BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC KINH TẾ-KỸ THUẬT CÔNG NGHIỆP

KHOA CÔNG NGHỆ THÔNG TIN

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO**

Môn học: **THỰC TẬP AN TOÀN THÔNG TIN**

**Đề Tài**

**Lập trình mã hóa và giải mã hệ mật mã Hill cipher và mã khóa công khai RSA**

**triển khai trên C# Winform**

Sinh viên thực hiện: Lê Văn Tiến

Nhóm: 8

Lớp học phần: 010100018120

Khóa: K15

Giảng viên hướng dẫn: Nguyễn Thu Hiền

*Hà Nội, ngày 8 tháng 1 năm 2024*

Contents

[**LỜI MỞ ĐẦU** 3](#_Toc182580958)

[**CHƯƠNG 1: HỆ MÃ HÓA HILL CIPHER** 4](#_Toc182580959)

[**I. Làm quen với hệ mã hóa Hill cipher** 4](#_Toc182580960)

[**1. Một số khái niệm liên quan** 4](#_Toc182580961)

[**2. Ma trận khả nghịch trên vành**  4](#_Toc182580962)

[**3. Các bước để mã hóa và giải mã thông tin** 5](#_Toc182580963)

[**4. Ví dụ** 6](#_Toc182580964)

[**5. Ưu nhược điểm** 6](#_Toc182580965)

[**6. Ứng dụng** 7](#_Toc182580966)

[**II. Triển Khai Hill Cipher bằng C# WinForm** 7](#_Toc182580967)

[**1. Thiết kế Giao diện Người dùng** 8](#_Toc182580968)

[**2. Sơ đồ khối của thuật toán:** 10](#_Toc182580969)

[**3. Quy trình chính của code:** 13](#_Toc182580970)

[**4. Chạy thử trên phần mềm Winform C#** 17](#_Toc182580971)

[**CHƯƠNG 2: HỆ MÃ HÓA RSA** 23](#_Toc182580972)

[**I. Làm quen với hệ mã hóa RSA** 23](#_Toc182580973)

[**1. Một số khái niệm liên quan** 23](#_Toc182580974)

[**2. Cơ sở toán học của RSA** 23](#_Toc182580975)

[**3. Các bước mã hóa và giải mã thông tin** 24](#_Toc182580976)

[**4. Ví dụ** 24](#_Toc182580977)

[**5.Ưu nhược điểm** 25](#_Toc182580978)

[**6.Ứng dụng** 25](#_Toc182580979)

[**II. Triển Khai RSA bằng C# WinForm** 25](#_Toc182580980)

[**1. Thiết kế Giao diện Người dùng** 26](#_Toc182580981)

[**2. Sơ đồ khối của thuật toán:** 28](#_Toc182580982)

[**3. Quy trình chính của code:** 32](#_Toc182580983)

[**4. Chạy thử trên phần mềm Winform C#** 33](#_Toc182580984)

# **LỜI MỞ ĐẦU**

Trong môi trường ngày nay, bảo mật thông tin đóng vai trò trung tâm trong mọi hoạt động từ kinh doanh, chính trị đến đời sống cá nhân. Khi ngày càng có nhiều dữ liệu được lưu trữ và trao đổi qua môi trường mạng thì nhu cầu mã hóa để bảo vệ thông tin nhạy cảm cũng ngày càng trở nên quan trọng. Các thuật toán mã hóa như Hill cipher và RSA được coi là phương pháp mã hóa phổ biến và hiệu quả trong việc bảo vệ thông tin.

Mật mã Hill, một thuật toán mã hóa cổ xưa dựa trên đại số tuyến tính, sử dụng các phép biến đổi ma trận để bảo vệ dữ liệu. Mặc dù đơn giản nhưng mật mã Hill vẫn được sử dụng rộng rãi trong các hệ thống yêu cầu mã hóa cơ bản do khả năng mã hóa nhanh và dễ sử dụng. Ngược lại, RSA là một trong những thuật toán mã hóa bất đối xứng hiện đại, có đặc điểm là bảo mật mạnh mẽ dựa trên các mẫu số và hàm số nguyên lớn. Thường được sử dụng trong các hệ thống bảo mật cao, RSA không chỉ mã hóa mà còn giúp xác thực và bảo vệ tính toàn vẹn của dữ liệu.

Báo cáo này sẽ tập trung vào việc triển khai lập trình mã hóa và giải mã bằng hai thuật toán Hill cipher và RSA, xem xét hiệu năng, ưu điểm và nhược điểm của từng phương pháp. Qua đó, độc giả sẽ cùng tìm hiểu chi tiết về biện pháp bảo mật thông tin của các thuật toán mã hóa phổ biến này.

# **CHƯƠNG 1: HỆ MÃ HÓA HILL CIPHER**

## **I. Làm quen với hệ mã hóa Hill cipher**

### **1. Một số khái niệm liên quan**

* **Mật mã khóa đối xứng**: Là loại mật mã trong đó cùng một khóa được dùng để mã hóa và giải mã thông tin. Hệ mật mã Hill là một ví dụ điển hình của loại này.
* **Mã hóa theo khối**: Phương pháp mã hóa dữ liệu theo từng khối ký tự, thay vì mã hóa từng ký tự một. Trong hệ mật mã Hill, dữ liệu được chia thành các khối có kích thước bằng bậc của ma trận khóa.
* **Ma trận khóa**: Ma trận vuông dùng làm khóa để mã hóa và giải mã thông tin. Ma trận này phải có khả năng nghịch đảo trên vành để có thể giải mã thành công.
* **Vành**  : Là tập hợp các số nguyên từ 0 đến 25, cùng với các phép toán cộng và nhân theo modulo 26. Hệ thống này dùng để ánh xạ các ký tự trong bảng chữ cái tiếng Anh.

### **2. Ma trận khả nghịch trên vành**

* Một ma trận khóa **𝐴** trong hệ mật mã Hill phải có định thức khác không và có ước chung lớn nhất với 26 là 1 (tức là, **gcd( det(A), 26) = 1**) để đảm bảo rằng ma trận có thể đảo ngược trên vành ​.
* Để tìm ma trận nghịch đảo của **A** trên vành , ta cần:

1. Tính định thức của **A : det(A)**
2. Tính nghịch đảo của định thức trong modulo 26, ký hiệu là  **mod 26**
3. Nhân ma trận liên hợp (adjugate) của **A** ký hiệu là **adj(A)**, với nghịch đảo của định thức (theo modulo 26) để thu được ma trận nghịch đảo:



### **3. Các bước để mã hóa và giải mã thông tin**

* **Mã hóa:**
  1. Chia văn bản rõ thành các khối ký tự có kích thước bằng bậc của ma trận khóa **A**.
  2. Chuyển đổi mỗi ký tự trong khối thành số tương ứng theo bảng chữ cái (A = 0, B = 1, …, Z = 25).
  3. Tạo vector **P** từ các số tương ứng trong khối văn bản rõ.
  4. Nhân ma trận khóa **A** với vector **P**, rồi lấy modulo 26 để có được vector mã hóa **C**:

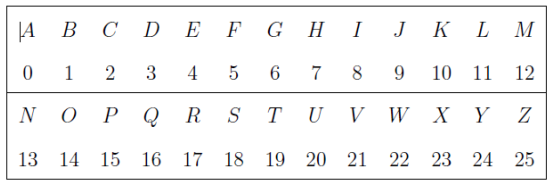


* 1. Ánh xạ các số trong vector **C** trở lại các ký tự để tạo thành khối văn bản mã hóa.
* **Giải mã:**
  1. Chuyển đổi các ký tự trong văn bản mã hóa thành các số và tạo vector **C**.
  2. Nhân ma trận nghịch đảo của khóa với vector **C** và lấy modulo 26 để thu được vector của văn bản rõ **P**:



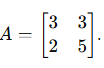
* 1. Ánh xạ các số trong vector **P** trở lại các ký tự để khôi phục văn bản gốc.

(\*): Quy trình thực hiện chuyển đổi giữa số ⬄ chữ được thực hiện qua bảng sau:



### **4. Ví dụ**

* Giả sử văn bản cần mã hóa là “HELP” với ma trận khóa



* Chia văn bản thành các khối: “HE” và “LP”.
* Chuyển đổi ký tự thành số: “H” = 7, “E” = 4; “L” = 11, “P” = 15.
* Mã hóa khối “HE”:



Ánh xạ 7 và 8 trở lại ký tự: “HE” mã hóa thành “HI”.

* Mã hóa khối “LP”:



Ánh xạ 0 và 19 trở lại ký tự: “LP” mã hóa thành “AT”.

* Kết quả mã hóa của “HELP” là “HIAT”.

### **5. Ưu nhược điểm**

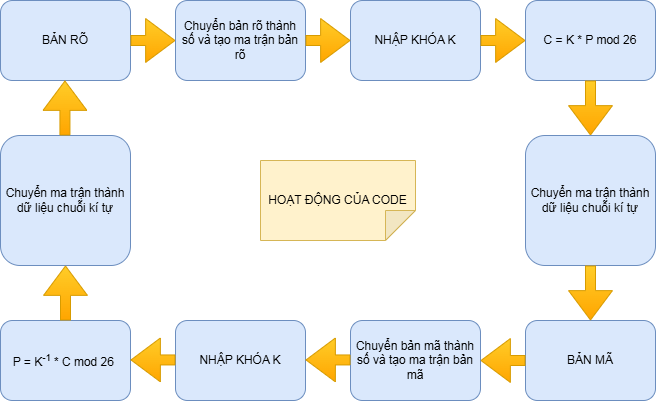
* **Ưu điểm**:
  + Đơn giản trong việc mã hóa và giải mã, dễ dàng thực hiện bằng các phép toán tuyến tính.
  + Do sử dụng mã hóa theo khối, hệ mật mã Hill bảo mật hơn so với các hệ mã hóa ký tự đơn.
* **Nhược điểm**:
  + Dễ bị tấn công bằng phân tích tuyến tính nếu kẻ tấn công có đủ cặp văn bản rõ và văn bản mã hóa.
  + Nếu ma trận khóa không khả nghịch trên , việc giải mã sẽ không thực hiện được.

### **6. Ứng dụng**

* Hệ mật mã Hill được dùng trong giảng dạy, nghiên cứu và học tập để minh họa các nguyên lý về mật mã hóa và đại số tuyến tính.
* Mặc dù không thường xuyên được sử dụng trong các ứng dụng thực tế yêu cầu bảo mật cao, hệ mật mã Hill giúp cung cấp cơ sở cho các hệ mã hóa phức tạp hơn.

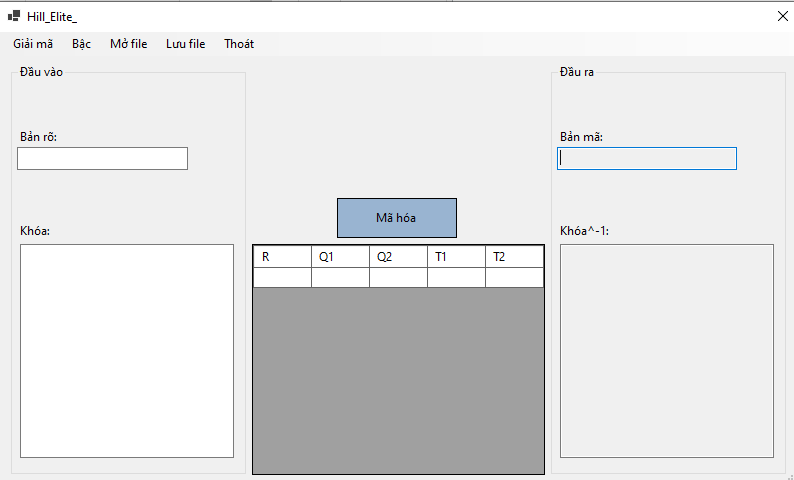
## **II. Triển Khai Hill Cipher bằng C# WinForm**

Việc triển khai Hill cipher trên C# WinForm sẽ giúp tạo ra một ứng dụng dễ sử dụng, cho phép người dùng nhập khóa và văn bản, sau đó thực hiện mã hóa và giải mã một cách trực quan. Quy trình tổng quan được thực hiện như sau:



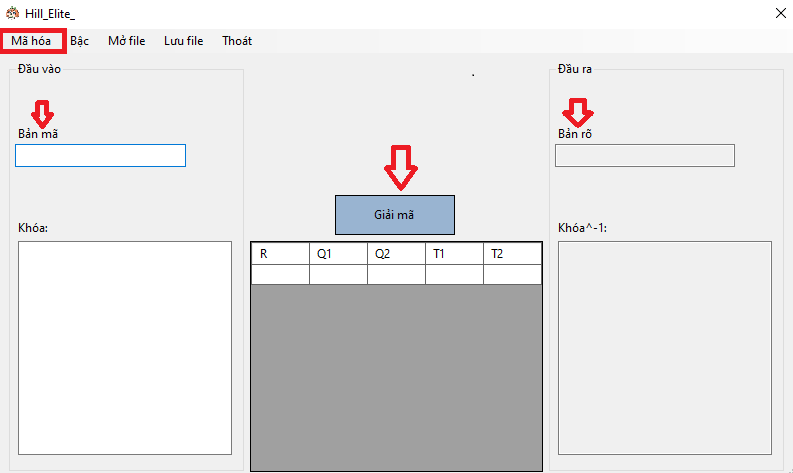
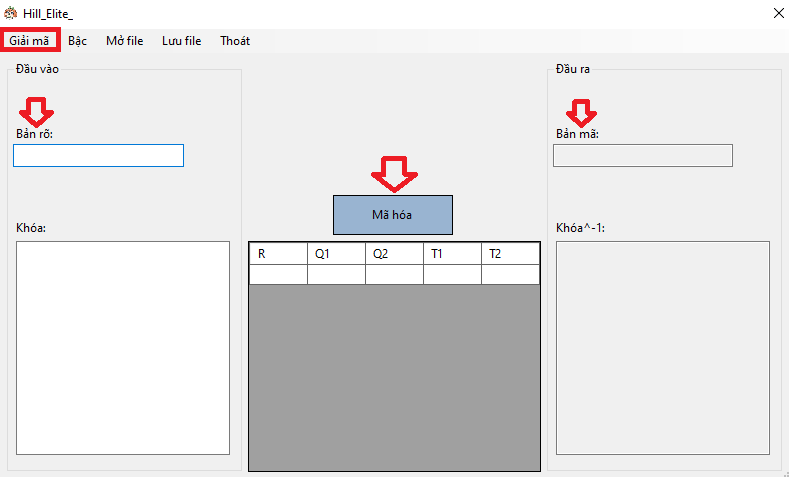
### **1. Thiết kế Giao diện Người dùng**

#### **1.1. Giao diện thiết kế cho hệ mật mã Hill cipher:**

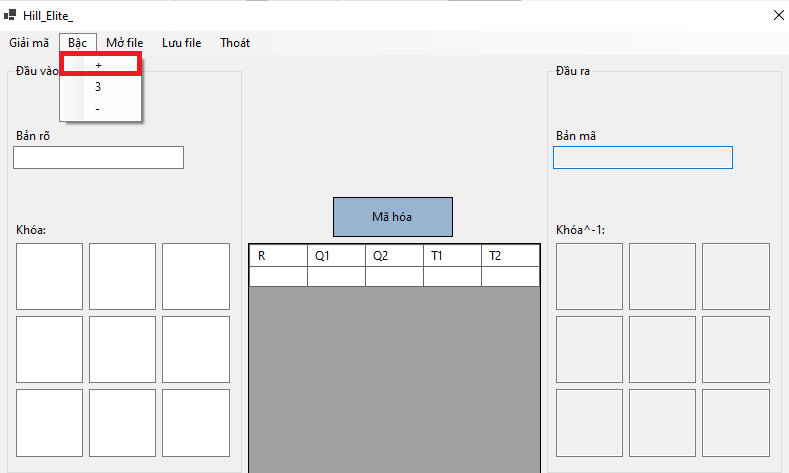
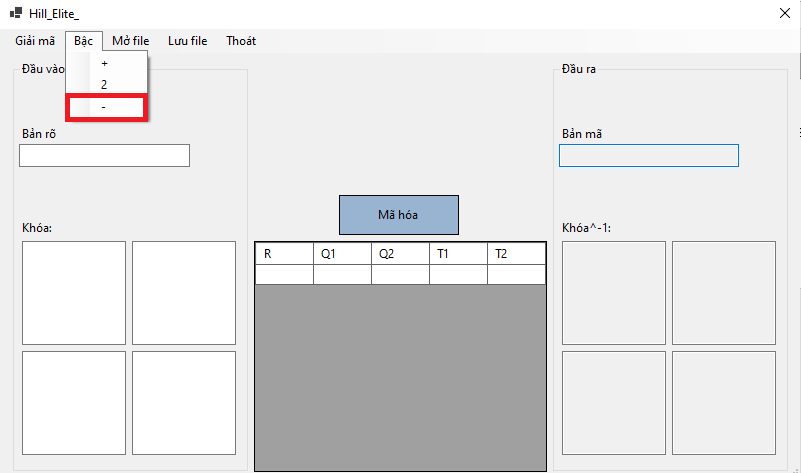


* MenuStrip: Tổ chức chức năng theo danh mục
* GroupBox: Chứa đầu vào, đầu ra
* Button: Thực hiện mã hóa, giải mã
* DataGridView: Hiển thị quy trình tìm

#### **1.2. Giao diện Giải mã/Mã hóa: Chọn chế độ hoạt động của chương trình, chuyển đổi giữa chế độ mã hóa và giải mã**

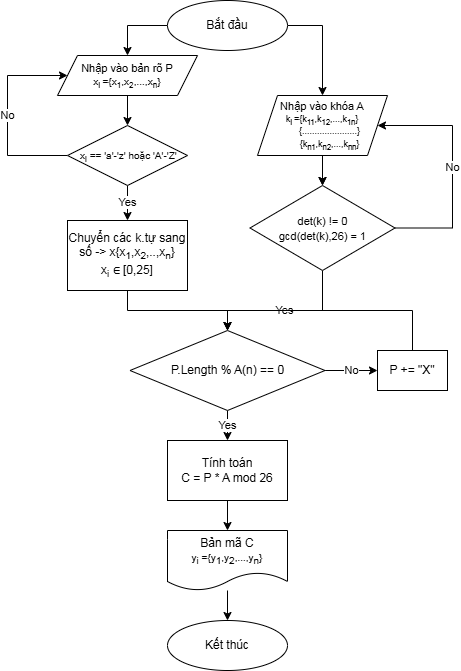


#### **1.3. Giao diện thay đổi Bậc: Tùy chọn này cho phép tăng hoặc giảm bậc của khóa. Bậc của khóa xác định kích thước của ma trận khóa, ví dụ 2x2, 3x3, v.v.,**



### **2. Sơ đồ khối của thuật toán:**

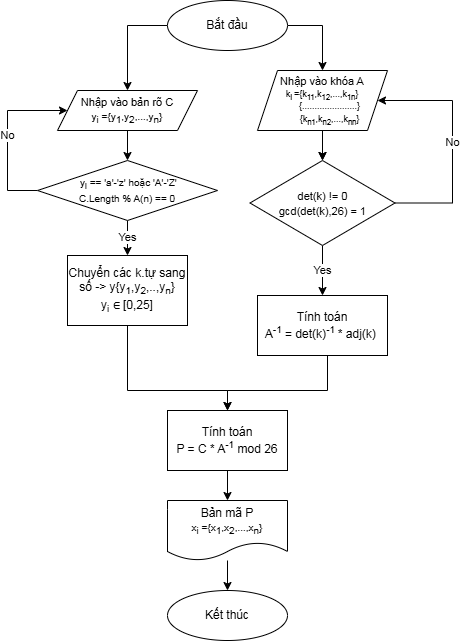
#### **2.1. Mã hóa:**



1. Sơ đồ khối này mô tả quy trình mã hóa bản rõ **P** bằng cách sử dụng một khóa **A** trong hệ mã. Đầu tiên, chương trình nhận bản rõ **P** dưới dạng chuỗi ký tự. Nếu chuỗi này chứa ký tự không nằm trong phạm vi từ 'a' đến 'z' hoặc 'A' đến 'Z', chương trình sẽ yêu cầu nhập lại. Khi bản rõ hợp lệ, các ký tự sẽ được chuyển thành dạng số trong phạm vi từ 0 đến 25.
2. Tiếp theo, chương trình nhận khóa **A**, là ma trận hệ số với điều kiện rằng định thức của ma trận phải khác 0 và ước chung lớn nhất của định thức với 26 phải bằng 1. Nếu khóa không đạt yêu cầu, người dùng sẽ phải nhập lại khóa mới.
3. Sau khi có khóa hợp lệ, chương trình kiểm tra độ dài của bản rõ **P**. Nếu độ dài không chia hết cho **A(n)** (số chiều của ma trận khóa), một ký tự 'X' sẽ được thêm vào cuối bản rõ để đảm bảo độ dài phù hợp.
4. Cuối cùng, chương trình tiến hành tính toán mã hóa bằng công thức **C = P ∗ A mod 26** để tạo ra bản mã **C**, với mỗi phần tử ​ thuộc bản mã. Sau đó, bản mã sẽ được xuất ra và quy trình kết thúc.

=> Sơ đồ khối mã hóa mô tả rõ ràng các bước cần thiết để thực hiện mã hóa một bản rõ thành bản mã, sử dụng phương pháp mã hóa ma trận. Quy trình kiểm tra đầu vào như độ dài và ký tự không hợp lệ đảm bảo tính chính xác của dữ liệu đầu vào. Việc thêm ký tự "X" nếu độ dài không đủ cũng giúp bản rõ tương thích với ma trận khóa. Nhìn chung, quy trình này dễ hiểu và hiệu quả cho việc mã hóa thông tin bằng khóa ma trận trong hệ mã modular 26.

#### **2.2. Giải mã:**

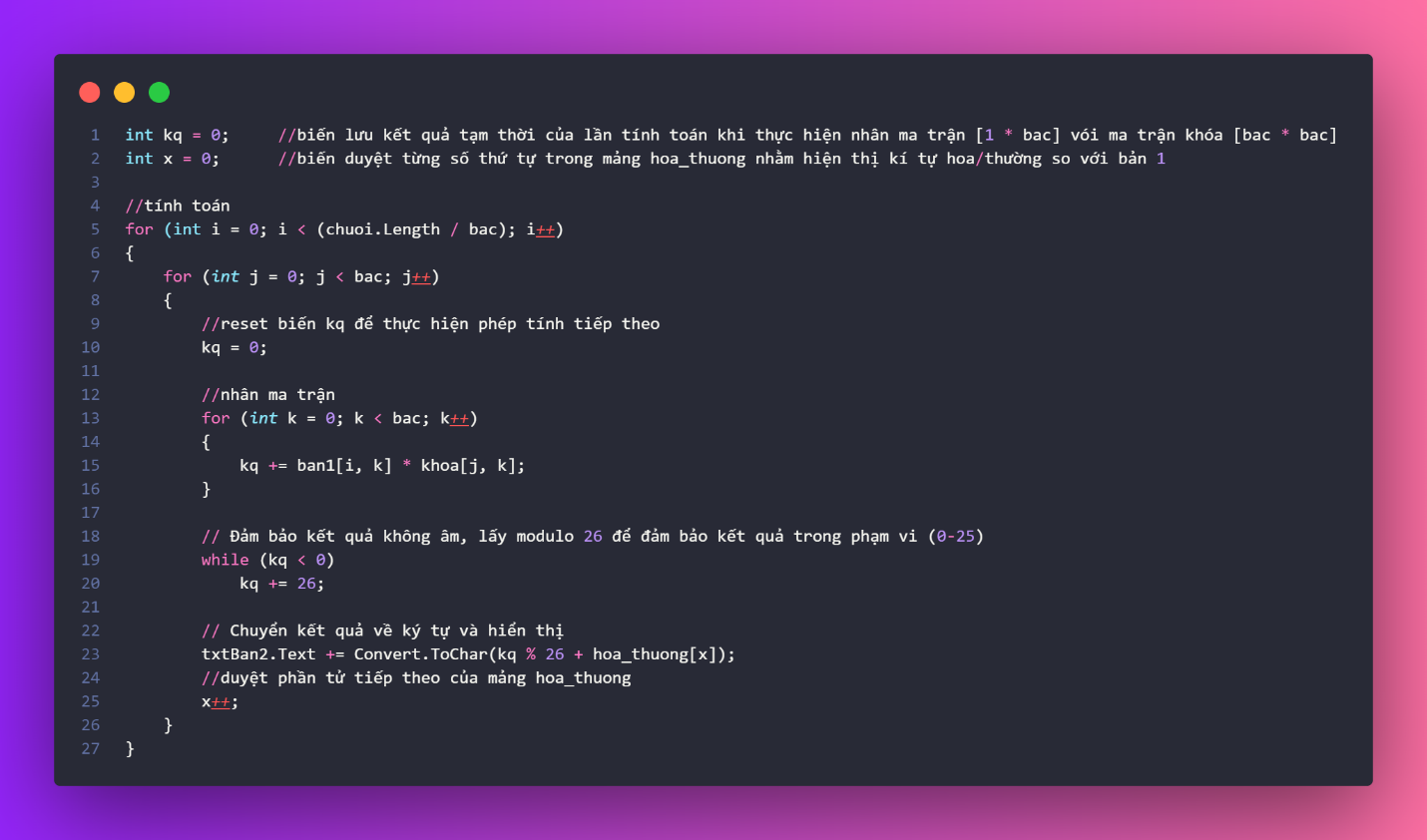


1. Sơ đồ khối này mô tả quy trình giải mã bản mã **C** để khôi phục bản rõ **P** bằng cách sử dụng khóa **A**. Đầu tiên, chương trình nhận bản mã **C** dưới dạng chuỗi ký tự. Nếu chuỗi này chứa ký tự không nằm trong phạm vi từ 'a' đến 'z' hoặc 'A' đến 'Z', hoặc độ dài của bản mã không chia hết cho kích thước ma trận khóa **A(n)**, chương trình sẽ yêu cầu nhập lại.
2. Sau khi bản mã hợp lệ, các ký tự sẽ được chuyển thành dạng số trong phạm vi từ 0 đến 25. Tiếp theo, chương trình nhận khóa **A**, là ma trận hệ số ​, với điều kiện rằng định thức của ma trận phải khác 0 và ước chung lớn nhất của định thức với 26 phải bằng 1. Nếu khóa không đạt yêu cầu, người dùng sẽ phải nhập lại khóa mới.
3. Sau khi có khóa hợp lệ, chương trình tính ma trận nghịch đảo của khóa bằng công thức  **= \* adj(k)**, trong đó **adj(k)** là ma trận phụ đại số của **k**.
4. Cuối cùng, chương trình tiến hành giải mã bằng công thức **P = C ∗ mod 26** để khôi phục bản rõ **P**, với mỗi phần tử ​ thuộc bản rõ. Sau đó, bản rõ sẽ được xuất ra và quy trình kết thúc.

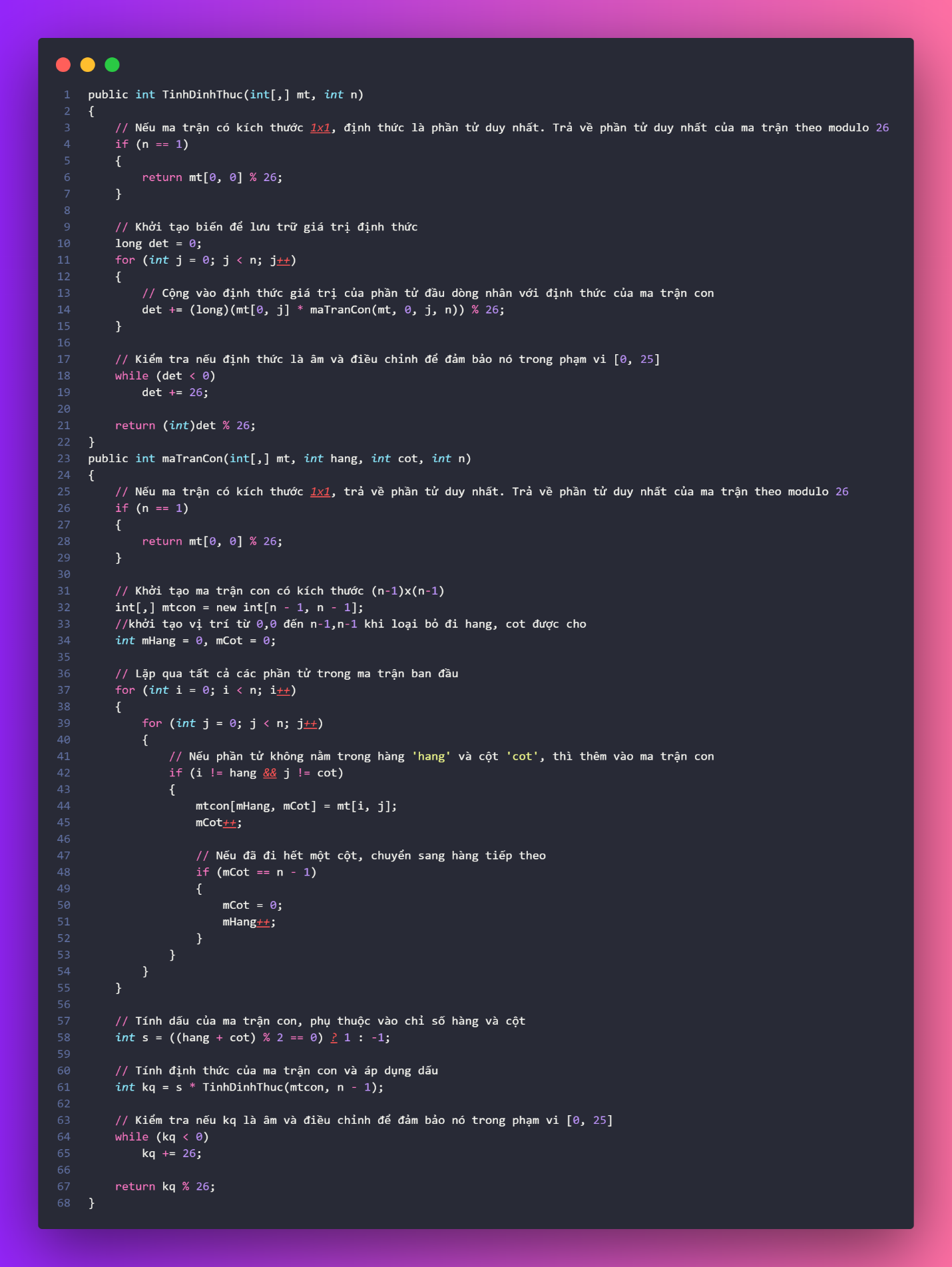
=> Sơ đồ khối giải mã chi tiết các bước cần thiết để giải mã bản mã thành bản rõ, sử dụng khóa và ma trận nghịch đảo. Việc kiểm tra đầu vào để đảm bảo tính hợp lệ của bản mã và khóa trước khi tiến hành giải mã là cần thiết để tránh lỗi trong quá trình xử lý. Sử dụng ma trận nghịch đảo giúp đảm bảo rằng quá trình giải mã sẽ khôi phục chính xác bản rõ ban đầu. Quy trình này chặt chẽ và hiệu quả cho việc giải mã trong hệ mã modular 26, đảm bảo tính chính xác và tính toàn vẹn của thông tin.

### **3. Quy trình chính của code:**

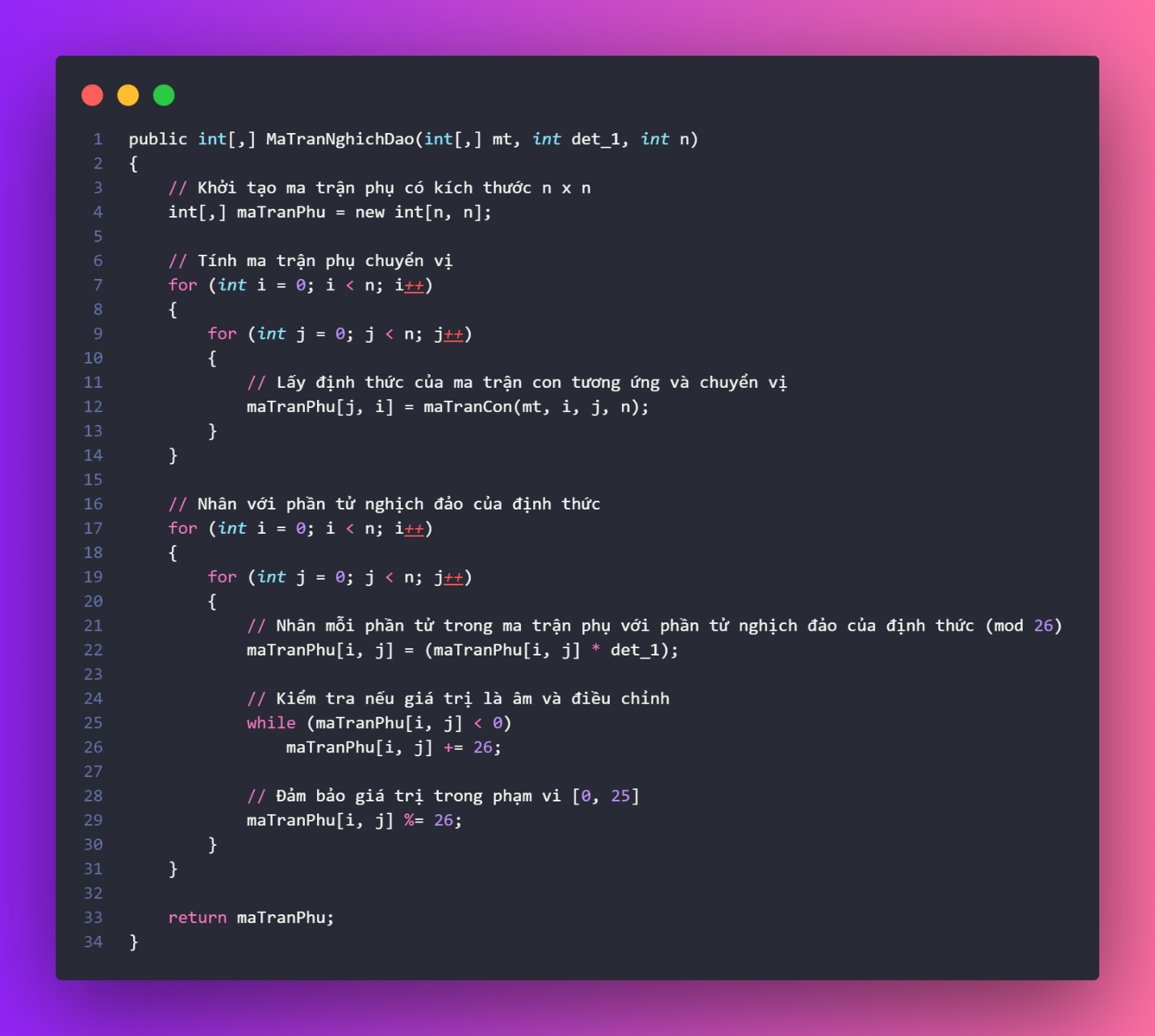
#### **3.1. Nhân ma trận:**



#### **3.2. Ma trận nghịch đảo:**



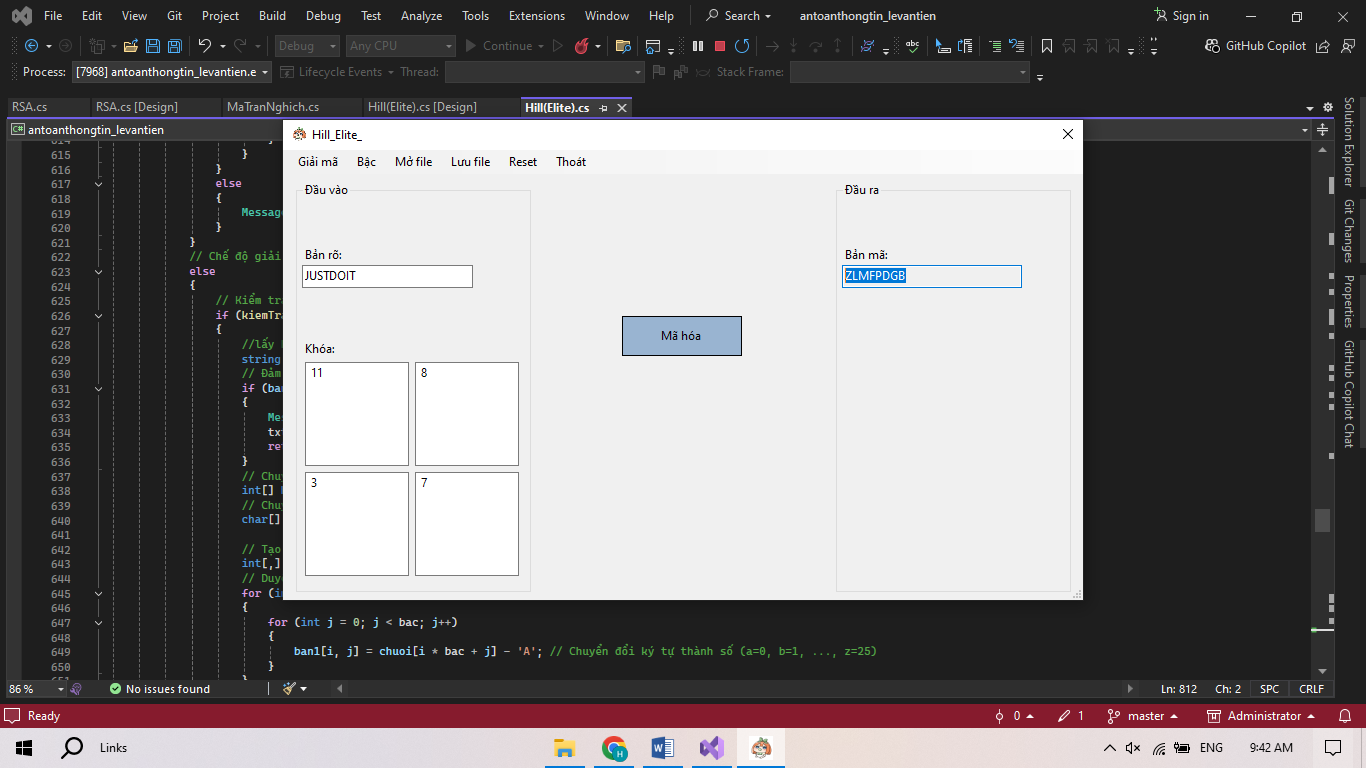
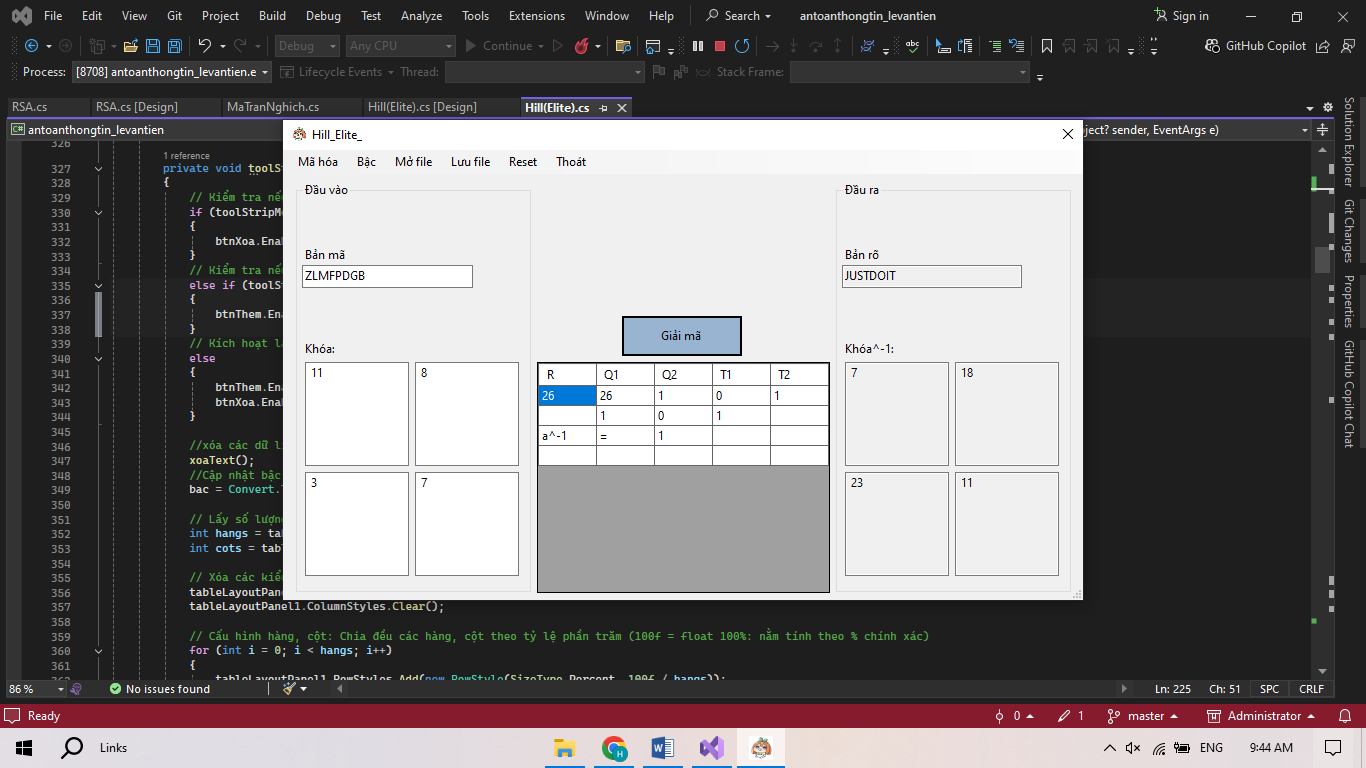
* **TinhDinhThuc(int[,] mt, int n):** Phương thức này tính định thức của một ma trận vuông kích thước n x n. Nếu ma trận có kích thước 1x1, định thức là phần tử duy nhất của ma trận, được tính theo modulo 26. Với ma trận lớn hơn, phương thức sử dụng công thức Laplace để tính định thức bằng cách cộng các giá trị của phần tử đầu dòng nhân với định thức của các ma trận con (lấy các phần tử khác ngoài dòng và cột chứa phần tử đó). Định thức cuối cùng được điều chỉnh để đảm bảo nằm trong phạm vi từ 0 đến 25.
* **maTranCon(int[,] mt, int hang, int cot, int n):** Phương thức này tính toán ma trận con của ma trận gốc bằng cách loại bỏ một hàng và một cột cho trước. Sau đó, phương thức tính định thức của ma trận con này và áp dụng dấu (-1)^(i+j) để có kết quả chính xác.



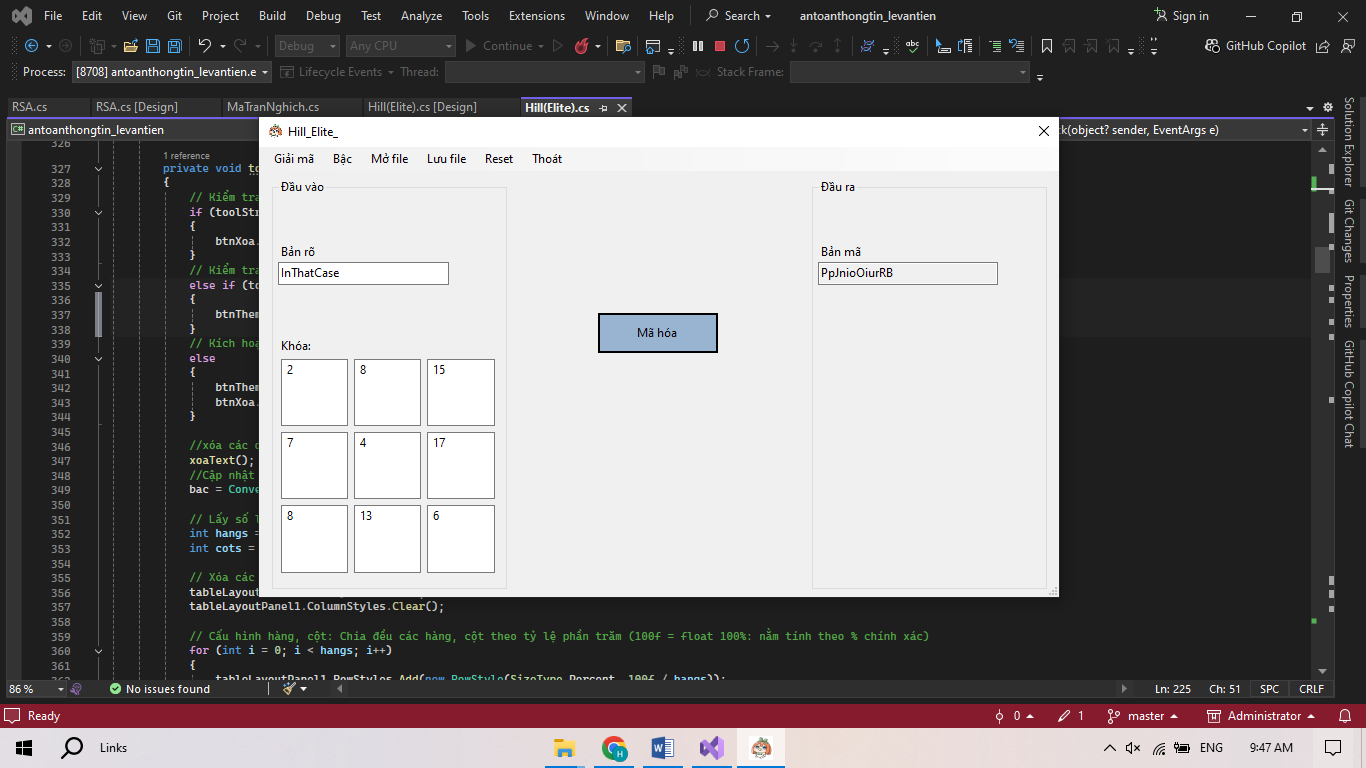
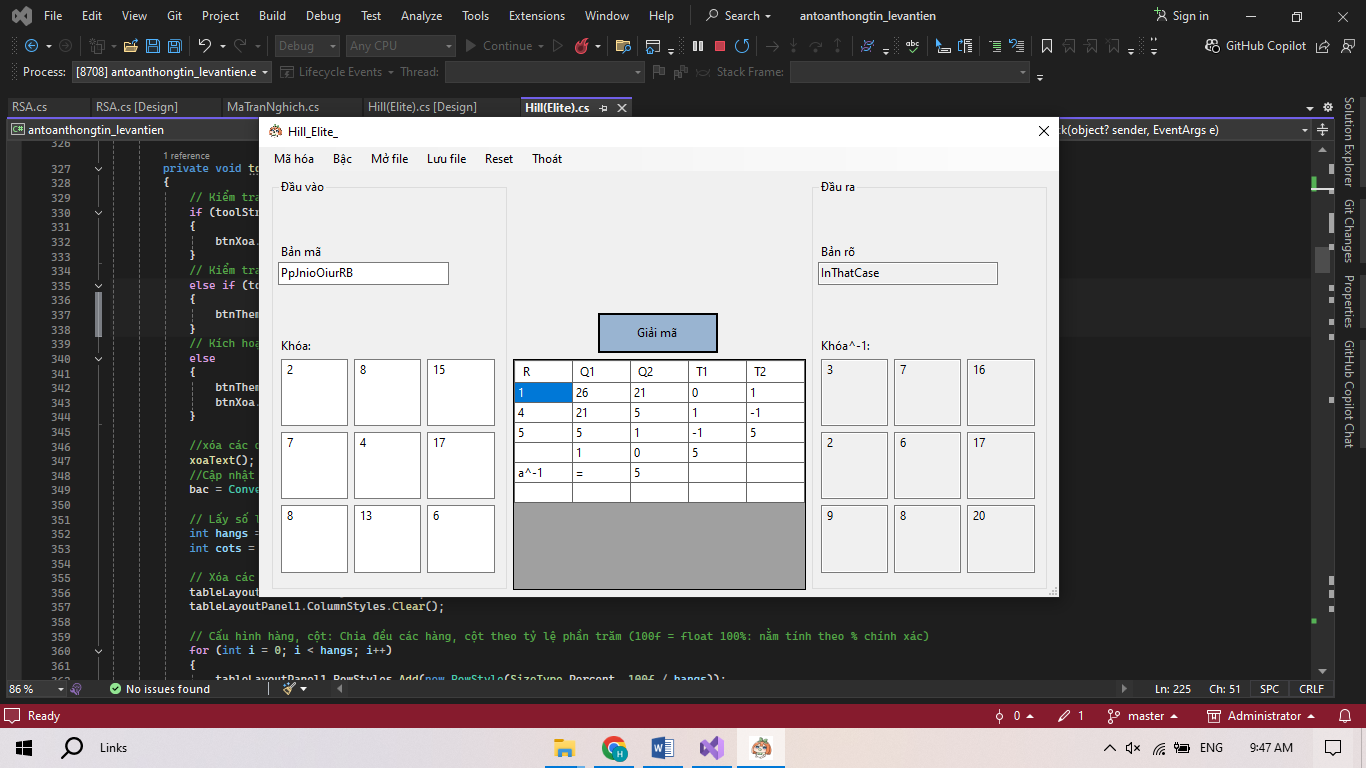
* **MaTranNghichDao(int[,] mt, int det\_1, int n)**: Phương thức này tính ma trận nghịch đảo của ma trận gốc. Đầu tiên, nó tạo ra ma trận phụ (cofactor matrix) của ma trận gốc bằng cách tính các định thức của các ma trận con. Sau đó, ma trận phụ được chuyển vị (transpose) và nhân với phần tử nghịch đảo của định thức modulo 26. Các giá trị trong ma trận nghịch đảo cuối cùng được điều chỉnh để đảm bảo trong phạm vi [0, 25].

### **4. Chạy thử trên phần mềm Winform C#**

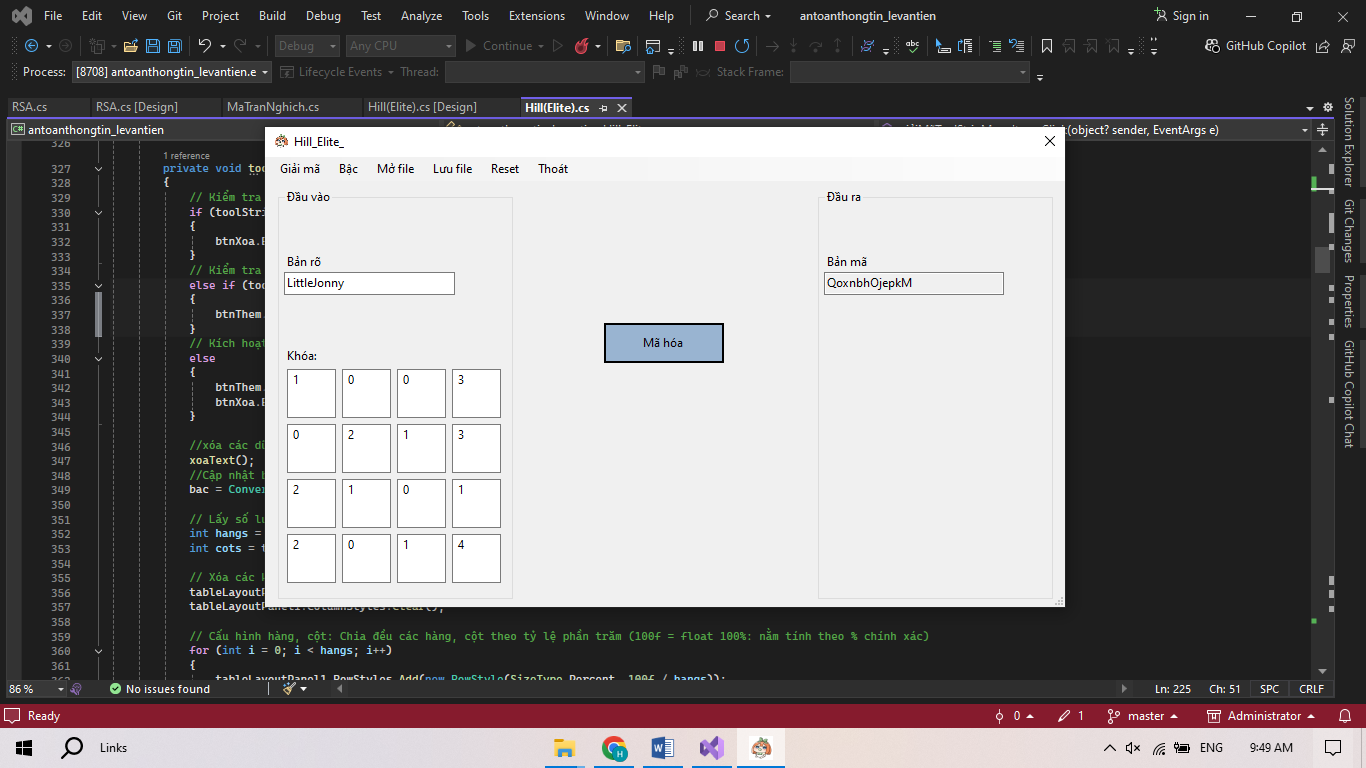
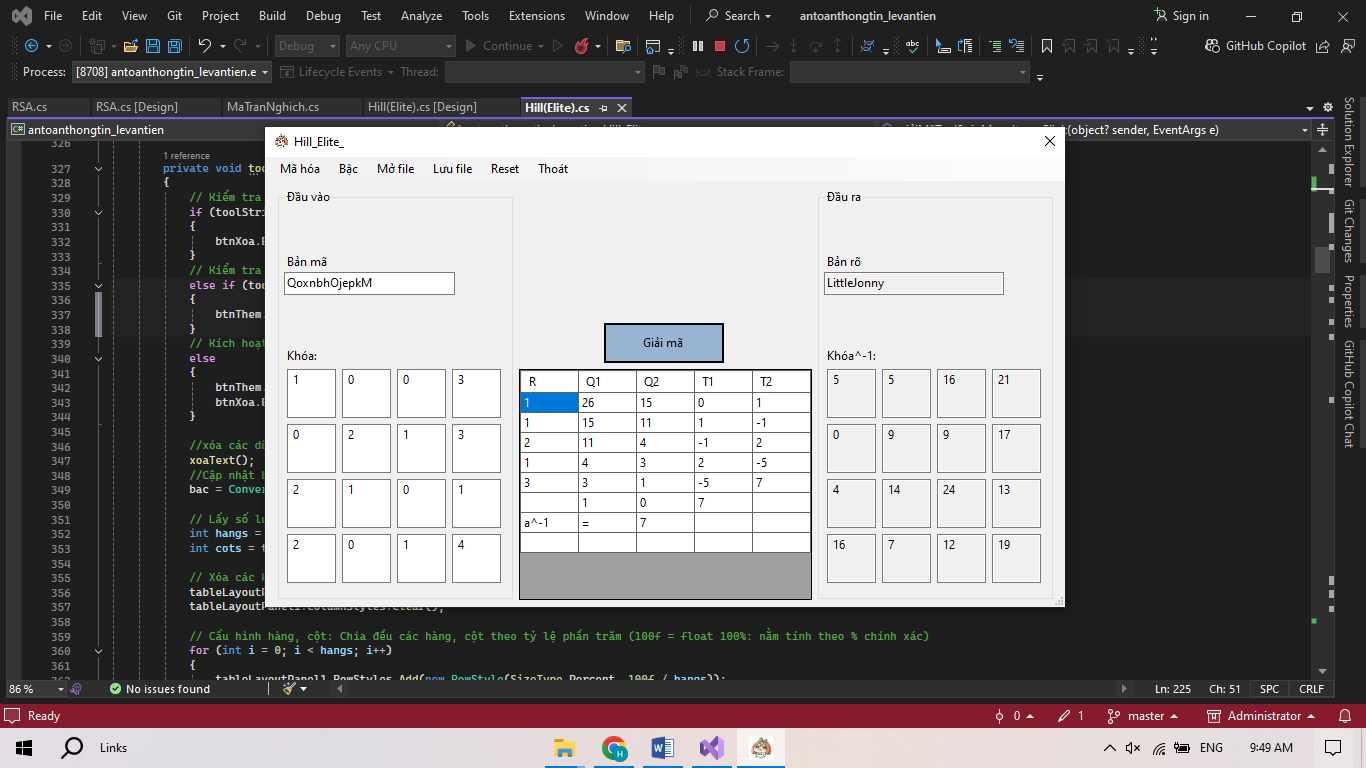
VD: Mã hóa thông điệp “JUSTDOIT” với khóa k = {11,8,3,7}

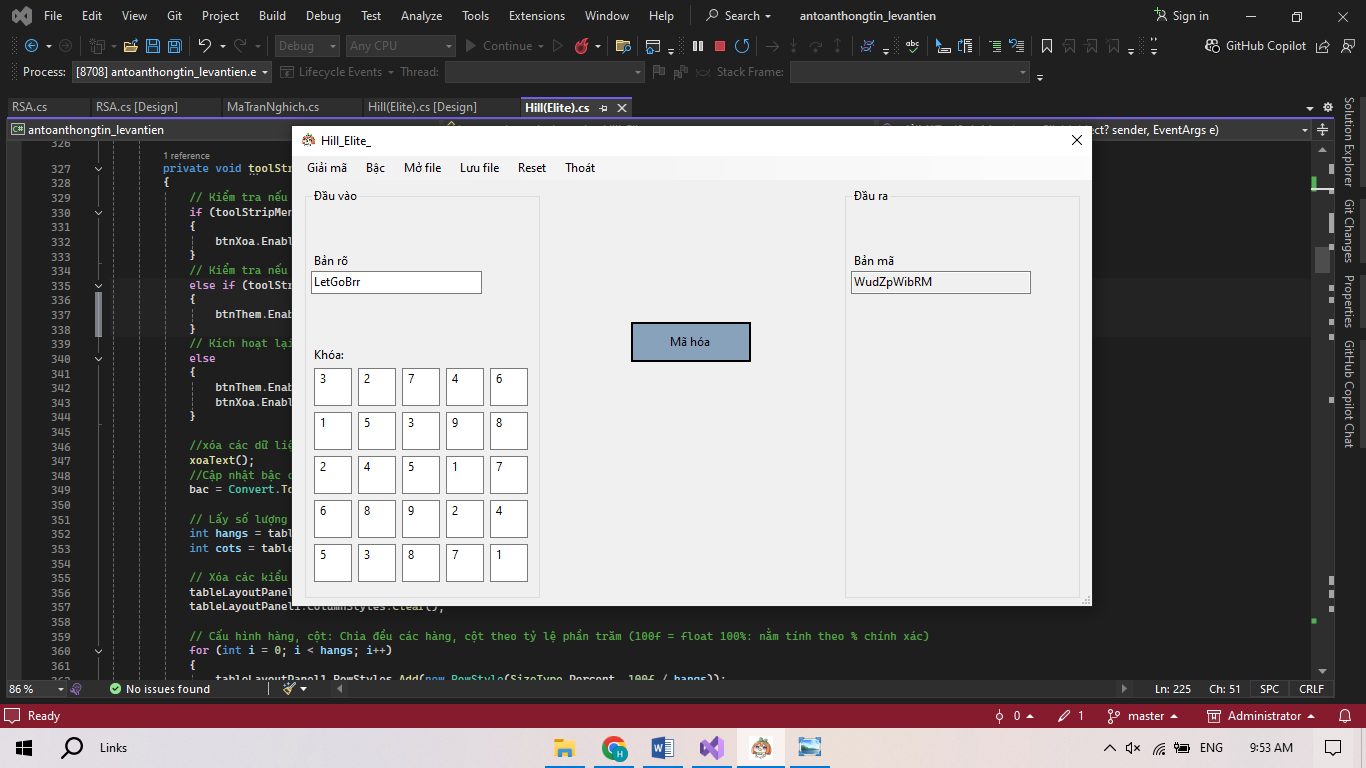
VD: Mã hóa thông điệp “InThatCase” với khóa k = {2,8,15,7,4,17,8,13,6}

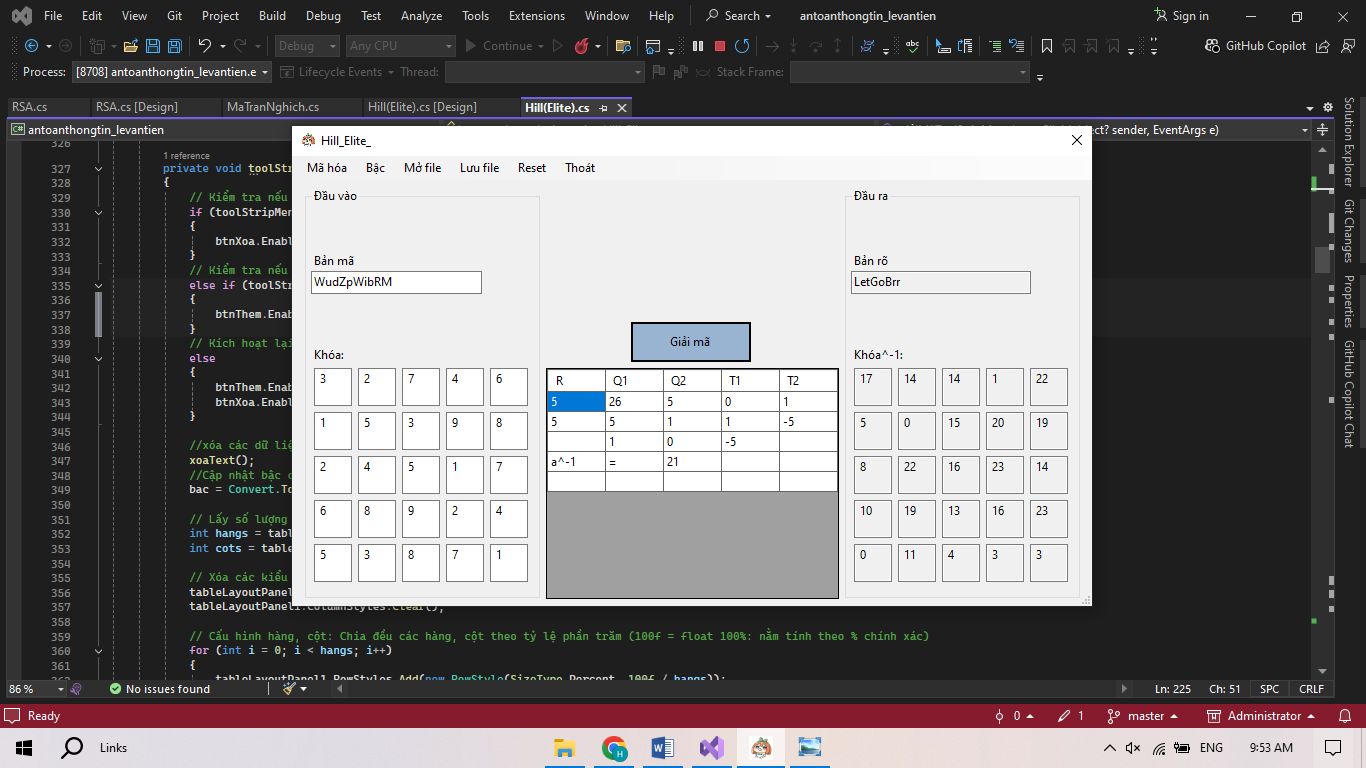
 

VD: Mã hóa thông điệp “LittleJonny” với khóa k = {1,0,0,3,0,2,1,3,2,1,0,1,2,0,1,4}

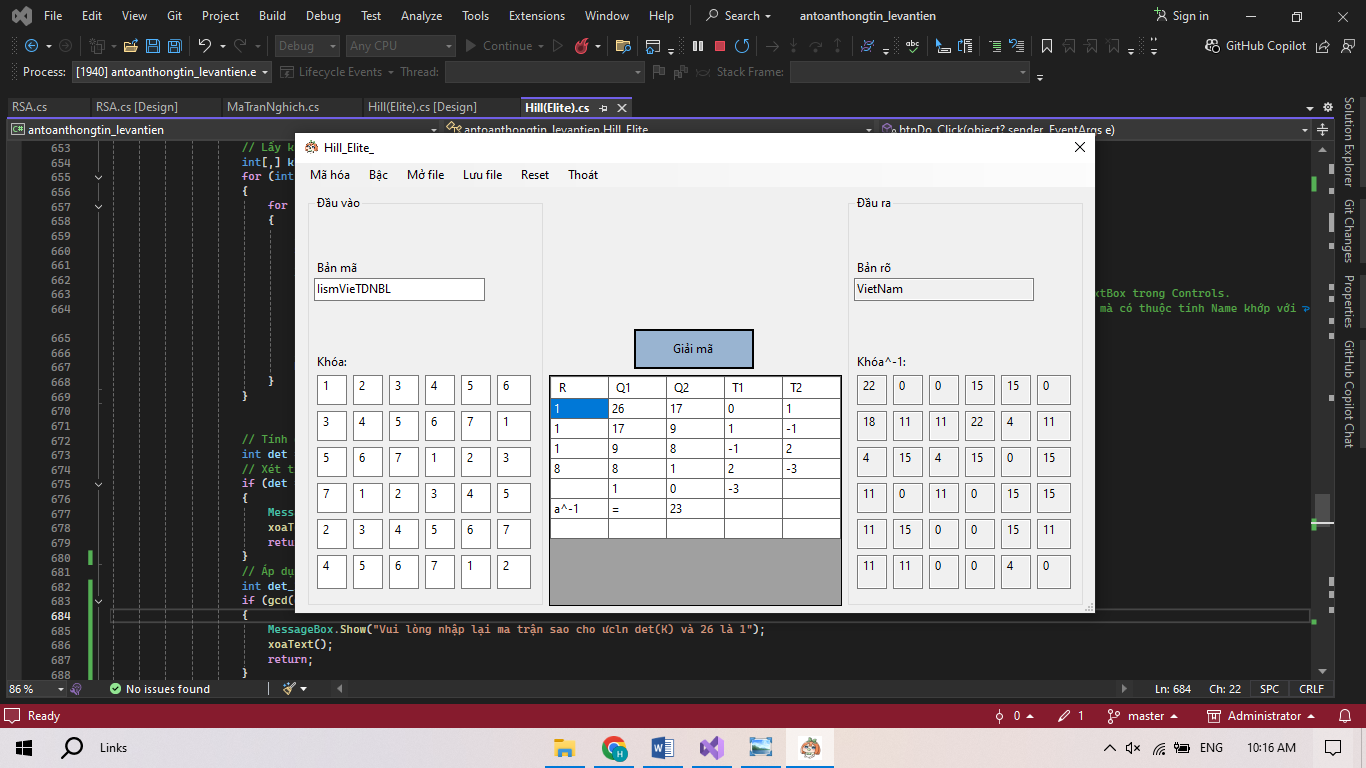
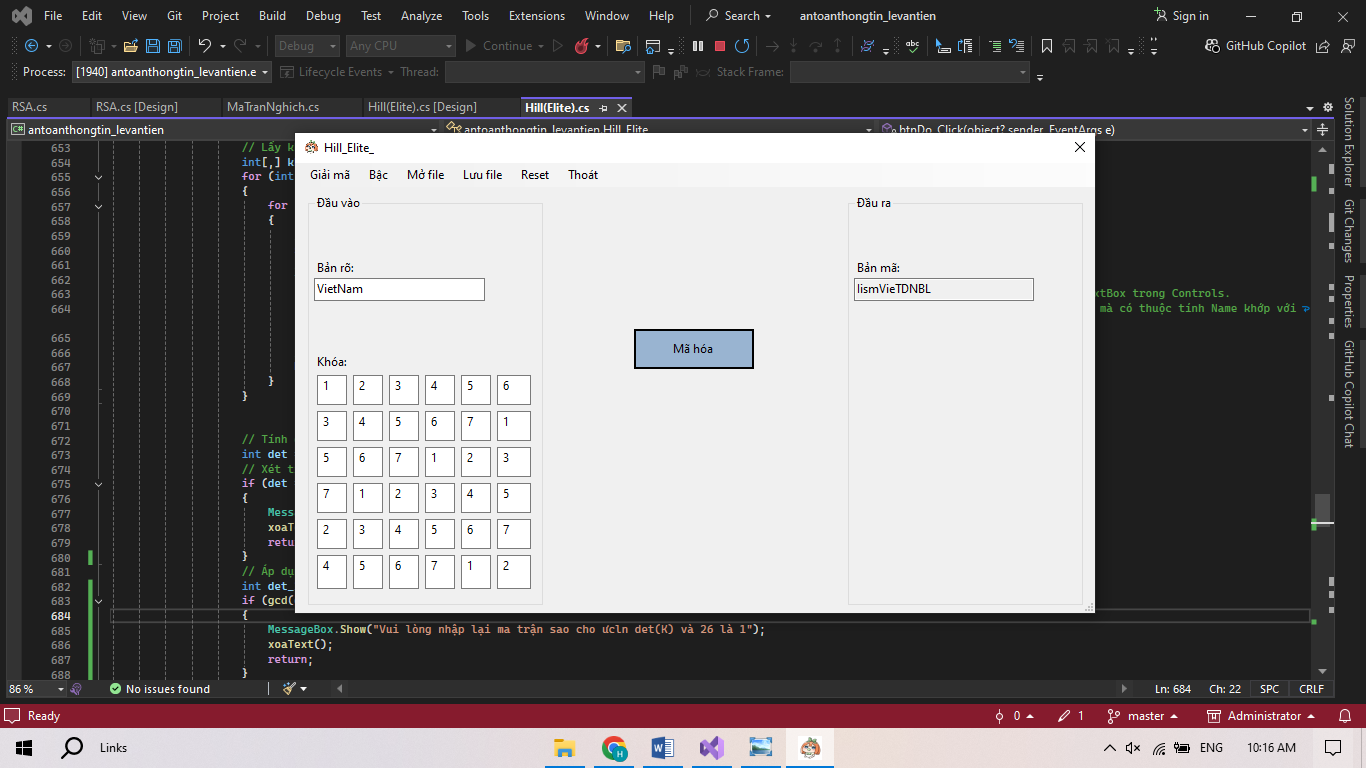
 

VD: Mã hóa thông điệp “LetGoBrr” với khóa k = {3,2,7,4,6,1,5,3,9,8,2,4,5,1,7,6,8,9,2,4,5,3,8,7,1}

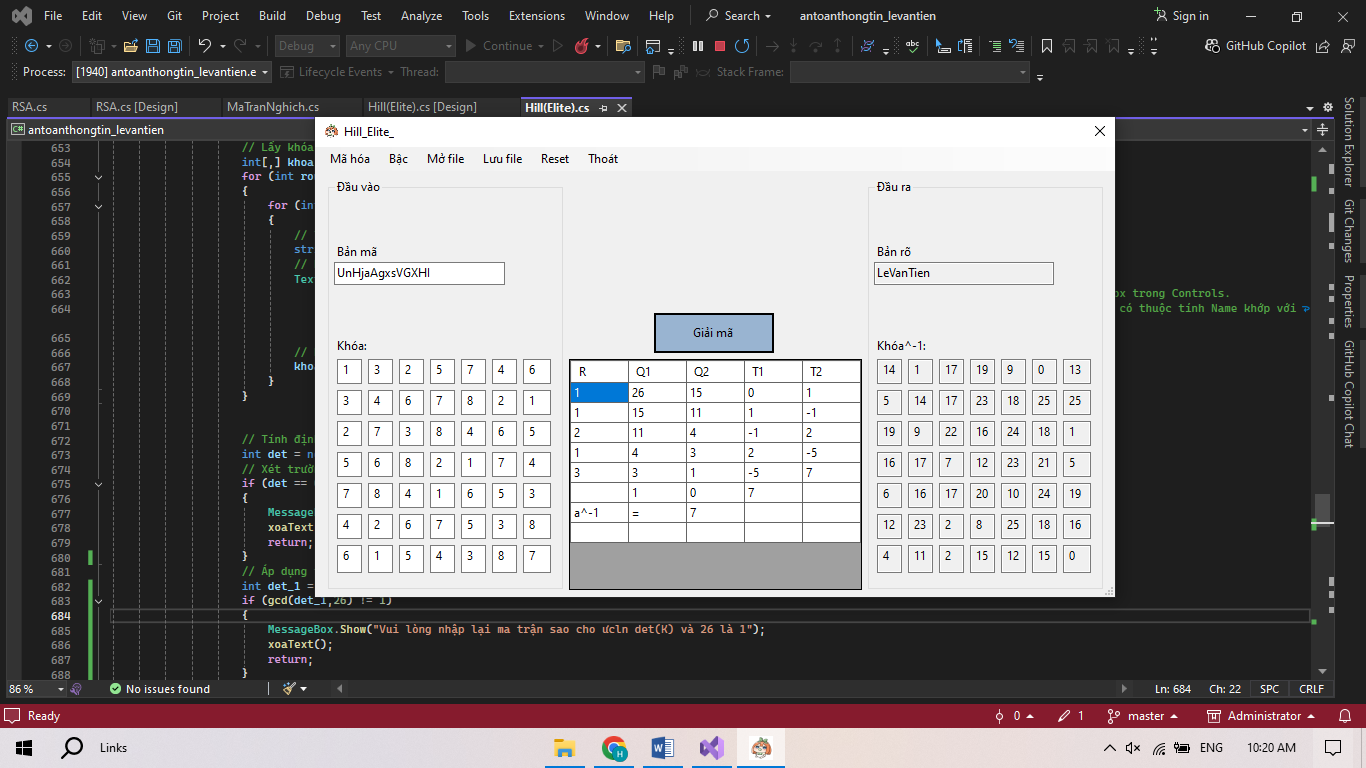
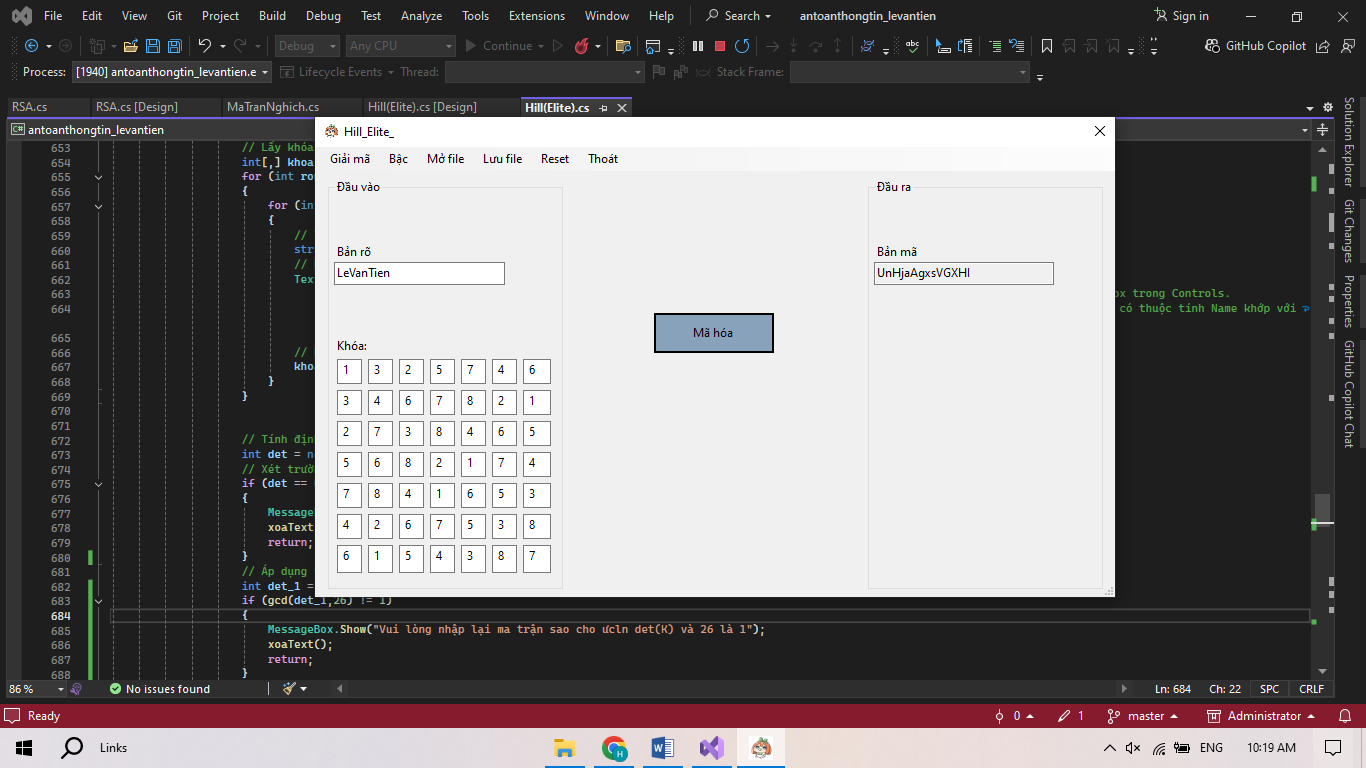




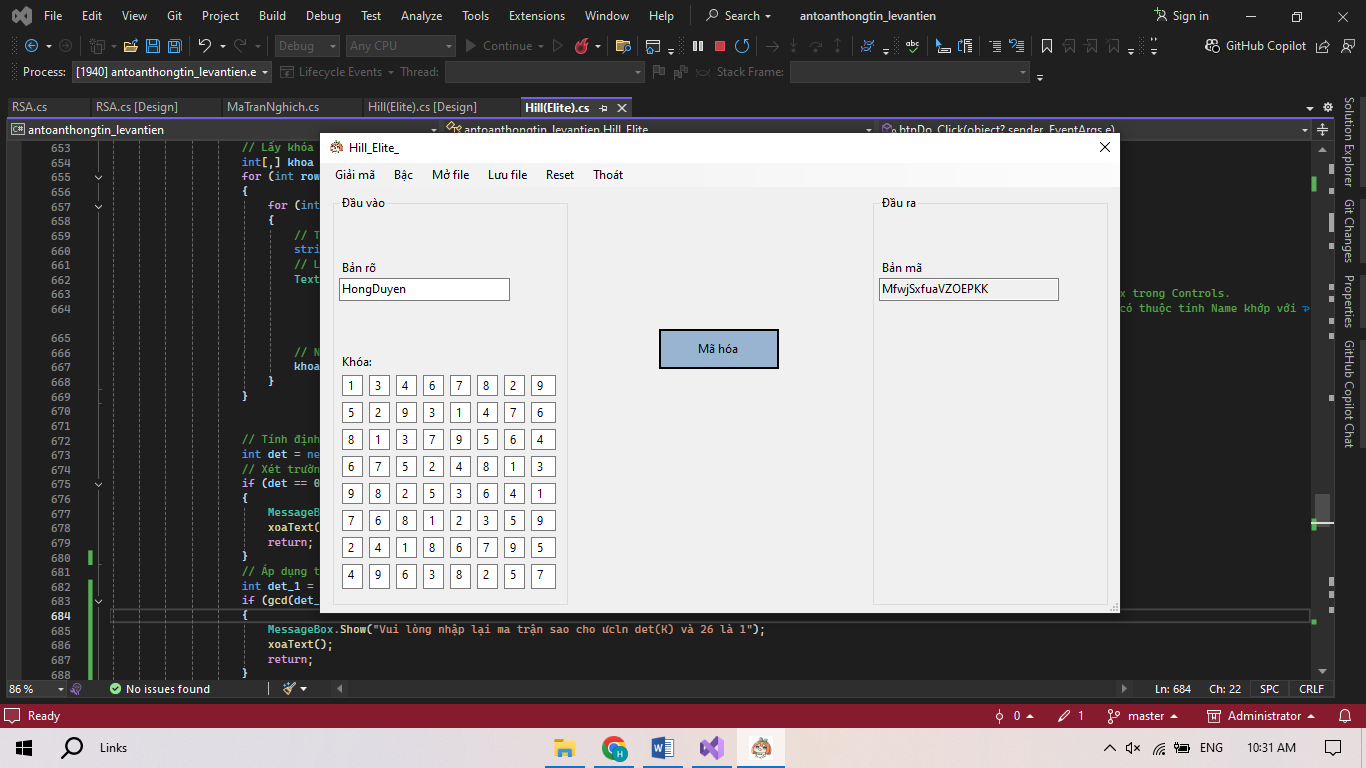
VD: Mã hóa thông điệp “VietNam” với khóa k = {1,2,3,4,5,6,3,4,5,6,7,1,5,6,7,1,2,3,7,1,2,3,4,5,2,3,4,5,6,7,4,5,6,7,1,2}



VD: Mã hóa thông điệp “LeVanTien” với khóa k = {1,3,2,5,7,4,6,3,4,6,7,8,2,1,2,7,3,8,4,6,5,5,6,8,2,1,7,4,7,8,4,1,6,5,3,4,2,6,7,5,3,8,6,1,5,4,3,8,7}



VD: Mã hóa thông điệp “HongDuyen” với khóa k = {1,3,4,6,7,8,2,9,5,2,9,3,1,4,7,6,8,1,3,7,9,5,6,4,6,7,5,2,4,8,1,3,9,8,2,5,3,6,4,1,7,6,8,1,2,3,5,9,2,4,1,8,6,7,9,5,4,9,6,3,8,2,5,7}



# **CHƯƠNG 2: HỆ MÃ HÓA RSA**

## **I. Làm quen với hệ mã hóa RSA**

### **1. Một số khái niệm liên quan**

* Mật mã khóa công khai (Asymmetric Encryption): Là loại mật mã sử dụng một cặp khóa, bao gồm khóa công khai (public key) dùng để mã hóa và khóa riêng (private key) dùng để giải mã. Hệ mật mã RSA là một ví dụ điển hình của mật mã khóa công khai.
* Khóa công khai: Là khóa được công bố và dùng để mã hóa thông tin. Mọi người có thể sử dụng khóa công khai này để gửi thông tin mã hóa đến chủ sở hữu của khóa riêng.
* Khóa riêng: Là khóa bí mật chỉ được biết bởi người sở hữu và dùng để giải mã thông tin đã được mã hóa bằng khóa công khai.
* Mã hóa với khóa công khai: Là phương pháp mà thông tin được mã hóa bằng khóa công khai của người nhận, và chỉ có thể giải mã bằng khóa riêng tương ứng.

### **2. Cơ sở toán học của RSA**

Hệ mật mã RSA dựa trên các khái niệm trong lý thuyết số, đặc biệt là phép toán về số nguyên và tính chất của các số nguyên tố. Các bước cơ bản của RSA gồm:

* **Chọn hai số nguyên tố lớn:** **p** và **q**.
* **Tính giá trị n = p × q:** Đây là một phần của khóa công khai và khóa riêng.
* **Tính hàm Euler φ(n)=(p−1)×(q−1):** Đây là một giá trị quan trọng trong việc chọn khóa công khai.
* **Chọn một số e sao cho 1 < e < φ(n) và gcd( e , φ(n) )=1:** Số **e** này sẽ là phần của khóa công khai.
* **Tính số d sao cho e × d ≡ 1 ( mod φ(n) ):**  Số **d** này là khóa riêng.

Khóa công khai sẽ là cặp (n,e) và khóa riêng là (n,d).

### **3. Các bước mã hóa và giải mã thông tin**

* **Mã hóa:**

1. Chuyển đổi văn bản rõ thành các số nguyên m sao cho 0 ≤ m < n.
2. Mã hóa mỗi số m bằng công thức c =  (mod n), trong đó c là số mã hóa.
3. Kết quả mã hóa là chuỗi các số c.

* **Giải mã:**

1. Sử dụng khóa riêng d và công thức giải mã m =  (mod n) để giải mã mỗi số c và thu lại văn bản rõ m.
2. Ánh xạ các số m trở lại ký tự để thu được văn bản gốc.

### **4. Ví dụ**

Giả sử chọn hai số nguyên tố p = 61 và q = 53, ta tính được:

* N = p × q = 61 × 53 = 3233.
* φ(n) = (p−1) × (q−1) = 60×52 = 3120.
* Chọn e = 17 (một số nguyên tố với φ(n)).
* Tính d = 2753 sao cho e × d ≡ 1 (mod 3120).

**Khóa công khai:** (e=17, n=3233)

**Khóa riêng:** (d=2753, n=3233)

**Mã hóa văn bản "HI":**

* Chuyển đổi "H" thành số 7 và "I" thành số 8.
* Mã hóa số 7: c =  (mod 3233) = 2197
* Mã hóa số 8: c =  (mod 3233) = 1928

Kết quả mã hóa là "2197 1928".

**Giải mã:**

* Giải mã số 2197: m =  (mod 3233) = 7 (tương ứng với "H").
* Giải mã số 1928: m =  (mod 3233) = 8 (tương ứng với "I").

Kết quả giải mã là "HI".

### **5.Ưu nhược điểm**

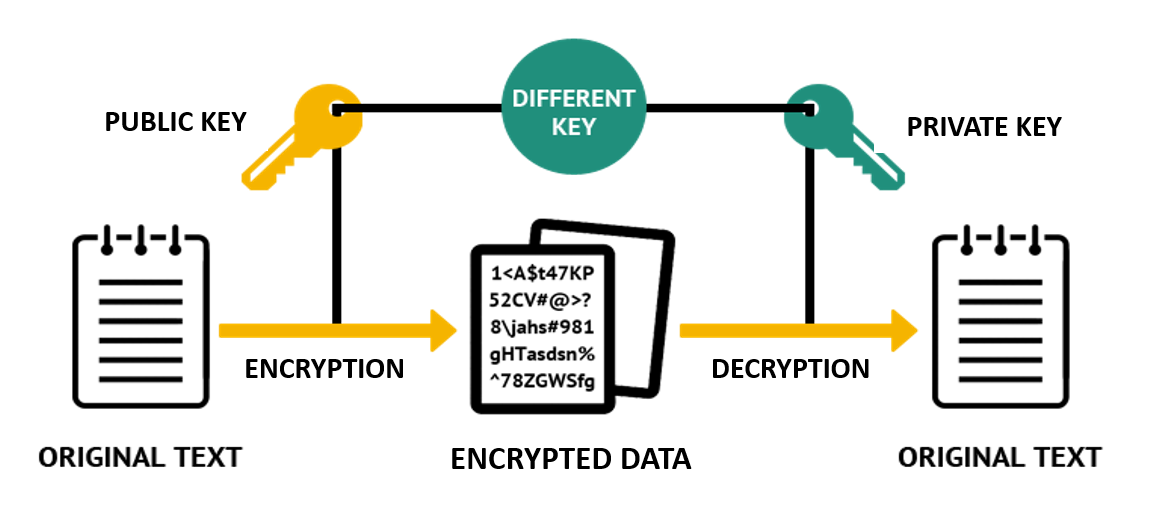
* **Ưu điểm:**
* An toàn trong việc bảo mật vì không cần chia sẻ khóa riêng.
* Dễ dàng thực hiện và có thể sử dụng cho cả việc ký số và mã hóa thông tin.
* **Nhược điểm:**
* Quá trình mã hóa và giải mã chậm hơn so với hệ mật mã đối xứng.
* Kích thước khóa thường lớn và yêu cầu tính toán phức tạp.

### **6.Ứng dụng**

* Hệ mật mã RSA được sử dụng rộng rãi trong bảo mật Internet, ví dụ như trong giao thức SSL/TLS để bảo vệ truyền tải dữ liệu qua mạng.
* Ngoài ra, RSA còn được ứng dụng trong các chữ ký số và xác thực điện tử.

## **II. Triển Khai RSA bằng C# WinForm**

Việc triển khai Hill cipher trên C# WinForm sẽ giúp tạo ra một ứng dụng dễ sử dụng, cho phép người dùng nhập khóa và văn bản, sau đó thực hiện mã hóa và giải mã một cách trực quan. Quy trình tổng quan được thực hiện như sau:



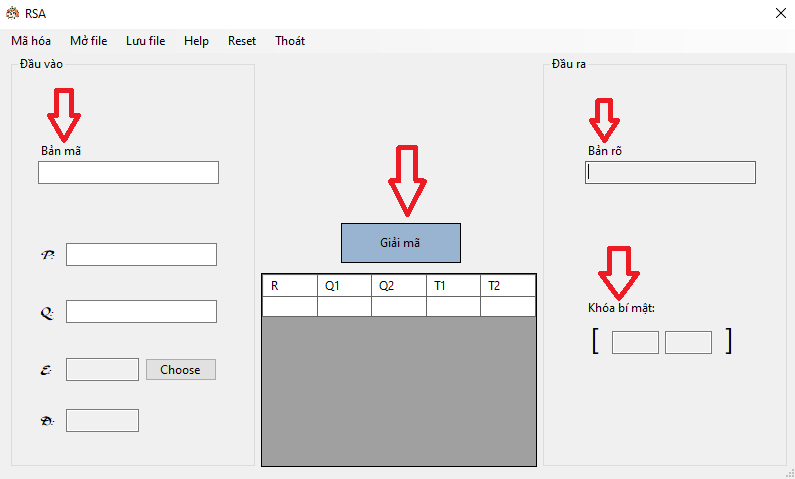
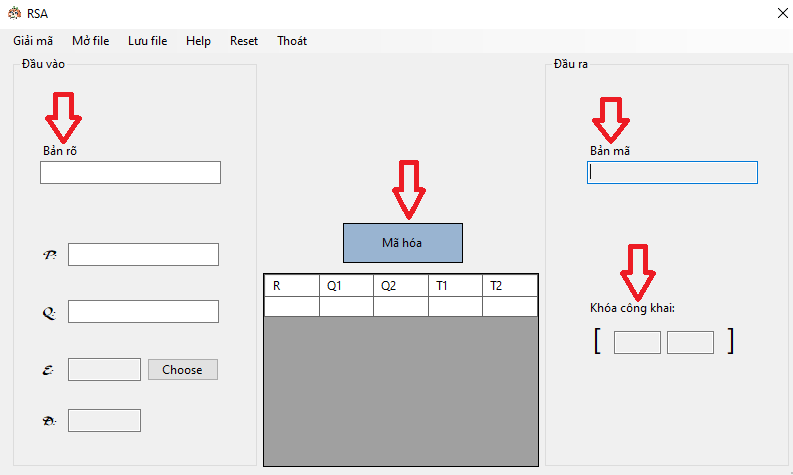
### **1. Thiết kế Giao diện Người dùng**

#### **1.1. Giao diện thiết kế cho hệ mật mã Hill cipher:**

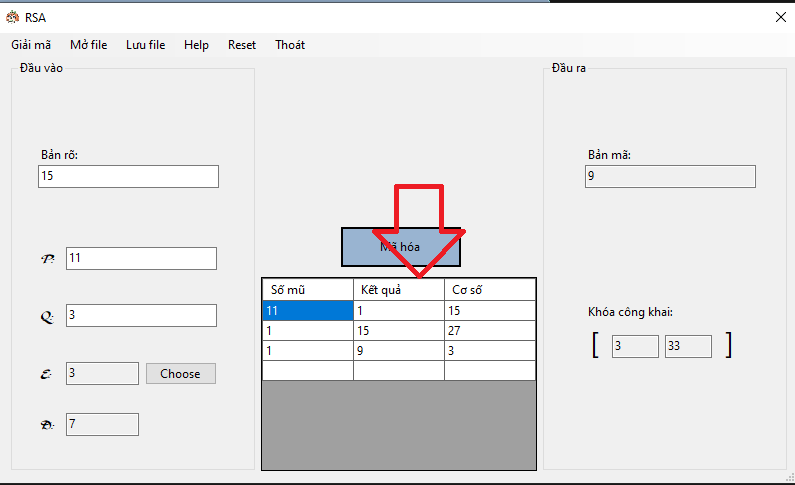
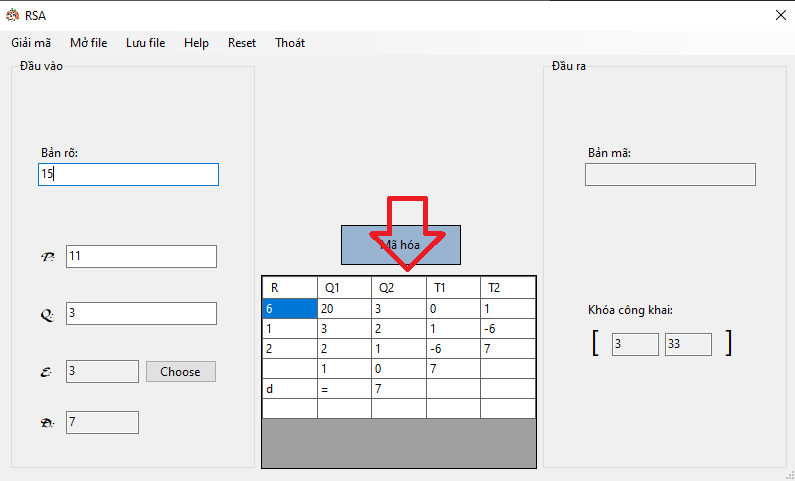


* MenuStrip: Tổ chức chức năng theo danh mục
* GroupBox: Chứa đầu vào, đầu ra
* Button: Thực hiện mã hóa, giải mã
* DataGridView: Hiển thị quy trình tìm D và Quy trình tính toán

#### **1.2. Giao diện Giải mã/Mã hóa: Chọn chế độ hoạt động của chương trình, chuyển đổi giữa chế độ mã hóa và giải mã**

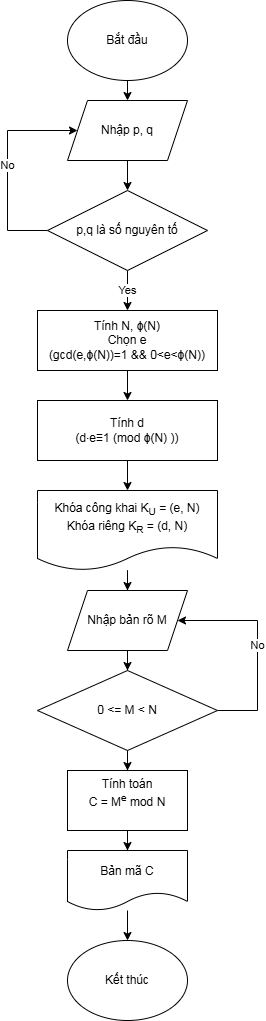
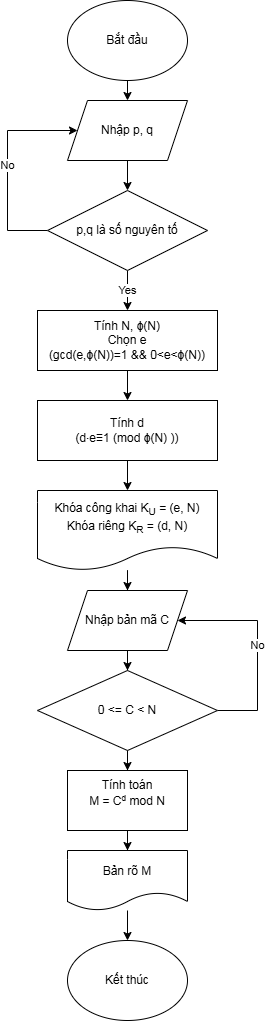


#### **1.3. Giao diện hiển thị tìm D và Quy trình thực hiện tính toán khi mã hóa và giải mã:**



### **2. Sơ đồ khối của thuật toán:**

#### **2.1. Mã hóa & giải mã**

1. **Bắt đầu**: Đây là điểm khởi đầu của quy trình mã hóa RSA.
2. **Nhập p, q**: Hai số nguyên tố **p** và **q** được chọn. Đây là các số nguyên tố bí mật được dùng để tạo khóa công khai và khóa riêng tư.
3. **Kiểm tra p, q là số nguyên tố**: Bước này kiểm tra xem **p** và **q** có phải là số nguyên tố không. Nếu không, quay lại để nhập lại **p** và **q**.
4. **Tính N và φ(N)**:
   * Tính **N = p × q**, đây sẽ là phần của khóa công khai và khóa riêng.
   * Tính **ϕ(N) = (p−1) × (q−1)**, là hàm số Euler của **N**.
5. **Chọn e**:
   * Chọn một số nguyên **e** sao cho **gcd( e, ϕ(N)) = 1** và **0 < e < ϕ(N).**
   * **e** là số nguyên tố cùng nhau với **ϕ(N)** và sẽ là thành phần của khóa công khai.
6. **Tính d**: Tìm số nguyên **d** sao cho **d × e ≡ 1 (mod ϕ(N))**. Số **d** này là nghịch đảo của **e** theo mod **ϕ(N)** và sẽ là thành phần của khóa riêng.
7. **Khóa công khai và khóa riêng**:
   * Khóa công khai KU=(e,N) được dùng để mã hóa.
   * Khóa riêng KR=(d,N) được dùng để giải mã.

Đối với mã hóa:

1. **Nhập bản rõ M**: Nhập thông điệp **M** cần mã hóa. **M** phải là một số trong khoảng 0 ≤ M < N. Nếu không thỏa mãn, cần nhập lại giá trị **M**.
2. **Tính toán C**: Mã hóa bản rõ **M** thành bản mã **C** bằng công thức C =  mod N.
3. **Bản mã C**: Bước này hiển thị bản mã **C** sau khi mã hóa.
4. **Kết thúc**: Hoàn thành quá trình mã hóa với bản mã **C** đã tạo ra.

Quy trình này bao gồm tất cả các bước chính để thiết lập và mã hóa một thông điệp bằng hệ mã RSA, từ việc chọn khóa đến quá trình mã hóa.

Đối với giải mã:

h. **Nhận bản mã C**: Bắt đầu với bản mã **C** đã được tạo ra từ quá trình mã hóa. Bản mã **C** này được truyền cho người có khóa riêng KR.

i. **Tính toán bản rõ M**:

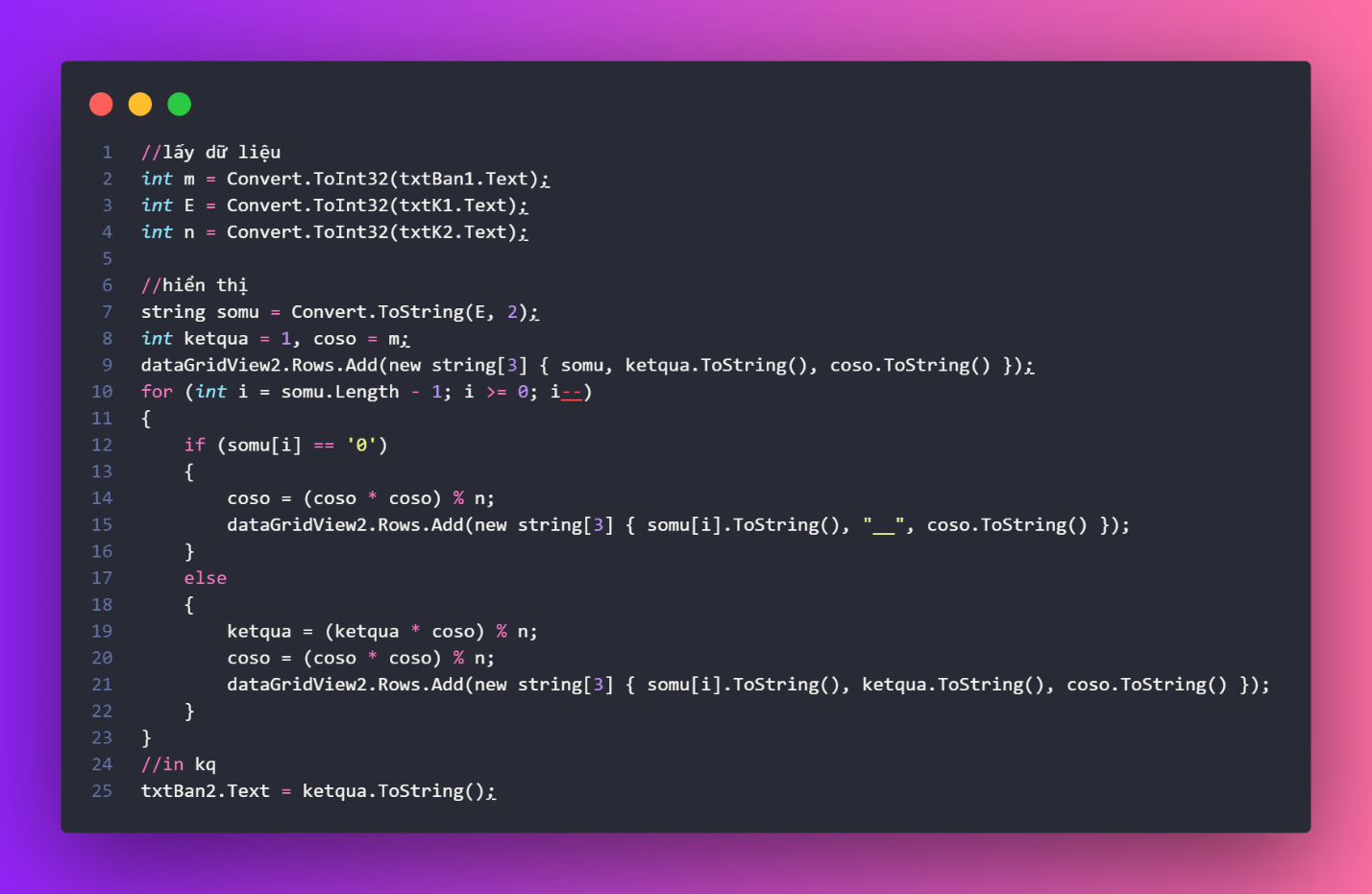
* Dùng khóa riêng **d** và **N** để tính lại bản rõ **M** từ bản mã **C** bằng công thức: M =  mod N.
* Kết quả **M** là thông điệp ban đầu trước khi mã hóa.

j. **Hoàn thành**: Sau khi giải mã, ta thu được bản rõ **M**, tức là thông điệp gốc mà chỉ người giữ khóa riêng ddd mới có thể giải mã được.

Quy trình này bao gồm tất cả các bước chính để thiết lập và giải hóa một thông điệp bằng hệ mã RSA, từ việc chọn khóa đến quá trình giải hóa.

### **3. Quy trình chính của code:**

#### **3.1. Nhân ma trận:**

**a.** **Lấy dữ liệu đầu vào**:

* Người dùng nhập ba số vào các ô đầu vào txtBan1, txtK1, và txtK2. Các giá trị này lần lượt được chuyển đổi sang số nguyên và lưu vào các biến m, E, và n, trong đó:
  + m là cơ số (số cần lũy thừa),
  + E là số mũ,
  + n là modulus (số chia để lấy phần dư).

**b.** **Chuyển đổi số mũ E sang dạng nhị phân**:

* Số mũ E được chuyển sang dạng nhị phân và lưu vào chuỗi somu. Việc chuyển đổi này giúp xử lý từng bit của E một cách dễ dàng trong quá trình tính toán.

**c.** **Khởi tạo giá trị ban đầu**:

* Biến ketqua được khởi tạo với giá trị 1, là kết quả cuối cùng sẽ được tính toán dần qua từng bước. Biến coso ban đầu là m, là cơ số cần lũy thừa.
* Hàng đầu tiên của bảng dataGridView2 được điền với giá trị nhị phân của E, ketqua, và coso. Đây là trạng thái ban đầu của phép tính.

**d.** **Xử lý từng bit của số mũ E từ phải sang trái**:

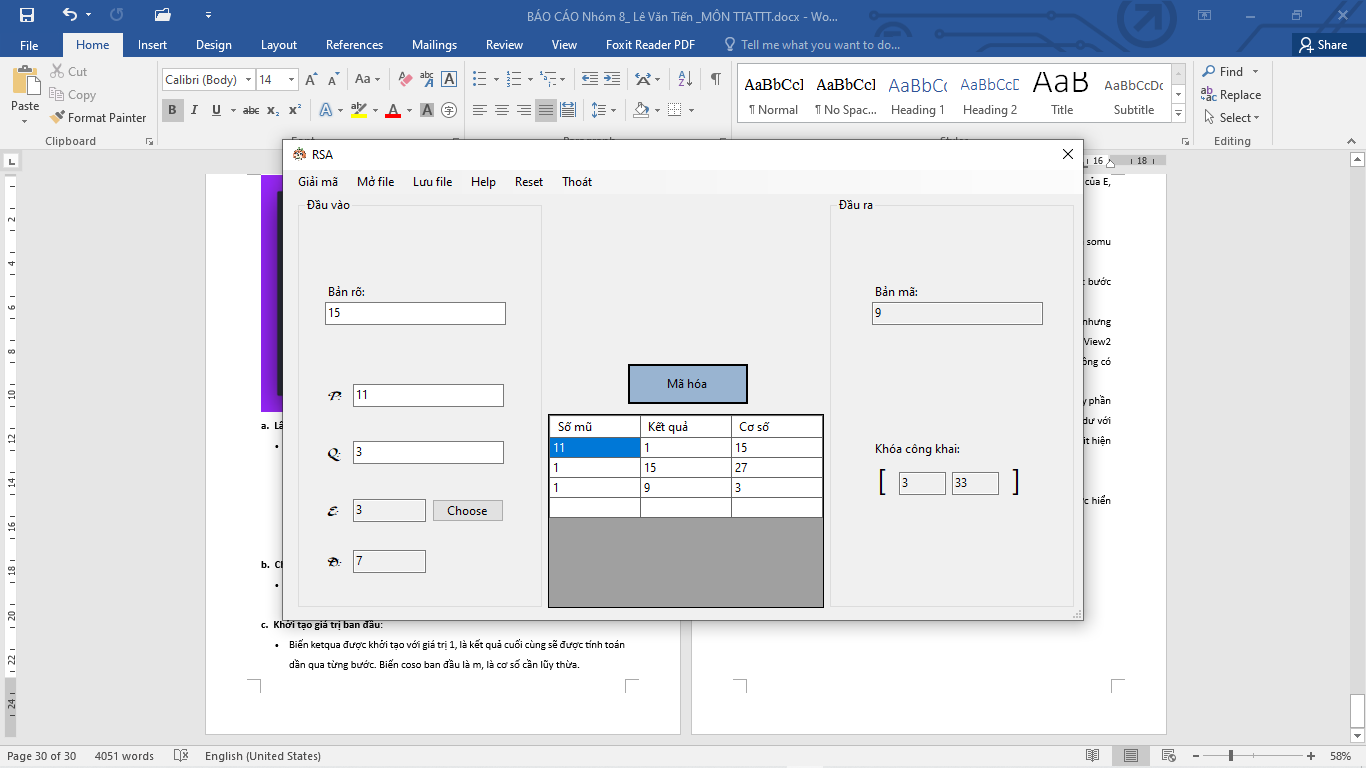
* Một vòng lặp được sử dụng để duyệt qua từng bit của chuỗi nhị phân somu từ phải sang trái (từ bit thấp nhất đến bit cao nhất).
* Tùy theo giá trị của bit hiện tại (0 hoặc 1), chương trình thực hiện các bước tính toán khác nhau:
  + Nếu bit là 0, coso được bình phương và lấy phần dư với n, nhưng ketqua vẫn giữ nguyên. Một hàng mới được thêm vào dataGridView2 để hiển thị bit hiện tại, dấu \_\_ ở cột ketqua để biểu thị rằng không có thay đổi, và giá trị mới của coso.
  + Nếu bit là 1, ketqua được cập nhật bằng cách nhân với coso và lấy phần dư với n. Sau đó, coso tiếp tục được bình phương và lấy phần dư với n. Một hàng mới cũng được thêm vào dataGridView2, hiển thị bit hiện tại, giá trị mới của ketqua, và coso.

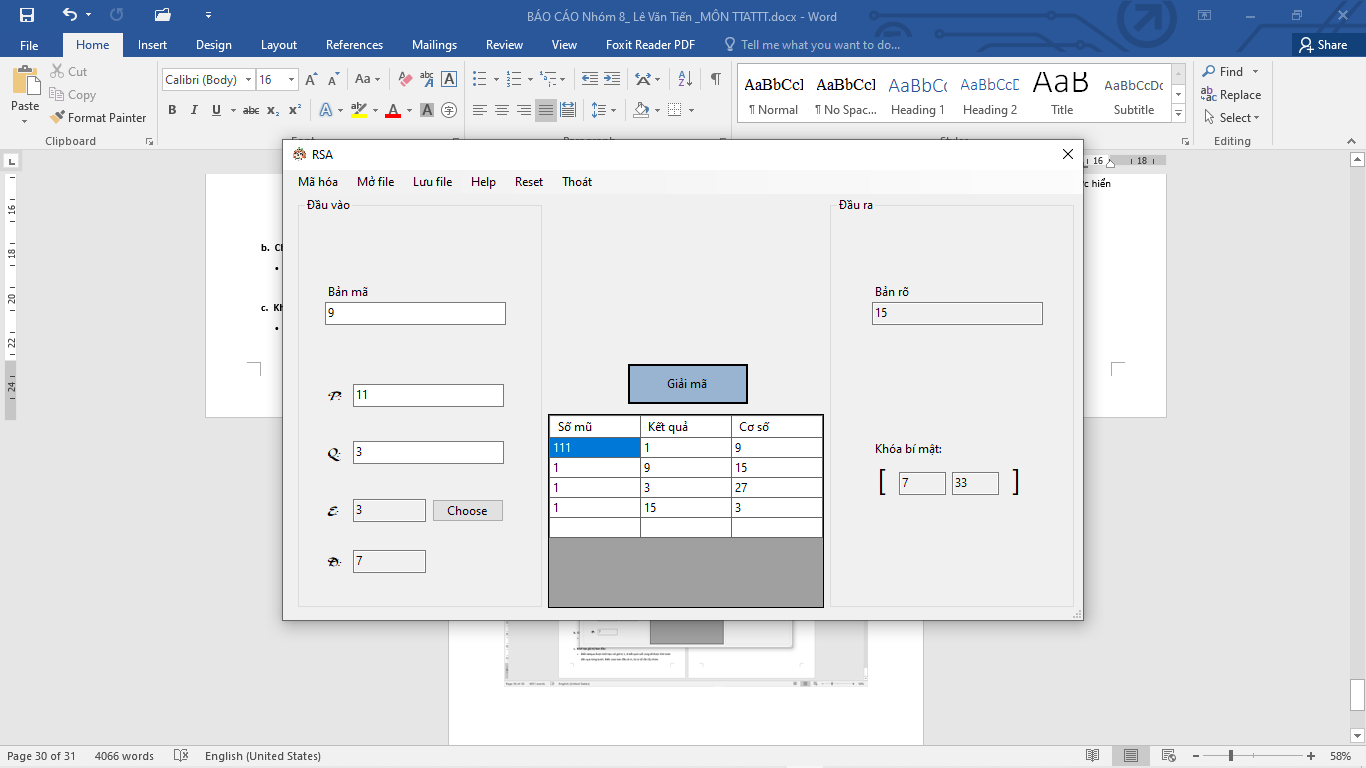
**e.** **Hiển thị kết quả cuối cùng**:

* Sau khi hoàn tất vòng lặp, giá trị cuối cùng là ketqua. Giá trị này được hiển thị trong ô txtBan2.

### **4. Chạy thử trên phần mềm Winform C#**

VD: Mã hóa bản rõ 15 với p = 11, q= 3, e = 3





VD: Mã hóa bản rõ 65 với p = 61, q= 53, e = 17

