**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**Viện Công nghệ Thông tin và Truyền thông**

**-----\*\*\*-----**



**BÁO CÁO PROJECT I**

Mã lớp học: 699554

Giảng viên hướng dẫn: **TS. Nguyễn Tuấn Dũng**

Sinh viên: **Đặng Quang Huy** – 20183551

Hà Nội, tháng 11 năm 2020

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH 1](#_Toc57675842)

[**MỞ ĐẦU** 2](#_Toc57675843)

[**CHƯƠNG I: VIẾT HÀM NHÂN HAI MA TRẬN TRONG C++ VÀ GỌI HÀM ĐÓ TRONG JAVA** 3](#_Toc57675844)

[1. Phương pháp chung thực hiện sử dụng Java gọi các hàm C/C++ qua JNI 3](#_Toc57675845)

[2. Các bước cụ thể thực hiện bài toán 3](#_Toc57675846)

[**CHƯƠNG 2: VIẾT HÀM NHÂN HAI MA TRẬN TRONG JAVA VÀ GỌI HÀM ĐÓ TRONG C** 7](#_Toc57675847)

[1. Phương pháp chung thực hiện sử dụng C/C++ gọi các hàm Java qua JNI 7](#_Toc57675848)

[2. Các bước cụ thể thực hiện bài toán 7](#_Toc57675849)

[**KẾT LUẬN** 11](#_Toc57675850)

# DANH MỤC HÌNH ẢNH

[Hình 1 Khai báo phương thức Native](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675829) 3

[Hình 2 Tải thư viện liên kết động](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675830) 4

[Hình 3 Gọi đến phương thức Native](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675831) 4

[Hình 4 Khai báo phương thức Native trong header file](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675832) 5

[Hình 5 Khai báo các Header file trong C++](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675833) 5

[Hình 6 Kết quả thực hiện dùng Java gọi C++ để nhân hai ma trận](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675834) 6

[Hình 7 Thực thi phương thức nhân ma trận trong Java](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675835) 7

[Hình 8 Khởi tạo máy ảo JavaVM và con trỏ JNIEnv](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675836) 8

[Hình 9 Gọi đến phương thức Java từ trong C](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675837) 8

[Hình 10 Chuyển đổi kiểu mảng 2 chiều giữa C và Java](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675838) 9

[Hình 11 Kết quả gọi Java từ trong C thực hiện nhân hai ma trận](../../../../C:/Users/ADMIN/Downloads/20183551_%C4%90%E1%BA%B7ng-Quang-Huy_-B%C3%A1o-c%C3%A1o-Project-I.docx" \l "_Toc57675839) 10

# **MỞ ĐẦU**

Báo cáo thực hành học phần Project I gồm 2 chương tương ứng với 2 bài toán được yêu cầu, cụ thể gồm:

* Chương I: Viết hàm nhân hai ma trận bằng C++ và gọi hàm đó trong Java
* Chương II: Viết hàm nhân hai ma trận bằng Java và gọi hàm đó trong C.

Nhìn chung, việc liên kết giữa hai ngôn ngữ được thực hiện thông qua Java Native Interface (JNI). JNI cung cấp các phương thức cho phép C/C++ thao tác được với các đối tượng của Java. Đồng thời, Java cũng có thể gọi đến các phương thức của C/C++ thông qua thư viện liên kết động.

# **CHƯƠNG I: VIẾT HÀM NHÂN HAI MA TRẬN TRONG C++ VÀ GỌI HÀM ĐÓ TRONG JAVA**

## 1. Phương pháp chung thực hiện sử dụng Java gọi các hàm C/C++ qua JNI

Các bước thực hiện gọi các hàm C/C++ từ trong Java:

- Khai báo những phương thức cần được thực thi trong C/C++ bằng từ khóa “native” và không thực thi các phương thức này trong Java (các phương thức chỉ có khai báo mà không có phần thân phương thức).

- Dịch chương trình Java, sử dụng công cụ javah để tạo ra header file quy định khuôn dạng (tên phương thức, kiểu trả về,..) của các native method được thực thi trong C/C++

- Thực thi các phương thức trên trong C/C++

- Cần dịch và sinh ra file thư viện liên kết động từ những file mã nguồn C/C++ thực thi các phương thức trên. Trong hệ điều hành Linux, các file đó có phần đuôi là “.so”.

- Liên kết các thư viện liên kết đông trước khi sử dụng các “native method”

- Chạy chương trình Java

## 2. Các bước cụ thể thực hiện bài toán

**Bài toán:**

Thực thi hàm nhân hai ma trận kích thước m x n và n x k trong C++ và trả về ma trận kết quả trong file mã nguồn “Matrix.cpp”.

Lớp “MatrixUtil” trong Java định nghĩa phương thức native “multiply” thực hiện nhân hai ma trận.

Hàm “main” trong chương trình sẽ gọi đến phương thức này để nhân hai ma trận và in kết quả ra màn hình

**Các bước:**

- Khai báo phương thức native “multiply”:



Hình 1 Khai báo phương thức Native

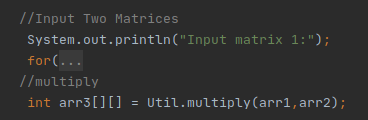
- Phương thức native trên sẽ được thực thi trong C++ và được liên kết thông qua file thư viện liên kết động “lib.so”.

Hình 2 Tải thư viện liên kết động

Khối lệnh static sẽ được hiện trước khi khởi chạy hàm main.

Lưu ý, chúng ta sử dụng lênh System.load() nên cần phải truyền vào đường dẫn tuyệt đối của file thư viện liên kết động.

- Thực hiện bài toán nhân hai ma trận, gọi hàm “multiply” để gọi C++ thực hiện hàm nhân và trả về kết quả:



Hình 3 Gọi đến phương thức Native

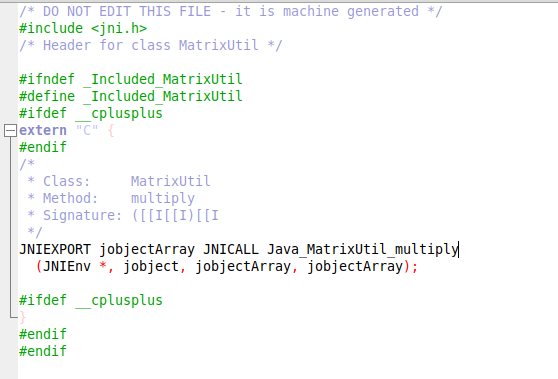
- Dịch chương trình:

***javac MatrixUtil.java***

- Tạo header file bằng công cụ javah:

***javah MatrixUtil***

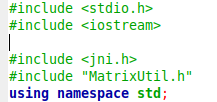
Thực hiện lệnh trên sẽ tạo ra file “MatrixUtil.h”

Phương thức native được thực thi trong file mã nguồn C++ cần phải có khai báo giống như phần khai báo trong file “MatrixUtil.h”.

Hình 4 Khai báo phương thức Native trong header file

- Thực thi phương thức multiply trong file mã nguồn C++: Matrix.cpp.

Lưu ý: Ngoài những header file cần thiết cho thực thi, file mã nguồn C++ cần phải include thêm file “jni.h” nằm trong Java JDK và header file “MatrixUtil.h” chúng ta tạo ra từ trước.



Hình 5 Khai báo các Header file trong C++

- Dịch file mã nguồn C++:

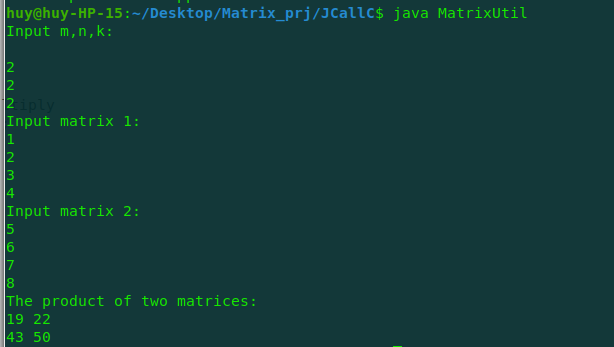
***gcc -I/path/to/your/jdk/include -I/path/to/your/jdk/include/linux -o lib.so -shared -fpic Matrix.cpp***

Các tham số của lệnh gcc ở đây bao gồm đường dẫn tới file “jni.h” trong Java JDK và các header file cần thiết khác. Đồng thời lệnh trên sinh ra file thư viện liên kết động “lib.so”.

- Chạy chương trình Java:

***java MatrixUtil***

Kết quả:



Hình 6 Kết quả thực hiện dùng Java gọi C++ để nhân hai ma trận

# **CHƯƠNG 2: VIẾT HÀM NHÂN HAI MA TRẬN TRONG JAVA VÀ GỌI HÀM ĐÓ TRONG C**

## 1. Phương pháp chung thực hiện sử dụng C/C++ gọi các hàm Java qua JNI

Các bước thực hiện:

- Khởi tạo một máy áo JavaVM và con trỏ JNIEnv để gọi đến các phương thức trong JNI.

- Sử dụng các phương thức trong JNI để tìm kiếm các class, khởi tạo các đối tượng, gọi đến các trường và các phương thức của các đối tượng, …

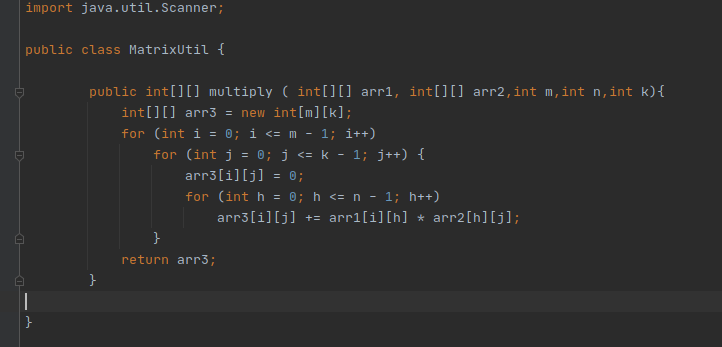
## 2. Các bước cụ thể thực hiện bài toán

**Bài toán:** Hàm “multiply” thực hiện nhân hai ma trận kích thước m x n và n x k được viết trong class MatrixUtil của Java.

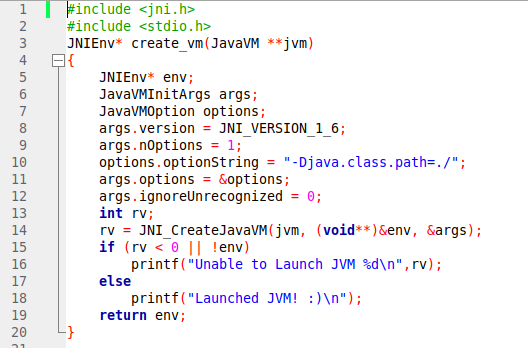
Hàm main trong file mã nguồn “Matrix.c” nhập vào hai ma trận tương ứng và gọi đến hàm “multiply” trong java để nhân hai ma trận, sau đó in kết quả ra màn hình.

**Các bước:**

- Khai báo và thực thi phương thức multiply trong Java

- Khởi tạo máy ảo JavaVM và con trỏ JNIEnv

Hình 7 Thực thi phương thức nhân ma trận trong Java

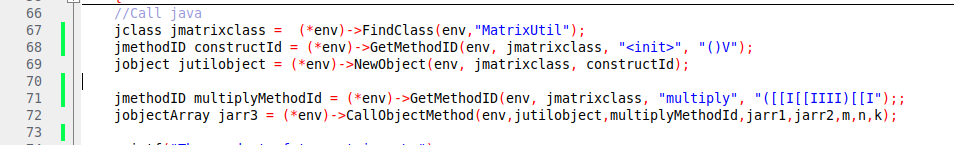


Hình 8 Khởi tạo máy ảo JavaVM và con trỏ JNIEnv

Cần phải include header file “jni.h”.

Nhập vào hai ma trận kích thước m x n và n x k.

Gọi phương thức Java trong C để nhân hai ma trận:



Hình 9 Gọi đến phương thức Java từ trong C

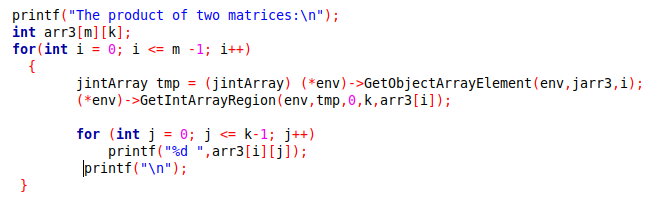
env là con trỏ JNIEnv\*.

Phương thức FindCLass() giúp chúng ta tìm kiếm class MatrixUtil (nơi chúng ta thực thi phương thức multiply trong java) và trả về một biến thuộc kiểu jclass đại diện cho class đó.

Để gọi được các phương thức trong Java, chúng ta cần phải có các jmethodID (tương tự là jfieldID đối với các thuộc tính của các lớp). Hàm GetMethodID() trả về một biến jmethodID tương ứng với tên của phương thức và chữ ký của các phương thức đó (được định nghĩa riêng bởi JNI).

Hàm CallObjectMethod() gọi đến phương thức multiply trong java và trả về một biến jobjectArray tương ứng với ma trận kết quả trả về.

- Chúng ta cần chuyển đổi biến jobjectArray trên về kiểu mảng hai chiều thông thường của C:

- Dịch và chạy chương trình C:

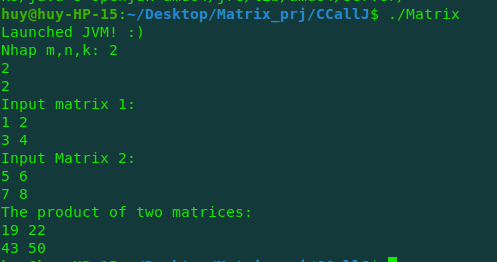
Hình 10 Chuyển đổi kiểu mảng 2 chiều giữa C và Java

Dịch: ***gcc -o Matrix -L/path/to/your/jdk/jre/lib/amd64/server/ -I/path/to/your/jdk/include -I/path/to/your/jdk/include/linux -L/path/to/your/jdk/bin/java Matrix.c -ljvm***

Thêm biến môi trường: ***export LD\_LIBRARY\_PATH="/path/to/your/jdk/jre/lib/amd64/server/"***

Chạy: ***./Matrix***

Kết quả:



Hình 11 Kết quả gọi Java từ trong C thực hiện nhân hai ma trận

# **KẾT LUẬN**

Báo cáo đã nêu được những bước chính cụ thể trong từng bài toán, thực hiện đúng yêu cầu mà giáo viên đặt ra.

Bản thân em khi thực hiện các chương trình trên cũng đã gặp những khó khăn nhất định, những lỗi trong chương trình do còn xa lạ với Ubuntu và JNI.

Báo cáo cũng như chương trình minh họa có thể vẫn có những thiếu sót, em rất mong nhận được sự góp ý từ phía thầy để có thể hoàn thiện hơn kĩ năng của bản thân.

Em xin chân thành cảm ơn!