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

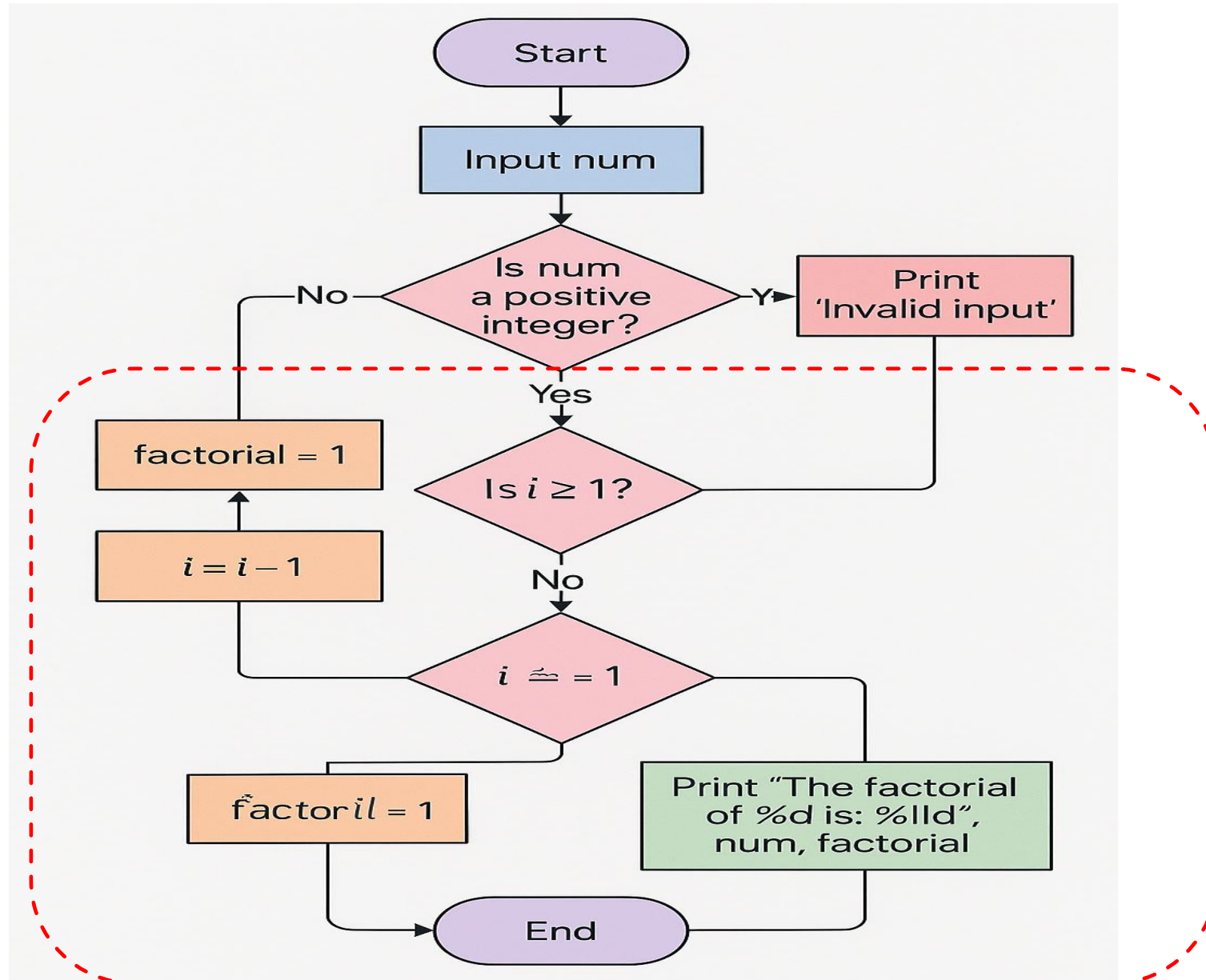| AI與我互不認識	AI 的回應正確嗎?	檢驗AI說的話

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;
}
```
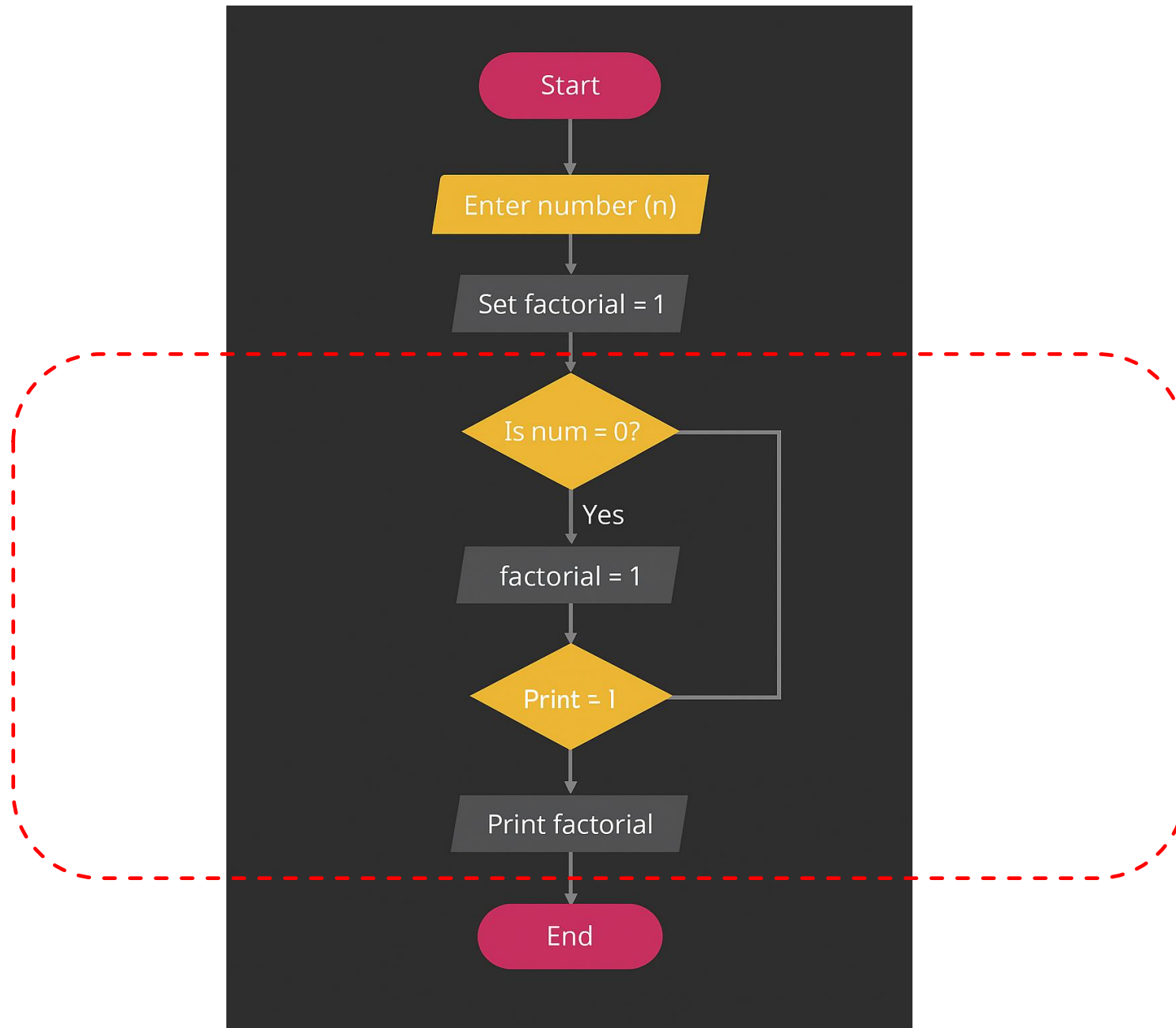
給予已知程式再詢問AI的題目

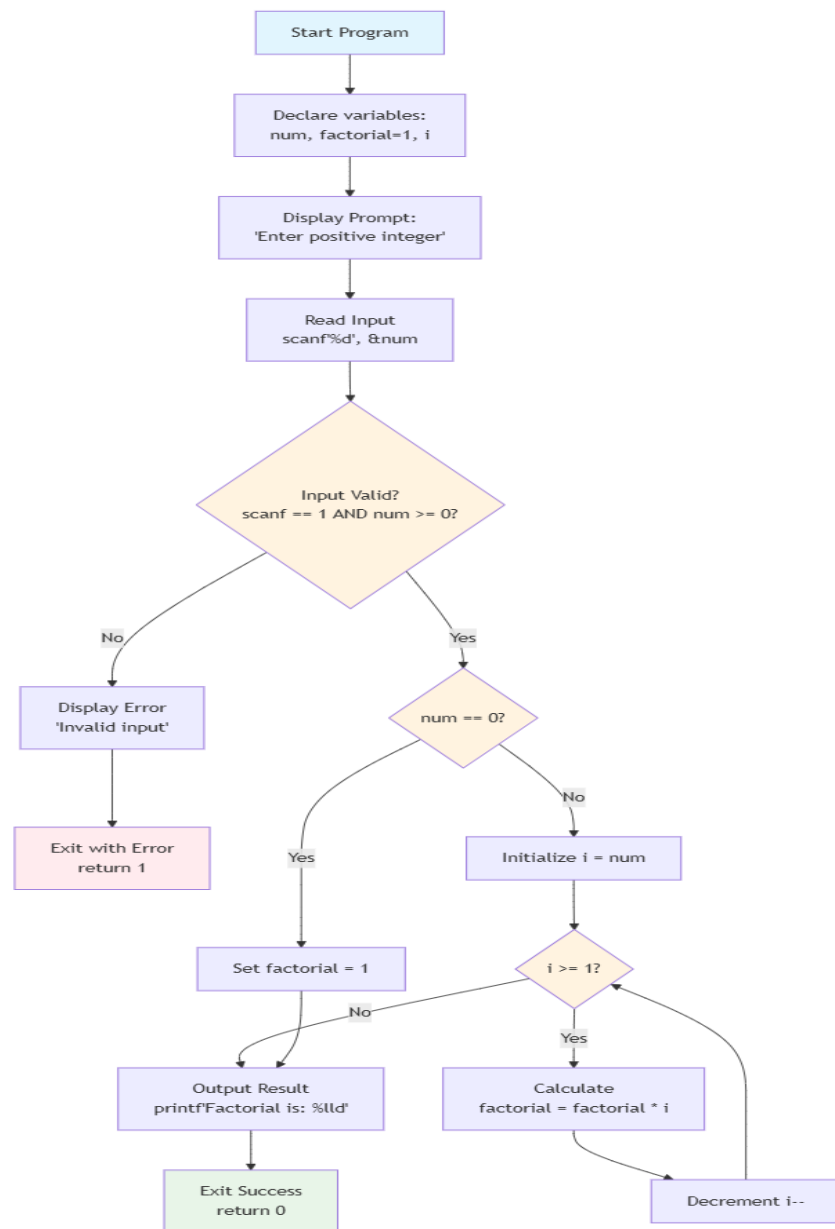- Draw a picture of execution processes with flow chart for this program

COPILOT 回覆

CHATGPT
回覆

DEEPSEEK 回覆

AI檢驗自己 說的話

Start Program

Declare variables: num, factorial=1, i

Display Prompt: 'Enter positive integer'

Read Input scanf'%d', &num

Input Valid? scanf == 1 AND num >= 0?

No → Display Error 'Invalid input' → Exit with Error return 1

Yes → num == 0?

Yes → Set factorial = 1

No → Initialize i = num

i >= 1?

No → Output Result printf'Factorial is: %lld' → Exit Success return 0

Yes → Calculate factorial = factorial * i → Decrement i--

```
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)
```

2025/10/20

主要議題(II) 用AI實踐跨語言跨系統的專案

溝通橋梁

專案建置機
+AI除錯

編譯+AI除錯

File Structure

Cmake

Multi-language compilers

檢驗AI說的話
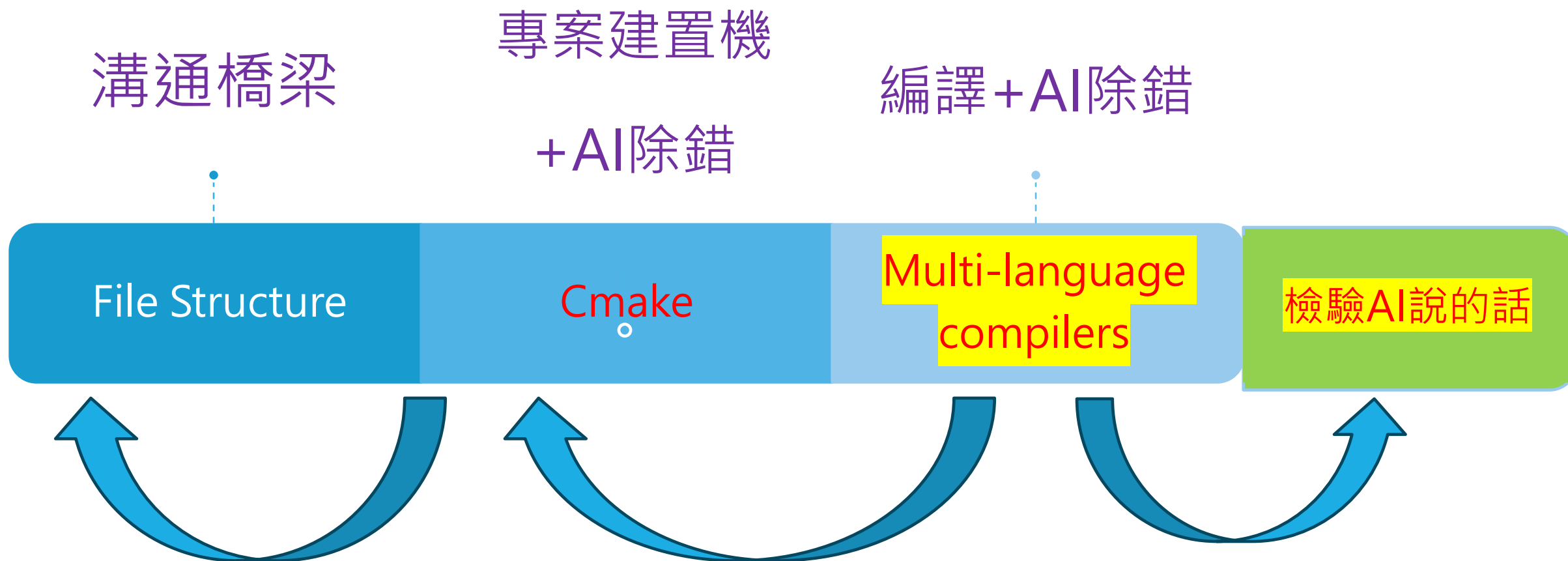
專案的主架構
FILE STRUCTURE

```
1   11VCPKG/                                # Project Root
2   ├── CMakePresets.json                   # CMake 预设配置 high priority
3   ├── CMakeLists.txt                      # Root CMake configuration
4   ├── vcpkg.json                          # vcpkg dependencies
5   ├── web/                                # Web client files
6   │   ├── index.html                      # Main HTML page
7   │   ├── css/                            # Stylesheets
8   │   │   └── homePageStyle.css           # CSS styles
9   │   └── script/                         # JavaScript files
10  │       └── AAA.js                      # Main JavaScript
11  ├── jni_bridge/
12  │       ├── JavaIntegration.cpp
13  │       └── JavaIntegration.h
14  │
15  ├── python_bridge/
16  │       ├── PythonModule.cpp
17  │       └── PythonModule.h
18  ├── cpp/                                # C++ Components
19  │   ├── include/
20  │   │
21  │   ├── src/
22  │   │   ├── 11VCPKG.cpp                  # Main C++ source file
23  │   │   └── 11VCPKG.h                    # Main C++ header file
24  │   │
25  │   └── lib/
26  │       └── static/
27  │           └── libjni_helpers.a
28  ├── python/                             # NEW: Python Components
29  │   ├── pyproject.toml                  # Python project config
30  │   ├── requirements.txt                # Python dependencies
31  │   ├── src/
32  │   │   └── my_python_pkg/
33  │   │       ├── __init__.py
34  │   │       ├── data_processor.py
35  │   │       └── utils.py
36  │   ├── scripts/
37  │   │   └── python_runner.py
38  │   └── tests/
39  │       └── test_processor.py
40  ├── java/                               # Java Components
41  │   ├── build.gradle.kts                # ← ADDED: 主要的构建配置文件 by gradle init
42  │   ├── settings.gradle.kts             # ← ADDED: 项目设置文件 by gradle init
43  │   ├── gradlew                         # ← ADDED: Gradle wrapper (Unix/Linux/Mac) by gradle init
44  │   ├── gradlew.bat                     # ← ADDED: Gradle wrapper (Windows) by gradle init
45  │   ├── gradle/                         # ← ADDED: Gradle wrapper 目录 by gradle init
46  │   │   └── wrapper/
47  │   │       ├── gradle-wrapper.jar      # ← ADDED: Wrapper JAR 文件
48  │   │       └── gradle-wrapper.properties # ← ADDED: Wrapper 配置
```

2025/10/20

專案自動建置機
CMAKE

```
1   # CMakeList.txt: 11VCPKG 的 CMake 專案，在此包含來源及定義
2   # 專案專屬邏輯。
3   # 在 project() 命令之前添加，設定 64 位元建置
4   cmake_minimum_required (VERSION 3.20)
5
6   # 只在 Visual Studio 生成器时设置平台
7   if(CMAKE_GENERATOR MATCHES "Visual Studio")
8       set(CMAKE_GENERATOR_PLATFORM x64)
9   endif()
10
11  add_compile_definitions(_WIN32_WINNT=0x0A00)  # Target Windows 10 or later
12
13  # 如果支援，則為 MSVC 編譯器啟用熱重新載入。
14  if (POLICY CMP0141)
15      cmake_policy(SET CMP0141 NEW)
16      set(CMAKE_MSVC_DEBUG_INFORMATION_FORMAT "$<IF:$<AND:$<C_COMPILER_ID:MSVC>,$<CXX_COMPILER_ID:MSVC>>,$<$<CONFIG:Debug,RelWithI
17  endif()
18
19  message(STATUS "ENV{VCPKG_ROOT}: $ENV{VCPKG_ROOT}")
20  message(STATUS "VCPKG_ROOT: ${VCPKG_ROOT}")
21  message(STATUS "CMAKE_SOURCE_DIR : ${CMAKE_SOURCE_DIR}")
22  message(STATUS "CMAKE_CURRENT_SOURCE_DIR : ${CMAKE_CURRENT_SOURCE_DIR}")       # Current directory of this CMakeLists.txt
23
24  project ("11VCPKG")
25
26  # 在 project() 命令之后添加架构检查 64 位元或 32 位元
27  if(CMAKE_SIZEOF_VOID_P EQUAL 8)
28      message(STATUS "Building for 64-bit architecture: This message indicates this system is 64-bit")
29  else()
30      message(STATUS "Building for 32-bit architecture")
31  endif()
32
33  # 在 project() 命令之后添加架构检查
34  if(CMAKE_SIZEOF_VOID_P EQUAL 8)
35      message(STATUS "✓ Building 64-bit architecture")
36  else()
37      message(FATAL_ERROR "✗ This project must be built as 64-bit. Please use x64 preset.")
38  endif()
39
40  # 或者更严格的检查
41  if(NOT CMAKE_CL_64)
42      message(FATAL_ERROR "This project requires 64-bit build. Use x64 preset.")
43  endif()
44
```