Рекурсія у програмуванні

Самусенко Петро Федорович
кандидат фізико-математичних наук, доцент кафедри теоретичних основ інформатики
Оніщенко Сергій Миколайович
старший викладач кафедри інформаційних технологій і програмування
Національний педагогічний університет імені М.П. Драгоманова

Анотація. У роботі проаналізовано доцільність використання апарату рекурсивних функцій у програмуванні.

Ключові слова: алгоритм, рекурсія, ітерація.

Із середини минулого століття апарат рекурсивних функцій стає однією з фундаментальних основ теорії алгоритмів. Завдяки засобам підтримки механізму рекурсивного виклику підпрограм в мовах програмування високого рівня, а також створенню спеціальних мов програмування, в яких рекурсивні обчислення є вбудованим базисом опису обчислювального процесу (Рефал, Лісп), рекурсію стали активно використовувати не лише, як теоретичний апарат дослідження загальних властивостей алгоритмів, але й для практичного програмування. В математиці і програмуванні рекурсія – це метод визначення або опису функції за допомогою цієї ж функції. Рекурсія як парадигма програмування – це метод створення програм з використанням програмних конструкцій функцій, що викликають самі себе безпосередньо або опосередковано. Рекурсія природно зустрічається у всіх областях комп'ютерної математики. Описи ідентифікаторів і імен у мовах програмування, записаних нотацією Бекуса-Наура, містять явні рекурсивні визначення. На принципах рекурсії будуються формальні граматики, що описують мови програмування. Багатьом динамічним структурам даних (список, стек, черга, дерево) також можна дати просте рекурсивне визначення.

Початківці програмісти обережно ставляться до використання рекурсії, що пояснюється не очевидним механізмом рекурсивного виклику, можливістю виникнення нескінченого циклу або аварійного завершення програм як наслідок звернення до адреси пам'яті за допустимими межами. Разом з тим, слід зазначити про переваги рекурсивного підходу.

- Існує достатньо велика кількість задач, які мають рекурсивну природу, і ряд методів розробки алгоритмів, які природно породжують рекурсивні алгоритми. Метод динамічного програмування рекурсивний за своєю суттю. Метод декомпозиції передбачає поділ задачі на кілька аналогічних задач меншої розмірності, з наступним об'єднанням розв'язків, і породжує алгоритми з рекурсивною структурою.
- Ряд структур даних (список, дерево), багато об'єктів сучасних мов програмування рекурсивні за визначенням. Повторення цілого в його частині – основна властивість фрактальних об'єктів. Ієрархія класів в об'єктно-орієнтованому програмуванні, різні деревовидні регулярні структури даних мають просте рекурсивне визначення. Програми для роботи з таким структурами ефективні саме в рекурсивній реалізації.
- Рекурсія в програмуванні є аналогом методу математичної індукції в математиці і тому має як внутрішню витонченість і простоту, так і широку область застосування.
- Рекурсивно описані алгоритми мають лаконічні описи. При цьому суттєво зменшуються витрати часу для їх налаштування і модифікації, в порівнянні з ітераційними описами.
- Аргумент з теоретичним підтекстом – якщо апарат рекурсивних функцій є одним з базисів сучасної теорії алгоритмів, то чому самі алгоритми не мають бути рекурсивними?

Для об'єктивності слід навести думки програмістів-професіоналів про рекурсивну ефективність програмних реалізацій.

- Рекурсивно реалізовані алгоритми програють ітераційним за часовою ефективністю. Але при цьому рекурсивні алгоритми, як правило, мають кращі асимптотичні оцінки.

- Ємнісна ефективність рекурсивних реалізацій менша за ємнісну ефективність ітераційних. Так, рекурсивно реалізований алгоритм потребує для використання більше пам'яті в області програмного стеку, але ці витрати визначаються глибиною рекурсії. Для більшості рекурсивних алгоритмів і практично значущих розмірностей задач ця глибина, при сучасному обсязі оперативної пам'яті комп'ютерів, не суттєво впливає на загальний потрібний обсяг пам'яті.

Згідно з теоретичними основами теорії будь-який алгоритм, який можна реалізувати у вигляді машини Тюрінга, може бути описаний за допомогою апарату рекурсивних функцій. На практиці це означає, що рекурсія і ітерація як способи розробки алгоритмів і програм - еквівалентні. Використовуючи теорію ресурсної ефективності, залежно від задач і діапазону довжин входу можна визначити, який підхід - ітераційний або рекурсивний є більш ефективним.

Рекурсія – це природний метод реалізації алгоритмів розв'язування багатьох математичних задач. У математичних методах аналізу рекурсивних алгоритмів, розроблених в теорії різницевих рівнянь, суттєво використовуються асимптотичні оцінки рекурсивно визначених функцій. Застосування таких оцінок дозволяє зробити висновок про часову ефективність рекурсивних алгоритмів.

Оскільки рекурсивний алгоритм визначає послідовність звернень до одного й того ж фрагменту програмного коду, потрібно забезпечити збереження структур даних, які будуть задіяні алгоритмом після повернення з рекурсивного виклику. Основних способів опису структур даних є три. У випадку використання глобальних структур даних, як аргумент функції може бути використано посилання на потрібну структуру. При використанні локальних структур даних в тілі рекурсивної функції, при її виклику створення копій структур забезпечується засобами мови програмування і компілятора. Такий спосіб призводить до значних витрат пам'яті програмного стеку. Третій підхід до створення потрібної структури для програмної реалізації рекурсивного алгоритму полягає в тому, що користувач засобами мови програмування звертається до операційної системи для виділення потрібного в даний момент ресурсу пам'яті. Цей підхід потребує від користувача обережного маніпулювання з динамічною пам'яттю, своєчасного її звільнення, організації коректного передавання аргументів при рекурсивних викликах і коректного повернення отриманих результатів після закінчення роботи функції.

Список використаних джерел

1. Андерсон Дж. Дискретная математика и комбинаторика: Пер. с англ. – М.: Изд. дом «Вильямс». 2004.
2. Баррон Д. Рекурсивные методы в программировании. – М.: Мир, 1974.
3. Беллман Р., Дрейфус Р. Прикладные задачи динамического программирования: Пер. с анг. – М.: Наука, 1965.
4. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. – 2-е изд., испр. – СПб.: Невский диалект, 2001.
5. Головешкин В.А., Ульянов М.В. Теория рекурсии для программистов. – М.: ФИЗМАТЛИТ, 2006.
6. Грин Д., Кнут Д. Математические методы анализа алгоритмов. – М.: Мир, 1987.
7. Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основание информатики: Пер. с англ. – М.: Мир, 1998.
8. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов. – М.: Мир, 1981.
9. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-е изд. – М.: Изд. дом "Вильямс", 2005.
10. Хаггарти Р. Дискретная математика для программистов. – М.: Техносфера, 2005.