МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ  
«ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ЛУЦЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ»

Випускова циклова комісія Компʼютерних систем та інформаційних технологій

**Звіт**

**з практичної роботи №2**

з дисципліни “Алгоритми та структури даних”

Виконав: студент **2** курсу, групи **22-ICT**

Синиця Назар

Викладач: Володимир Остапюк

Здано:

Луцьк – 2024

**Зміст**

1. **Практичні завдання**
   1. Напишіть функцію на Python, яка обчислює n-не число Фібоначчі за допомогою ітеративного підходу (цикл).
   2. Напишіть функцію на Python, яка обчислює n-не число Фібоначчі за допомогою рекурсії.
   3. Напишіть функцію на Python, яка перевіряє, чи є задане число членом послідовності Фібоначчі.
   4. Напишіть функцію на Python, яка генерує перші n чисел послідовності Фібоначчі.
2. **Дайте відповіді на запитання**
   1. Який з реалізованих вами алгоритмів обчислення чисел Фібоначчі є найбільш ефективним з точки зору часової складності? Чому?
   2. У чому полягають переваги та недоліки рекурсивного підходу в порівнянні з ітеративним?
   3. Які проблеми можуть виникнути при обчисленні великих чисел Фібоначчі за допомогою рекурсивного алгоритму? Як їх можна уникнути?
   4. Як можна оптимізувати обчислення чисел Фібоначчі за допомогою динамічного програмування?
   5. Які інші послідовності, подібні до послідовності Фібоначчі, ви знаєте? Наведіть приклади та опишіть їх властивості.
   6. Як послідовність Фібоначчі пов'язана з поняттям рекурсії в програмуванні?
   7. Чи можете ви навести приклади реальних задач, де знання про послідовність Фібоначчі може бути корисним?

**Висновок**

**1. Практичні завдання**

* 1. Напишіть функцію на Python, яка обчислює n-не число Фібоначчі за допомогою ітеративного підходу (цикл).(Рис. 1.1.)

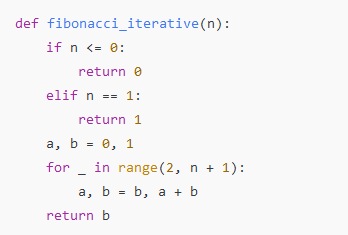
****

Рисунок - 1.1.

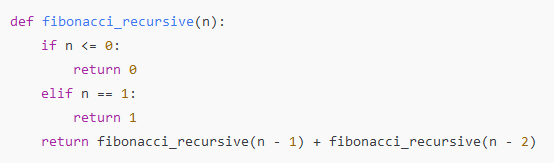
* 1. Напишіть функцію на Python, яка обчислює n-не число Фібоначчі за допомогою рекурсії.(Рис. 1.2.)

Рисунок - 1.2.

* 1. Напишіть функцію на Python, яка перевіряє, чи є задане число членом послідовності Фібоначчі. (Рис. 1.3.)

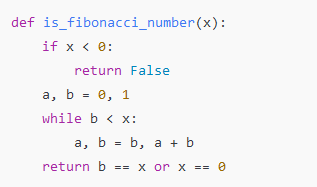


Рисунок - 1.3.

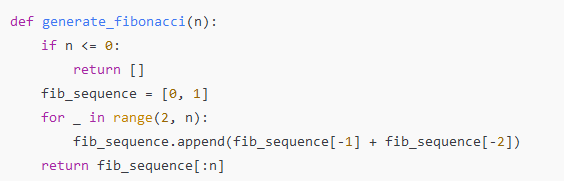
* 1. ****Напишіть функцію на Python, яка генерує перші n чисел послідовності Фібоначчі. (Рис. 1.4.)

Рисунок - 1.4.

**2. Дайте відповіді на запитання**

* 1. Який з реалізованих вами алгоритмів обчислення чисел Фібоначчі є найбільш ефективним з точки зору часової складності? Чому?

**Ітеративний** підхід найефективніший, тому що його швидкість зростає пропорційно до числа nn (часова складність (O(n)), і він використовує дуже мало пам’яті (постійний обсяг (O(1)).

**Рекурсивний** підхід без покращень працює набагато повільніше, оскільки повторно рахує ті самі значення багато разів. Час виконання росте дуже швидко, приблизно вдвічі з кожним збільшенням n (експоненційна складність (O(2n)).

* 1. У чому полягають переваги та недоліки рекурсивного підходу в порівнянні з ітеративним?

**Переваги:** Простота реалізації, особливо для задач, які природно описуються рекурсією.

**Недоліки:** Висока витрата пам’яті (стек викликів) і повторні обчислення, що знижує ефективність.

* 1. Які проблеми можуть виникнути при обчисленні великих чисел Фібоначчі за допомогою рекурсивного алгоритму? Як їх можна уникнути?

Якщо число дуже велике, функція викликається так багато разів, що програма «ламається» через переповнення пам’яті.

Цього можна уникнути, якщо зберігати вже знайдені значення (мемоізація) або використовувати ітеративний підхід.

* 1. Як можна оптимізувати обчислення чисел Фібоначчі за допомогою динамічного програмування?

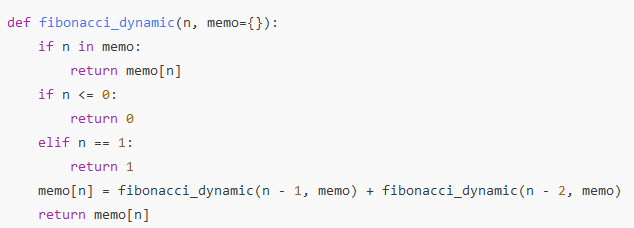
Щоб не рахувати те саме число багато разів, ми зберігаємо вже знайдені результати в пам’яті. Це значно прискорює роботу.(Рис. 2.4)

Рисунок - 2.4

* 1. Які інші послідовності, подібні до послідовності Фібоначчі, ви знаєте? Наведіть приклади та опишіть їх властивості.
* **Послідовність Люка:** Починається з 2 та 1; кожен наступний член є сумою двох попередніх.
* **Трибоначчі:** Кожен член є сумою трьох попередніх.
* **Падован:** Задає кількість способів розбиття довжини на сегменти 2 та 3.
* **Перрон:** Використовується в математичній біології.
  1. Як послідовність Фібоначчі пов'язана з поняттям рекурсії в програмуванні?

**Послідовність Фібоначчі** ідеально пояснює, що таке рекурсія. Ми використовуємо правило: щоб знайти число, потрібно скласти два попередні. Це виглядає як виклик функції, яка викликає сама себе.

* 1. Чи можете ви навести приклади реальних задач, де знання про послідовність Фібоначчі може бути корисним?
* **Комп’ютерна графіка:** Використовується для створення красивих форм, побудованих за «золотим перетином».
* **Криптографія:** Допомагає створювати захищені коди та ключі.
* **Біологія:** Наприклад, для моделювання росту популяцій тварин.
* **Економіка:** Для аналізу руху цін на ринку та прогнозування трендів (рівні Фібоначчі).

**Висновок**

У цій роботі Я розглянув різні способи обчислення чисел Фібоначчі: ітеративний, рекурсивний та оптимізований за допомогою динамічного програмування. Ітеративний підхід виявився найефективнішим завдяки простоті та низьким вимогам до пам’яті. Рекурсивний метод наочно демонструє принцип рекурсії, але є менш ефективним через повторні обчислення. Використання мемоізації значно покращує його продуктивність.

Я також навчився генерувати послідовність і перевіряти, чи належить число до неї. Послідовність Фібоначчі знаходить застосування у багатьох сферах: від комп’ютерної графіки до біології та фінансів. Робота з цією послідовністю є важливим прикладом застосування алгоритмічного мислення та ефективного програмування.