Деякі аспекти програмної реалізації системи для роботи з багатовимірними масивами даних

Іванова Г.А., Іванов В.О.
Україна, Київ, Національний технічний університет України "КПІ"

В статье рассмотрены некоторые аспекты программной реализации системы для работы с многомерными массивами данных. Системы подобного типа эффективно используются в учебном процессе высших учебных заведений технического профиля.

У сучасному суспільстві важко переоцінити роль комп'ютерних технологій в освіті. Зараз неможливо уявити вивчення таких фундаментальних дисциплін, як лінійна алгебра, аналітична геометрія, математичний аналіз, диференціальне числення без використання якісного програмного забезпечення. Для вивчення майже всіх технічних дисциплін необхідно приділяти значну увагу роботі з багатовимірними масивами даних, зокрема з матрицями. Звичайно, навчальні програми для технічних спеціальностей розробляються таким чином, щоб студент оволодів знаннями, вміннями та навичками роботи з матрицями ще на першому курсі. Але студентів в процесі навчання необхідно підготувати до вирішення реальних задач, в яких, як правило, використовуються багатовимірні масиви даних та матриці з великою кількістю елементів. Студентів необхідно навчити правильно обирати методи та створювати оптимальні алгоритми для вирішення задач. Але не менш важливим є навчити студентів користуватися програмним забезпеченням, яке допоможе виконувати рутинні операції з багатовимірними масивами даних та матрицями з великою кількістю елементів.

Сьогодні на ринку програмного забезпечення існує великий вибір відповідних програмних продуктів. Але слід зазначити, що метою створення таких програмних продуктів є використання їх при проведенні наукових досліджень. Як правило, подібні програмні продукти включають багато функцій, які мало придатні для навчальних цілей. Крім того, їх ціна є нерентабельною для бюджетів вищих навчальних закладів України.

Враховуючи фактори, зазначені вище, автори створили систему для виконання операцій з багатовимірними масивами даних та матрицями, більш ефективну і рентабельну для навчальних цілей. Особливістю цього програмного продукту є те, що вектор та матриця є базовими типами даних, що робить швидкість роботи програми значно вищою. Ще однією позитивною рисою є наявність можливості роботи з цією програмою в мережі Інтернет. У цьому випадку всі обчислення виконуються на сервері, не завантажуючи процесор персонального комп'ютера користувача.

Робота з програмою проходить в інтерактивному режимі, що забезпечує .оперативність зворотного зв'язку. Програма створена за допомогою технології ASP.NET на мові програмування C++.

Програма складається з шести модулів. В одному з цих модулів реалізовані різноманітні функції роботи з багатовимірними масивами даних, зокрема з матрицями. Кожен з інших п'яти модулів містить реалізацію відповідного блоку інтерпретатора.

Інтерпретатор складається із п'яти блоків, кожен з яких реалізований у окремому модулі.

1) Блок лексичного аналізу зчитує вхідну строку, введену користувачем, і перетворює її в потік токенів (або термінальних символів). Токеном може бути ключове слово, знак пунктуації або ідентифікатор (тобто будь-яка послідовність символів, яка є коректною для даної програми). Послідовність токенів називається нетермінальним символом. На етапі лексичного аналізу виконується видалення пропусків та коментарів і розпізнавання ідентифікаторів, ключових слів та знаків пунктуації, тобто токенів. Для того, щоб визначити, чи є послідовність символів токеном, необхідно продивитися вхідний потік на кілька символів за межами токена. Після того, як буде визначено, що дана послідовність символів є токеном, необхідно повернути прочитані символи, що не належать токену, у вхідний потік. Оскільки на переміщення символів може бути витрачено досить багато часу, для зменшення витрат часу на обробку вхідного потоку в програмі, що описується, використовується спеціальна буферезуюча технологія. В програмі використовується буфер, поділений на 2N-символьні половини. В кожну половину буфера зчитується N вхідних символів за допомогою однієї команди (при цьому не викликається команда читання для кожного вхідного символу окремо). Якщо залишилось зчитати з вхідного потоку менше ніж N символів, то в буфер вноситься спеціальний символ EOF, який вказує на кінець вхідного файлу. В процесі роботи підтримуються 2 вказівники. Між цими вказівниками міститься низка символів, що вже прочитані з вхідного потоку для визначення поточного токена. Спочатку ці вказівники відмічають перший символ низки символів, що формують наступний токен. Один вказівник переміщується вперед, скануючи вхідний потік, до тих пір, поки не буде визначено, що дана послідовність символів є токеном. Після обробки токена обидва вказівники вказують на символ, наступний після тих, що формують токен. Якщо перший вказівник перетне границю між половинами буфера, права половина буфера заповнюється N новими символами з вхідного потоку. Якщо цей вказівник виходить за межі правої границі буфера, то нові N символів зчитуються в ліву половину буфера, а вказівник переміщується на початок буфера.

2) Блок синтаксичного аналізу. Функція, в якій реалізовано синтаксичний аналізатор, отримує строку токенів від лексичного аналізатора і перевіряє цю строку на наявність синтаксичних помилок. Синтаксичний аналізатор також повідомляє про всі знайдені ним помилки, що дає змогу користувачу досить швидко і легко їх виправити. Таким чином, в цьому блоці перевіряються запити користувачів на наявність синтаксичних помилок. Результатом роботи синтаксичного аналізатора також є побудова дерева розбору. Вузлами дерева розбору є нетермінальні символи граматики мови, яку сприймає програма, що описується. Коренем цього дерева є стартовий символ граматики, а його листям – термінальні символи. При проходженні дерева розбору до кожного з його вузлів прикріплюються атрибути. Таким чином, кожен з вузлів дерева розбору містить таку інформацію:

- Термінальний чи нетермінальний символ, який відповідає даному вузлу.
- Інформацію про тип числа чи змінної, яка необхідна даному вузлу.
- Інформацію про значення числа чи змінної, яка необхідна даному вузлу.

3) Блок семантичного аналізу. Необхідно перевірити коректність команд, введених користувачем, не тільки з точки зору синтаксису, але й з точки зору семантики.

В цьому блоці здійснюється перевірка таких семантичних правил:

- Оператор не повинен застосовуватись до несумісного з ним операнду.
- Тип операнду повинен відповідати типу, що очікується в даному контексті.
- Індексування може бути застосовано лише до масиву.

Інформація про тип кожного з операндів виразу, введеного користувачем, зберігається в одному з атрибутів відповідного вузла дерева розбору цього виразу. Для реалізації роботи блока семантичного аналізу використовується система типів. Система типів являє собою набір правил для присвоєння типів виразам. У відповідності з цими правилами, використовуючи інформацію про типи операндів, інтерпретатор визначає типи підвиразів і тип виразу. При цьому виконується перевірка сумісності типів операндів. Таким чином, при перевірці типів виконується синтез типу кожного виразу з типів його підвиразів.

4) Блок генерації проміжного коду. Після проведення лексичного, синтаксичного та семантичного аналізу команди користувача необхідно перевести у проміжне представлення. Для цього використовується триадресний код. Триадресні інструкції являють собою абстрактний вигляд проміжного коду. В інтерпретаторі ці інструкції реалізовані, як записи з полями для операндів і оператора. В програмі, що описується, для представлення триадресних інструкцій використовуються трійки, тобто триадресні

інструкції, представленні записами з трьома полями: два поля для збереження аргументів і одне – для збереження операції.

5) Обчислювальний блок. В цьому блоці реалізовано функції, які забезпечують виконання команд користувача.

Розглянутий у роботі програмний продукт допоможе виконувати рутинні операції з багатовимірними масивами даних та матрицями з великою кількістю елементів, може бути використаний у наукових дослідженнях і у навчальному процесі з технічних дисциплін, що сприятиме підвищенню ефективності цього процесу.