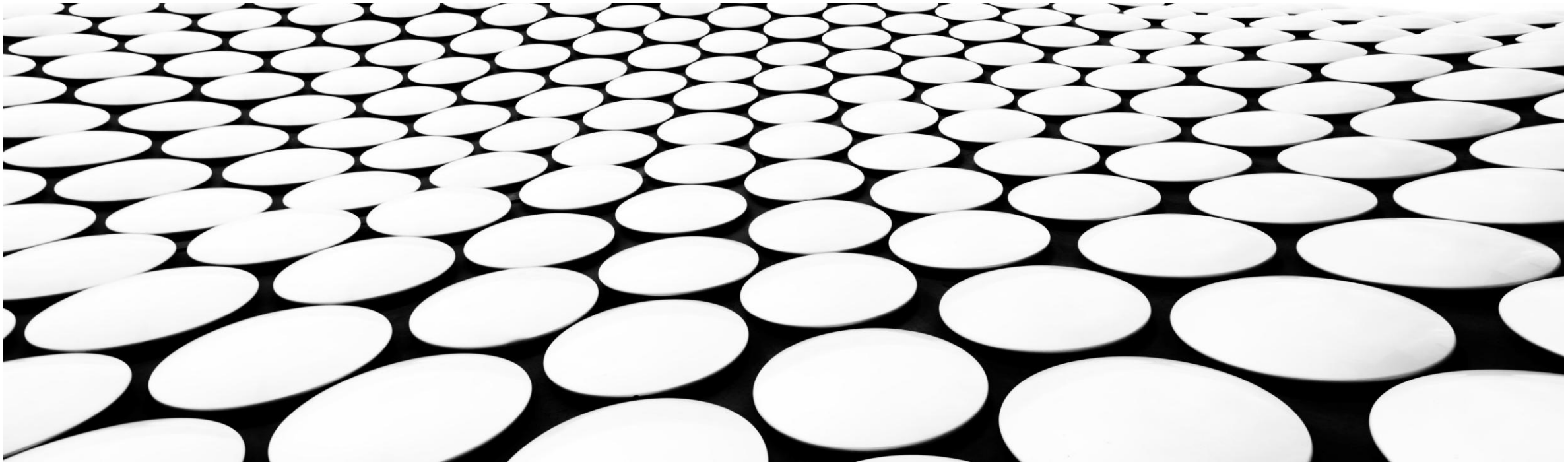
AI生成的方法與應用

講者: 柯宇謙



主要議題(I)

陌生

積極面對

保留

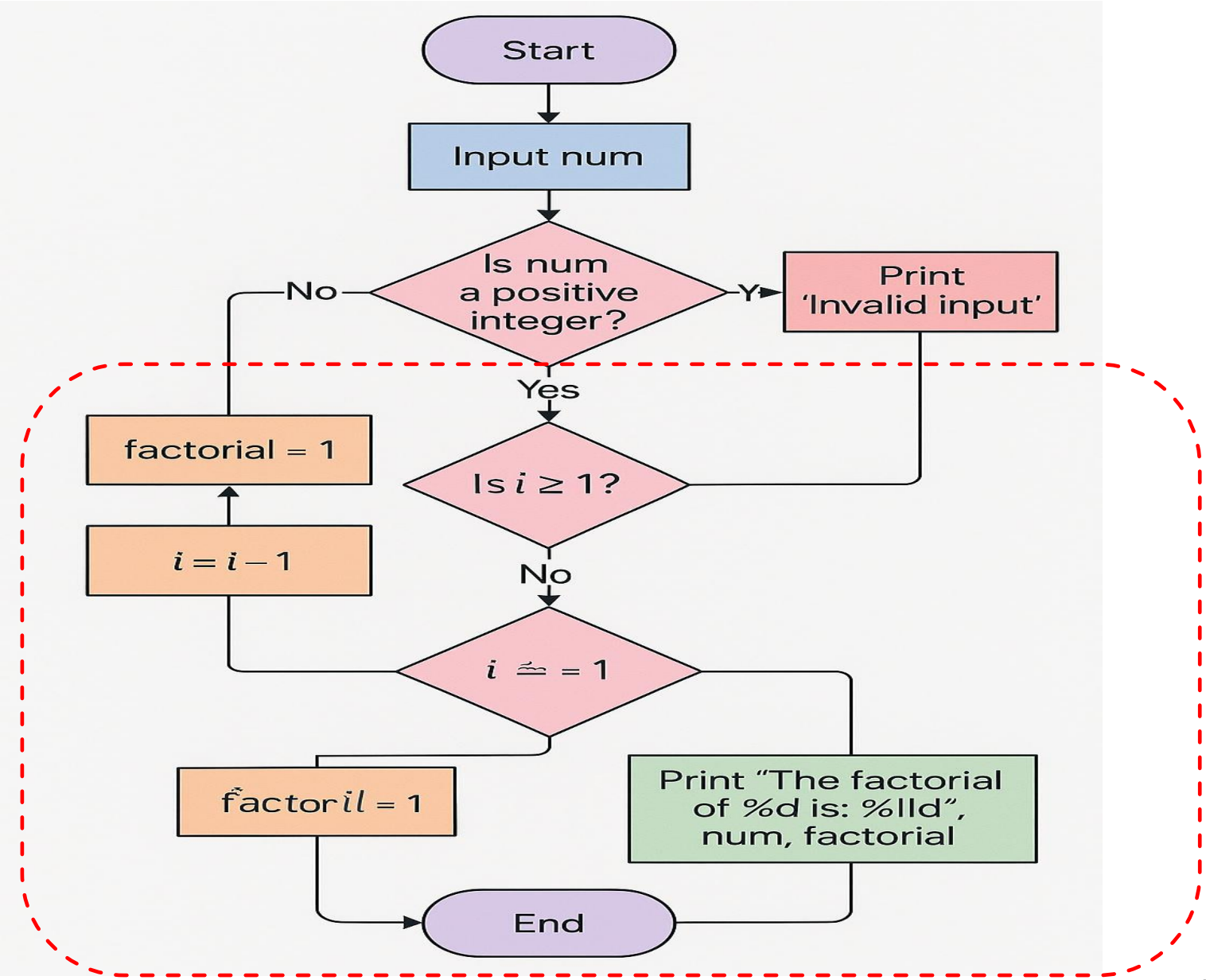
已知正確的程式

```c
#include <stdio.h>
int main() {
    int num;        // The number to calculate the factorial
    long long factorial = 1;
    int i;              // Loop counter
    printf("Enter a positive integer (max 12 for standard int size): ");
    if (scanf("%d", &num) != 1 || num < 0) {
        printf("Invalid input. Please enter a positive integer.\n");
        return 1; // Return error code
    }
    if (num == 0) factorial = 1;
    else for (i = num; i >= 1; i--) factorial = factorial * i;
    printf("The factorial of %d is: %lld\n", num, factorial);
    return 0;
}
```

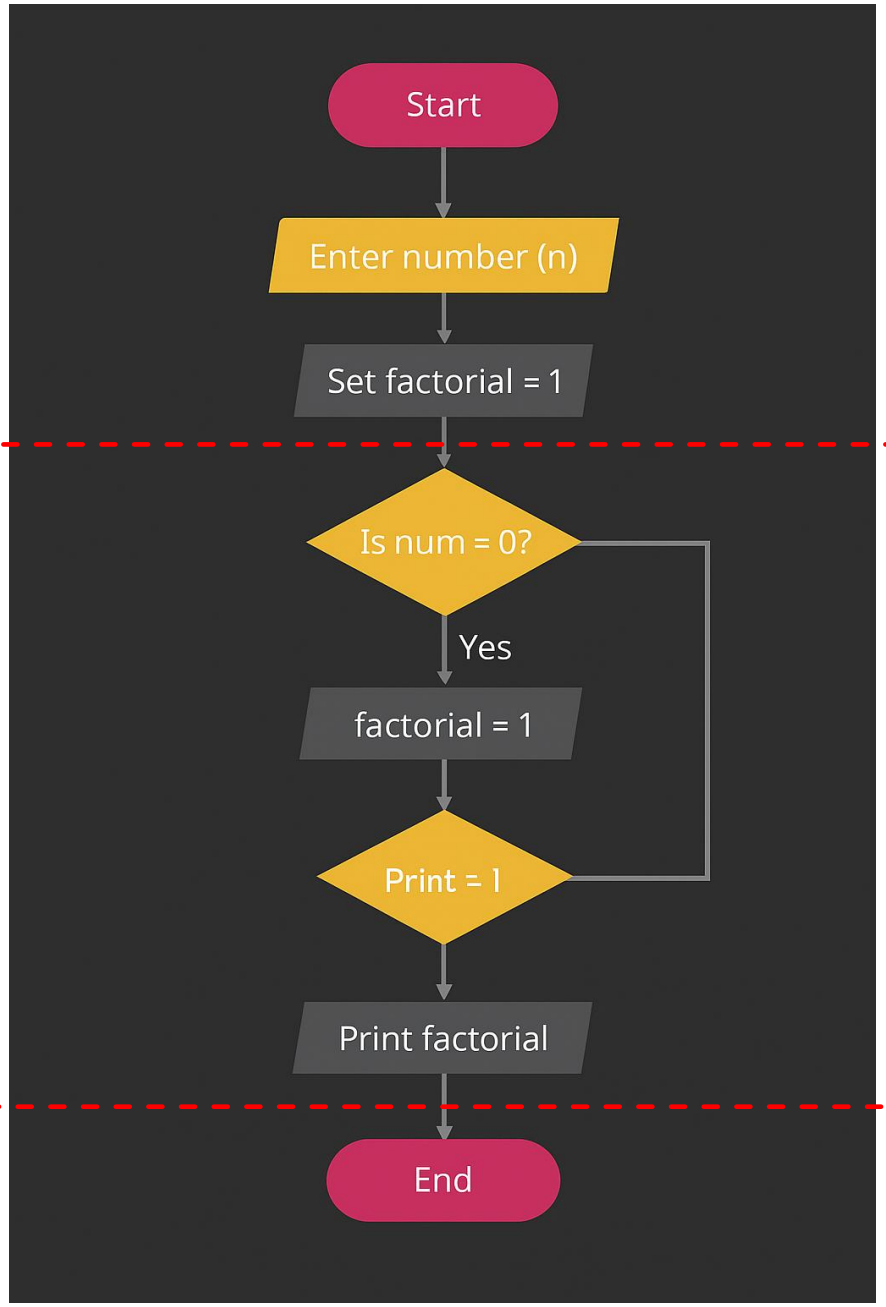
給予已知程式再詢問AI的題目

- Draw a picture of execution processes with flow chart for this program

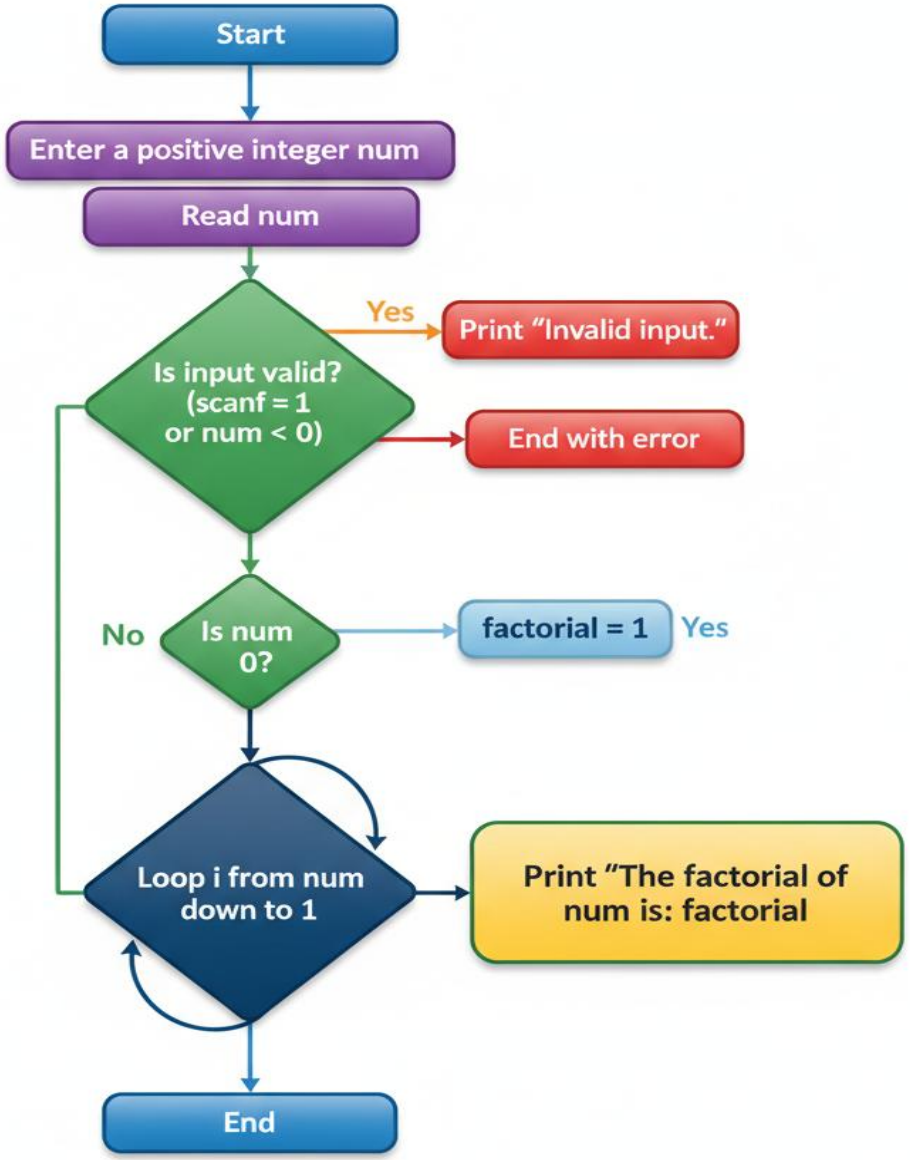
COPILOT 回覆


Start
Input num
Is num a positive integer?
No
Yes
Print 'Invalid input'
factorial = 1
Is $i \geq 1$?
$i = i - 1$
No
i == 1
factorial = 1
Print "The factorial of %d is: %lld", num, factorial
End

CHATGPT 回覆

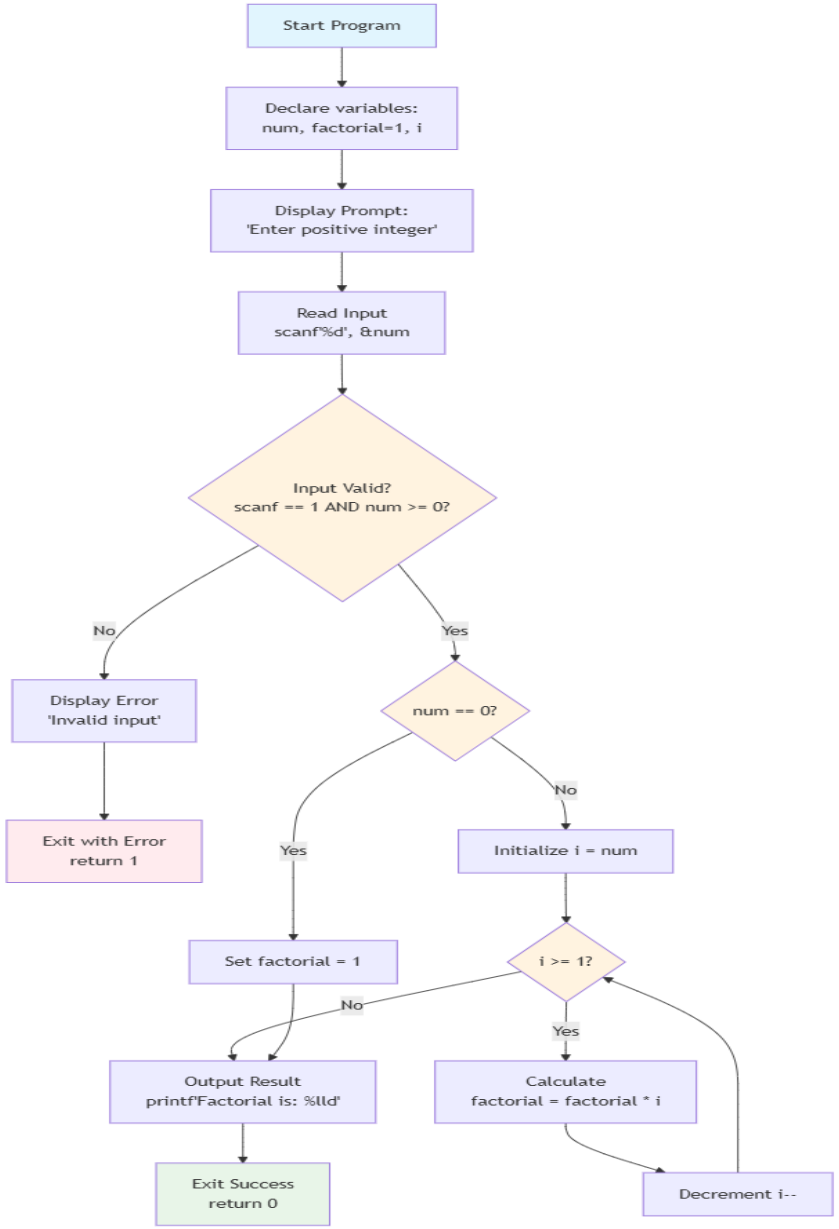

Start
Enter number (n)
Set factorial = 1
Is num = 0?
Yes
factorial = 1
Print = 1
Print factorial
End

GEMINI 回覆


Start
Enter a positive integer num
Read num
Is input valid? (scanf = 1 or num < 0)
Yes
Print "Invalid input."
End with error
No
Is num 0?
factorial = 1
Yes
Loop i from num down to 1
Print "The factorial of num is: factorial
End

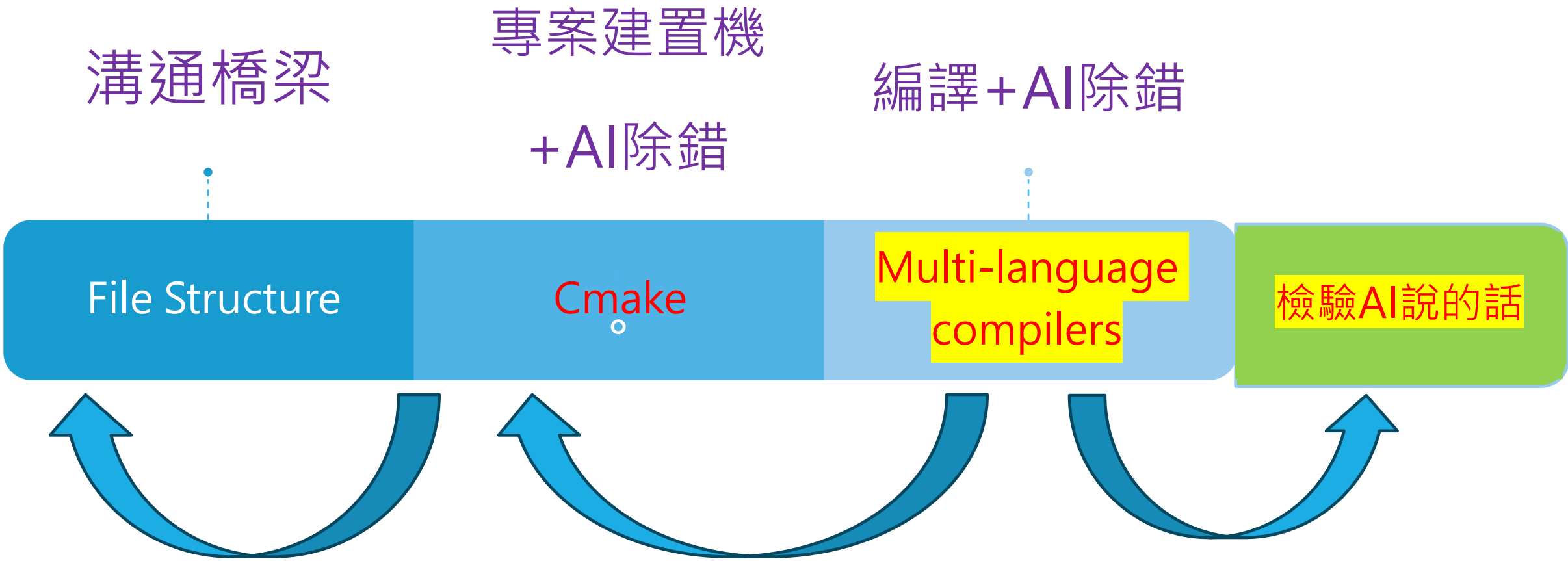
DEEPSEEK 回覆

AI檢驗自己說的話




```
Loop Start: i = 5
┌──Iteration 1
│  i = 5, factorial = 1
│  factorial = 1 * 5 = 5
│  i-- → i = 4
└──────────────────────

┌──Iteration 2
│  i = 4, factorial = 5
│  factorial = 5 * 4 = 20
│  i-- → i = 3
└──────────────────────

┌──Iteration 3
│  i = 3, factorial = 20
│  factorial = 20 * 3 = 60
│  i-- → i = 2
└──────────────────────

┌──Iteration 4
│  i = 2, factorial = 60
│  factorial = 60 * 2 = 120
│  i-- → i = 1
└──────────────────────

┌──Iteration 5
│  i = 1, factorial = 120
│  factorial = 120 * 1 = 120
│  i-- → i = 0
└──────────────────────

Loop End: i = 0 (0 >= 1? → False)
```

主要議題(II) 用AI實踐跨語言跨系統的專案



專案的主架構
FILE STRUCTURE

```
11VCPKG/                              # Project Root
├── CMakePresets.json                 # CMake 预设配置 high priority
├── CMakeLists.txt                    # Root CMake configuration
├── vcpkg.json                        # vcpkg dependencies
├── web/                              # Web client files
│   ├── index.html                    # Main HTML page
│   ├── css/                          # Stylesheets
│   │   └── homePageStyle.css         # CSS styles
│   └── script/                       # JavaScript files
│       └── AAA.js                    # Main JavaScript
├── jni_bridge/
│       ├── JavaIntegration.cpp
│       └── JavaIntegration.h
│
├── python_bridge/
│       ├── PythonModule.cpp
│       └── PythonModule.h
├── cpp/                              # C++ Components
│   ├── include/
│   │
│   ├── src/
│   │   ├── 11VCPKG.cpp               # Main C++ source file
│   │   └── 11VCPKG.h                 # Main C++ header file
│   │
│   └── lib/
│       └── static/
│           └── libjni_helpers.a
├── python/                           # NEW: Python Components
│   ├── pyproject.toml                # Python project config
│   ├── requirements.txt              # Python dependencies
│   ├── src/
│   │   └── my_python_pkg/
│   │       ├── __init__.py
│   │       ├── data_processor.py
│   │       └── utils.py
│   ├── scripts/
│   │   └── python_runner.py
│   └── tests/
│       └── test_processor.py
├── java/                             # Java Components
│   ├── build.gradle.kts              # ← ADDED: 主要的构建配置文件 by gradle init
│   ├── settings.gradle.kts           # ← ADDED: 项目设置文件 by gradle init
│   ├── gradlew                       # ← ADDED: Gradle wrapper (Unix/Linux/Mac) by gradle init
│   ├── gradlew.bat                   # ← ADDED: Gradle wrapper (Windows) by gradle init
│   ├── gradle/                       # ← ADDED: Gradle wrapper 目录 by gradle init
│   │   └── wrapper/
│   │       ├── gradle-wrapper.jar  # ← ADDED: Wrapper JAR 文件
│   │       └── gradle-wrapper.properties # ← ADDED: Wrapper 配置
```

專案自動建置機
CMAKE

```
# CMakeList.txt: 11VCPKG 的 CMake 專案，在此包含來源及定義
# 專案專屬邏輯。
# 在 project() 命令之前添加，設定 64 位元建置
cmake_minimum_required (VERSION 3.20)

# 只在 Visual Studio 生成器时设置平台
if(CMAKE_GENERATOR MATCHES "Visual Studio")
    set(CMAKE_GENERATOR_PLATFORM x64)
endif()

add_compile_definitions(_WIN32_WINNT=0x0A00)  # Target Windows 10 or later

# 如果支援，則為 MSVC 編譯器啟用熱重新載入。
if (POLICY CMP0141)
  cmake_policy(SET CMP0141 NEW)
  set(CMAKE_MSVC_DEBUG_INFORMATION_FORMAT "$<IF:$<AND:$<C_COMPILER_ID:MSVC>,$<CXX_COMPILER_ID:MSVC>>,$<$<CONFIG:Debug,RelWithI
endif()

message(STATUS "ENV{VCPKG_ROOT}: $ENV{VCPKG_ROOT}")
message(STATUS "VCPKG_ROOT: ${VCPKG_ROOT}")
message(STATUS "CMAKE_SOURCE_DIR : ${CMAKE_SOURCE_DIR}")
message(STATUS "CMAKE_CURRENT_SOURCE_DIR : ${CMAKE_CURRENT_SOURCE_DIR}")      # Current directory of this CMakeLists.txt

project ("11VCPKG")

# 在 project() 命令之后添加架构检查 64 位元或 32 位元
if(CMAKE_SIZEOF_VOID_P EQUAL 8)
    message(STATUS "Building for 64-bit architecture: This message indicates this system is 64-bit")
else()
    message(STATUS "Building for 32-bit architecture")
endif()

# 在 project() 命令之后添加架构检查
if(CMAKE_SIZEOF_VOID_P EQUAL 8)
    message(STATUS "✓ Building 64-bit architecture")
else()
    message(FATAL_ERROR "✗ This project must be built as 64-bit. Please use x64 preset.")
endif()

# 或者更严格的检查
if(NOT CMAKE_CL_64)
    message(FATAL_ERROR "This project requires 64-bit build. Use x64 preset.")
endif()

```