Міністерство освіти і науки України
Сумський державний університет

А. В. Загорулько, Д. О. Кайота

Програмування задач механіки

Конспект лекцій

Суми
Сумський державний університет
2019

Міністерство освіти і науки України
Сумський державний університет

Програмування задач механіки

Конспект лекцій

для студентів спеціальності

131 «Прикладна механіка»
денної форми навчання

Затверджено
на засіданні кафедри
загальної механіки
та динаміки машин
як конспект лекцій
із дисципліни «Програмування
задач механіки».
Протокол No 9 від 04.06.2019 р.

Суми
Сумський державний університет
2019

Програмування задач механіки : конспект лекцій / укладачі:
А. В. Загорулько, Д. О. Кайота. – Суми : Сумський державний університет, 2019. – 48 с.

Кафедра загальної механіки та динаміки машин

ЗМІСТ

С

Основні відомості про програмування

Комп'ютерна програма (вона ж додаток) - зв'язка численних рядків спеціального тексту. Він є спеціальним, тому що створений таким чином, щоб машині було зрозуміло, які дії повинні бути виконані. Найпростіші додатки містять близько ста рядків коду, а в складних і масштабних додатках кількість рядків коду доходить до мільярда.

Комп'ютерний код - це спеціальний текст, що складається з набору покрокових інструкцій. Він не завжди містить в собі нулі і одиниці, також в ньому є певні слова і додаткові символи. Комп'ютер зчитує код, який повідомляє йому, які операції слід виконати з даними. Ви знайомі з Twitter? Уявіть, що Twitter - одна велика квартира, в якій розташовані мільйони комп'ютерів, що зберігають ваші твіти та вашу дату народження (день народження), як і твіти та дати народження мільйонів інших користувачів.

Всі ці твіти та дати - дані. Однак, комп'ютери не обробляють і не перечитують ваші твіти весь час. Навіть якби вони перечитували ваші твіти, повірте, їм було б боляче. Комп'ютер починає обробляти і отримувати код, коли ви входите в Twitter. У цей момент він завантажує з серверів ті дані, які користувачі внесли на сайт: твіти, дати народження. Ці дії виконуються в тому порядку, який прописаний в коді.

Якщо інструкції добре продумані, то все буде відносно добре працювати. Чому відносно? Розробникам часто доводиться шукати і виправляти баги (помилки) в проектах, навіть незважаючи на те, що вони робочі. Адже баг може перетворитися у велику вразливість.

Якщо в інструкціях буде будь-який недолік, наприклад, помилка або неправильне тлумачення даних, то додаток буде працювати нестійко або не буде запускатися взагалі, показуючи повідомлення про помилку.

Пам'ятайте, не так складно писати код, як вміти терпіти і приймати всі невдачі і провали, з якими вам, безсумнівно, доведеться зіткнутися. Можете взяти приклад в плані терпіння з програмістів з Індії: їх код не завжди

хороший, але їх оптимізму можуть позаздрити багато новачків, які розчарувалися в собі.

Одна з найбільш цікавих і захоплюючих частин програмування - рішення різних завдань і проблем, які представляють собою щось на зразок пазла або головоломки. Але це буде цікавим тільки тоді, коли ви розберетеся в коді і почнете розуміти його.

Інструментарій типового програміста найчастіше складається з наступних речей:

- комп'ютер;
- інтернет (перш за все він потрібен для пошуку всього невивченого і невідомого в будь-якому з відомих пошукових сервісів);
- редактор коду (або IDE - комплекс програмних засобів, який використовується програмістами для розробки програмного забезпечення), який допоможе впорядкувати все, що ви створюєте;
- компілятор або інтерпретатор. Це програма, яка читає ваш код і намагається знайти в ньому помилки. Потім він збирає ваш код в єдиний пакет і передає комп'ютеру для виконання;
- навушники. Можливо, вас будуть відволікати зовнішні шуми, а навушники - один з простих способів захистити себе від шумів.

Ви можете використовувати вільне програмне забезпечення для початку роботи з кодом. Таким є Atom і Notepad++. Ви також можете спробувати SublimeText, однак цей редактор є платним програмним забезпеченням.

У світі існує більше тисячі різних мов програмування. Багато з них повністю марні. У вас може бути чудова кар'єра, якщо ви володієте 3-4 мовами програмування. Але не лякайтеся, вони не такі складні, як людські мови.

Мови програмування часто описуються своєї парадигмою, яка є способом категоризації видів функцій. Наприклад, JavaScript - мова, яка може бути запущена в будь-якому веб-браузері, також вона заслужено займає

лідируючі позиції в рейтингу популярності. Вона має різноманітний набір функцій: підтримує імперативні, структуровані, об'єктно-орієнтовані і керовані подіями парадигми. Поки що це може звучати дивно для вас, але якщо ви заглибитеся в мови програмування, то зрозумієте, що означають всі ці поняття.

У мов програмування є одна особливість - якщо дві мови мають схожість в парадигмах, то, як правило, вони схожі і в синтаксисі. Після вивчення JavaScript ви зможете зрозуміти до 75% коду на Python або Ruby, так як вони схожі.

Досвідчені розробники розглядають проблеми з точки зору алгоритмів - серії кроків, що потрібно виконати для досягнення певної мети, навіть якщо деталі міняються. Давайте розберемо це на основі реальних прикладів. Розглянемо сервіс покупок Amazon. Кожен раз, коли ви що-небудь купуєте, то робите приблизно однакові кроки: відкладання покупок в корзину, вибір способу оплати, прописування адреси для доставки, вибір швидкості доставки і підтвердження замовлення. Для кожної покупки використовується один і той же код (він же алгоритм в даному випадку). Різняться лише дані (деталі). Отже, можна прийти до думки, що алгоритми можуть бути створені на будь-якій нормальній мові програмування. Ви ж пам'ятаєте, що не всі вони гарні й корисні? Коли ви навчитеся вибудовувати своє мислення у вигляді алгоритмів, то мова програмування буде всього лише вашим інструментом. Інакше кажучи, у вас можуть бути інструменти для створення космічного корабля, але через відсутність правильних думок і ідей у вас виходить створити тільки двері для дому, які повинні бути космічним кораблем. Звичайно, добре знання мови на високому рівні грає важливу роль. Але важливіше вміти описувати правильно всі процеси.

Мова програмування С++

Історія появи мови С++.

Співробітник фірми Bell Labs Денис Рітчі створив мову С (читається "сі") в 1972 р під час спільної роботи з Кеном Томпсоном над операційною системою UNIX. Прообразом послужила мова В ("бі"), розроблена Томпсоном. А свій початок мова В бере від мови APL ("ей пі ель"). Мова С була розроблена як інструмент для програмістів-практиків. Велика частина ОС UNIX була написана на цій мові. У відповідності з потребою вирішувати таку головну задачу як написання коду ОС, метою авторів було створення зручної і корисної мови.

Ці критерії, звичайно, враховувалися і при розробці безлічі інших мов. Але розробка інших мов переслідувала і інші цілі, наприклад:

- Pascal – мова, на основі якої, можна було б навчати фундаментальним основам і принципам програмування;
- Basic - синтаксис мови близький до англійської мови; призначений для швидкого освоєння програмування непрофесіоналами.

В епоху розвитку об'єктно-орієнтованого програмування і появи об'єктних мов, така універсальна мова як С, теж отримала розвиток в цій області. Нова мова, що включає в себе об'єктно-орієнтоване розширення, отримала назву С++ ("сі плас плас" від англійського "c plus plus"; "сі плюс плюс").

Переваги мови С++.

До основних переваг мови С++ необхідно віднести наступні моменти:

- С++ - сучасна мова. Вона включає в себе керуючі конструкції, рекомендовані теоретичним і практичним програмуванням.
- С++ - ефективна. Структура дозволяє найкращим чином використовувати можливості сучасних ЕОМ. Програми відрізняються компактністю і швидкістю виконання.

- С++ - стерпна або мобільна. Програма, написана для однієї обчислювальної системи, може бути перенесена майже без змін на іншу. Компілятори реалізовані майже на 40 типах обчислювальних систем, починаючи від 8-ми розрядних процесорів і закінчуючи CRAY-1 один з найпотужніших суперкомп'ютерів.
- С++ - потужна і гнучка. Велика частина ОС UNIX написана на С.
- С++ - зручна мова. Досить структурована, щоб підтримувати хороший стиль програмування, в той же час не накладає великих обмежень.
- С++ - мова "компілюючого" типу.

Структура програми.

Вихідна програма на мові С складається з наступних частин:

- директив препроцесора;
- вказівок компілятору;
- оголошень;
- визначень.

Ці частини мають різне призначення в тексті програми:

Директиви - специфікують дії препроцесора по перетворенню тексту програми перед компіляцією. Вказівки - це спеціальні інструкції, яким компілятор С++ слідує під час компіляції.

Оголошення - задають ім'я і атрибути даних, їх початкові значення явно або за замовчуванням.

Визначення - функція специфікує тіло функції, яка представляє собою складений оператор (блок операторів), що містить оголошення та оператори.

Оголошення типу - дозволяє програмісту створювати власний тип даних. Для типу поняття оголошення і визначення збігаються. Вихідна програма може містити довільне число директив, вказівок, оголошень і визначень. Але при цьому їх порядок істотний.

Константи в С. Константи використовуються для завдання постійних величин безпосередньо в тексті програми. У мові С++ розрізняють кілька типів констант.

Цілі константи специфікують позитивні значення. Складаються з послідовності цифр.

Знак "-" розглядається як унарна арифметична операція.

Приклади: 134 2 67894.

Довгі цілі позначаються як <цифра> ... <цифра> L. Якщо константа велика для типу *int*, то вона вважається довгою.

Приклади: 134L 2L 67894L.

Крім цього існують правила для запису константи в форматі 8-их і 16-их системах числення:

- якщо число починається з цифри <0> (нуль), то це ціле число задано в 8-ій системі числення.

Приклади: 037 037L 12345L.

Якщо число починається з *0x* то це ціле число задано в 16-ій системі числення.

Приклади: 0x3EA7 0x3D7L 0x12AF4L.

Зауваження: між цифрами числових констант прогалини неприпустимі.

Константи з плаваючою комою або речові константи завжди представляються числом з подвійною точністю, тобто як тип double. У повному форматі ці константи складаються з наступних частин:

- цілої частини - послідовність цифр;
- десяткового дробу;
- дробової частини - послідовність цифр;
- символу експоненти e і E;
- експоненти у вигляді цілої константи (може бути зі знаком "-")

Одна частина (але не обидві відразу) з нижченаведених пар може бути опущена:

- ціла чи дрібна частина;
- десяткова точка або символ e (E) і експонента у вигляді цілої частини.

Приклади: 345. 3.1415926 2.1E5 .123E3 4037e-54.

Експонента позначає, що мантиса числа (дійсне число до символу e) повинна бути помножена на 10 у ступені цього ступеня. Так запис 12.98e-3 буде еквівалентний 12.98 · 10-3 в звичному записі на папері. Слід зазначити, що це ж число можна було записати і як: 1.298e-2 129.8e-4 0.01298 і т.п.

Зауваження: використовувати пробіл під час запису констант забороняється.

Символьні константи - складаються з одного текстового символу укладеного в одинарні апострофи: *'x', 'o', 'Z'*. Деякі символи не мають графічного представлення (спеціальні символи, які неможливо ввести з клавіатури), можна набрати, використовуючи спеціальні комбінації цифрові або символьні. Такі комбінації починаються зі зворотної косої риски, а число вказується в 8-ій системі числення:

- '\ 007' - код символу в 8-ій системі числення;
- '\ n' - код символу новий рядок;
- '\ t' - код символу табуляції;
- '\ 0' - код символу зі значенням 0;
- '\\' - код символу зворотна похила риска;
- '\' '- код символу одинарні лапки;

Вся ця комбінація складається з пари символів або більше, але компілятор замінює її на один символ.

Рядкові константи - це послідовність символів укладена в подвійні лапки.

Необхідно розуміти, що конструкції *'x'* і *"x"* формують різні константи. Справа в тому, що рядкова константа при розміщенні в пам'яті закінчується символом <0> (нуль), як покажчик закінчення тексту. Тому конструкція з одного символу, але зазначена як рядкова константа, зажадає пам'яті в два байта - для самого символу і для завершального нуля.

Мова програмування Python

Однією з важливих переваг мови Python є наявність великої бібліотеки модулів і пакетів, що входять в стандартну поставку. Як кажуть, до Python "прикладені батарейки".

Python - інтерпретована, об'єктно-орієнтована високорівнева мова програмування з динамічною семантикою. Вбудовані високорівневі структури даних в поєднанні з динамічною типізацією і зв'язуванням роблять мову привабливою для швидкої розробки додатків (RAD, Rapid Application Development). Крім того, її можна використовувати в якості сценарної мови для зв'язку програмних компонентів. Синтаксис Python простий у вивченні, в ньому надається особливе значення читання коду, а це скорочує витрати на супровід програмних продуктів. Python підтримує модулі та пакети, заохочуючи модульність і повторне використання коду. Інтерпретатор Python і велика стандартна бібліотека доступні безкоштовно у вигляді вихідних і виконуваних кодів для всіх основних платформ і можуть вільно поширюватися.

У процесі вивчення буде розкритий сенс цього визначення, а зараз досить знати, що Python - це універсальна мова програмування. Вона має свої переваги і недоліки, а також сфери застосування. В поставку Python входить велика стандартна бібліотека для вирішення широкого кола завдань. В Інтернеті доступні якісні бібліотеки для Python по різних предметних областях: засоби обробки текстів і технології Інтернет, обробка зображень, інструменти для створення додатків, механізми доступу до баз даних, пакети для наукових обчислень, бібліотеки побудови графічного інтерфейсу і т.п. Крім того, Python має досить прості засоби для інтеграції з мовами C, C++ (і Java), як шляхом вбудовування (embedding) інтерпретатора в програми на цих мовах, так і навпаки, за допомогою використання бібліотек, написаних на цих мовах, в Python-програмах. Мова Python підтримує кілька парадигм

програмування: імперативне (процедурний, структурний, модульний підходи), об'єктно-орієнтоване і функціональне програмування.

Можна вважати, що Python - це ціла технологія для створення програмних продуктів (і їх прототипів). Вона доступна майже на всіх сучасних платформах (як 32-бітових, так і на 64-бітних) з компілятором C і на платформі Java.

Може здатися, що в програмній індустрії немає місця для чогось іншого крім C/C++, Java, Visual Basic, C #. Однак це не так. Можливо, завдяки цьому курсу лекцій і практичних занять у Python з'являться нові прихильники, для яких він стане незамінним інструментом.

Синтаксис - повністю формалізована частина: його можна описати на формальній мові синтаксичних діаграм (що і робиться в довідкових посібниках). Виявом прагматики є сам інтерпретатор мови. Саме він читає записане відповідно до синтаксису "послання" і перетворює його в дії по закладеному в ньому алгоритму. Неформальним компонентом залишається тільки семантика. Саме в перекладі сенсу в формальний опис і криється найбільша складність програмування. Синтаксис мови Python має потужні засоби, які допомагають наблизити розуміння проблеми програмістом до її "розуміння" інтерпретатором.

Створення Python було розпочато Гвідо ван Россум (Guido van Rossum) в 1991 році, коли він працював над розподіленою ОС Амеба. Йому була потрібна розширювана мова, яка б забезпечила підтримку системних викликів. За основу були взяті ABC і Модула-3. Як назву він вибрав Python в честь комедійних серій BBC "Літаючий цирк Монті-Пайтона", а зовсім не за назвою змії. З тих пір Python розвивався за підтримки тих організацій, в яких Гвідо працював. Особливо активно мова вдосконалюється в даний час, коли над нею працює не тільки команда творців, а й ціле співтовариство програмістів з усього світу. І все-таки останнє слово про напрямок розвитку мови залишається за Гвідо ван Россум.

Програма на мові Python може складатися з одного або декількох модулів. Кожен модуль являє собою текстовий файл в кодуванні, сумісний з 7-бітним кодуванням ASCII. Для кодувань, що використовують старший біт, необхідно явно вказувати назву кодування. Наприклад, модуль, коментарі або рядкові літерали якого записані в кодуванні KOI8-R, повинен мати в першому або другому рядку наступну специфікацію:

```
# -*- coding: koi8-r -*-
```

Завдяки цій специфікації інтерпретатор Python знатиме, як коректно переводити символи літералів Unicode-рядків в Unicode. Без цього рядка нові версії Python будуть видавати попередження на кожен модуль, в якому зустрічаються коди з встановленим восьмим бітом.

Послідовні дії описуються послідовними рядками програми. Варто, щоправда, додати, що в програмах важливі відступи, тому всі оператори, що входять у послідовність дій, повинні мати один і той же відступ:

```
a = 1
b = 2
a = a + b
b = a - b
a = a - b
print a, b
```

При роботі з Python в інтерактивному режимі якби вводиться одна велика програма, що складається з послідовних дій. В наведеному вище прикладі використані оператори присвоювання і оператор print.

Зрозуміло, одними тільки послідовними діями в програмуванні не обійтися, тому при написанні алгоритмів використовується ще й розгалуження:

```
if a > b:
 c = a
 else:
 c = b
```

Цей шматок коду на Python інтуїтивно зрозумілий кожному, хто пам'ятає, що *if* - на англійській мові означає "якщо", а *else* - "інакше". Оператор розгалуження має в даному випадку дві частини, оператори кожної з яких записуються з відступом вправо щодо оператора розгалуження. Більш загальний випадок - оператор вибору – можна записати за допомогою наступного синтаксису (приклад обчислення знака числа):

```
if a < 0:
s = -1
elif a == 0:
s = 0
else:
s = 1
```

Варто зауважити, що *elif* - це скорочений *else if*. Без скорочення довелося б застосовувати вкладений оператор розгалуження:

```
if a < 0:
s = -1
else:
if a == 0:
s = 0
else:
s = 1
```

На відміну від оператора *print*, оператор *if-else* - складовий оператор.

Третьою необхідною алгоритмічною конструкцією є цикл. За допомогою циклу можна описати повторювані дії. В Python є два види циклів: цикл ПОКИ (виконується деяка умова) і цикл ДЛЯ (всіх значень послідовності). Наступний приклад ілюструє цикл ПОКИ на Python:

```
s = "abcdefghijklmnop"
while s != "":
print s
s = s[1:-1]
```

Оператор while говорить інтерпретатору Python: "поки вірно умова циклу, виконувати тіло циклу". У мові Python тіло циклу виділяється відступом. Кожне виконання тіла циклу буде називатися ітерацією. У

наведеному прикладі забирається перший і останній символ рядка до тих пір, поки не залишиться порожній рядок.

Для більшої гнучкості при організації циклів застосовуються оператори *break* (перервати) і *continue* (продовжити). Перший може скасовувати цикл, а другий - продовжити цикл, перейшовши до наступної ітерації (якщо, звичайно, виконується умова циклу).

Наступний приклад читає рядки з файлу і виводить ті, у яких довжина більше 5:

```
f = open("file.txt", "r")
while 1:
l = f.readline()
if not l:
break
if len(l) > 5:
print l,
f.close()
```

У цьому прикладі організований нескінченний цикл, який переривається тільки при отриманні з файлу порожнього рядка (l), що позначає кінець файлу.

У мові Python логічне значення несе кожен об'єкт: нулі, порожні рядки і послідовності, спеціальний об'єкт None і логічний літерал False мають значення "брехня", а інші об'єкти значення "істина". Для позначення істини зазвичай використовується 1 або True.

Цикл ДЛЯ виконує тіло циклу для кожного елемента послідовності. В наступному прикладі виводиться таблиця множення:

```
for i in range(1, 10):
for j in range(1, 10):
print "%2i" % (i*j),
print
```

Тут цикли *for* є вкладеними. Функція *range* () породжує список цілих чисел з напіввідкритого інтервалу (1, 10). Перед кожною ітерацією лічильник циклу отримує чергове значення з цього списку. Напіввідкриті діапазони

загальноприйняті в Python. Вважається, що їх використання більш зручне і викликає менше помилок у програмістів. Наприклад, *range (len (s))* породжує список індексів для списку *s* (в Python-послідовності перший елемент має індекс 0). Для красивого виведення таблиці множення застосована операція форматування відсотків (для цілих чисел той же символ використовується для позначення операції взяття остачі від ділення). Рядок форматування (задається зліва) будується майже як рядок форматування для *printf* з C.

Програміст може визначати власні функції двома способами: з допомогою оператора def або прямо в вираженні, за допомогою *lambda*.

```
def cena(rub, kop=0):
return "%i руб. %i коп." % (rub, kop)
print cena(8, 50)
print cena(7)
print cena(rub=23, kop=70)
```

У цьому прикладі визначена функція двох аргументів (у тому числі друга має значення за замовчуванням - 0). Варіантів виклику цієї функції з конкретними параметрами також кілька. Варто лише зауважити, що при виконанні функції спочатку повинні йти позиційні параметри, а потім, іменовані. Аргументи зі значеннями за замовчуванням повинні слідувати після звичайних аргументів. Оператор *return* повертає значення функції. З функції можна повернути тільки один об'єкт, але він може бути кортежем з кількох об'єктів.

Основи мови програмування APDL ANSYS

Програма на командній мові APDL ANSYS являє собою послідовність команд, записаних в звичайному тестовому файлі.

Оскільки ANSYS є інтерпретатором, то команди виконуються послідовно з урахуванням операторів змін до порядку виконання, що задаються операторами циклу і переходу. Якщо ж при виконанні програми на мові APDL ANSYS виникає помилка, то видається повідомлення про помилку, і, як правило, програма далі не функціонує належним чином і в подальшому призводить до зупинки.

Мова APDL ANSYS схожа на мову FORTRAN, яка зараз, проте, не так добре відома і популярна, як раніше.

Команда APDL ANSYS записується в окремому рядку, причому максимальне число символів не повинно перевищувати 80. Команда складається з імені команди і набору аргументів, або операндів, відокремлених один від одного і від імені команди запитом. Рядкові і заголовні букви еквіваленти.

Імена команд можуть складатися з набору символів від 1 до 8. Стандартні команди APDL мають зарезервовані імена, з яких перші 4 символи є значущими (не рахуючи початкових символів / і *).

Таким чином, наприклад, записи /SOLU, /SOLUT і /SOLUTION еквіваленти.

Деякі команди є макрокомандами (тобто складаються з послідовно належним чином оформлених команд APDL). Ряд макрокоманд входять в комплект поставки програмного забезпечення ANSYS, але макроси можуть бути створені і користувачем. Імена макрокоманд повинні вводитися повністю без скорочень.

Формат введення команд вільний, тобто прогалини не враховуються. При введенні команд застосовується правило умовчання. За цим правилом,

якщо якийсь з аргументів не заданий явно, програма намагається привласнити цьому аргументу значення за замовчуванням.

Наприклад, ANSYS завжди працює з об'єктами, які задаються в тривимірному просторі. Але якщо вирішується плоска задача, то третя координата не використовується і за замовчуванням дорівнює нулю. Аналогічно, для одновимірних задач не потрібна і друга координата. Тому, якщо аргументи координат не задані, то ANSYS приймає їх рівними значенням за замовчуванням. Ці значення вказуються для кожної команди в керівництві по командам ANSYS (Command Reference). У більшості випадків (але не завжди!) ці значення рівні нулю.

Тому команди K, 1,10,0,2 і K, 1,10,, 2 еквіваленти; команда N, 2,3.5, в дійсності означає, що вузлу з номером 2 присвоєні координати (3.5,0,0) у відповідній системі координат, і т.д.

Правила умовчання прийняті для більшості команд ANSYS, є зазвичай «найприродніше розумними», дозволять часто не вводити великого числа аргументів.

Кілька команд можна записати в одному рядку, відділяючи їх один від одного знаком $, наприклад:

L, 1,2 $ L, 2,3 $ L, 3,4.

Деякі команди починаються зі знаку «/», і зазвичай призначені для управління основною програмою і контролю. Є ще команди, що починаються зі знаку «*». Зазвичай це і є команди власне мови APDL.

Коментарі починаються зі знаку «!» або, на початку рядка, з символів /COM, або C ***,. Відмінність полягає в тому, що коментарі, записані в полі /COM, або C ***, будуть міститися і файлі виведення ANSYS.

У APDL ANSYS зручно використовувати параметри або змінні. При визначенні параметрів в ANSYS не потрібно вказувати їх тип, за винятком багатовимірних даних у формі масивів.

Всі чисельні змінні мають цілий або дійсний тип і зберігаються з подвійною точністю. Якщо параметру не присвоєно ніякого значення, то

вважається, що він дорівнює нулю. (Тому, якщо розділити на невизначений раніше параметр будь-яке число, то програма видає помилку.) Є також символьні параметри, які використовуються для завдання імен файлів, рядків виводу, заголовків тощо. Значення цих параметрів повинні бути укладені в одинарні апострофи. Букви кирилиці в них не допускаються.

Масиви, які перед використанням потрібно обов'язково визначати, можуть мати чисельний, текстовий або табличний типи.

Імена параметрів повинні починатися з літери і містити не більше 32 знаків, що включають тільки букви, цифри і символ підкреслення. Бажано, щоб ім'я параметра не збігалося з зарезервованими іменами ANSYS для команд, ступенів свободи, ключових слів і т.п.

Для визначення параметрів ANSYS пропонує використовувати або команду * SET, або звичайний оператор рівності. Остання можливість здається найбільш природною. Таким чином, синтаксис оператора завдання параметра виглядає наступним чином:

Name = *Value* - де *Name* - ім'я параметра, *Value* - його значення, яке може бути чисельною величиною, рядком (в апострофі), ім'ям іншого параметра, математичним параметричним виразом або функцією. Приклади завдання параметрів наведені нижче:

```
HL = 0.4
F_R = 'Mod_ANS_1'
RO1 = 7.86e3
PEL = (HAL / 2) / (WL-WWL)
```

Для визначення параметрів, пов'язаних з величинами з бази даних ANSYS, що містить опис твердотільних і кінцево-елементних моделей, надзвичайно корисна команда * GET. Опис цієї команди займає кілька сторінок фірмового керівництва. Наступні приклади ілюструють приклади використання цієї важливої команди:

PLNSOL, U, X! Виведення переміщень по осі X.

**** GET, UXMAX, PLNSOL, 0, MAX! UXMAX - максимальні переміщення по осі X***

**** GET, EL_MAX, ELEM,, COUNT! EL_MAX - число елементів***

**** GET, ND_MAX, NODE,, COUNT! ND_MAX - число вузлів***

ETAB, S11, S, X

**** Get, c11_el, elem, ii, etab, s11***

Деякі з корисних варіантів використання команди * GET реалізовані також в скороченій формі GET-функцій.

Параметричні вираження APDL ANSYS мають точно такий же синтаксис, як в мові FORTRAN. При цьому допустимими операторами є наступні: додавання (+), віднімання (-), множення (*), ділення (/), зведення в ступінь (**), порівняння «менше ніж» (<) і порівняння «більше ніж» (>).

Застосовується стандартний порядок виконання операцій:

1) дії всередині дужок;
2) піднесення до ступеня (в порядку зліва направо);
3) множення і ділення (в порядку зліва направо);
4) додавання і віднімання (в порядку зліва направо);
5) логічна рівність (в порядку зліва направо).

Очевидно, щоб бути впевненим у правильності порядку виконання операцій, потрібно просто використовувати дужки і розбивати складні вирази на більш прості.

Деякі математичні функції, допустимі в програмі ANSYS, зібрані в таблиці.

Позначення функції	Дія
ABS(x)	Абсолютне значення величини *x*.

SIGN(x,y)	Абсолютне значення *x* зі знаком *y*. При *y = 0* результат невід'ємний
EXP(x)	Експонента від *x* (e^x).
LOG(x)	Натуральний логарифм *x* (*ln (x)*).
LOG10(x)	Десятковий логарифм *x* ($log_{10}(x)$).
SQRT(x)	Квадратний корінь із *x*.
NINT(x)	Найближче ціле до *x*.
MOD(x,y)	Залишок від часткового *x/y*. При *y = 0* повертається значення (0).
RAND(x,y)	Випадкова величина (має постійний розподіл) в межах від *x* до *y* (*x*= нижня межа, *y* = верхня межа).
GDIS(x,y)	Випадкова величина з Гауссовим (нормальним) розподілом з середнім значенням *x* і стандартним відхиленням *y*.
SIN(x), COS(x), TAN(x)	Синус, косинус і тангенс *x*. За замовчуванням *x* задається в радіанах, але одиниці виміру *x* можуть бути змінені командою *AFUN.

Крім скалярних параметрів в програмах ANSYS можуть використовуватися масиви параметрів. Масиви можуть мати розмірності від 1 до 5 і мати такі типи: ARRAY (цілі і дійсні числа), CHAR (текстовий), TABLE і STRING (строковий).

Тип TABLE є спеціальним типом числового масиву з автоматичним виконанням процедури лінійної інтерполяції між елементами масиву. Масиви визначаються командою *DIM.

Для зміни порядку виконання команд в програмі APDL ANSYS пропонує стандартні можливості програмування:

- оператор безумовного переходу *GO
- умовний оператор (команда *IF)

- оператор повторення (команда *REPEAT)
- оператор циклу (команда *DO)
- додаткове управління в операторі циклу (команда *DOWHILE)

Оператор безумовного переходу може бути використаний в такий спосіб:

```
*GO, :LABEL1
- - -
- - -
LABEL1
- - -
```

Оператор умовного переходу *IF має наступний синтаксис:

IF,** ***VAL1, Oper, VAL2, Base

де

VAL1 - перший порівнюваний вираз;

Oper - оператор порівняння, задається ключовим словом;

VAL2 - другий порівнюваний вираз;

Base - ключове слово, що визначає дію, яка буде відбуватися при виконанні умови.

Оператори порівняння Oper можуть бути наступними: EQ - логічна рівність; NE - логічна нерівність; LT - логічне менше ніж; GT - логічне більше ніж; LE - логічне менше або дорівнює; GT - логічне більше або дорівнює; ABLT - логічне менше за абсолютним значенням; ABGT - логічне більше за абсолютним значенням.

Значним Base може бути ключове слово THEN, що перетворює команду *IF в конструкцію IF_THEN-ELSE.

Після виконання блоку, наступного за THEN, можуть розташовуватися ключові слова: *ELSEIF (може бути відсутнім) і наступний за ним блок; *ELSE (може бути відсутнім) і наступний за ним блок; *ENDIF (повинен бути присутнім обов'язково).

Наступний приклад ілюструє дану конструкцію:

```
*IF,A,GT,1,THEN
  ! Block 1
   ---
  ----
  *ELSEIF,A,GT,0
  ! Block 2
   ---
  ----
  *ELSEIF,A,EQ,0
  ! Block 3
   ---
  ----
  *ELSE
  ! Block 4
   ---
  ----
*ENDIF
```

Оператор циклу *DO призначений для виконання блоку команд певне число раз. Цей блок команд повинен бути укладений між командами *DO і *ENDDO. Наступний привіт ілюструє можливість застосування оператора циклу:

```
*DO,II,1,10
  ---
  ---
*ENDDO
```

Таким чином, мова APDL є цілком розвиненим засобом програмування, що дозволяє разом з набором команд ANSYS створювати призначені для користувача програми, що реалізують різні типи кінцево-елементного аналізу.

Мова програмування MATLAB

В наші дні комп'ютерна математика отримала належну популярність і інтенсивно розвивається як передовий науковий напрямок на стику математики та інформатики. Програмовані мікрокалькулятори, а потім вже і персональні комп'ютери давно застосовуються для математичних розрахунків. Для підготовки програм таких розрахунків ще зовсім недавно використовувалися різні універсальні мови програмування. Але вже на початку 90-х років на зміну їм прийшли спеціалізовані системи комп'ютерної математики (СКМ). Найбільшу популярність здобули системи Eureka, Mercury, Mathcad, Mathematica, Maple та ін. Щорічно з'являються їх нові версії.

Серед ряду сучасних СКМ особливе місце займає математична система MATLAB. Зо два десятки років робота сотень вчених і програмістів була спрямована на постійне розширення його можливостей і вдосконалення закладених алгоритмів. В даний час з його бібліотекою чисельних методів ні за обсягом, ні за якістю не може зрівнятися жодна з систем.

Система MATLAB увібрала в себе не тільки передовий досвід розвитку та комп'ютерної реалізації чисельних методів, накопичений за останні три десятиліття, але і весь досвід становлення математики за всю історію людства. На базі ядра MATLAB створені численні розширення, щоб забезпечити моделювання і аналіз систем в різноманітних сферах людської діяльності. Спектр проблем, дослідження яких може бути здійснено за допомогою MATLAB і його розширень, охоплює: матричний аналіз, обробку сигналів і зображень, проектування і аналіз радіочастотних ланцюгів, завдання математичної фізики, оптимізаційні задачі, фінансові завдання, обробку та візуалізацію даних, роботу з картографічними зображеннями, нейронні мережі, нечітку логіку і багато іншого.

Її охоче використовують в своїх наукових проектах провідні університети і наукові центри світу. Для сучасного інженера і науково-

технічного працівника MATLAB є незамінним інструментом моделювання та дослідження різних прикладних систем, перш за все, за рахунок використання готових рішень. У пакеті MATLAB ретельно відпрацьовані засоби візуалізації результатів обчислень і відображення різних графічних об'єктів.

Система MATLAB створена таким чином, що будь-які (іноді дуже складні) обчислення можна виконувати в режимі прямих обчислень, тобто без підготовки програми. Це перетворює MATLAB в надзвичайно потужний калькулятор, який здатний виробляти не тільки звичайні для калькуляторів обчислення (наприклад, виконувати арифметичні операції і обчислювати елементарні функції), але і операції з векторами і матрицями, комплексними числами, рядами і поліномами. Можна майже миттєво задати і вивести графіки різних функцій - від простої синусоїди до складної тривимірної фігури.

Робота з системою в режимі прямих обчислень (або в командному режимі) носить діалоговий характер і відбувається за правилом «задав питання - отримав відповідь». Користувач набирає на клавіатурі обчислювальний вираз, редагує його (якщо потрібно) в командному рядку і завершує введення натисканням клавіші ENTER.

Середовище розробки MATLAB.

Навіть з таких простих прикладів можна зробити деякі висновки:

- для вказівки введення вихідних даних використовується символ >>;
- дані вводяться за допомогою найпростішого рядкового редактора;
- для блокування виведення результату обчислень деякого виразу після нього треба встановити знак «;» (крапка з комою);
- якщо не вказана змінна для значення результату обчислень, то MATLAB призначає таку змінну з ім'ям *ans*;

- знаком присвоювання є звичний математикам знак рівності =, а не комбінований знак: =, як у багатьох інших мовах програмування і математичних системах;

- вбудовані функції (наприклад *sin*) записуються малими буквами, і їхні аргументи вказуються в круглих дужках;

У деяких випадках математичний вираз, що вводиться, може виявитися настільки довгим, що для нього не вистачить одного рядка. У цьому випадку частину виразу можна перенести на новий рядок за допомогою знака крапки «...» (3 або більше точок), наприклад:

```
c = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 - ...
1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

Цей прийом може бути вельми корисним для створення наочних документів, у яких не допускається захід рядків в невидиму область вікна.

Основні об'єкти MATLAB.

Центральним поняттям всіх математичних систем є математичний вираз. Він задає те, що повинно бути обчислено в чисельному (рідше символьному) вигляді. Ось приклади простих математичних виразів:

```
2+3
2.301*sin(x)
4+exp(3)/5
sqrt(y)/2
sin(pi/2)
```

Математичні вирази будуються на основі чисел, констант, змінних, операторів, функцій і різних спецзнаків. Нижче даються короткі пояснення суті цих понять.

Число - найпростіший об'єкт мови MATLAB, що представляє кількісні дані. Числа можна вважати константами, імена яких збігаються з їх значеннями. Числа використовуються в загальноприйнятому поданні для них. Вони можуть бути цілими, дробовими, з фіксованою і плаваючою крапкою. Можливо уявлення чисел в добре відомому науковому форматі із

зазначенням мантиси і порядку числа. Нижче наводяться приклади представлення чисел:

```
0
2 -3
2.301
0.00001
123.456e-24 -234.456e10
```

Як неважко помітити, в мантисі чисел ціла частина відділяється від дробової НЕ комою, а точкою, як прийнято в більшості мов програмування. Для відділення порядку числа від мантиси використовується символ *e*. Знак «плюс» у чисел не проставляється, а знак «мінус» у числа називають унарним мінусом. Прогалини між символами в числах не допускаються. Числа можуть бути комплексними: *z = Re (z) + Im (z)* * і. Такі числа містять дійсну *Re (z)* і уявну *Im (z)* частини. Уявна частина має множник *i* або *j*, що означає корінь квадратний з *-1*:

```
3i
2+3i
-123.456+2.7e-3i
```

Функція *real (z)* повертає дійсну частину комплексного числа, *Re (z)*, а функція *imag (z)* - уявну, *Im (z)*. Для отримання модуля комплексного числа використовується функція *abs (z)*, а для обчислення фази - *angle (Z)*. Нижче дані найпростіші приклади роботи з комплексними числами:

```
>> i ans =
0 + 1.0000i
>> j ans =
        0 + 1.0000i >> z=2+3i
z =
2.0000 + 3.0000i
>>abs(z) ans =
3.6056
>>real(z) ans =
2
```

```
>>imag(z) ans =
3
>>angle(z) ans =
0.9828
```

В MATLAB операції над числами зазвичай виконуються в форматі, який прийнято вважати форматом з подвійною точністю. Такий формат задовольняє переважній більшості вимог до чисельних розрахунків.

За замовчуванням MATLAB видає числові результати в нормалізованій формі з чотирма цифрами після десяткової точки і однієї до неї. Багатьох така форма уявлення не завжди влаштовує. Тому при роботі з числовими даними можна задавати різні формати представлення чисел. Однак в будь-якому випадку всі обчислення проводяться з граничною, так званою подвійною точністю. Для установки формату представлення чисел використовується команда `>> format name`.

де *name* - ім'я формату. Для числових даних *name* може бути наступним повідомленням:

- *short* - коротке представлення в фіксованому форматі (5 знаків);
- *short e* - коротке представлення в експоненційному форматі (5 знаків мантиси і 3 знаки порядку);
- *long* - довге подання у фіксованому форматі (15 знаків);
- *long e* - довге подання в експоненційному форматі (15 знаків мантиси і 3 знаки порядку);
- *hex* - уявлення чисел в шістнадцятковій формі;
- *bank* - представлення для грошових одиниць.

Для ілюстрації різних форматів розглянемо число $x = 4/3$. У різних форматах воно матиме такий вигляд:

```
format short     1.3333
format short e   1.3333E+000
format long      1.333333333333338
format long e    1.333333333333338E+000
```

```
format bank       1.33.
```

Завдання формату позначається тільки на формі виведення чисел. Обчислення все одно відбуваються в форматі подвійної точності, а введення чисел можливо в будь-якому зручному для користувача вигляді.

Константи і системні змінні.

Константа - це попередньо визначене числове або символьне значення, представлене унікальним ім'ям. Числа (наприклад, 1, -2 і 1.23) є безіменними числовими константами.

Інші види констант в MATLAB прийнято назвати системними змінними, оскільки, з одного боку, вони задаються системою при її завантаженні, а з іншого - можуть перевизначатися. Основні системні змінні, що застосовуються в системі MATLAB, вказані нижче:

i чи *j* - уявна одиниця (корінь квадратний з -1); *Pi* - число π=3.1415926...;

eps - похибка операцій над числами з плаваючою точкою (2-52);

realmin - найменше число з плаваючою точкою (2-1022);

realmax - найбільше число з плаваючою точкою (21023)

inf - значення машинної нескінченності;

ans - змінна, що зберігає результат останньої операції і зазвичай викликає його відображення на екрані дисплея;

NaN - вказівка на нечисловий характер даних *(Not-a-Number).*

Ось приклади застосування системних змінних:

```
>>2*pi ans =
6.2832
>>eps ans =
2.2204e-016
>>realmin ans =
2.2251e-308
>>realmax ans =
1.7977e+308
>>1/0
```

```
Warning: Divide by zero. ans =
   Inf >> 0/0
Warning: Divide by zero. ans =
NaN
```

Як зазначалося, системні змінні можуть перевизначатися. Можна задати системній змінній *eps* інше значення, наприклад *eps = 0.000l*. Однак важливо те, що їх значення за замовчуванням задаються відразу після завантаження системи. Тому невизначеними, на відміну від звичайних змінних, системні змінні не можуть бути ніколи.

Змінні.

Змінні - це об'єкти, що мають імена, здатні зберігати деякі, звичайно різні за значенням, дані. Залежно від цих даних змінні можуть бути числовими або символьними, векторними або матричними. В системі MATLAB можна задавати змінним певні значення. Для цього використовується операція присвоювання, що вводиться знаком рівності.

Типи змінних заздалегідь не декларуються. Вони визначаються виразом, значення якого присвоюється змінній. Так, якщо цей вислів - вектор або матриця, то змінна буде векторною або матричною.

Ім'я змінної (її ідентифікатор) може містити скільки завгодно символів, але запам'ятовується і ідентифікується тільки 31 початковий символ. При цьому істотний регістр, ім'я будь-якої змінної не повинно збігатися з іменами інших змінних, функцій і процедур системи, тобто воно повинно бути унікальним. Ім'я повинно починатися з літери, може містити букви, цифри і символ підкреслення _. Неприпустимо включати в імена змінних прогалини і спеціальні знаки, наприклад +, -, *, / і т. д., оскільки в цьому випадку правильна інтерпретація виразів стає неможливою.

Бажано використовувати змістовні імена для позначень змінних, наприклад *speed_l* для змінної, що позначає швидкість першого об'єкта. Змінні можуть бути звичайними і індексованими, тобто елементами векторів

або матриць. Можуть використовуватися і символьні змінні, причому символьні значення полягають в апострофі, наприклад *s = 'Demo'*.

Оператори і функції.

Оператор - це спеціальне позначення для певної операції над даними - операндами. Наприклад, найпростішими арифметичними операторами є знаки суми +, віднімання -, множення * і розподілу /. Оператори використовуються спільно з операндами. Наприклад, у виразі 2 + 3 знак + є оператором складання, а числа 2 і 3 - операндами.

Слід зазначити, що більшість операторів відноситься до матричних операцій, що може служити причиною серйозних непорозумінь. Наприклад, оператори множення * і розподілу / обчислюють твір і частка від ділення двох масивів, векторів або матриць. Є ряд спеціальних операторів, наприклад, оператор \ означає розподіл справа наліво, а оператори .* і ./ означають, відповідно, поелементне множення і поелементне ділення масивів. Повний список операторів можна отримати, використовуючи команду

```
>> help ops
```

Поступово ми розглянемо всі оператори системи MATLAB і обговоримо особливості їх застосування. А поки наведемо лише частину повного списку операторів, що містить арифметичні оператори:

+ складання,

- віднімання,

* матричне множення,

/ матричний розподіл,

\ зворотний розподіл,

^ піднесення до ступеня,

' транспонування,

. * поелементне множення,

./ поелементне ділення,

.\ зворотне поелементне ділення,

.^ поелементне спорудження до рівня,

. ' поелементне транспонування.

Пріоритет у виконанні арифметичних операцій звичайний: спочатку (тобто найвищий пріоритет) - зведення в ступінь, потім - множення і ділення, і потім - додавання і віднімання. Операції однакового пріоритету виконуються зліва направо, але круглі дужки можуть змінити цей порядок.

Функції – це об'єкти, що мають унікальні імена, які виконують певні перетворення своїх аргументів і при цьому повертають результати цих перетворень. Повернення результату - відмінна риса функцій. При цьому результат обчислення функції з одним вихідним параметром підставляється на місце її виклику, що дозволяє використовувати функції в математичних виразах, наприклад функцію *sin* в *2 * sin (pi/2).*

Функції в загальному випадку мають список аргументів (параметрів), взятих у круглі дужки. Наприклад, функція Бесселя записується як *bessel (NU, X)*. В даному випадку список параметрів містить два аргументи - *NU* у вигляді скаляра і *X* у вигляді вектору. Багато функцій допускають ряд форм запису, що відрізняються списком параметрів. Якщо функція повертає кілька значень, то вона записується у вигляді

```
[Y1,Y2,…]=func(X1,X2,…)
```

де *Y1, Y2,* ... - список вихідних параметрів і *X1,* X2, ... - список вхідних аргументів (параметрів). Зі списком елементарних функцій можна ознайомитися, виконавши команду *help elfun*, а зі списком спеціальних функцій - за допомогою команди *help specfun*. Функції можуть бути вбудованими (внутрішніми) і зовнішніми, або *T*-функціями. Так, вбудованими є найбільш розповсюджені елементарні функції, наприклад, *sin(x)* і *exp(y)*, тоді як функція *sinh(x)* є зовнішньою функцією. Зовнішні функції містять свої визначення в *m*-файлах.

Функції користувача.

Хоча в ядро MATLAB останніх версій вбудовано вже близько тисячі операторів і функцій, користувачеві завжди може знадобитися та чи інша

функція, проста або складна, відсутня в ядрі. Мова програмування систем MATLAB надає ряд ефективних можливостей для завдання функцій користувача.

Одна з таких можливостей полягає в застосуванні функції *inline*, аргументом якої треба в апострофах задати вираз, що задає функцію однієї або декількох змінних. У наведеному нижче прикладі задана функція двох змінних - суми квадратів *sin(x)* і *cos(y)*:

```
>> sc2 = inline('sin(x).^2+cos(y).^2') sc2 =
Inline function:
sc2(x,y) = sin(x).^2+cos(y).^2
```

Можна також задавати свої функції у вигляді *m*-файлів. Наприклад, можна у вікні редактора *m*-файлів (відкривається командою *New* в меню *File*) створити *m*-файл з ім'ям *sc2* і лістингом:

```
function y=sc2(x,y)
y=sin(x).^2+cos(y).^2;
```

Записавши його на диск, можна командою *type sc2* вивести лістинг створеної функції:

```
>> type sc2 function y=sc2(x,y)
y=sin(x).^2+cos(y).^2;
```

Звернення до функції, створеної описаними методами, задається як *sc2(x, y)*, де на місце *x* і *y* підставляються значення змінних - аргументів функції користувача. Наприклад:

```
>>sc2(1,2)
ans =
0.8313
>>sc2(2,1)
ans =
1.1187
```

Мова програмування Maple

Системи класу Maple були створені корпорацією Waterloo Maple, Inc. (Канада) як системи комп'ютерної алгебри (СКА) з розширеними можливостями в області символьних (аналітичних) обчислень. Уже перші версії системи Maple V показали себе лідерами в області символьних обчислень. Ядро і вбудовані пакети розширення цих систем налічували до 3500 вбудованих функцій для виконання різних обчислень і символьних перетворень. На відміну від мов програмування високого рівня, Maple може вирішувати велику кількість математичних задач шляхом введення команд, без будь-якого додаткового програмування. Крім того, Maple може оперувати не тільки наближеними числами, а й точними цілими і раціональними числами. Рішення задач може бути отримано аналітично, тобто у вигляді формул, що складаються з математичних символів. Внаслідок цього Maple називають пакетом символьної математики.

Головною перевагою системи Maple є її здатність виконувати арифметичні дії. При роботі з дробами і корінням вони не наводяться в процесі обчислення до десяткового виду, що дозволяє уникнути помилок при округленні. При необхідності роботи з десятковими еквівалентами в системі Maple є команда, апроксимуюча значення виразу в форматі чисел з плаваючою комою. Система Maple обчислює кінцеві і нескінченні суми і твори, виконує обчислювальні операції з комплексними числами, легко призводить комплексне число до числа в полярних координатах, числові значення елементарних функцій, а також багатьох спеціальних функцій і констант.

Розробники інших відомих математичних пакетів, таких як MathCad і MathLab використовують символьний процесор Maple в своїх програмах.

Maple - типова інтегрована програмна система. Вона об'єднує в собі:

- потужна мова програмування (вона ж мова для інтерактивного спілкування з системою);

- редактор для підготовки і редагування документів і програм;
- сучасний багатовіконний інтерфейс з можливістю роботи в діалоговому режимі;
- потужну довідкову систему з багатьма тисячами прикладів;
- словник математичних понять і термінів з алфавітною організацією;
- ядро алгоритмів і правил перетворення математичних виразів;
- чисельний і символьний програмні процесори;
- систему діагностики;
- бібліотеки вбудованих і додаткових функцій;
- пакети розширення як вбудовані, так і сторонніх виробників;
- засоби підтримки деяких мов програмування та інтеграції з широко поширеними програмами.

Центральне місце в структурі Maple займає ядро системи, яке складається з безлічі заздалегідь відкомпільованих функцій і процедур, представлених в машинних кодах і забезпечують досить представницький набір вбудованих функцій і операторів системи. Специфікою СКА є наявність в ядрі безлічі правил перетворень математичних виразів і функцій і їх визначень в символьному вигляді.

Ядро СКМ ретельно оптимізується, оскільки від цього залежить швидкість обчислень, які забезпечуються тією чи іншою системою комп'ютерної математики. Цьому сприяє і компіляція ядра. Доступ в ядро користувача для його модифікації, як правило, виключений. Обсяг ядра досягає декількох мегабайт. Пишеться ядро на мові реалізації системи - в Maple це мова *C*.

Основний режим роботи в пакеті - режим командного рядка або інтерактивний режим. При завантаженні програми автоматично завантажується новий робочий лист (worksheet), на якому є запрошення для введення команди >. В командний рядок можна записати будь-яке вираження алгебри. Якщо в кінці виразу поставити знак ;, то при натисканні клавіші

Enter вираз буде оброблено програмою, а результат виведений на дисплей, наприклад:

```
>2*3^5-x^2*sin(y-Pi);
```

$486+x^2\sin(y)$

Таким чином, можна отримувати обчислені значення виразів, введених в командний рядок, тобто працювати з програмою, як з калькулятором. Також можна присвоювати імена, що вводяться виразами за допомогою оператора присвоювання

```
>r:=5*sin(y);
```

$r:=5\sin(y)$

Тепер можна вивести попередній вираз, записавши

```
>r;
```

$5\sin(y)$

Алфавіт мови і його синтаксис.

Визначення мови можна розбити на 4 частини: символи (characters), висловлювання (tokens), синтаксис (syntax) і семантика (semantics) - тлумачення.

Алфавіт мови містить 26 великих і малих латинських букв (від *a* до *z*) (великі та малі літери розрізняються) 10 арабських цифр (від 0 до 9) і 32 спеціальних символів:

- арифметичні оператори +, -, *, /;
- знак зведення в ступінь ^;
- 5 пар альтернативних символів: ^ - **, [- (|,] - |), {- (*,} - *);
- : - фіксатор виразу, що запобігає висновок результату обчислення в комірку виводу;

- ; - фіксатор виразу, що дає висновок результату обчислення в комірку виводу;
- # - покажчик програмного коментаря;
- '- обмежувач рядка;
- : = - оператор присвоювання;
- ;; - порожній оператор;
- :: - покажчик типу змінної (n :: integer або z :: complex).

Висловлюваннями (лексемами) є ключові слова, оператори програмування, рядки, натуральні числа і знаки пунктуації.

Зарезервовані слова - мають спеціальне значення і їх не можна застосовувати в якості ідентифікаторів.

Оператори програмної мови. Є 3 типи операторів (binary, unary, nullary): двомісні (бінарні), одномісні (унарні), і нульарні - не мають операндів. Останніх всього 3 (ditto-оператори) звернення до попереднього обчислення (%, %%, %%%)

Висловлювання можна розділяти порожніми операторами або знаками пунктуації.

Порожні розділювачі - це прогалини, знаки табуляції і повернення каретки. Прогалини можна використовувати всередині висловлювань (лексем), але можна між лексемами. У рядках, взятих в зворотні лапки, вони стають частиною висловлювання.

```
>a*x+x*y; # коментар
```

$ax+xy$

Перейти на новий рядок з продовженням записи команди можна, натиснувши клавіші Shift-Enter.

Мова програмування Fortran

Мова Фортран займає лідируюче положення серед мов програмування, орієнтованих на вирішення науково-технічних завдань, що вимагають великого обсягу обчислень. Особливо актуальним є застосування Фортран при вирішенні великомасштабних обчислювальних задач з використанням сучасних супер ЕОМ. Рішення таких завдань потрібно в різних сферах фундаментальних наукових досліджень і в багатьох прикладних областях.

Фортран постійно розвивається і вдосконалюється відповідно до розвитку обчислювальної техніки, мов і технології програмування. Однією з найбільш важливих причин популярності і живучості Фортран є величезний фонд прикладних програм, який накопичено за десятиліття існування мови. Цьому в значній мірі сприяли зручність і ефективність виконання програм, а також стандартизація мови на міжнародному рівні.

Розробка нових мобільних засобів мови є результатом фундаментальних досліджень (виконуваних фахівцями багатьох країн) в області теорії і практики програмування, в області розвитку архітектури сучасних обчислювальних систем і технології програмування, а також є результатом аналізу та узагальнення досвіду використання обчислювальної техніки при вирішенні завдань, що вимагають великого обсягу обчислень.

Існуюча думка про те, що Фортран застарів - помилкова. Звичайно, в зв'язку з широким впровадженням персональних ЕОМ в багатьох сферах людської діяльності, питома вага Фортран в загальному обсязі програмного забезпечення знизилася. Однак при вирішенні великих обчислювальних задач перевага віддається сучасному Фортран.

Деякий спад інтересу до Фортран був викликаний великою затримкою в розробці Фортран 90. Однак в даний час після завершення розробки, вступу в дію сучасних міжнародних стандартів Фортран (Фортран 90 і Фортран 95) і реалізації їх практично для всіх обчислювальних систем, а також у зв'язку з

використанням високопродуктивних паралельних комп'ютерів, інтерес до Фортран знову зріс.

Пропонований огляд присвячений аналізу сучасного стану та перспективам розвитку мови Фортран.

Сучасні стандарти Фортран представляють собою сімейство стандартів, що складається з декількох частин. Чинний стандарт Фортран 95 складається з трьох частин. Перша частина - основна (базова) мова. Решта частини є додатковими. При цьому не потрібно, щоб компілятор, що відповідає стандарту, обов'язково реалізовував додаткові частини. Друга частина стандарту містить опис засобів для роботи з символьними рядками змінної довжини. Третя частина Фортран 95 визначає опис мови умовної компіляції.

У Фортран 90/95, поряд з фіксованим форматом вихідного тексту програми, дозволений вільний формат. Вільний формат допускає приміщення більше одного оператора в рядку, при цьому як роздільник крапку з комою. Ознака продовження оператора на рядок продовження - символ & - вказується в кінці того рядка, який треба продовжити. Коментарі записуються після символу знак оклику на початку рядка або в будь-якій позиції рядка після оператора. У вільному форматі прогалини є значущими.

Імена можуть містити до 31 символу; в іменах допускається символ підкреслення. Можна використовувати малі латинські букви, які всюди, крім символьних констант і символьних специфікацій формату, вважаються рівнозначними у відповідних прописних буквах.

У Фортран 90/95 властивості об'єктів можуть описуватися в відповідних операторах специфікації атрибутів (як і раніше), або за допомогою атрибутів в операторі оголошення типу. В одному операторі оголошення типу можна описати кілька атрибутів об'єктів, перерахованих в даному операторі після подвійної двокрапки; в цих операторах допускається також ініціалізація об'єктів.

Приклад:

```
REAL, DIMENSION (3), PARAMETER :: VECTOR = ( / 1., 2., 3. /)

REAL, SAVE :: XY (10, 10)
```

Для всіх вбудованих типів введені (необов'язкові) кошти параметризації, тобто кожен вбудований тип може мати кілька різновидів. Параметризація вбудованих типів забезпечує можливість завдання способу подання даних, що дозволяє компіляторам підтримувати короткі цілі, більше двох видів точності для дійсних і комплексних даних, а також багатобайтові символьні дані (для мов з великим набором символів, таких як китайський або японський) або використовувати додаткові набори символів для конкретних додатків.

Наведений нижче приклад ілюструє спосіб завдання точності і діапазону для речових даних.

Приклад:

```
! Два варіанти для завдання способу подання, що забезпечує

! точність не менше 10 знаків і діапазон десяткового порядку не менш

! ніж (-30, +30)
INTEGER, PARAMETER :: NUMS = SELECTED_REAL_KIND (10, 30)

REAL (KIND = NUMS) R1, R2          ! NUMS - параметр типа

REAL (SELECTED_REAL_KIND (10, 30)) :: R1, R2
```

Для наведеного прикладу значення змінних R1 і R2 на одних комп'ютерах можуть відображатися як звичайні дійсні числа, що займають одне машинне слово, на інших - як "довгі" (подвійної точності) речові числа. Однак при перенесенні даної програми на комп'ютер з іншою формою представлення даних не потрібно міняти ні опису змінних R1 і R2, ні інші оператори, пов'язані з цими змінними, так як в будь-який реалізації (якщо компілятор не видав відмови) повинні бути забезпечені відповідні точність і діапазон. Таким чином, зазначені кошти більше відповідають вимогам

мобільності програм, ніж використання непараметризованних традиційних для Фортран типів REAL і DOUBLE PRECISION.

Для мови Фортран традиційно був характерний принцип умовчання, тобто автоматичне приписування об'єктам типу, якщо тип об'єкта не вказано явно. У Фортран 90/95 введений оператор IMPLICIT NONE. При наявності такого оператора типи об'єктів в даній програмній одиниці повинні бути оголошені явно. Явне оголошення об'єктів забезпечує можливість контролю типів, що, очевидно, корисно для підвищення надійності створюваних програм. IMPLICIT NONE є більш гнучким інструментом, ніж просто обов'язковість описів, що має місце в інших мовах.

Введені засоби, що дозволяють визначити нові типи даних - похідні типи (структури даних), які представляють собою сукупність компонентів. Компоненти похідного типу можуть мати вбудований або раніше описаний похідний тип; компонентами можуть бути скаляри, масиви і покажчики. Компонент похідного типу може мати такий же тип як описуваний, якщо він є покажчиком (тобто має атрибут POINTER). Це корисно для створення пов'язаних списків.

Скалярний об'єкт похідного типу являє собою структуру. Допускаються масиви об'єктів похідного типу. Компоненти об'єкта похідного типу можуть використовуватися в звичайних виразах і операторах. Для об'єктів однакового похідного типу визначена операція присвоювання; крім того, такі об'єкти можуть використовуватися в операторах введення/виведення, в якості параметрів процедур і як результат функції.

Приклад:

```
! Описання типу PERSON

TYPE PERSON

INTEGER AGE

CHARACTER (LEN = 10) NAME

END TYPE PERSON
```

```
! Оголошення об'єктів описаного типу
TYPE (PERSON):: STUDENT, TEACHER, MEMBER (10)
! Операція над компонентами похідного типу
N = TEACHER%AGE - STUDENT%AGE
! Компонент елемента масиву похідного типу
MEMBER (1) % AGE = 50
! Присвоєння об'єкту похідного типу
STUDENT = (20, '     SMITH')
! Приклад пов'язаного списку
TYPE LINK
REAL VALUE
TYPE (LINK), POINTER :: PREVIOUS
TYPE (LINK), POINTER :: NEXT
END TYPE LINK
```

У мові є апарат, що дозволяє програмісту поширити для похідних типів вбудовані операції (перевантаження) або описати нові операції для даних вбудованих і похідних типів. Ці можливості дозволяють програмісту визначити абстрактні типи даних, що є одним з елементів об'єктно-орієнтованого програмування.

У мову введено великий набір засобів для роботи з масивами і секціями масивів як з цілими об'єктами. Істотним є той факт, що операції над масивами неявно специфікують паралелізм дій над компонентами масивів (масиву). Нові засоби дозволяють програмісту описати алгоритми обробки масивів в більш лаконічній і наочній формі (з елементами Непроцедурного) ніж при традиційному способі з використанням вкладених циклів і умовних

операторів. Крім того, ці кошти дозволяють компілятору згенерувати ефективний код з урахуванням особливостей апаратури, оскільки явно специфікують векторні операції.

Секція масиву забезпечує можливість адресуватися до частини масиву. Секція задає послідовність індексів і тим самим виділяє частину масиву, що складається з елементів з відповідними індексами. Секція масиву може складатися не тільки з розташованих підряд елементів вихідного масиву, але також і з незв'язаних областей цього масиву.

Семантика операцій і вбудованих функцій Фортран 77 розширена таким чином, що їх можна застосовувати не тільки до скаляр, а й до масивів; при цьому операції і стандартні функції виконуються поелементно над відповідними елементами масиву. Операнди, які беруть участь в одній операції, повинні бути узгоджені з конфігурацією (тобто повинні мати однакову розмірність і розмір по кожному виміру; скаляр вважається узгодженим з будь-яким масивом). Масив може бути також значенням виразів і значенням функцій (як вбудованих, так і обумовлених користувачем).

Аналогічним чином розширена семантика операторів присвоювання; результат операції такої, якщо б спочатку виконувалися всі операції в правій частині, а потім - присвоювання, яке може виконуватися в довільному порядку.

Приклад:

```
REAL X (10, 20), Y (40), Z (-9: 10), W (10)

! Присвоєння значень елементам 10-го рядка матриці X

X (10, :) = 2.8 * Y (2: 40: 2) + SQRT (Z)

! Зміна порядку елементів масиву на зворотний

W (1:10) = W (10: 1: -1)
```

Використовуючи оператор WHERE або конструкцію WHERE-END WHERE, можна виконати присвоєння значень тільки деяких елементів одного масиву відповідних елементів іншого масиву, тобто виконати присвоєння під керуванням логічної маски.

Приклад:

```
REAL X (10, 20), Z (-9: 10), V (20)

! Обчислення логарифма для позитивних елементів масиву V

WHERE (V> 0.0) Z = LOG (V)

! конструкція WHERE

WHERE (V / = 0)

  X (10, :) = 3.3 * Z / V

ELSEWHERE

  X (10, :) = 0.0

END WHERE
```

Одним з найбільш важливих нововведень (в Фортран 95) є оператор і конструкція FORALL. FORALL допускає можливість специфікувати більший клас секцій масивів, ніж в Фортран 90. Наприклад, діагональ масиву в Фортран 95 може бути представлена за допомогою FORALL.

Приклад:

```
FORALL (I = 1: N) A (I, I) = B (I)
```

FORALL функціонально схожий на цикл, але виконується інакше. Так, оператор

```
FORALL (I = 2: N) C (I, I) = C (I-1, I-1)
```

дає результат, відмінний від результату наведеної нижче послідовності операторів:

```
DO I = 2, N
        C(I, I) = C(I-1, I-1)
END DO
```

Це пояснюється тим, що ітерації DO-циклу виконуються послідовно так, що кожен наступний елемент діагоналі модифікується до його використання в наступній ітерації. На противагу цьому, в FORALL всі діагональні елементи вибираються і використовуються до збереження модифікованого значення в пам'яті. Зауважимо, що такий результат не залежить від того, чи підтримує компілятор паралельне виконання FORALL.

Є також конструкція FORALL - END FORALL. Тема конструкції може містити логічну маску. Конструкції можуть бути вкладеними і можуть містити конструкції WHERE.

У мові є великий набір стандартних функцій для роботи з масивами. Функції редукції (ALL, ANY, COUNT, MINVAL, MAXVAL, PRODUCT, SUM) виконують арифметичні і логічні операції над масивами. Є функції множення матриць, транспонування матриці, обчислення скалярного добутку векторів, функції, які виконують зрушення позицій елементів масиву, функції конструювання масиву з елементів інших масивів, різні довідкові функції і ін.

Приклад:

```
! Обчислення суми позитивних елементів масиву X

C = SUM (X, MASK = X> 0.0)
```

СПИСОК ЛІТЕРАТУРИ

1. Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. «Приемы объектно-ориентированного проектирования. Паттерны проектирования»
2. Литвиненко Н. А. - Технология программирования на C++
3. The Python Tutorial – Yeradis P. Barbosa Marrero
4. Федорова Н.Н., Вальгер С.А., Данилов М.Н., Захарова Ю.В.- «Основы работы в ANSYS»
5. Дьяконов В. П. MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012. – 768 с.: ил
6. Васильев А.Н. – «Maple 8. Самоучитель», 352 с.
7. О.В. Бартеньев – «Современный Фортран», 2000. – 450 .

Навчальне видання

Загорулько Андрій Васильович,
Кайота Дмитро Олегович

Програмування задач механіки

Конспект лекцій

для студентів спеціальності
131 «Прикладна механіка»
денної форми навчання

Відповідальний за випуск А. В. Загорулько
Редактор С. М. Симоненко
Комп’ютерне верстання С. О. Міщенко

Формат 60×84/8. Ум. друк. арк. 3,72. Обл.-вид. арк. 2,62.

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, м. Суми, 40007
Свідоцтво суб’єкта видавничої справи ДК No 3062 від 17.12.2007.