УДК 004:005:51

ОСОБЛИВОСТІ ВИКОРИСТАННЯ СТРУКТУР ДАНИХ У ВИГЛЯДІ ЗВ'ЯЗНИХ СПИСКІВ

О. С. Бурківський, Н. А. Потапова

Анотація. У статті розкривається тема використання однозв'язних і двозв'язних списків в алгоритмізації та обробці даних. Висвітлюються основні принципи побудови та функціонування зв'язних списків, їх переваги та недоліки, порівняно з іншими структурами даних. Досліджується практичне застосування в імплементації алгоритмів сортування, управління пам'яттю, обробки даних у базах даних, графічних інтерфейсів користувача тощо. Досліджено сучасні тенденції в розвитку цих структур даних, зокрема оптимізацію під конкретні завдання, використання у паралельному та розподіленому програмуванні, а також вплив технологій штучного інтелекту та машинного навчання на їх використання.

Ключові слова: алгоритми, структури даних, однозв'язні списки, двозв'язні списки, програмування.

Вступ. Стрімке зростання обсягів обумовлюють важливість їх ефективної організації та критичність обробки інформації загалом. Одним із ключових аспектів функціонування цих процесів є вибір оптимальних структур даних, які дають змогу ефективно зберігати, організовувати доступ та маніпулювати даними.

У цьому контексті однозв'язні та двозв'язні списки займають важливе місце. Структури даних є основою для розробки різноманітних програмних застосунків у сфері інформаційних технологій. Використання однозв'язних та двозв'язних списків є однією з ключових стратегій у цьому процесі. Вони дають змогу ефективного управління даними в пам'яті комп'ютера, що є критичним для реалізації багатьох алгоритмів та програм. Актуальною є організація даних, що становлять множини та перерахування конкретного інформаційного наповнення. До таких структур належить базовий тип – зв'язані списки даних.

Однозв'язні та двозв'язні списки – це базові структури даних, які дають змогу зберігати та організовувати дані у вигляді послідовностей, забезпечуючи ефективний доступ, вставку та видалення елементів [1]. Основні переваги однозв'язних та двозв'язних списків використовують під час формування доступу до динамічної пам'яті, а втрата гнучкості залежить від відслідковування категорії зв'язаності елементів.

Метою статті є висвітлення основних питань організації даних за структурою однозв'язних і двозв'язних списків та можливостей їх реалізації на мові C#.

Основна частина. Суть організації списків полягає у формуванні механізму звернення до множини елементів за визначеними ключами та вказівниками. Вказівники слугують для встановлення зв'язності між елементами списку. Окремий елемент структури визначений як вузол списку. Отже, загалом тип списку може визначатись за кількістю вказівників на зв'язність структури. Однозв'язний список складається з вузлів, кожен із яких містить дані та посилання на наступний вузол у списку. Останній вузол має посилання, що вказує на пустий вузол (NULL) або на кінець списку. Тобто формування зв'язку відбувається за посиланням на наступний структурний елемент. Така структура дає змогу просто переглядати елементи списку, але доступ до будь-якого елементу потребує проходження через весь список від початку [2].

Більш швидкий доступ до елементів забезпечує двозв'язний список. У цьому випадку розширення функціональності, порівняно з однозв'язним списком, виконується завдяки включенню додаткового посилання на попередній вузол. Це дає змогу здійснювати обхід списку у будь-якому напрямку, а також більш ефективно виконувати операції вставки та видалення елементів, оскільки для цього не потрібно пройти весь список від початку [3].

Обидва типи списків мають свої переваги та недоліки, і вибір між ними залежить від конкретних потреб програми та характеристик задачі. Так, однозв'язні списки зазвичай використовуються там, де потрібен простий список без необхідності звертатися до попередніх елементів, тоді як двозв'язні списки корисні у випадках, коли потрібно виконувати часті операції вставки та видалення елементів у середині списку.

Організація списків розглядається на побудованому алгоритмі пошуку відрізка ізоляції деякої нелінійної диференційованої функції. Алгоритм обчислення значень функції та її похід-

ної на вказаному відрізку використовується для аналізу поведінки функції у заданому діапазоні значень. Використання алгоритму обумовлено класичною задачею пошуку відрізку ізоляції під час аналізу поведінки функцій у межах побудованих математичних моделей, розв'язання рівнянь та оптимізації функцій методами обчислень, а також у різних областях науки та інженерії, де важливо аналізувати та робити висновки про функціональні залежності.

Представлений код на мові програмування C# є прикладом використання програмування для аналізу функцій та їх похідних у визначеному діапазоні значень аргументу. За допомогою цього коду ми можемо обчислити значення функції та її похідної на заданому відрізку, а також вивести результати у зручному для аналізу форматі.

У наведеному коді спочатку визначаються параметри для обчислення функції та її похідної: крок зміни аргументу, мінімальне та максимальне значення аргументу. Потім створюються три списки, у які будуть зберігатися значення аргументу, значення функції та її похідної. Далі проводиться заповнення цих списків значеннями функції та похідної на кожному кроці аргументу. На наступному кроці виконується вивід таблиці результатів обчислень на екран, де кожен рядок таблиці містить номер обчислення, значення аргументу, значення функції та її похідної. Після цього виводиться інформація про відрізки, на яких функція може мати розв'язок y(x) = 0. Для цього перевіряються знаки значень функції на сусідніх точках. Якщо значення функції мають протилежні знаки, то це означає зміну знака функції між цими точками, що може вказувати на наявність кореня рівняння.

Вигляд розробленого коду на мові програмування C#, що демонструє використання зазначених алгоритмів для обчислення значень функції та її похідної на заданому відрізку, такий:

```
namespace Приклад_однозвязного_списку
{
   class Program
   {
      static void Main(string[] args)
      {
         double step = 1;
         double minX = -10;
         double maxX = 10;
         List<double> xValues = new List<double>();
         List<double> yValues = new List<double>();
         List<double> derivativeValues = new List<double>();

         // Заповнення списків значеннями функції та похідної
         for (double x = minX; x <= maxX; x += step)
         {
            double y = FunctionY(x);
            double derivative = DerivativeFunctionY(x);
            xValues.Add(x);
            yValues.Add(y);
            derivativeValues.Add(derivative);
         }
         Console.OutputEncoding = System.Text.Encoding.UTF8;
         Console.WriteLine("Таблиця результатів обчислень");
         Console.WriteLine("№\tx(i)\ty(i)\ty'(x)");
         for (int i = 0; i < xValues.Count; i++)
         {
            Console.WriteLine($"{i}\t{xValues[i]}\t{yValues[i]}\t{derivativeValues[i]}");
         }
}
  }
}
```

Основні характеристики однозв’язаного списку можна визначити так [4, 5]:

1. Кожен елемент списку містить інформаційну та посилальну частини, тобто дані та вказівник на наступний елемент.

2. Під час опису елементу порядок розташування інформаційних та полів вказівника є довільним.

3. Інформаційна частина може містити поля з інформацією різних типів.

4. Вказівники є однотипними, але їх кількість може бути різною залежно від типу списку. Тобто елемент повинен містити щонайменше два поля: одне поле типу «покажчик», інше – для зберігання даних користувача.

Реалізація двозв’язного списку для задачі пошуку відрізка ізоляції матиме вигляд:

```
namespace Приклад_однозвязного_списку
{
   class Program
   {
      static void Main(string[] args)
      {
         double step = 1;
         double minX = -10;
         double maxX = 10;
         List<double> xValues = new List<double>();
         List<double> yValues = new List<double>();
         List<double> derivativeValues = new List<double>();

         // Заповнення списків значеннями функції та похідної
         for (double x = minX; x <= maxX; x += step)
         {
            double y = FunctionY(x);
            double derivative = DerivativeFunctionY(x);
            xValues.Add(x);
            yValues.Add(y);
            derivativeValues.Add(derivative);
         }
         Console.OutputEncoding = System.Text.Encoding.UTF8;
         Console.WriteLine("Таблиця результатів обчислень");
         Console.WriteLine("№\tx(i)\ty(i)\ty'(x)");
         for (int i = 0; i < xValues.Count; i++)
         {
            Console.WriteLine($"{i}\t{xValues[i]}\t{yValues[i]}\t{derivativeValues[i]}");
         }
}
  }
}
```

Ознаки двозв’язного списку можна визначити так: зв’язок елементів у визначеному порядку; ідентифікація першого елементу; ідентифікація останнього елементу; наявність двох вказівників (один вказує на наступний елемент, а інший на попередній); перший елемент має попереднім вказівником на невизначений елемент None; None буде наступним елементом для останнього елементу списку.

Для обох видів списків можна виділити такі основні операції з даними:

1. Додавання елементу. Порядок реалізації цієї операції включає дії:

– створення нового вузла;

– встановлення посилань нового вузла на попередній та наступний вузли відповідно до позиції, на яку потрібно вставити елемент;

– оновлення посилань попереднього та наступного вузлів на новий вузол.

2. Видалення елементу. Порядок реалізації цієї операції включає дії:

– пошук вузла, який потрібно видалити;

– оновлення вказівників наступного та попереднього вузлів, щоб вони вказували один на одного, обминаючи вузол, який видаляють;

– видалення посилань вузла на інші вузли (ті, що залишаються), щоб уникнути витоку пам'яті.

3. Пошук елементу у двозв'язному списку виконується шляхом ітераційного порівняння значень елементів з шуканим значенням просуваючись через усі вузли.

Зазначимо, що використання двозв'язних списків може проводитись за такими напрямами:

1. Побудова ітераторів. Двозв'язний список у багатьох мовах програмування є основною для ітераторів, що дає змогу здійснювати швидкий перебір елементів колекцій даних.

2. Засіб програмної реалізації структур даних. Зокрема, реалізації черги, стеку та складних списків, орієнтованих на двонаправлене відслідковування позицій елементів.

3. Хешування даних. Двозв'язні списки можуть використовуватися для реалізації хешів для швидкого доступу до останніх даних, а також їх видалення за необхідністю.

4. Обробка текстових даних. За потреби зберігання та обробки текстів у вигляді списків рядків двозв'язні списки можуть бути корисними для вставки та видалення елементів всередині тексту.

Висновки. Використання однозв'язних і двозв'язних списків є ключовим аспектом алгоритмізації, програмування та управління даними. Ці структури даних здатні забезпечувати швидкий доступ до елементів, можливість їх вставки та видалення, а також зручного обходу. Однозв'язні списки треба застосовувати під час організації структур простих списків без необхідності звертання до попередніх елементів, тоді як двозв'язні списки надають більше можливостей у випадках проведення багаторазових операцій вставок та видалень із середини списку. Використання цих структур даних є засобом для підвищення ефективності алгоритмів та оптимізації ресурсів використання пам'яті завдяки організації динамічного звернення.

Abstract. This article discusses the topic of using singly and doubly linked lists in algorithmization and data processing. The main principles of construction and operation of linked lists, their advantages and disadvantages compared to other data structures are highlighted. Practical applications in such areas as implementation of sorting algorithms, memory management, data processing in databases, graphical user interfaces, etc. are explored. Modern trends in the development of these data structures, zokerms, such as optimization for specific tasks, use in parallel and distributed programming, as well as the impact of artificial intelligence and machine learning technologies on their use, are studied.

Keywords: algorithms, data structures, singly linked lists, doubly linked lists, programming.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Креневич А. П. Алгоритми і структури даних: підручник. Київ: ВПЦ «Київський Університет», 2021. 200 с.
2. Лінійний однозв'язний список. URL: https://erudyt.net/navchalni-predmety/informatika/prohramuvannya/linijnyj-odnozvyaznyj-spysok.html (дата звернення 03.02.2024).
3. Двозв'язкові списки. URL: https://krypton.com.ua/rozdil-2-struktury-danyh/dvozvyazkovi-spysky/ (дата звернення 03.02.2024).
4. Korman T. H. Algorithms. Unlocked. The MIT Press, Massachusetts Institute of Technology, 2013. 207 p.
5. Stephens R. Essential Algorithms: A Practical Approach to Computer Algorithms Using. John Wiley & Sons, 2013. 624 p.
6. Sedgewick R., Wayne K. Algorithms. Addison-Wesley Professional, 2020. 956 p.
7. Bhargava A. Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People. Manning Publications, 2016. 256 p.
8. Schildt H. C#: The Complete Reference. McGraw-Hill Osborne Media, 2002. 933 p.
9. Goodrich M. T., Tamassia R., Goldwasser M. H. Data Structures and Algorithms in Python. WILEY, 2013. 768 p.
10. Aho A. V., Ullman J. D., Hopcroft J. E. Data Structures and Algorithms. Addison Wesley, 1982. 448 p.