіперболічного косинуса комплексного аргумента **z**;

Cot[z] – повертається значення котангенса комплексного аргумента **z**;

Coth[z] – повертається значення гіперболічного котангенса комплексного аргумента **z**;

Exp[z] – повертається значення exp(**z**);

Log[z] – повертається натуральний логарифм аргументу **z** (логарифм за основою e);

Log[b,z] – повертається логарифм аргумента **z** за основою **b**;

Sin[z] – повертається значення синуса комплексного аргумента **z**;

Sinh[z] – повертається значення гіперболічного синуса комплексного аргументу **z**;

Sqrt[z] – повертається значення квадратного кореня з аргумента **z**;

Tan[z] – повертається значення тангенса комплексного аргумента **z**;

Tanh[z] – повертається значення гіперболічного тангенса комплексного аргумента **z**.

Завдання до виконання лабораторної роботи „Основи синтаксису. Стандартні математичні функції та операції”

1. За допомогою генератора псевдовипадкових чисел задати тризначне та чотиризначне натуральне число. Знайти середнє арифметичне чотиризначного числа суму цифр тризначного числа.
2. Йде k-та секунда доби (задати k за допомогою генератора псевдовипадкових чисел). Визначити, скільки повних годин h, повних хвилин m ($m < 60$) і секунд s ($s < 60$) пройшло до цього моменту.
3. Поміняти місцями значення змінних *a*, *b*, *c* так, щоб *a* набуло значення *b*, *b* набуло значення набуло значення *c*, *c* набуло значення *a*, не використовуючи проміжних змінних.
4. Знайти мінімальне та максимальне з трьох чисел.
5. Задати натуральне число. Перевірити, чи число є ідеальним (ідеальним називається число, сума дільників якого дорівнює самому числу.).
6. Задати довільне число. Перевірити, чи число є членом арифметичної прогресії з першим елементом 10 і різницею 13.
7. Обчислити значення виразу $z = \dfrac{e^x \lg y + \sin(tgx)^{\arccos y}}{\left|x - ctg(\log_2(x+y))\right| + \cos(arstg(2x))}$ при $x = \dfrac{\pi}{4}$, $y = \dfrac{\pi}{2}$, використовуючи локальні та глобальні підстановки.
8. Заокруглити: а) число 3,14526 до тисячних; б) число 63541654 до сотень.

§2 РОЗВ’ЯЗУВАННЯ РІВНЯНЬ, НЕРІВНОСТЕЙ ТА ЇХ СИСТЕМ

Для аналітичного розв’язування рівнянь та їх систем використовується вказівка **`Solve`**, опис якої має вигляд **`Solve[eqns,var]`**. За цією вказівкою відшукуються корені рівняння (системи рівнянь) **`eqns`** відносно змінної (змінних) var. Результатом обчислення є список, елементами якого є одноелементні списки. Ці елементи є підстановками виду $\text{var} \to value_i$. Під виразом $value_i$ розуміється розв’язок рівняння (системи рівнянь). Наприклад, за допомогою команди **`Solve`** можна відшукати корені рівняння $x^2 - x - 6 = 0$ так:

```
eqns=x^2-x-6==0; (*задання рівняння*)
rozv=Solve[eqns,x] (*розв'язок рівняння*)
```

$$\{\{x \to -2\}, \{x \to 3\}\}$$

Якщо у рівняння **eqns** зробити підстановки, отримані за допомогою команди **Solve**, то якщо корені знайдено правильно, результатом буде **True**:

```
eqns/.rozv (*перевірка*)
```

$$\{\text{True}, \text{True}\}$$

Для того щоб отримати список коренів, досить зробити підстановку:

```
x/.rozv
```

$$\{-2, 3\}$$

Інше подання коренів рівняння (системи рівнянь) можна отримати за вказівкою **Roots**:

```
rozv1=Roots[eqns,x]
```

$$x == -2 \,||\, x == 3$$

Таке подання коренів рівняння можна перетворити до підстановок виду $\text{var} \to value_i$ за вказівкою **{ToRules}**:

```
{ToRules[rozv1]}
```

$$\{\{x \to -2\}, \{x \to 3\}\}$$

За вказівками **Solve** та **Roots** відшукуються розв'язки рівнянь (систем рівнянь) в аналітичному поданні, не даючи значень розв'язків при певних значеннях параметра (якщо такі входять в рівняння):

```
Solve[a*x^2+b*x+c==0,x]
```

$$\left\{\left\{x \to \frac{-b-\sqrt{b^2-4ac}}{2a}\right\}, \left\{x \to \frac{-b+\sqrt{b^2-4ac}}{2a}\right\}\right\}$$

За вказівкою **Reduce** відшукуються корені рівняння при певних значеннях параметрів:

```
Reduce[a*x^2+b*x+c==0,x]
```

$$x == \frac{-b-\sqrt{b^2-4ac}}{2a} \;\&\&\; a \neq 0 \;||$$

$$x == \frac{-b+\sqrt{b^2-4ac}}{2a} \;\&\&\; a \neq 0 \;||$$

$$a == 0 \;\&\&\; b == 0 \;\&\&\; c == 0 \;||\; a == 0 \;\&\&\; x == -\frac{c}{b} \;\&\&\; b \neq 0$$

Цей результат потрібно розуміти так: $x=\frac{-b-\sqrt{b^2-4ac}}{2a}$ при умові $a\neq 0$; $x=\frac{-b+\sqrt{b^2-4ac}}{2a}$, при умові $a\neq 0$; $x\in R$ при умові $a=0, b=0, c=0$; $x=-\frac{c}{b}$ при умові $a=0, b\neq 0$.

Системи рівнянь та змінні, відносно яких відшукуються корені, задаються у вигляді списку:

```
Solve[{x+y==2,x-y==0},{x,y}]
```

```
{{x → 1, y → 1}}
```

Для чисельного розв'язування рівняння (системи рівнянь) використовується вказівка **NSolve**, опис якої подібний до **Solve**. Проте не кожне рівняння (систему рівнянь) можна чисельно розв'язати за цією вказівкою:

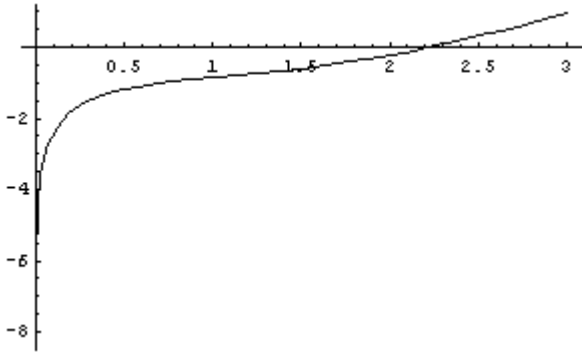
```
NSolve[Log[x]-Sin[x]==0,x]
```

```
Solve::tdep : The equations appear
   to involve the variables to be solved
   for in an essentially non-algebraic way.

NSolve[Log[x] == Sin[x], x]
```

У такому випадку при розв'язування трансцендентних рівнянь (систем рівнянь) застосовується вказівка **FindRoot**, яку зручно використовувати разом з вказівкою **Plot** – побудови двовимірних графічних об'єктів. Наприклад, знайдемо корені рівняння $\ln x-\sin x=0$:

```
Plot[Log[x]-Sin[x],{x,0,3}](*побудова графіка*)
```




```
FindRoot[Log[x]-Sin[x]==0,{x,2.2}](*відшукання кореня*)
```

```
{x → 2.21911}
```

У попередньому прикладі спочатку побудували графік функції $f(x)=\ln x-\sin x$ для того, щоб приблизно визначити, де міститься корінь. У команді **FindRoot**, першим аргументом якої є рівняння $\ln x-\sin x=0$, у

фігурних дужках вказується, відносно якої змінної відшукуються корені та початкове наближення кореня (у даному випадку вказано відносно змінної **x**, початкове наближення **2.2**).

Для розв'язування нерівностей використовується вказівка **InequalitySolve** з пакету **Algebra**:

InequalitySolve[expr,var] – розв'язується нерівність (система нерівностей) **expr** відносно змінної (змінних) **var**. Наприклад, розв'яжемо нерівність $x^2-5x-7>0$:

```
<< Algebra`InequalitySolve`
InequalitySolve[x^2-5*x-7>0, x]
```

$$x < \frac{1}{2}\left(5-\sqrt{53}\right) \;||\; x > \frac{1}{2}\left(5+\sqrt{53}\right)$$

Відповідь, яку отримали у системі Mathematica, слід розуміти так: $x \in \left(-\infty;\frac{1}{2}\left(5-\sqrt{53}\right)\right)\cup\left(\frac{1}{2}\left(5-\sqrt{53}\right);+\infty\right)$. Зауважимо, що у системі Mathematica5.1 нерівності можна розв'язувати також за допомогою вказівки **Reduce**.

За вказівкою **DSolve[expr,fun,vars]** відшукуються загальний розв'язок звичайних диференціальних рівнянь (системи рівнянь) **expr**, розв'язки задачі Коші та крайових задач відносно функції (функцій) fun, що залежить від змінної (змінних) **vars**. Якщо не вдається знайти розв'язок в аналітичному поданні, тоді використовується вказівка **NDSolve**, за якою наближений розв'язок відшукується за чисельними методами.

Завдання до виконання лабораторної роботи „Команди для розв'язування рівнянь, нерівностей та їх систем"

1. Розв'язати систему рівнянь $\begin{cases} x^2+y^2=5, \\ xy=-4. \end{cases}$. Зробити перевірку.
2. Студент прийшов до банкомату зняти 230 грн. У банкоматі є купюри номіналом 10 грн., 20 грн., 50 грн. Якими способами (кількість купюр по 10 грн., 20 грн., 50 грн.) студент може отримати 230 грн.?
3. При яких значеннях параметра a обидва корені квадратного рівняння $(a-2)x^2-2ax-a+3=0$ додатні?

4. При яких значеннях параметра a корені рівняння $4x^2-(3a+1)x-a-2=0$ належать інтервалу $(-1;2)$?

5. Знайти суму коренів рівняння $\ln\frac{x}{5}=\sin(4x)$ на проміжку $(5;10)$.

6. Знайти добуток найбільшого та найменшого коренів рівняння $|\sin(5x)|=\log_2 x$.

7. Розв'язати рівняння $x^2-5x+6=0$ та зробити перевірку. Знайти суму та добуток коренів цього рівняння, не використовуючи теореми Вієта.

§3 КОМАНДИ ДЛЯ РОБОТИ З ВИРАЗАМИ

Для спрощення виразів використовується функція **Simplify**:

Simplify[expr] – виконуються послідовно алгебраїчні перетворення виразу **expr** і повертається найпростіша з відшуканих форм.

Функція **Simplify** може використовуватися для спрощення різноманітних математичних виразів: многочленів, елементарних та спеціальних функцій, алгебраїчних, раціональних та тригонометричних виразів і т.д. Як правило, вираз зводиться до нормального вигляду, що автоматично означає і зведення до вигляду достатньо простих виразів.

Використання функції **FullSimplify** надає ширші можливості, ніж функції **Simplify**. Зокрема, забезпечується спрощення виразів, що містять спеціальні математичні функції. Наприклад:

```
Simplify[Gamma[x]*Gamma[1-x]]
```

```
Gamma[1 - x] Gamma[x]
```

```
FullSimplify[Gamma[x]*Gamma[1-x]]
```

```
π Csc[π x]
```

Як видно з наведеного прикладу, за вказівкою **FullSimplify** виконується спрощення навіть в тому випадку, коли за вказівкою функції **Simplify** цього не можна зробити.

Розглянемо вказівки, призначені для розкривання дужок у виразі:

ComplexExpand[expr] – розкриваються дужки у виразі **expr** за умови, що всі змінні є дійсними;

FunctionExpand[expr] – розкривають дужки у виразі **expr**, що містить спеціальні функції;

Expand[**expr**] – розкривають дужки у виразі **expr**, що містить добуток та додатні цілі степені;

ExpandAll[expr] – розкривають дужки у виразі **expr**, що містить добутки та цілочисельні степені;

ExpandDenominator[expr] – розкривають дужки у знаменнику виразу **expr**, що містить добуток та додатні цілі степені;

ExpandNumerator[expr] – розкривають дужки у чисельнику виразу expr, що містить добуток та додатні цілі степені.

За вказівкою **Colltect[expr, var]** зводяться подібні доданки у виразі **expr** відносно змінної **var**.

За вказівкою **Factor[expr]** вираз **expr** розкладається на множники. Слід зазначити, за вказівкою **Factor** вираз розкладається на множники над числовим полем, якому належать коефіцієнти полінома. Це означає, що якщо всі коефіцієнти цілі, то і в отриманих співмножниках будуть тільки цілі коефіцієнти.

За вказівкою **PolynomialDivision[p,q,var]** повертається список частки та остачі, отриманих в результаті ділення многочлена **p** на многочлен **q** як многочленів від змінної **var**.

За вказівкою **PolynomialQuotient[p,q,var]** відшукується частка від ділення многочлена **p** на многочлен **q** як многочленів від змінної **var**.

За вказівкою **PolynomialRemainder[p,q,var]** відшукується остача від ділення многочлена **p** на многочлен q як многочленів від змінної **var**.

За командами **PolynomialGCD[poly1,poly2,...]** та **PolynomialLCM[poly1,poly2,...]** відшукуються відповідно найбільший спільний дільник та найменше спільне кратне кількох многочленів **poly1,poly2,...**.

За вказівкою **Apart[expr]** дріб **expr** подається у вигляді суми елементарних дробів.

Зауважимо, що для виконання перетворень над тригонометричними виразами є кілька спеціальних вказівок. Призначення функцій **TrigExpand** та **TrigFactor** аналогічне до **Expand** та **Factor** відповідно.

Завдання до виконання лабораторної роботи „Команди для роботи з виразами”

1. Розкласти на множники наступні вирази x^3-8, x^3+9, x^3+9, $x^2+3x+15$.
2. Подати дріб $\frac{x^2-5x+6}{x^3-1}$ у вигляді суми елементарних дробів. Перевірити отриманий результат за допомогою команди **`Apart`**.
3. Використовуючи команди для перетворення виразів обчислити границю $\lim\limits_{x\to 1}\frac{x^4-3x+2}{x^5-4x+3}$.
4. Знайти частку та остачу від ділення многочлена $p(x)=x^4-16$ на многочлен $q(x)=x^2-13x+5$. Виконати перевірку.
5. Довести тотожність $\frac{8\cos 2x}{ctg^2x-tg^2x}=1-\cos 4x$, використовуючи команди для перетворення виразів для однієї з частин тотожності.

§4 СПИСКИ

4.1. Способи задання та дії зі списками

Список у системі Mathematica – це набір будь-яких елементів, поміщених у фігурних дужках. На відміну від системи Maple, у системі Mathematica не передбачено використання таких типів даних як множина (**`set`**) чи набір виразів (**`exprseq`**). Списки можна створювати безпосередньо, поміщаючи набір елементів у фігурні дужки або можна генерувати списки. Елементами списку можуть бути будь-які вирази, допустимі в системі Mathematica, в тому числі і списки (вкладені списки). Змінним можна надавати значення списку.

Для генерації списків часто використовується команда Table, за якою створюється таблиця-список:

`Table[expr,{imax}]` – утворюється список, що містить imax елементів виразу **`expr`**;

`Table[expr,{i,imax}]` – утворюється список значень виразу **`expr`**, що залежить від параметра **`i`**, $i=\overline{1,i\max}$;

Table[expr,{i,imin,imax}] – утворюється список значень виразу **expr**, що залежить від параметра **i**, $i = \overline{i\min, i\max}$;

Table[expr,{i,imin,imax,di}] – утворюєтья список значень виразу **expr**, який залежить від параметру **i**, що змінюється від значення **imin** до значення **imax** з кроком **di**;

Table[expr,{i,imin,imax,di},{j,jmin,jmax,dj},...] – утворює вкладений список. Зовнішнім є список за змінною **i**.

За вказівкою **Table** зручно задавати вектори та матриці. Наприклад:

```
Table[i^2,{i,-1,1,0.5}]
```

```
{1, 0.25, 0., 0.25, 1.}
```

```
Table[a^(i+j),{i,1,3},{j,1,3}]
```

$$\{\{a^2, a^3, a^4\}, \{a^3, a^4, a^5\}, \{a^4, a^5, a^6\}\}$$

Для створення списків використовується також вказівка **Range**:

Range[n] – створюється список **{1,2,..., n}**. Наприклад:

```
Range[5]
```

```
{1, 2, 3, 4, 5}
```

Range[m,n] – створюється список **{m,m+1,..., m+k·i}**, причому числа **m** та **n** можуть бути дробовими (**m+k·i<n<m+(k+1)i**). Наприклад:

```
Range[3.3,8.5]
```

```
{3.3, 4.3, 5.3, 6.3, 7.3, 8.3}
```

Range[m,n,di] – створюється список **{m,m+di,..., m+k·di}**, причому числа **m** та **n** можуть бути дробовими (**m+k·di<n<m+(k+1)·di**). Наприклад:

```
Range[2.1,4.5,0.5]
```

```
{2.1, 2.6, 3.1, 3.6, 4.1}
```

За вказівкою **Array[a,n]** утворюється список з елементів **a[i]**, $i = \overline{1,n}$. Ітератор заданий у вигляді **{n1,n2,...}** призводить до вкладених списків з елементами **a[i1,i2,...]**. Наприклад:

```
Array[a,{3,4}]
```

```
{{a[1, 1], a[1, 2], a[1, 3], a[1, 4]},
 {a[2, 1], a[2, 2], a[2, 3], a[2, 4]},
 {a[3, 1], a[3, 2], a[3, 3], a[3, 4]}}
```

Елементам списку можна надавати значення.

За виразом **list[[k]]** виокремлюється **k**-ий елемент зі списку **list**. За вказівкою **Length[list]** визначається довжина (кількість елементів) списку **list**.

Для додавання елемента **elem** до списку **list** використовуються команди **Append[list,elem]** та **Prepend[list,elem]**. За вказівкою **Append**

елемент **elem** ставиться на останнє місце у списку **list**, за вказівкою **Prepend** – на перше. Наприклад, додамо елемент **d** у кінець та на початок списку **{a,b,c}**:

```
Append[{a,b,c},d]
```

```
{a, b, c, d}
```

```
Prepend[{a,b,c},d]
```

```
{d, a, b, c}
```

Кожен елемент списку однозначно визначається номером: додатним, якщо рахунок проводиться зліва направо, і від'ємним, якщо справа наліво. За вказівкою **Insert[list,elem,k]** елемент **elem** вставляється у список **list** на **k**-те місце. Інші елементи зберігаються у списку. Наприклад, вставимо елемент **d** у список **{a,b,c}** на друге місце:

```
Insert[{a,b,c},d,2]
```

```
{a, d, b, c}
```

За вказівкою **ReplacePart[list,elem,k]** елемент зі списку **list** з номером **k** замінюється елементом elem. Наприклад, замінимо елемент **b** (з номером 2) у списку **{a,b,c}** на елемент **d**:

```
ReplacePart[{a,b,c},d,2]
```

```
{a, d, c}
```

Вилучення елементу з номером **k** зі списку **list** здійснюється за вказівкою **Delete[list,k]**. Зазначимо, що **k** може набувати значення списку **{k1,k2,...}** (список list багатовимірний), що призводить до вилучення елемента, який міститься внутрі **k1**–го елемента (списку), в ньому – внутрі **k2**–го елемента і т.д. Наприклад, зі списку **{{5,7},{3,4}}** вилучимо елемент з номером **{1,2}**, (другий елемент першого внутрішнього списку):

```
Delete[{{5,7},{3,4}},{1, 2}]
```

```
{{5}, {3, 4}}
```

За вказівкою **Drop[list,k]** зі списку вилучаються перші **k** елементів (якщо **k** додатне) або останні **k** елементів (якщо **k** від'ємне). При цьому якщо **k** задане у вигляді списку **{k1,k2}**, то будуть вилучені елементи з номерами від **k1** до **k2**. Наприклад:

```
Drop[{13,6,26,132,a},{2,4}]
```

```
{13, a}
```

За вказівкою **Count[list,patt]** визначається кількість елементів у списку **list**, що відповідають заданому шаблону **patt**. Наприклад, визначимо кількість комплексних чисел у списку **{I,3,0.2,3/5+6I}**:

```
Count[{I,3,0.2,3/5+6I},_Complex]
```
```
2
```

Для комбінування кількох списків використовуються вказівки:

Complement[A,B,C,...] – повертається список **A**, що не містить елементів із списках **B,C,...** (операція віднімання для множин);

Intersection[A,B,...] – повертається список елементів, що належать до всіх списків **A,B,...** (операція перетину для множин);

Union[A,B,...] – вилучаються елементи, які повторюються у списках, і повертається відсортований список всіх різних між собою елементів, що належать до списків **A,B,...** (операція об'єднання для множин).

За вказівкою **Sort[list]** елементи у списку **list** розташовуються наступним чином: спочатку числа в порядку зростання, далі букви в алфавітному порядку. Крім того, за вказівкою **Sort[list,rules]** можна розташовувати елементи списку **list** за правилом **rules** . Наприклад:

```
Sort[{a,5,g,-7,9,z}]
```
```
{-7, 5, 9, a, g, z}
```
```
Sort[{2,-63,6,5,8,5},Greater](*сортування чисел за спаданням*)
```
```
{8, 6, 5, 5, 2, -63}
```

За вказівкою **Select[list,crit]** проводиться вибірка зі списку **list** за критерієм crit. Наприклад, зі списку **{1,5,3/5,2.05,53}** потрібно вибрати всі цілі числа (критерій **IntegerQ**):

```
Select[{-1,5,3/5,2.05,53},IntegerQ]
```
```
{-1, 5, 53}
```

4.2 Команди для роботи з векторами та матрицями

У системі Mathematica вектор – це одновимірний список, наприклад **a={a1,a2,a3}**; матриця – це двовимірний список, наприклад, **m={{a11,a12},{a21,a22}}**. Операції додавання та віднімання для векторів та матриць виконуються поелементно так само, як для чисел.

За вказівкою **Dot** обчислюються скалярний добуток двох векторів, добуток вектора на матрицю, а також добуток двох матриць. Наприклад, обчислимо скалярний добуток $\bar{a}\cdot\bar{b}$, де $\bar{a}=(1;2)$, $\bar{b}=(3;4)$ і добуток матриць **c** і **d**:

```
a={1,2};b={3,4};
Dot[a,b]
```

```
ll
c={{c11,c12},{c21,c22}};d={{d11,d12},{d21,d22}};
Dot[c,d]//MatrixForm
```

$$\begin{pmatrix} c11\,d11 + c12\,d21 & c11\,d12 + c12\,d22 \\ c21\,d11 + c22\,d21 & c21\,d12 + c22\,d22 \end{pmatrix}$$

За вказівкою **MatrixForm** список подається у вигляді матриці.

Вказівку **Dot[a,b]** можна подати у вигляді **a.b** (операція **.**), де **a,b** можуть бути векторами або матрицями.

За вказівкою **Det[A]** обчислюється визначник квадратної матриці **A**.

За вказівкою **Eigenvalues[A]** відшукуються власні значення для квадратної матриці **A**. За вказівкою **Eigenvectors[A]** відшукуються власні вектори для квадратної матриці **A**. За вказівкою **Eigensystem[A]** відшукуються власні значення та власні вектори для квадратної матриці **A**.

За вказівкою **Inverse[A]** відшукується обернена матриця для матриці **A**. За вказівкою **Transpose[A]** відшукується транспонована матриця для матриці **A**.

За вказівкою **LinearSolve[A,b]** відшукується вектор – розв'язок матричного рівняння $A\bar{x} = \bar{b}$, де A – матриця коефіцієнтів лівої частини системи лінійних рівнянь, $\bar{x}$ – вектор невідомих, $\bar{b}$ – вектор вільних членів правої частини системи рівнянь. Наприклад, знайти корені системи лінійних рівнянь $\begin{cases} 2x_1 + 3x_2 = -13, \\ 3x_1 - 11x_2 = 6. \end{cases}$

```
A={{2,3},{3,-11}}; b={-13,6};
x=LinearSolve[A, b]
```

$$\left\{-\frac{125}{31}, -\frac{51}{31}\right\}$$

Зробимо перевірку:

```
A.x-b
```

```
{0, 0}
```

Завдання на виконання лабораторної роботи „Списки"

1. Задати матриці A і B розмірності $n \times n$. Елементи матриць випадкові числа.
2. Знайти визначник заданих матриць, використовуючи розклад за k- м рядком.
3. Обчислити обернену матрицю до заданих. Зробити перевірку.
4. Обчислити $(A - 2E)(B^{-1} + 3E)$.

5. Розв'язати систему лінійних рівнянь $AX = B$. Зробити перевірку.
6. Знайти власні значення матриці A. Зробити перевірку.
7. Обчислити власні вектори матриці B. Зробити перевірку.

§5 Операції математичного аналізу

5.1 Обчислення границь

Для знаходження границі функції у системі Mathematica використовується вказівка **Limit[expr, x->x0]**, за якою обчислюється границя виразу expr при **x->x0**. За допомогою опції Direction вказується напрям, в якому відбувається наближення до границі. Опція використовується у вигляді **Direction->1** (або **-1**), за замовчуванням **Automatic**. Значення **1** вказує на обчислення лівосторонньої границі, а **-1** – правостороньої. Наведемо приклади:

```
Limit[Sin[x]/x,x->0]
      1
Limit[Tan[x],x->Pi/2,Direction->1] (*лівостороння границя*)
      ∞
Limit[Tan[x],x->Pi/2,Direction->-1] (*правостороння границя*)
      -∞
```

5.2 Обчислення похідних

Засобами для символьного обчислення похідних, що містяться в ядрі системи Mathematica, охоплюються практично всі важливі типи математичних виразів. До них можуть входити як елементарні, так і спеціальні математичні функції.

Для обчислення похідних у системі Mathematica використовуються наступні функції:

D[f,x] – повертається частинна похідна функції **f** за замінною **x**;

D[f,{x,n}] – повертається частинна похідна **n**-го порядку функції **f** за замінною **x**;

D[f,x1,x2,...] – повертається змішана похідна функції **f** за змінними **x1,x2,...**;

Dt[f,x] – повертається повна похідна функції **f** за змінною **x** за умови, що всі змінні, які входять до **f**, залежать від x;

Dt[f] – повертає повний диференціал функції **f**.

Наведемо кілька прикладів.

```
f=Sin[x]*Cos[y];
D[f,x](*похідна функції f за змінною x*)
      Cos[x] Cos[y]
D[f,{x,3}](*похідна 3-го порядку функції f за змінною x*)
      -Cos[x] Cos[y]
D[f,x,y](*змішана похідна 2-го порядку функції f за змінними x та y*)
      -Cos[x] Sin[y]
Dt[f,x](*похідна функції f за змінною x, y=f(x)*)
      Cos[x] Cos[y] - Dt[y, x] Sin[x] Sin[y]
Dt[f](*повний диференціал функції f*)
      Cos[x] Cos[y] Dt[x] - Dt[y] Sin[x] Sin[y]
```

5.3 Обчислення первісних та визначених інтегралів

Для інтегрування в системі Mathematica використовуються наступні функції:

Integrate[f,x] – відшукується первісна для підінтегральної функції $f(x)$ за замінною **x**;

Integrate[f,{x,a,b}] – обчислюється значення визначеного інтеграла $\int_a^b f(x)dx$;

Integrate[f,{x,a,b},{x,c,d},...] – обчислюється значення кратного інтеграла $\int_a^b\int_c^d ...f(x,y...)dxdy....$

Зауважимо, що визначений інтеграл може бути знайдений як аналітично, так і чисельно. Для чисельного відшукання визначених інтегралів використовується наступна функція.

NIntegrate[f,{x,xmin,xmax}] – повертається чисельне наближення інтеграла $\int_{x\min}^{x\max} f(x)dx$.

Опишемо призначення опцій команди **NIntegrate**:

AccuracyGoal – задається кількість цифр, що визначають точність проміжних результатів;

MaxPoints – визначається максимальна кількість точок при інтегруванні;

Method – визначається метод (**GaussKronrod**, **DoubleExponential**, **Trapezoidal**, **Oscillatory**, **MultiDimensional**, **MonteCarlo**, **QuasiMonteCarlo**), що буде використовуватися при обчисленні інтегралу;

WorkingPrecision – визначається, кількість цифр, що використовуються у внутрішніх обчисленнях.

Наведемо приклади обчислення інтегралів.

```
Integrate[x^2*Log[x],x]
```

$$-\frac{x^3}{9}+\frac{1}{3}x^3\,\text{Log}[x]$$

```
Integrate[x^2*Log[x],{x,1/E,E}]
```

$$\frac{2\,(2+e^6)}{9\,e^3}$$

```
Integrate[Cos[Sin[x/2]],{x,0,1}]
```

$$\int_0^1 \text{Cos}\left[\text{Sin}\left[\frac{x}{2}\right]\right]dx$$

```
NIntegrate[Cos[Sin[x/2]],{x,0,1}]
```

```
0.960828
```

5.4 Розвинення функцій в степеневий ряд

Для розвинення функцій в ряди призначені наступні вказівки системи Mathematica:

Series[f,{x,x0,n}] – виконується розвинення в степеневий ряд функції **f** в околі точки **x=x0** за степенями **(x-x0)^n**;

Series[f,{x,x0,nx},{y,y0,ny}] – послідовно виконується розвинення в ряд функції **f** за змінною **y**, потім за **x**;

SeriesCoefficient[s,n] – повертається коефіцієнт при змінній **n**-ого степеня в ряді **s**;

Суть розвинення функції в степеневий ряд добре видно з розвинення узагальненої функції $f(x)$:

```
Series[f[x],{x,0,5}]
```

$$f[0]+f'[0]\,x+\frac{1}{2}f''[0]\,x^2+\frac{1}{6}f^{(3)}[0]\,x^3+\frac{1}{24}f^{(4)}[0]\,x^4+\frac{1}{120}f^{(5)}[0]\,x^5+0[x]^6$$

```
Series[f[x],{x,x0,5}]
```

$$f[x0]+f'[x0]\,(x-x0)+\frac{1}{2}f''[x0]\,(x-x0)^2+\frac{1}{6}f^{(3)}[x0]\,(x-x0)^3+$$
$$\frac{1}{24}f^{(4)}[x0]\,(x-x0)^4+\frac{1}{120}f^{(5)}[x0]\,(x-x0)^5+0[x-x0]^6$$

У першому випадку функція $f(x)$ розвивається в околі точки **x=0**, а в другому – в околі точки **x=x0**. Відповідно з прийнятою математичною символікою похибка (залишковий член ряду) визначається як **O[x]** з показником степеня, який вказує на порядок похибки.

У зв'язку з наявністю залишкового члена у розвиненні функції в степеневий ряд результат розвинення явно не можна використовувати для наступних розрахунків, наприклад для побудови графіка функції за її розвиненням в ряд. Для отримання прийнятних для розрахунків виразів можна використовувати функції **Normal** або **Collect**. Наприклад, подамо функцію $\cos x$ у вигляді степеневого ряду Тейлора без залишкового члена, використовуючи команду **Normal**:

```
Normal[Series[Cos[x], {x, 0, 6}]]
```

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720}$$

та команду **Collect:**

```
Collect[Series[Cos[x],{x,0,6}],x]
```

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720}$$

5.5 Пошук мінімуму та максимуму функцій

Для відшукання локального мінімуму деякої аналітичної функції використовується команда:

FindMinimum[f,{x,x0}] – виконується пошук локального мінімуму функції **f** в околі точки **x=x0** і повертається його значення.

Для відшукання локального мінімуму функції кількох змінних використовується команда **FindMinimum[f,{x,x0},{y,y0},...]**.

Щоб відшукати локальні максимуми, потрібно функцію **f** помножити на **-1**.

Для пошуку глобального максимуму та мінімуму аналітично заданої функції використовуються відповідно команди:

ConstrainedMax[f,{inequalities},{x,y,...}] – відшукується глобальний максимум функції **f** в області, що визначається нерівностями **inequalities**. Вважається, що всі змінні **x,y,...** невід'ємні.

ConstrainedMin[f,{inequalities},{x,y,...}] – відшукується глобальний мінімум функції **f** в області, що визначається нерівностями **inequalities**. Вважається, що всі змінні **x,y,...** невід'ємні.

За допомогою останніх двох команди розв'язують типові задачі лінійного програмування. До них можна ще додати наступну команду:

LinearProgramming[c,m,b] – відшукується вектор $\bar{x}$, що мінімізує величину $\bar{c}\cdot\bar{x}$ у відповідності до умов $m\cdot\bar{x}\geq 0$ та $x\geq 0$.

Завдання до лабораторної роботи „Операції математичного аналізу”

1. Дослідити функцію $f(x)$ на неперервність і вияснити характер точок розриву, якщо

а) $f(x)=\begin{cases}x^2-1,\ 0\leq x\leq 1,\\ 3-x,\ 1<x\leq 2.\end{cases}$ б) $f(x)=\begin{cases}e^{-\frac{1}{x^2}},\ x\neq 0,\\ 0,\ \textit{інакше.}\end{cases}$ в) $f(x)=\dfrac{\cos\frac{\pi}{x}}{\cos\frac{\pi}{x}}$.

2. При якому значенні параметра a функція $f(x)=\begin{cases}x^2-2ax,\ x>1,\\ a-x,\ x\leq 1.\end{cases}$ буде неперервною.

3. Засобами диференціального числення дослідити функцію $f(x)=3x-x^3$.

4. Обчислити інтеграл $\int x\arctan x dx$, використовуючи формулу інтегрування частинами.

5. Обчислити площу фігури, обмеженої лініями $y_1=x^2$ та $y_2=x+2$.

6. Побудувати на одному рисунку графіки функції $f(x)=(1+x)e^{-x}$ та її розвинення в ряд Тейлора в околі точки $x_0=2$.

7. Розвинути в ряд Фур’є функцію $f(x)=x\cos x$ на проміжку $\left(-\frac{\pi}{2};\frac{\pi}{2}\right)$.

§6 ГРАФІКА

6.1 ДВОВИМІРНА ГРАФІКА

Усі вбудовані функції, призначені для побудови як дво-, так і тривимірних графічних об’єктів, крім параметрів, значення яких обов’язково задаються користувачем, містять велику кількість необов’язкових параметрів, або опцій. Значення опцій встановлені за замовчуванням, проте вони можуть змінюватися користувачем. Першим обов’язковим параметром є вираз або список виразів, що визначають лінії або поверхні. За допомогою другого обов’язкового параметра вказуються, які змінні є аргументами

функцій та проміжки їх зміни. За допомогою опцій визначаються стиль побудови та додаткові елементи графічних об'єктів. Опції задаються у вигляді правил підстановок, наприклад **AspectRatio→Automatic**. При відсутності явного задання опцій приймаються їх значення за замовчуванням.

Охарактеризуємо кілька опцій.

AspectRatio – визначається відношення висоти до ширини двовимірного рисунка. Його можна задати рівним будь-якому числу. За замовчуванням це число дорівнює **1/GoldenRatio**, де **GoldenRatio**=$\frac{1+\sqrt{5}}{2}$. При заданні **AspectRatio→Automatic** опція буде визначатися за алгоритмами системи.

Axes – визначається, які з координатних осей будуть нарисовані. Якщо **False** – жодної з осей не буде; якщо **True** – дві осі нарисовані; **{Bool, Bool}**, де **Bool** – може набувати значень **True** або **False**. Наприклад, якщо задати **Axes→{True,False}**, то буде нарисована тільки вісь **OX**.

AxesLabel – надається назва осям координат. Наприклад, за опцією **AxesLabel→{"X","У"}** осям координат надаються імена **X** та **У**. Назва рисунку надається за опцією **PlotLabel→"Назва"**.

AxesStyle – задається стиль побудови осей координат за допомогою графічних директив, серед яких **Thickness[d]**, за якою визначається відносна товщина осей координат (відносно ширини рисунка), **RGBColor[d1,d2,d3]** – колір лінії, **Dashing[{d1,d2...}]** – розміри послідовних сегментів розривної лінії (розміри повторюються циклічно). Числа **d,d1,d2,d3** належать проміжку **[0,1]**. Стилі побудови для кожної з осей можна задати як різні, так і однакові. Стиль побудови двовимірних графіків визначається за опцією **PlotStyle**, причому можливе задання директив для кожного з графіків. Наведемо приклад задання опції **AxesStyle**:

```
AxesStyle → {{Dashing[{0.02}], RGBColor[1, 0, 0]},
  {Thickness[0.008], RGBColor[0, 1, 0]}}
```

У наведеному прикладі вісь **OX** буде зображена пунктирною лінією червоним кольором, вісь **OУ** – лінією з відносною товщиною **0.008** зеленим кольором.

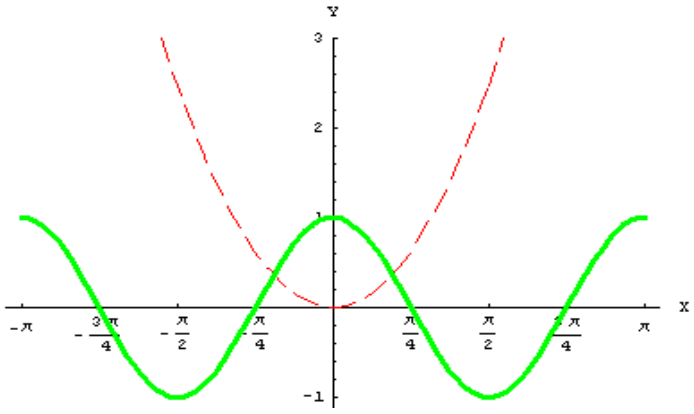
Ticks – задаються відмітки на координатних осях. За замовчуванням (**Ticks→Automatic**) мітки визначаються автоматично. Якщо користувач хоче розставити їх сам, то потрібно задати опцію **Ticks→{{data1},{data2}}**, де **data1**, **data2** – списки значень координат на осях **OX** та **OY** відповідно. Можна замість списку значень координат задавати **Automatic**.

PlotRange – визначається, яка область координати у повинна відображатися на рисунку з метою включення найбільш цікавих характеристик графічного об'єкта.

Для побудови двовимірних графічних об'єктів використовуються наступні команди.

Plot[f,{x,x1,x2},opts] – будується графік функції f, що залежить від аргументу x, який змінюється від **x1** до **x2**. Необов'язковий параметр opts визначає стиль побудови рисунка. За вказівкою **Plot** можна будувати графіки кількох функцій, задавши першим аргументом список функцій **{f1,f2,...}**. Наприклад, зобразимо графіки функцій $y(x)=x^2$ та $g(x)=\cos(2x)$ на проміжку $[-\pi,\pi]$, для кожного з графіків задамо стиль побудови та вкажемо мітки на осі OX $-\pi$, $-\frac{3\pi}{4}$, $-\frac{\pi}{2}$, $-\frac{\pi}{4}$, 0, π, $\frac{3\pi}{4}$, $\frac{\pi}{2}$, $\frac{\pi}{4}$ (на осі **OY Automatic**):

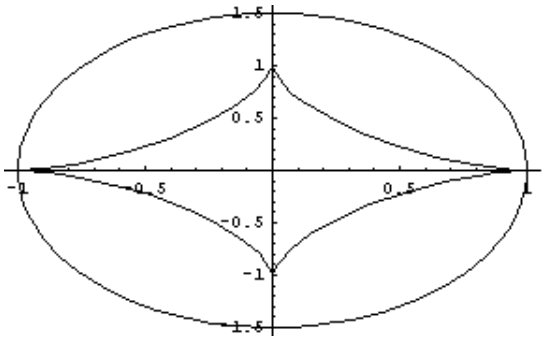
```
Plot[{x^2, Cos[2*x]}, {x, -Pi, Pi}, PlotRange→{-1.1, 3},
 PlotStyle→{{Dashing[{0.04, 0.02}], RGBColor[1, 0, 0]},
   {Thickness[0.009], RGBColor[0, 1, 0]}}, AxesLabel→{"X", "Y"},
 Ticks→{{-Pi, -3*Pi/4, -Pi/2, -Pi/4, 0, Pi, 3*Pi/4, Pi/2, Pi/4}, Automatic}]
```



ParametricPlot[{fx,fy},{t,t1,t2},opts] – будується параметрично задано крива, що визначається рівняннями **x=fx** та **y=fy**, де **fx** та **fy** – функції, що залежать від параметра **t**, який змінюється від **t1** до **t2**.

Необов'язковий параметр opts визначає стиль побудови рисунка. За вказівкою **ParametricPlot** можна будувати графіки кількох параметрично заданих функцій, задавши першим аргументом список функцій **{{fx,fy},{gx,gy}...}**. Наприклад, побудуємо в одній системі координат графіки функцій $\begin{cases} x = \cos t, \\ y = 1{,}5\sin t \end{cases}$ та $\begin{cases} x = \cos^3 t, \\ y = \sin^3 t \end{cases}$.

```
ParametricPlot[{{Cos[t],1.5*Sin[t]},{Cos[t]^3,Sin[t]^3}},{t,0,2*
Pi}]
```



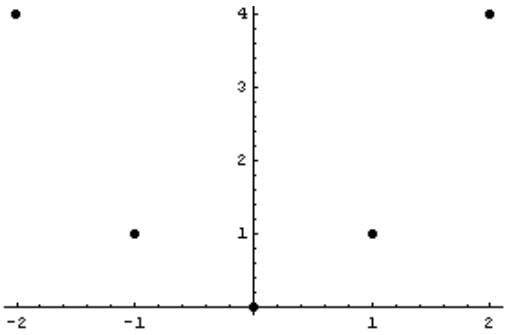
ListPlot[data] – будується графік за точками **data** (таблично задана функція). Аргумент data може визначати наступні набори точок:

1) **{y1,y2,...}** – задається набір значень функції, координата **x** для кожного зі значень **yi** набуває значення **i**;

2) **{{x1,y1},{x2,y2},...}** – задається набір пар точок {xi,yi}, за якими будується графік.

У найпростішому випадку за вказівкою **ListPlot** будуються тільки точки, задані в **data**. Наприклад:

```
ListPlot[{{-2, 4}, {-1, 1}, {0, 0}, {1, 1}, {2, 4}}, PlotStyle -> PointSize[0.02]]
```



Для з'єднання точок відрізками задається опція **PlotJioned→True**.

Для побудови неявно заданих функцій використовується команда **ImplicitPlot** з пакету розширень Graphics, опис якої є аналогічний до запису команди **Plot**.

6.2 ТРИВИМІРНА ГРАФІКА

Для побудови тривимірних графіків залежностей виду $z = f(x, y)$ використовується команда **Plot3D[z,{x,x1,x2},{y,y1,y2},opts]**, де другим та третім параметрами вказані змінні, від яких залежить функція **z**, та межі їх зміни. За замовчуванням графік поверхні міститься в обмежуючому паралелепіпеді.

Для модифікації тривимірних графіків використовуються відповідні опції. Опції **Axes**, **AspectRatio**, **AxesLabel**, **AxesStyle**, **PlotLabel**, **PlotRange**, **Ticks** застосовуються і для тривимірної графіки. Крім них, розглянемо кілька інших опцій:

Boxed – вказується, чи потрібно поміщати графік поверхні в обмежуючий паралелепіпед (за замовчуванням **Boxed→True**);

BoxStyle – визначається стиль побудови обмежуючого паралелепіпеда;

AxesEdge – вказується, на яких ребрах обмежуючого паралелепіпеда повинні виводитися осі;

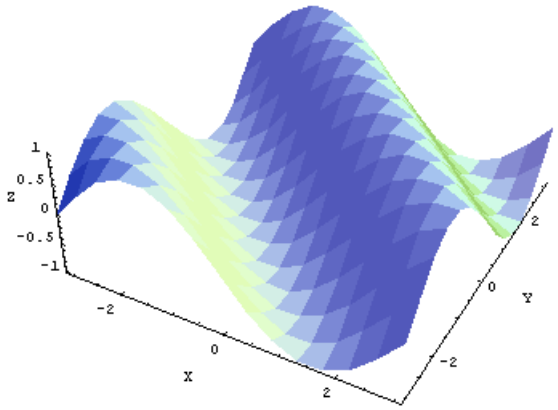
Mesh – вказується, чи потрібна криволінійна сітка на поверхні (за замовчуванням **Mesh→True**).

MeshRange – вказується діапазон зміни координат **x** та **y**, що відповідає масиву заданих величин z;

ViewPoint – задається точка, з якої оглядається побудована поверхня. Положення цієї точки визначається відносними координатами, в яких центр коробки рисунка має координати **{0,0,0}**, а лінійний розмір її найбільшої сторони не перевищує одиниці. Відносні розміри інших сторін визначаються опцією **BoxRotation**. За замовчуванням **ViewPoint→{1.3,-2.4,2}**.

Наприклад, побудуємо поверхню $z = \sin(x + y)$ без обмежуючого паралелепіпеда (**Boxed→False**), без криволінійної сітки на поверхні (**Mesh→False**) та надамо імена осям координат (**AxesLabel→{"X","Y","Z"}**):

```
Plot3D[Sin[x+y],{x,-Pi,Pi},{y,-Pi,Pi},Boxed->False,
  AxesLabel->{"X","Y","Z"},Mesh->False]
```



Зауважимо, що за вказівкою **Plot3D** можна побудувати графік тільки однієї поверхні.

Для побудови графічного подання параметрично заданих залежностей використовується команда **ParametricPlot3D**, а побудови графічного подання таблично заданих функцій – команда **ListPlot3D**.

6.3 ГРАФІЧНІ ПРИМІТИВИ (СТРУКТУРИ)

У системі Mathematica використовуються наступні примітиви двовимірної графіки:

Circle[{x,y},r] – визначається коло з центром у точці **{x,y}** та радіусом **r**;

Circle[{x,y},{rx,ry}] – визначається еліпс з центром у точці **{x,y}** та півосями **rx** та **ry**;

Circle[{x,y},r,{φ1,φ2}] – визначається дуга кола з радіусом **r**, з центром у точці **{x,y}** та кутами кінцевих точок **φ1** та **φ2**;

Circle[{x,y},{rx,ry},{φ1,φ2}] – визначається дуга еліпса з півосяси **rx** та **ry**, з центром у точці **{x,y}** та кутами кінцевих точок **φ1** та **φ2**.

Для визначення круга, замальованого еліпсу, секторів круга та замальованого еліпса використовується вказівка **Disk** з параметрами аналогічними до параметрів вказівокі **Circle**.

Point[{x,y}] – визначається точка з координатами **{x,y}**;

Line[{{x1,y1},{x2,y2},...}] – визначається ламана, відрізки якої послідовно з'єднують точки **{x1,y1},{x2,y2},...**;

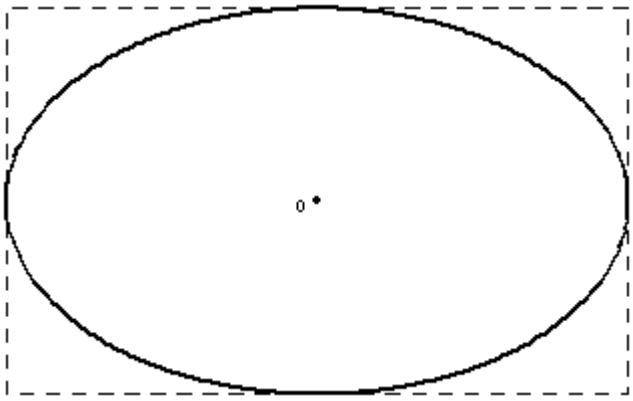
Polygon[{{x1,y1},{x2,y2},...}] – визначається замальований многокутник, обмежений замкненою лінією, яка проходить через точки **{x1,y1},{x2,y2},....**

Rectangle[{xmin,ymin},{xmax,ymax}] – визначається замальований прямокутник, де **{xmin,ymin}** та **{xmax,ymax}** координати протилежних кутів;

Text[expr,{x,y}] – визначається текст **expr**, який буде центруватися відносно точки з координатами **{x,y}**.

Для виведення зображення графічного примітиву двовимірної графіки спочатку треба застосувати функцію **Graphics**, а потім команду **Show**. Крім графічних примітивів, можна за допомогою директив визначати розмір, колір та стиль побудови примітивів. Директива (може бути кілька директив) записується перед графічним примітивом, і обидва об'єкти поміщаються у фігурних дужках. Крім того директива може стосуватися і кількох однорідних графічних примітивів. Уся група також поміщається у фігурні дужки. Наприклад:

```
pr1=Graphics[{Thickness[0.009],Circle[{0,0},{1,2}]}];
pr2=Graphics[{Dashing[{0.02}],Line[{{-1,-2},{-1,2},{1,2},{1,-2},
{-1,-2}}]}];
pr3=Graphics[Text[O,{-0.05,-0.05}]];
pr4=Graphics[{PointSize[0.015],Point[{0,0}]}];
Show[{pr1,pr2,pr3,pr4}]
```



Внести будь-який графічний примітив на рисунки, створені за допомогою вказівок **Plot**, **ListPlot**, **ParametricPlot**, **Plot3D**, **ListPlot3D**, **ParametricPlot3D** можна за опціями **Prolog** або **Epilog**.

Якщо замість функції **Graphics** використати функцію **Graphics3D**, то можна отримувати тривимірні графічні об'єкти. Примітиви **Line**, **Point**, **Polygon**, **Text** з відповідними змінами переносяться на тривимірний випадок. Єдиним суто тривимірним графічним примітивом є:

Cuboid[{xmin,ymin,zmin},{xmax,ymax,zmax}] – визначається паралелепіпед з протилежними кутами у точках **{xmin,ymin,zmin}** та **{xmax,ymax,zmax}**.

Завдання на виконання лабораторної роботи „Графіка”

1. Побудуйте графіки функції $f(x)=\dfrac{x^2-5x+6}{2x-1}$, дотичної у точці $x_0=1.5$, похилих асимптот. Графік функції $f(x)$ зробіть синім кольором, з відносною товщиною 0.008; графік дотичної – зеленим, з відносною товщиною 0.006, асимптоти – червоним, пунктиром. Дайте назву осям координат та рисунку.
2. Побудуйте графіки залежності $\dfrac{x^2}{2}+\dfrac{y^2}{4}=1$ та дотичних у точці $x_0=1.5$. Підпишіть графічні побудови.
3. Побудувати площину, яка проходить через точки $A(1;-5;6)$, $B(-2;3;-3)$, $F(-3;5;5)$. До даної площини провести перпендикулярну пряму, яка проходить через точку $G(5;5;6)$. Позначити літерою M точку перетину прямої з площиною.
4. Побудувати поверхню $z=\sqrt{x^2+y^2}$. Провести до даної поверхні дотичну площину і нормаль у точці $(0.8;0.6;1)$.
5. За допомогою графічних примітивів побудувати еліпс $\dfrac{x^2}{4}+\dfrac{y^2}{5}=1$ синім кольором з відносною товщиною 0.008. На рисунку позначити центр еліпса (точка О). До еліпса провести дві вертикальні дотичні. Вивести осі координат.

§7 ІНТЕРПОЛЯЦІЯ ДАНИХ ТА АПРОКСИМАЦІЯ ФУНКЦІЙ

Для розв’язування задач інтерполяції та апроксимації функцій, заданих набором вузлових точок, використовуються наступні функції:

InterpolatingPolynomial[data,var] – повертає поліном за змінною **var**, значення якого у вузлових точках співпадають з даними зі списку **data**.

Список може мати форму: **{{x1,y1},{x2,y2},...}** або **{y1,y2,...}**, тоді у другому випадку **xi** набувають значень **1,2,...**;

Interpolation[data] – будується об'єкт **InterpolatingFunction**, за яким повертається інтерполююча функція, за допомогою якої можна обчислювати проміжні значення в заданому діапазоні.

Приклад застосування функції Interpolation:

```
data=Table[{x,Sin[x]},{x,-3.,3.,0.5}](*задання вузлових точок*)
```

```
{{-3., -0.14112}, {-2.5, -0.598472}, {-2., -0.909297},
 {-1.5, -0.997495}, {-1., -0.841471}, {-0.5, -0.479426}, {0., 0.}, {0.5, 0.479426},
 {1., 0.841471}, {1.5, 0.997495}, {2., 0.909297}, {2.5, 0.598472}, {3., 0.14112}}
```

```
f=Interpolation[data](*створення інтерполюючої функції*)
```

```
InterpolatingFunction[{{-3., 3.}}, <>]
```

```
{f[-2.7],f[-1.5],f[0.3],f[1.2]}(*обчислення функції f у деяких точках*)
```

```
{-0.428853, -0.997495, 0.295168, 0.930742}
```

Таким чином, за допомогою команди **Interpolation** на заданому інтервалі зміни **x** можна знайти будь-яке проміжне значення функції **f[x]**, в тому числі і значення у вузлових точках.

Наведемо приклад поліноміальної інтерполяції. Зазначимо, що степінь інтерполюючого полінома завжди на одиницю менший від кількості вузлових точок. У вузлах інтерполяції значення інтерполюючого многочлена точно співпадає зі значеннями вихідних даних. Наприклад:

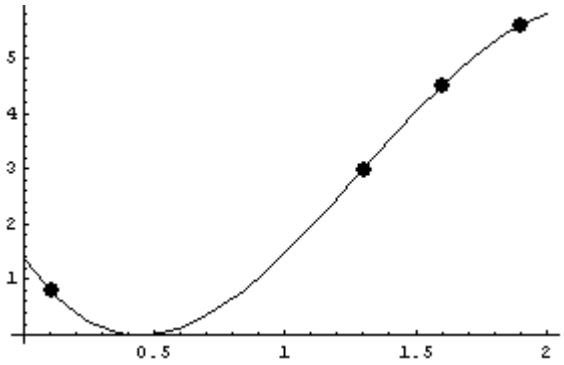
```
(*Поліноміальна інтерполяція*)
data1={{0.1,0.8},{1.3,3},{1.6,4.5},{1.9,5.6}}
```

```
{{0.1, 0.8}, {1.3, 3}, {1.6, 4.5}, {1.9, 5.6}}
```

```
p=Expand[InterpolatingPolynomial[data1,x]]
```

$$1.39185 - 6.82778\,x + 9.33333\,x^2 - 2.40741\,x^3$$

```
g1:=Plot[p,{x,0,2}];
g2:=ListPlot[data1,PlotStyle->PointSize[0.03]];Show[g1,g2]
```



При великій кількості вузлових точок практично більш корисною є регресія, що полягає у відшуканні параметрів деякої функції регресії, при якій графік функції проходить „біля" вузлових точок, чим забезпечується найменша середньоквадратична похибка обчислень. На відміну від інтерполяції, при регресії функція у вузлових точках не набуває точного значення ординат –просто мінімізується похибка обчислень у цих точках.

Для розв'язування задач регресії використовується наступна функція:

`Fit[data,funs,vars]` – для списку даних **`data`** відшукується методом найменших квадратів наближення у вигляді лінійної комбінації функцій funs від змінних vars. Дані data можуть мати форму: **`{{x1,y1},{x2,y2},...} або {y1,y2,...}`**, тоді у другому випадку **`xi`** набувають значень **`1, 2, ...`**. Як аргумент **`funs`** може бути вказаний будь-який список функцій, що залежать тільки від об'єктів vars. Наприклад:

```
(*Регресія*)
data1={{0.1,0.8},{1.3,3},{1.6,4.5},{1.9,5.6}}
```

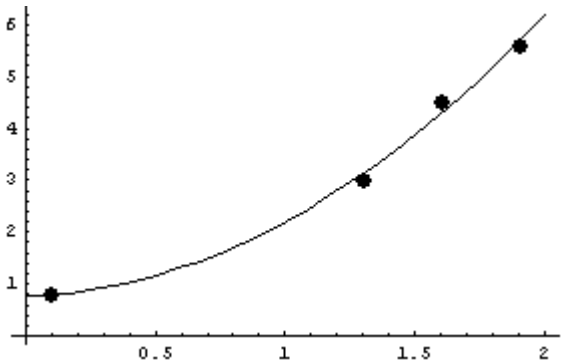
```
{{0.1, 0.8}, {1.3, 3}, {1.6, 4.5}, {1.9, 5.6}}
```

```
Out[17]= {{0.1, 0.8}, {1.3, 3}, {1.6, 4.5}, {1.9, 5.6}}
```

```
p1=Fit[data1,{1,x,x^2},x]
```

$$0.763513 + 0.149914\,x + 1.2848\,x^2$$

```
g1:=Plot[p1,{x,0,2}]
g2:=ListPlot[data1,PlotStyle->PointSize[0.03]]
Show[g1,g2]
```



Завдання до виконання лабораторної роботи «Інтерполяція даних та апроксимація функцій»

Функція задана таблично:

x	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
$f(x)$	0,35	0,4	0,65	0,7	0,75	0,82	0,874	0,945	0,853	0,739	0,65

Використовуючи інтерполяцію функцій, знайти:

а) значення даної функції у точці x=0,412532;

б) значення першої похідної функції у точці x_0=0,534;

в) визначений інтеграл функції від точки a=0,53 до точки b=0,6;

г) апроксимуючий та інтерполюючий многочлени, що відповідає даній функції.

§8 ПРОГРАМУВАННЯ У СИСТЕМІ MATHEMATICA

Система Mathematica в основному призначена для розв'язування математичних задач користувачами, які не володіють навичками програмування. Проте система Mathematica містить власну мову програмування, що дозволяє розв'язувати задачі, які вимагають наявності вбудованих функцій і/або функцій з пакетів розширень, необхідних для створення та реалізації власних алгоритмів. Основою системи Mathematica є проблемно-орієнтована на математичні обчислення мова програмування надвисокого.

8.1 ФУНКЦІОНАЛЬНЕ ПРОГРАМУВАННЯ

Система Mathematica містить більше тисячі вбудованих функцій. Проте інколи при виконанні обчислень певні комбінації вбудованих функцій повторюються. У такій ситуації корисно замінити такий набір функцій однією. Цей принцип лежить в основі функціонального програмування: функції застосовується до аргументу, потім інші функції застосовуються до функцій від аргументів, потім нові функції застосовуються до функцій від функцій від аргументів і т.д.

8.1.1 Шаблони

Шаблони в системі Mathematica використовуються для задання виразів різних класів і надання змінним певних властивостей, необхідних для створення функцій користувача. Ознакою шаблону є знак підкреслення **_**. Шаблони використовуються для виокремлення класів виразів Mathematica. Загальний шаблон має вигляд **_**, відповідний клас складається зі всіх виразів Mathematica. Шаблону _ можна надати ім'я, яке повинно бути символьного типу. Іменований шаблон **_** має вигляд **symbol_**. Крім того, можна вказувати заголовок шаблону у вигляді **_head** (вказується тип даних).

Крім шаблонів, які визначаються символом **_**, у системі Mathematica містяться шаблони виду __ – подвійне підкреслення та потрійне підкреслення ___. Подвійним підкресленням визначається клас виразів, що складаються з одного або кількох виразів Mathematica, що розділяються комами. Потрійним підкресленням визначається клас, що складається з нуля або з кількох виразів, які розділяються комами. Шаблони використовуються тільки у списках параметрів – у правій частині виразів вони не використовуються. Наприклад, функція:

```
fts[x_,y_]:=y*Tan[x]+x*Sin[y]
```

містить у списку параметрів два шаблони **x_** та **y_**. У правій частині функції **fts** змінні **x** та **y** є локальними змінними.

Задання за допомогою шаблонів типів даних робить програми більш строгими та наочними і дозволяє уникнути помилок, пов'язаних з невідповідністю даних.

8.1.2 Функції користувача

Для задання функцій користувача можна визначити кілька правил:

– така функція має ідентифікатор – ім'я, яке повинно бути унікальним і зрозумілим (призначення функції);

– список параметрів функції повинен містити шаблони змінних, а не самі змінні;

– можна використовувати відкладене := або невідкладене = надання значень;

– тіло функції може містити кілька виразів, поміщених у круглих дужках, при цьому повертається значення останнього виразу;

– змінні шаблонів у списку параметрів функції є локальними.

Наприклад, задамо функцію **seraryf**, за якою обчислюється середнє арифметичне значень із списку:

```
seraryf[x_List]:=Apply[Plus,x]/Length[x]
```

і використовуючи функцію **seraryf** обчислимо середнє арифметичне значень із списку **{1,2,6,4,5,6}**:

```
seraryf[{1,2,6,4,5,6}]
```

```
4
```

Після задання функції користувача використовуються аналогічно до вбудованих функцій.

8.1.3 Чисті функції

Функції користувача доцільно створювати у випадках, коли вони використовуватимуться в програмі або сеансі роботи. У випадку, коли нова функція виникає в процесі виконання обчислень, яка в наступному не знадобиться, то корисно використовувати так звані *чисті* функції, до яких звертаються в момент їх створення. Для створення чистих функцій використовується вбудована функція Function. Наприклад, за допомогою чистої функції створимо функцію, за якою обчислюється середнє арифметичне значень елементів списку:

```
Function[{x}, Apply[Plus,x]/Length[x]]
```

$$\text{Function}\left[\{x\}, \frac{\text{Plus@@x}}{\text{Length[x]}}\right]$$

У цьому визначенні x – формальний параметр (локальна змінна). Застосування чистої функції до аргументу здійснюється стандартним чином – після запису функції в квадратних дужках задається фактичний параметр:

```
Function[{x}, Apply[Plus,x]/Length[x]][{1,2,6,4,5,6}]
```

```
4
```

За допомогою функції **Function** можна визначити чисту функцію від будь-якого виразу **expr** від змінних **x1,x2,...**: **Function[{x1,x2,...},expr]**.

Існує можливість більш компактного та зручного задання чистої функції, яку називають *анонімною*. Такі функції не мають ні назви, ні звичного визначення і задаються виразами спеціального виду. У цих виразах замість змінних використовуються символи **#** (для однієї змінної) і **#1, #2,...** (для кількох змінних). Закінчується тіло функції символом &. Для обчислення значення такої функції після її запису у квадратних дужках вказуються фактичні параметри. Наприклад, задамо анонімну функцію піднесення x до степеня n і обчислимо 4^5:

```
#1^#2&[4,5]
```

```
1024
```

Анонімні функції зручно використовувати з командами **Sort** та **Select**.

8.1.4 Суперпозиція функцій

При функціональному програмуванні часто використовується послідовне застосування функцій, або суперпозиція функцій. За вказівкою **Nest[fun,x,n]** застосовується функція **fun** до аргументу x, потім до результату застосування функції до аргументу і так **n** разів. Така ситуація часто виникає при ітераційних обчисленнях. Для прикладу, використовуючи команду **Nest**, задамо функцію **iter**, яка реалізує метод простої ітерації для розв'язування нелінійних рівнянь. У методі простої ітерації для знаходження нуля функції $f(x)$, який міститься поблизу точки x_0, **n**-е наближення обчислюється за формулою $x_n = f(x_{n-1}), n = 1,2...$. Функція **iter** залежить трьох аргументів: функції **f** (права частина рівняння), початкового наближення x0 та кількості ітерацій **n**.

```
iter[f_,x0_,n_]:=Nest[(f[#])&,x0,n]
```

Використовуючи створену функцію **iter**, розв'яжемо рівняння $e^x - x - 2 = 0$. Для зручності використання методу задамо рівняння у вигляді

$x = e^x - 2$ ($f = e^x - 2$). Функцію f потрібно задавати як чисту (за допомогою **Function**).

```
iter[Function[{x},Exp[x]-2],-1.1,10]
```

```
-1.84141
```

Щоб оцінити збіжність і не повторювати вручну обчислення, які відрізняються тільки кількістю ітерацій, поступимо наступним чином:

```
iter[Function[{x},Exp[x]-2],-1.1,#]&/@Range[10]
```

```
{-1.66713, -1.81121, -1.83654, -1.84063, -1.84128,
 -1.84139, -1.8414, -1.84141, -1.84141, -1.84141}
```

У цьому випадку у функції **iter** визначена анонімна функція аргументу, який вказує кількість ітерацій. Вона застосовується до всіх елементів списку **Range[10]**. Як видно з результату, після виконання сьомої ітерації відбувається стабілізація п'яти знаків після коми, тому з такою точністю наближене значення кореня рівняння $e^x - x - 2 = 0$ дорівнює **-1,841410**.

За функцією FixedPoint здійснюються ітерації до тих пір, поки з машинною точністю результат не перестане змінюватися. Запишемо з її допомогою функцію **iter**:

```
iter[f_,x0_]:=FixedPoint[(f[#])&,x0]
iter[Function[{x},Exp[x]-2],-1.1]
```

```
-1.84141
```

За вказівкою **NestList[f,x0,n]** отримується список, першим елементом якого є аргумент **x0**, другим – результат застосування функції **f[x0]**, третім – **f[f[x0]]** і т.д. Задамо функцію **iterlist**, за допомогою якої реалізується метод простої ітерації, і застосуємо її до рівняння $x = e^x - 2$:

```
iterlist[f_,x0_,n_]:=NestList[(f[#])&,N[x0],n]
iterlist[Function[{x},Exp[x]-2],-1.1,10]
```

```
{-1.1, -1.66713, -1.81121, -1.83654, -1.84063,
 -1.84128, -1.84139, -1.8414, -1.84141, -1.84141, -1.84141}
```

За вказівкою **FoldList[f,x,{a,b,...}]** утворюється список **{x,f[x,a],f[f[x,a],b]}**, а за вказівкою **Fold[f,x,{a,b,...}]** повертається останній елемент цього списку. За вказівкою **ComposeList[f1,f2,...][expr]** утворюється список **{expr,f1[expr],f2[f1[expr]],...}**, а за вказівкою

Composition[f1,f2,...][expr] повертається останній елемент цього списку.

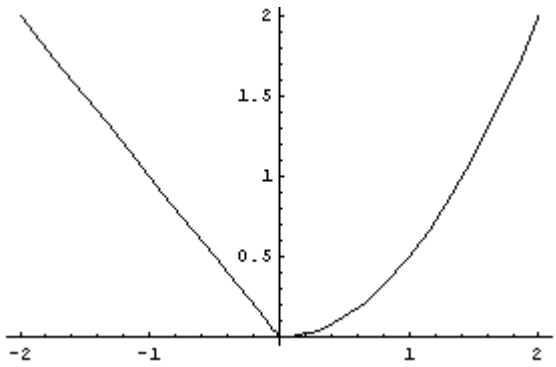
8.2 ПРОГРАМУВАННЯ, ЩО БАЗУЄТЬСЯ НА ПРАВИЛАХ ПЕРЕТВОРЕНЬ

Використання шаблонів у правилах перетворень як глобальних, так і локальних, дозволяє створювати короткі за довжиною, проте змістовні програми. Наприклад, визначимо функцію $f(x)=\begin{cases}-x, x<0,\\ \frac{1}{2}x^2, x\geq 0.\end{cases}$ Для задання такої функції використовуються два предикати, тому у визначенні виду f**[x_]:=expr** слід використовувати той чи інших з двох виразів в залежності від виконання тієї чи іншої умови. Для цього використовується символи **/;**. Наприклад:

```
f[x_]:=-x/;x<0
f[x_]:=1/2x^2/;x>=0
```

Щоб переконатися у правильності задання функції, побудуємо графік функції $f(x)$:

```
Plot[f[x],{x,-2,2}]
```



Визначимо функцію **fact** від цілого невід'ємного аргумента **n** наступним чином: **fact[n]=n* fact[n-1]**, що надає можливість обчислювати значення цієї функції, початкове значення якої є **fact[0]=1**. Така тотожність є найпростішим рекурсивним визначенням функції:

```
fact[0]=1;fact[n_Integer?Positive]:=n*fact[n-1]
{fact[3],fact[2.5],fact[-5]}
```

```
{6, fact[2.5], fact[-5]}
```

Функція **fact[n]** обчислюється у відповідності з заданими правилами для цілих невід'ємних чисел, для інших значень аргументів вона не визначена. У системі Mathematica послідовно застосовуєтьсяі основне

правило, зменшуючи значення аргументу доти, поки воно не дійде до заданого початкового значення **fact[0]=1**, а потім проводяться обчислення в зворотному порядку. У цьому можна переконатися, використавши команду **Trace**:

```
Trace[fact[3]]
```

```
{fact[3], {Positive[3], True}, 3 fact[3 - 1],
 {{3 - 1, 2}, fact[2], {Positive[2], True}, 2 fact[2 - 1], {{2 - 1, 1}, fact[1],
   {Positive[1], True}, 1 fact[1 - 1], {{1 - 1, 0}, fact[0], 1}, 1 1, 1}, 2 1, 2}, 3 2, 6}
```

Таким само, як і функцію для обчислення факторіала, можна визначати функцію для відшукання чисел Фібоначчі.

В якості іншого прикладу визначимо функцію lg, за якою обчислюється логарифми дійсних чисел за основою 10. Щоб виконати це завдання, можна задати наступний набір перетворень:

```
lg[10]=1;lg[1]=0;lg[b_^a_]:=a*lg[b];
lg[a_*b_]:=lg[a]+lg[b];
```

за якими визначаються властивості логарифмічної функції. Щоб знаходити чисельні значення функції lg, слід задати правило:

```
lg[x_Real]:=FixedPoint[(#-(10^#-x)/(10^#*Log[10.]))&,0.9]
```

яке базується на методі дотичних.

Для прикладу обчислимо значення виразу $\lg 0.31$ і порівняємо з точним значенням, відшуканим за допомогою функції **Log**:

```
{lg[10.31],Log[10,10.31]}
```

```
{1.01326, 1.01326}
```

8.3 ПРОЦЕДУРНЕ ПРОГРАМУВАННЯ

Незважаючи на те, що більш природними для системи Mathematica є функціональне програмування та програмування, яке базується на правилах перетворень, вона містить засоби і конструкції програмування у традиційному процедурному стилі мов Basic, Pascal, Fortran і т.п. Крім того, деякі функції (наприклад, **If**, **Which**) процедурного програмування часто використовуються і поза ним.

8.3.1 Умовні оператори

Опис умовного оператор **If** наступний:

If[test,expr1,expr2,expr3] – обчислюється вираз expr1, якщо умова **test** набуває значення істинно, **expr2**, якщо хибно і **expr3**, якщо засобами системи Mathematica не можливо визначити, чи умова є істинною, чи хибною (наприклад, в умові порівнюється число зі змінною, якій не надано числового значення). Найчастіше використовується класична форма умовного оператора **If**: **If[test,expr1,expr2]**. Можлива також неповна форма умовного оператора.

Умовна функція **Which[test1, value1, test2, value2,...]** містить парну кількість **2n** аргументів, серед яких **n** умов **testi** та **n** виразів **valuei**. При цьому обчислюється перший з виразів **valuei**, для якого умова (яка є перед виразом) **testi** набуває значення **True**. Якщо перші **k** умов набувають значення **False**, а **(k+1)**-ша є небулевим значенням, то умовна функція **Which** буде необчисленою.

8.3.2 Циклічні структури

Організація циклів здійснюється за допомогою циклів **Do**, **For**, **While**.

За допомогою циклу **Do[expr,{imax}]** значення виразу expr обчислюється imax разів. Після виконання обчислень у результат не виводиться, хоч вираз значення **expr** обчислений. Наприклад:

```
s=0;Do[s=s^2+1,{5}]
s
```

```
677
```

Якщо потрібно відзразу вивести на екран результат, то слід ввести наступний вираз:

```
s=0;Do[s=s^2+1,{5}];s
```

```
677
```

За допомогою циклу **Do[expr,{i,imax}]** значення виразу **expr**, який може містити параметр **i**, обчислюється imax разів ($i = \overline{1, i\max}$). За допомогою циклу **Do[expr,{i,imin,imax}]** значення виразу **expr**, який може містити змінну **i**, обчислюється від **imin** до imax з кроком **1**. За допомогою циклу **Do[expr,{i,imin,imax,di}]** значення виразу **expr**, який може містити змінну i, обчислюється від **imin** до **imax** з кроком **di**. У циклі **Do** може бути

кілька параметрів: **Do[expr,{i,imin,imax},{j,jmin,jmax},...]**. Розглянемо приклад.

Приклад. Скласти програму, використовуючи оператор циклу **Do**, за якою обчислюється добуток двох квадратних матриць розмірності $n \times n$.

Запишемо формулу, за якою обчислюється добуток двох матриць $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$, $i, j = \overline{1,n}$.

```
n=3;
A=Array[a,{n,n}]; (*оголошення масиву A*)
B=Array[b,{n,n}];(*оголошення масиву B*)
C1=Array[c,{n,n}];(*оголошення масиву C1*)
Table[{a[i,j]=i+j,b[i,j]=i-j^2},
{i,1,n},{j,1,n}];(*надання значень елементам матриць A та B*)
Do[s=0;Do[s=s+a[i,k]*b[k,j],{k,n}];
c[i,j]=s,{i,n},{j,n}];(*програмна реалізація формули множення матриць*)
C1//MatrixForm (*виведення результату*)
```

$$\begin{pmatrix} 11 & -16 & -61 \\ 14 & -22 & -82 \\ 17 & -28 & -103 \end{pmatrix}$$

Зупинити виконання циклу **Do** можна за допомогою директив **Abort** або **Break**, які використовується разом з умовним оператором **If**. Різниця між директива **Abort** та **Break** полягає у тому, що при застосуванні директиви **Abort** всі обчислення перериваються, а при застосуванні **Break** – тільки вихід з циклу і продовження виконання обчислень (якщо такі є). Наприклад:

```
s=0; Do[s=s+i^2;If[s>100,Abort[]],{i,100}];s
```

```
$Aborted
```

```
s=0; Do[s=s+i^2;If[s>100,Break[]],{i,100}];s
```

```
140
```

Розглянемо структуру циклу **For**, опис якої наступний:

For[start,test,step,body], де **start** – початкові дані, **test** – умова (якщо умова істинна, то цикл виконується), **step**, **body** – вирази, значення яких обчислюються, поки умова істинна. Наприклад, використовуючи оператор циклу, обчислимо значення виразу $1^2+2^2+3^2+...+12^2$:

```
For[i=1;s=0,i<=12,i=i+1,s=s+i^2];s
```

```
650
```

Розглянемо структуру циклу **While** опис якої наступний:

While[test,body] – обчислюються значення виразів body, поки умова **test** істинна. Наприклад, за алгоритмом Евкліда, обчислимо найбільший спільний дільник чисел **a** та **b**, використовуючи оператор циклу **While**:

```
a=54;b=36;
While[a!=b,If[a>b,a=a-b,b=b-a]];b
```

```
18
```

Змінні, яким надаються певні значення у процесі виконання обчислень у циклах, є глобальними (за винятком ітераторів у циклі **Do**). Для надання змінним локального характеру використовуються команди **Block** та **Module**, у яких першим аргументом є список локальних змінних (можна змінним надавати початкових вхідних значеннь), а другим – тіло процедури. Для прикладу, виконаємо попереднє завдання за умови, що змінні **a** та **b** – локальні:

```
Block[{a=54,b=36},While[a!=b,If[a>b,a=a-b,b=b-a]];b]
{a,b}(*змінним a,b не надано ніяких значень*)
```

```
18

{a, b}
```

Завдання на виконання лабораторної роботи „Програмування у системі Mathematica”

1. Створити функцію для обчислення середнього арифметичного списку.
2. Створити функцію для обчислення чисел Фібоначчі.
3. Створити функцію, за якою відшукувався корінь рівняння методом простої ітерації (дотичних).
4. Скласти програму для обчислення визначеного інтегралу, використовуючи розклад підінтегральної функції у ряд Тейлора, з заданою точністю.
5. Скласти програму для обчислення визначника квадратної матриці n-го порядку, використовуючи розклад за k-тим рядком (стовпцем).
6. Скласти програму для визначення кількості взаємно простих чисел з n, що не перевищують числа n.
7. Задано натуральне число n. Знайти всі натуральні числа, які не перевищують числа n і кожна цифра якого є дільником числа n.

8. Скласти процедуру (модуль) для розвинення функції у ряд Фур’є.
9. Створити функцію для вибірки зі списку членів арифметичної прогресії за заданими першим членом та різницею

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Воробьев Е.М. Введение в систему „Математика”. – М: Финансы и статистика, 1998. – 262 с.
2. Говорухин В., Цибулин В. Компьютер в математических исследованиях. – СПб.: Питер, 2001. – 624 с.
3. Дьяконов В.П. Системы символьной математики Mathematica 2 и Mathematica 3. Справочное издание. – М.: СК ПРЕСС.- 1998.– 328 с.
4. Кирьянов Д. В. Самоучитель MathCAD 13. – С-Пб: БХВ-Петербург, 2006. – 528 с.
5. Потемкин В. Г. Система MatLab. Справочное пособие. – М.: ДИАЛОГ-МИФИ, 1997. – 352 с.
6. Прохоров Г.В., Леденев М.А., Колбеев В.В. Система аналитических вычислений Maple. – М.: Петит, 1997. – 200 с.
7. Сайт http://maxima.sourceforge.net
8. Сайт фірми Maple Waterloo Inc. – http://www.maplesoft.com
9. Сайт фірми Mathsoft Engineering & Education. Inc. – http://www.mathsoft.com/
10. Сайт фірми MathWorks Inc. – http://www.mathwork.com
11. Сайт фірми SciFace Software GmbH & Co. – http://www.sciface.com
12. Сайт фірми Wolfram Research Inc. – http://www.wolfram.com
13. Steinhaus S. Comparison of Mathematical Programs for Data Analysis (Edition 5.03) [Електронний ресурс] — Munchen/Germany. — 64 p. — Режим доступу : http://www.scientificweb.de/ ncrunch/.

ПРЕДМЕТНИЙ ПОКАЖЧИК

ЗМІСТ