дає змогу оцінювати ефективність роботи дамб, шлюзів, мостів та інших об'єктів, аналізувати наслідки ерозії, акумуляції осадів і змін якості води. Застосування HEC-RAS дає змогу створювати детальні математичні моделі для відображення потоків води, визначати оптимальні стратегії управління водними системами, планувати заходи зі зменшення затоплення та прогнозувати вплив інженерних рішень на навколишнє середовище.

Програма активно використовується у сфері муніципального управління для аналізу інфраструктури, запобігання затопленням і планування водопостачання, а також в екологічних дослідженнях, які стосуються оцінки впливу змін у водних системах на екосистеми. Завдяки розвинутому інтерфейсу HEC-RAS допомагає візуалізувати результати моделювання, що є важливим для прийняття ефективних рішень на основі отриманих даних.

Отже, комп'ютерне моделювання виступає незамінним інструментом для аналізу та управління наслідками руйнування екологічних систем, спричинених воєнними діями. Воно дає змогу оцінювати масштаби катастроф, прогнозувати ризики затоплення та забруднення, а також розробляти ефективні стратегії реагування. Завдяки детальним симуляціям можна точно визначати зони ризику, оцінювати вплив інженерних рішень і заздалегідь планувати вчасні заходи для мінімізації шкоди екосистемам. Це сприяє більш ефективному управлінню водними ресурсами та відновленню природного балансу в умовах екологічних криз.

Список використаних джерел

1. Системний підхід до оцінки ризиків надзвичайних ситуацій в Україні / В. Д. Калугін, В. В. Тютюник, Л. Ф. Чорногор, Р. І. Шевченко. *Східноєвропейський журнал передових технологій*. 2012. № 1/6(55). Харків. С. 59–70.

2. Черенкевич О. С. Статистичне моделювання екологічних ризиків як фактора екологічної безпеки. *Статистика України.* 2020. № 2–3. С. 59–67. DOI: 10.31767/ su.2-3(89-90) 2020.02-03.07.

3. Яке призначення HEC-RAS? URL: https://bioglobin.com.ua/ukraincyam/yake-priznachennya-hec-ras.html (дата звернення: 04.12.2024).

УДК 004.422.5.

Синіцький В. Ю., здобувач вищої освіти,
Поремський Ю. В., канд. техн. наук, старший викладач кафедри інформаційних технологій

РЕКУРСИВНІ АЛГОРИТМИ В СУЧАСНОМУ ПРОГРАМУВАННІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Рекурсія – це метод розв'язання задачі шляхом розбиття її на підзадачі, що є спрощеними варіантами тієї ж задачі. Ключовою ідеєю цього методу є поетапне зведення задачі до її найпростішої версії, що має кінцеве рішення. Найпростішу версію називають базовим випадком. Базовий випадок є обов'язковим елементом у рекурсивних алгоритмах. Як приклад рекурсії можна навести обчислення чисел Фібоначчі, де кожне наступне число є сумою двох попер-

ніх. Такий алгоритм обчислення яскраво відображає те, як рекурсія допомагає будувати рішення подібних задач [1].

Існує декілька форм рекурсії, розглянемо деякі з них. Хвостова рекурсія є формою рекурсивного алгоритму, в якому остання дія, виконувана у функції перед поверненням значення, – її власний рекурсивний виклик. Такий підхід використовується для ефективного використання пам'яті, оскільки компілятор може оптимізувати стек викликів. Інша форма рекурсії – взаємна (обопільна) рекурсія. Вона виникає тоді, коли дві або більше функцій викликають одна одну по черзі. Такий тип рекурсії може бути корисним у задачах, пов'язаних із пошуком шляхів, симуляцією складних систем.

Рекурсія є інструментом, який має як переваги, так і недоліки. Серед основних переваг можна виділити стислість і зрозумілість, що сприяє написанню компактного коду. До того ж рекурсія широко застосовується у функціональному програмуванні. Проте її використання має і свої недоліки. Для задач із занадто великою кількістю рівнів рекурсії може значно знижуватись продуктивність, порівняно з циклічними алгоритмами. Рекурсія в програмуванні не універсальна, оскільки певні мови програмування або середовища мають обмеження на її використання через проблеми з пам'яттю [2].

Метод рекурсії має у собі кілька небезпек. Найбільш очевидна небезпека – нескінченна рекурсія. У разі неправильної побудови алгоритму функція може пропускати або не досягати умови зупинки алгоритму і виконуватися нескінченно. Проблема полягає в тому, що нескінченність насправді означає роботу функції доти, поки не буде вичерпаний стековий простір. Такий алгоритм буде приводити до переповнення стека й аварійного завершення роботи програми [3].

Розглянемо реалізацію рекурсії у програмуванні. Більшість сучасних мов програмування підтримують рекурсію: Python, C#, JavaScript тощо. Але її ефективність буде залежати від правильного проєктування алгоритму. Рекурсивні алгоритми часто бувають доволі компактними, хоча можуть вимагати додаткових ресурсів пам'яті. Код на C#, що представлено нижче, демонструє рекурсивний алгоритм, який виконує обчислення факторіалу числа.

```
using System;
using System.Text;

0 references
class Recursion
{
    //Рекурсивна функція для обчислення факторіалу
    2 references
    static int Factorial(int n)
    {
        //Базовий випадок
        if (n == 0)
            return 1;

        //Рекурсивний випадок
        return n * Factorial(n - 1);
    }

    0 references
    static void Main(string[] args)
    {
        Console.OutputEncoding = Encoding.UTF8;

        Console.Write("Введіть число для обчислення факторіалу: ");
        int number = int.Parse(Console.ReadLine());

        if (number < 0)
        {
            Console.WriteLine("Факторіал визначається тільки для додатніх чисел!");
        }
        else
        {
            int result = Factorial(number);
            Console.WriteLine($"Факторіал числа {number} дорівнює {result}.");
        }
    }
}
```

Рисунок 1 – Код роботи рекурсивного алгоритму

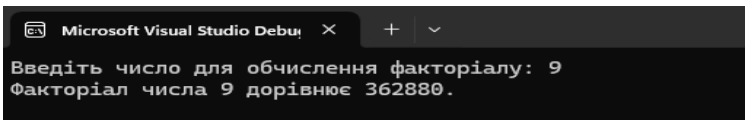

Рисунок 2 – Результат роботи коду

Список використаних джерел

1. Федорін І. В. Проектування та аналіз обчислювальних алгоритмів: Вступ до алгоритмів: навч. посіб. КПІ ім. Ігоря Сікорського, 2022. С. 19–24.

2. Foxminded. Рекурсія в програмуванні. URL: https://foxminded.ua/rekursiia-v-prohramuvanni/ (дата звернення 20.11.2024).

3. Основи інформатики та технологій програмування: навч. посіб. / М. Є. Рогоза, С. К. Рамазанов, А. В. Велігура, С. М. Танченко. Луганськ: Вид-во СНУ ім. В. Даля, 2012. С. 164–168.

УДК 004.9

Тищенко О. М., здобувач вищої освіти,
Поремський Ю. В., канд. техн. наук, старший викладач кафедри інформаційних технологій

АЛГОРИТМІЗАЦІЯ В ПОСТАНОВЦІ І РОЗВ'ЯЗАННІ ВІЙСЬКОВИХ ЗАВДАНЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Алгоритмізація є важливим аспектом сучасних військових технологій, які дають змогу розробляти ефективні стратегії та вирішувати складні проблеми, що виникають під час планування та виконання воєнних операцій. Він включає використання математичних моделей, спеціалізованих алгоритмів і комп'ютерних систем для оптимізації процесів управління, оцінки ризиків, ресурсного забезпечення та прийняття рішень [1].

В армії завдання варіюються від стратегії на полі бою до логістики постачання та обслуговування обладнання. Алгоритмізація дає змогу формулювати ці завдання в математичній або логічній формі, допомагаючи використовувати комп'ютерні програми для пошуку оптимальних рішень [2].

До основних категорій військових завдань, які потребують алгоритмізації, належать:

- Оптимальний розподіл ресурсів: наприклад, визначення оптимального розподілу боєприпасів, обладнання та персоналу.
- Планування операції: розробка ефективного плану нападу або захисту з урахуванням бойової ситуації, ресурсів і часу.
- Симуляція бою: розрахування результатів бою за допомогою математичних моделей, включно з мобілізацією військ, втратами, часом реакції та іншими факторами.

Використовуваний алгоритм може відрізнятися залежно від типу завдання. Ось кілька прикладів алгоритмів, які використовуються для вирішення військових завдань: