**🙢 ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH 🙠**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN TPHCM**

**KHOA: CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN MÔN**

**KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ**

**ĐỀ TÀI: BIỂU DIỄN VÀ TÍNH TOÁN SỐ HỌC TRÊN MÁY TÍNH**

**GVHD:** Thầy Lê Viết Long

**Nhóm: MSSV**

1. Bùi Phạm Phương Khanh 1712067
2. Hứa Mỹ Nghi 1712102
3. Nguyễn Hoàn Tuyết Nhi 1712637
4. Nguyễn Uyển Nhi 1712639

**LỚP: 17CTT4**

**KHÓA: 2017-2021**

I / Phân công và Đánh giá mức độ hoàn thành

# 1/ Mức độ hoàn thành ứng với từng yêu cầu:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Phần** | **Hàm** | **1712067** | **1712102** | **1712637** | **1712639** | **Đánh giá mức độ hoàn thành (%)** |
| **Số Nguyên Lớn** | **Nhập** |  |  | **x** |  | **100%** |
| **Xuất** |  |  | **x** |  | **100%** |
| **Chuyển đổi số QInt thập phân sang nhị phân** |  |  | **x** |  | **100%** |
| **Chuyển đổi số QInt nhị phân sang thập phân** |  |  | **x** |  | **100%** |
| **Chuyển đổi số QInt nhị phân sang thập lục phân** |  |  | **x** |  | **100%** |
| **Chuyển đổi số QInt thập phân sang thập lục phân** |  |  | **x** |  | **100%** |
| **Các operator toán tử** |  |  |  | **x** | **100%** |
| **Các toán tử so sánh và gán** |  |  |  | **x** | **100%** |
| **Các toán tử: &, | , ~** |  |  |  | **x** | **100%** |
| **Các toán tử: <<, >>, rol, ror** |  |  |  | **x** | **95%** |
| **Số Chấm Động Chính Xác Cao** | **Nhập** | **x** | **x** |  |  | **95%** |
| **Xuất** | **x** | **x** |  |  | **85%** |
| **Chuyển đổi số QFloat nhị phân sang thập phân** | **x** | **x** |  |  | **90%** |
| **Chuyển đổi số QFloat thập phân sang nhị phân** | **x** | **x** |  |  | **85%** |
|  | **Đọc tham số dòng lệnh command line** |  |  | **x** |  | **100%** |
| **Viết và chỉnh sửa báo cáo** | **x** | **x** |  |  | **95%** |

# 2/ Mức độ hoàn thành trên toàn bộ project:

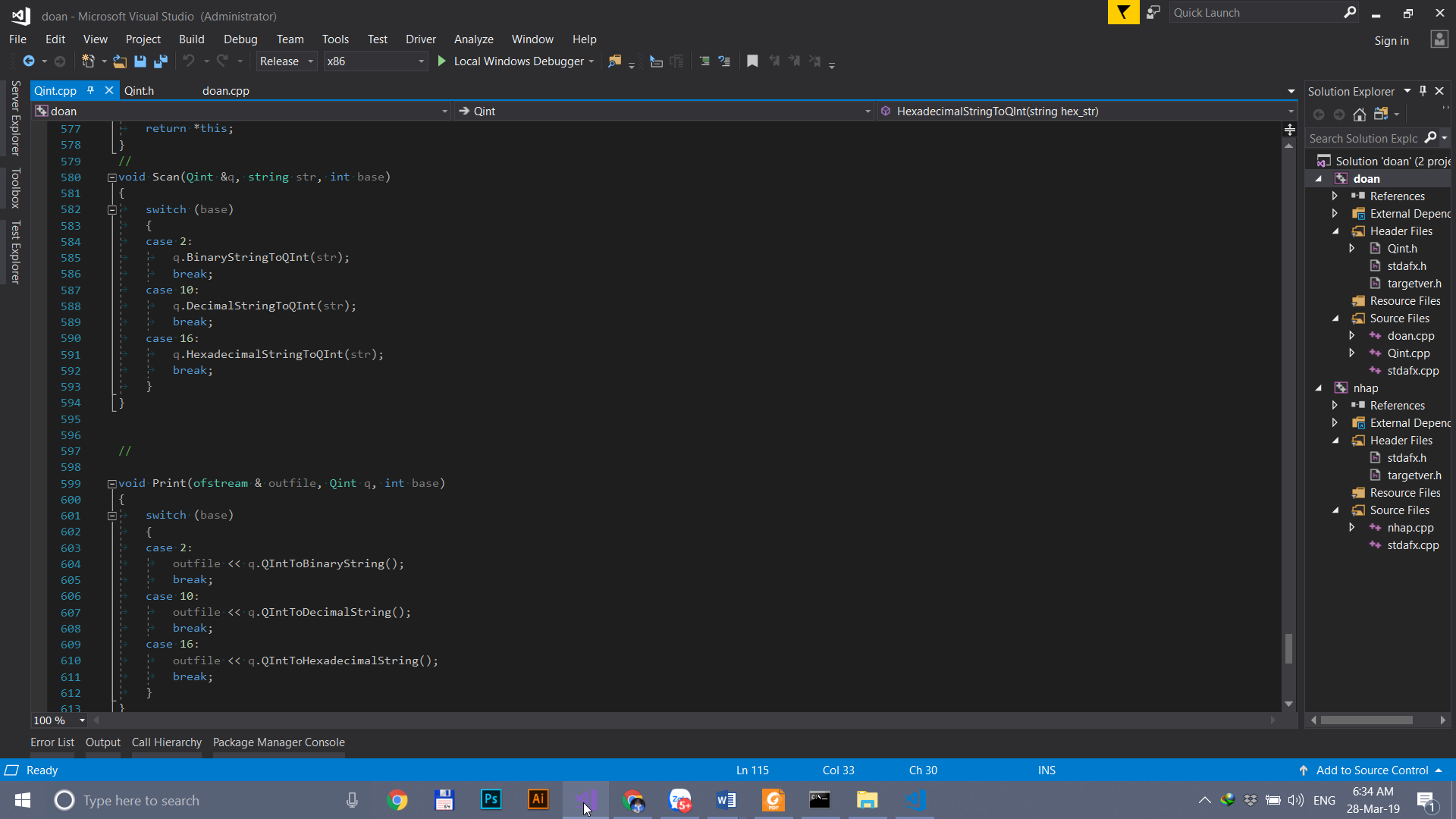
* Phần Số nguyên lớn: hoàn thành gần như 100% so với toàn bộ project.
* Phần Số chấm động chính xác cao: phần chuyển từ Qfloat nhị phân sang thập phân và Qfloat thập phân sang nhị phân có kết quả có thể chưa được làm tròn hoàn chỉnh.  
  Kết quả: hoàn thành gần như 80% so với toàn bộ project.

# 3/ Phạm vi biểu diễn của các kiểu dữ liệu đã thiết kế:

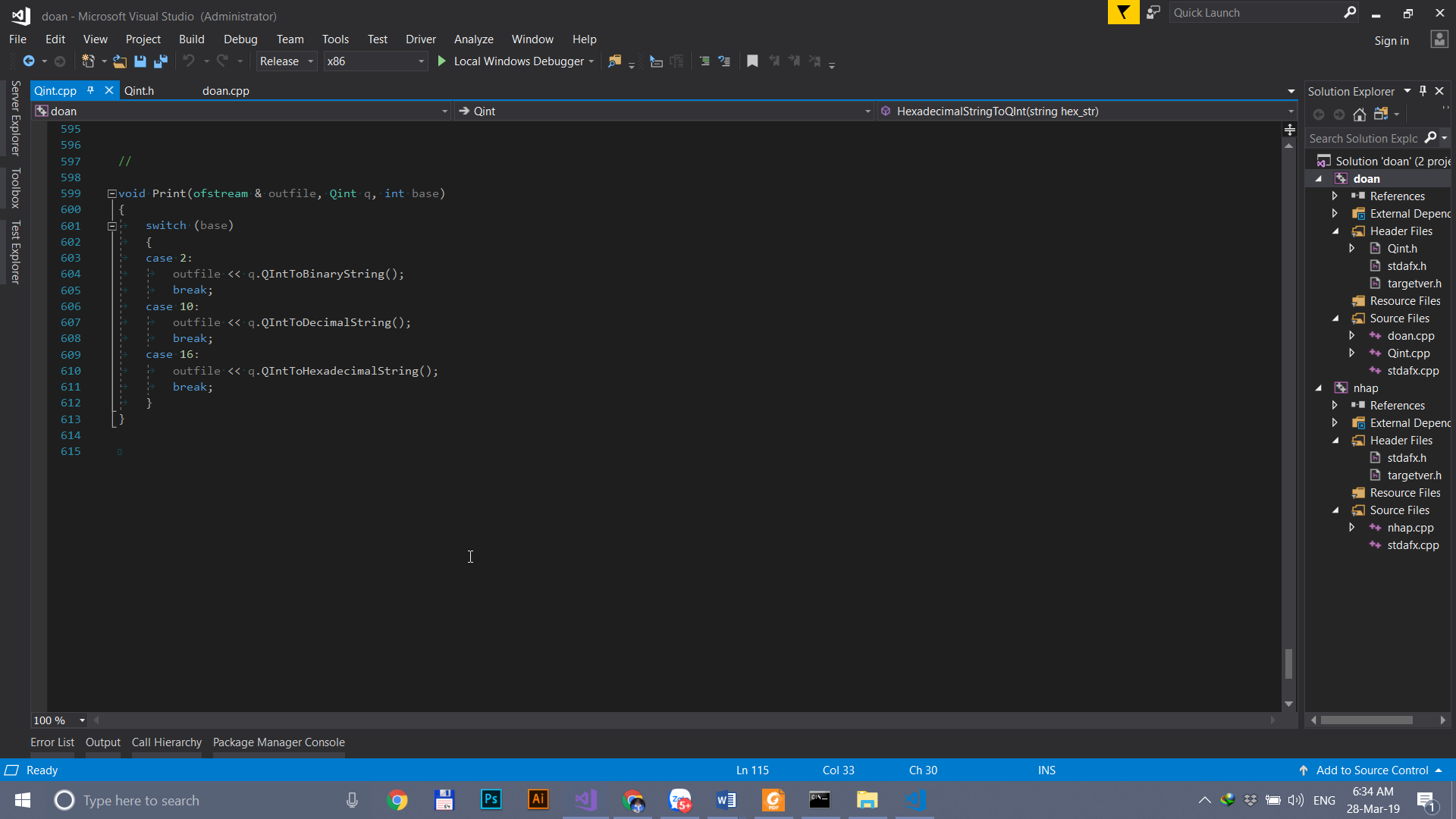
* Phần số nguyên lớn: số nguyên có dấu từ (-2^128)/2 đến [(2^128)/2]-1 (trừ số 0) .
* Phần số chấm động chính xác cao: [-2^16383; -(2-2^-112)\*2^16383] và [(2-2^-112)\*2^16383; 2^16383]

II / Giao diện chương trình ứng với các testcase

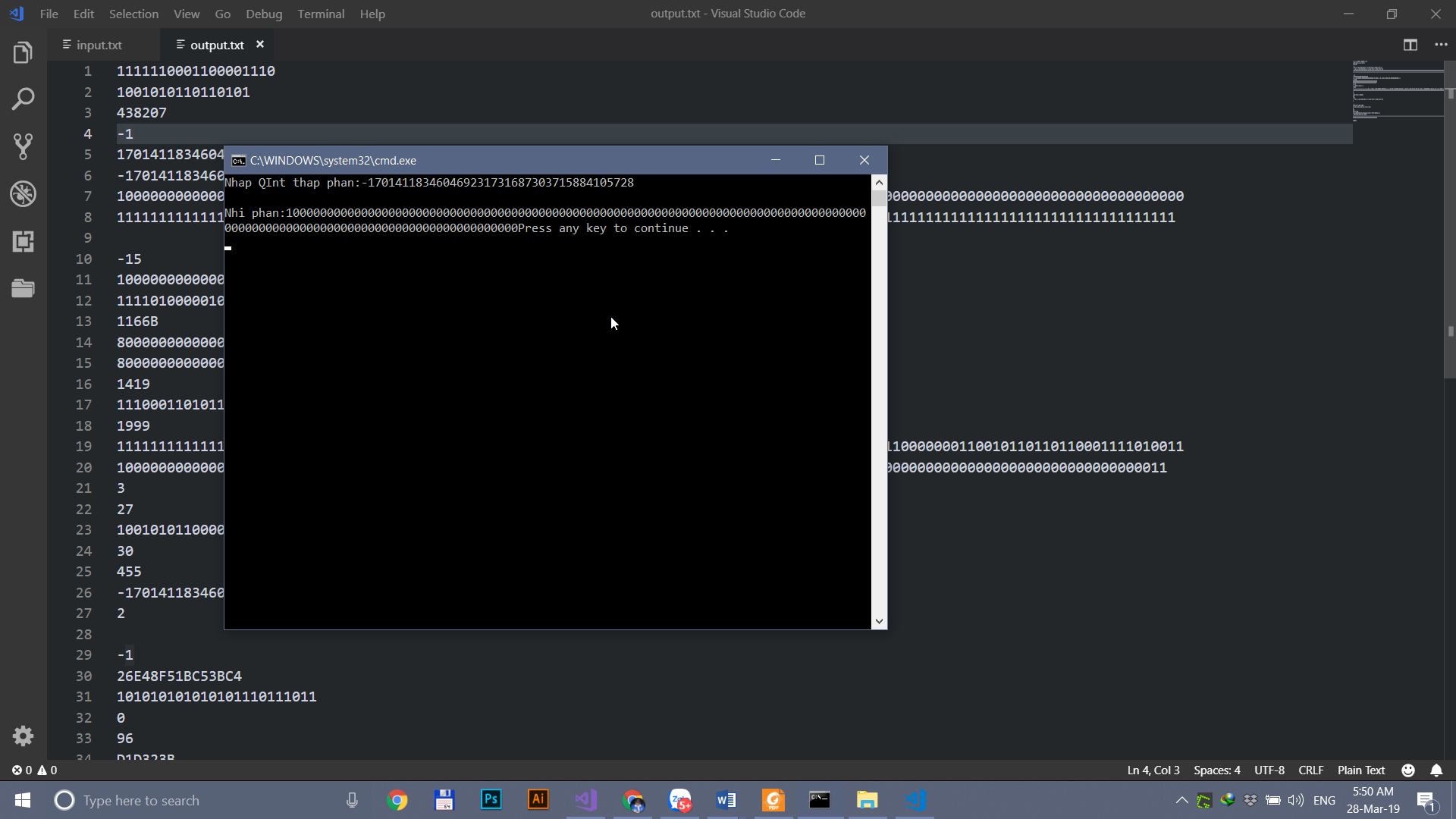
1. **Số nguyên lớn:**
2. **Hàm Nhập:**



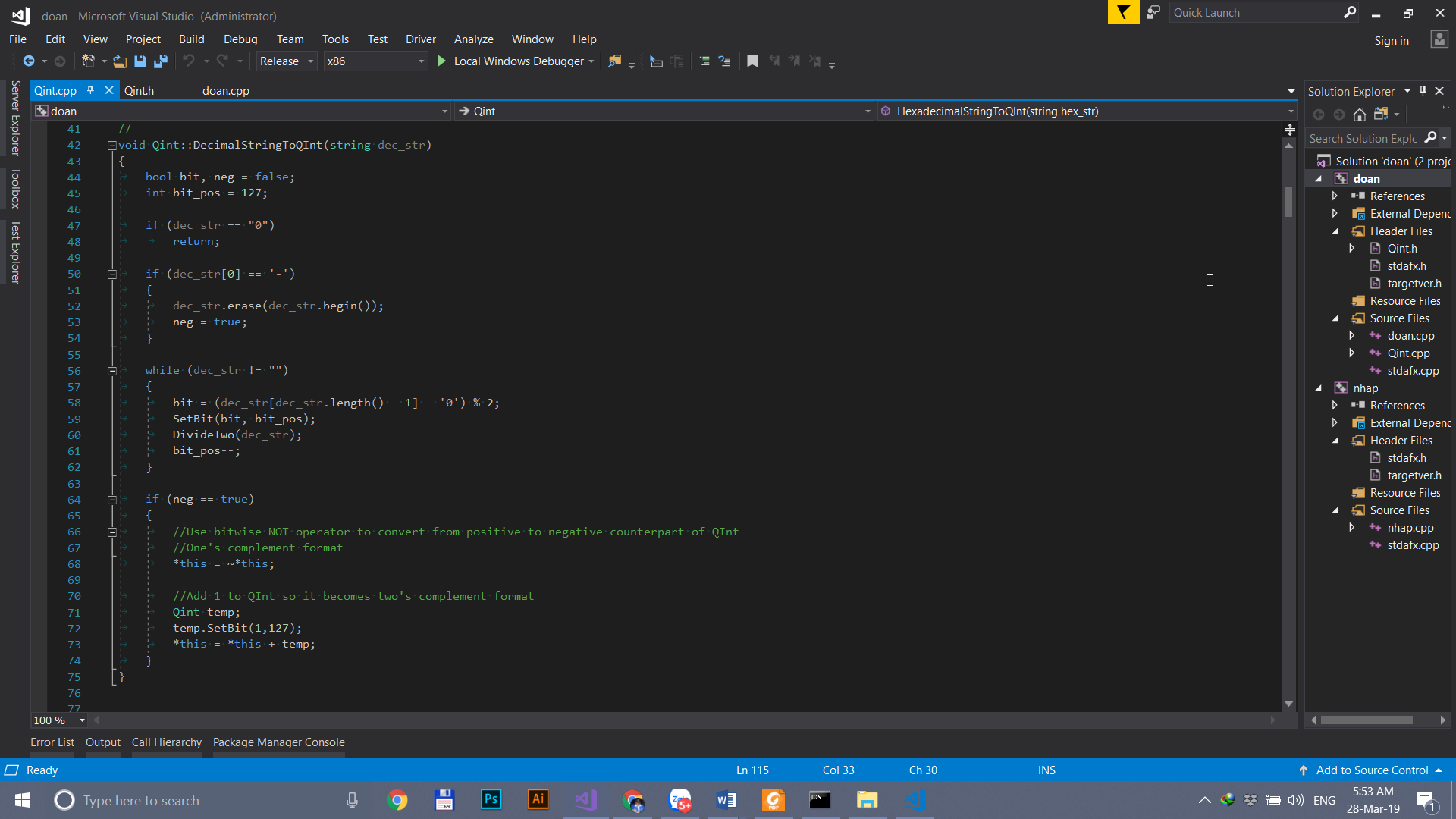
1. **Hàm Xuất:**



1. **Hàm chuyển Qint thập phân sang nhị phân:**

**Test:**

**Code:**



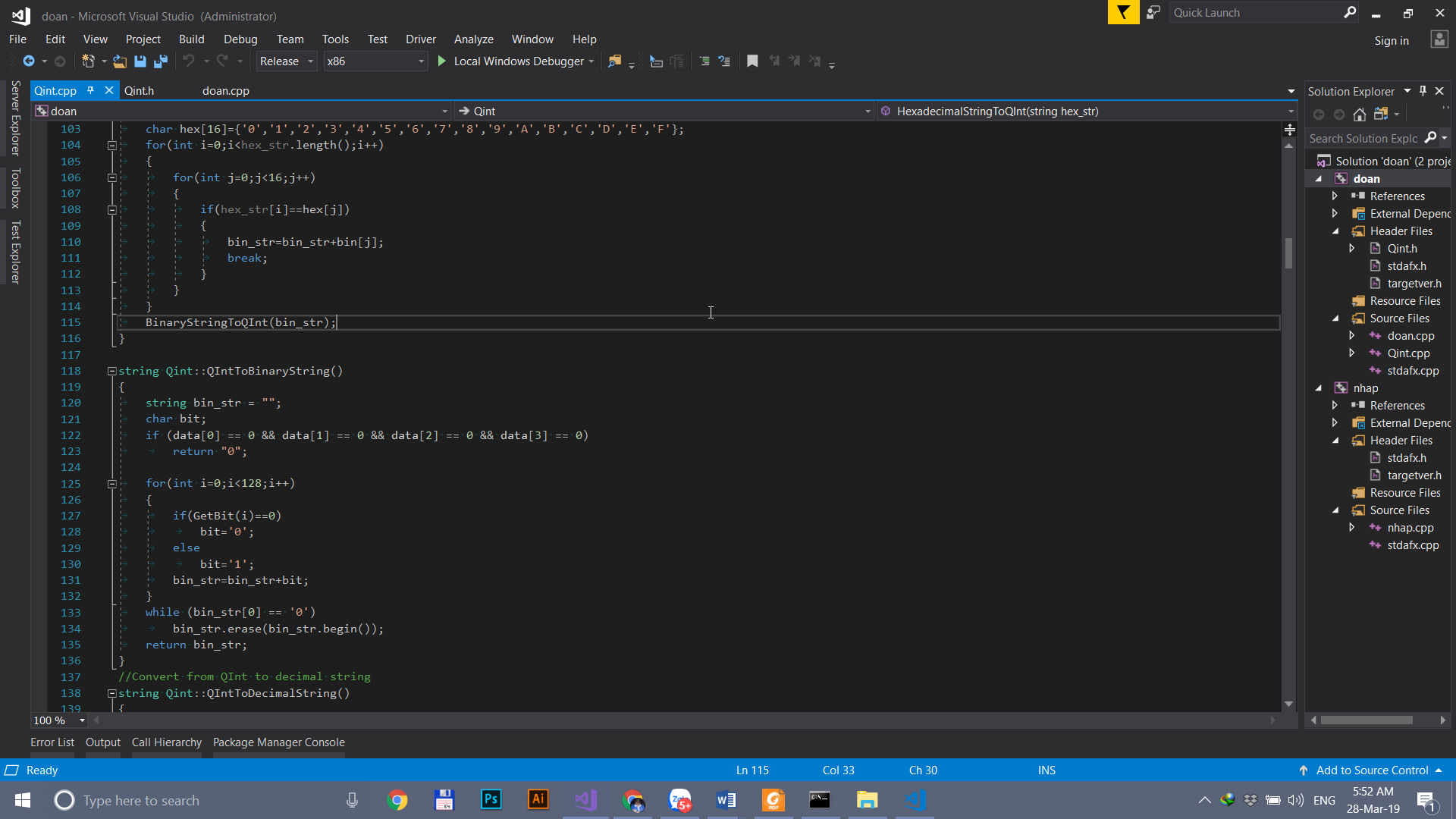
Hàm **DecimalStringToQint**: (chuyển chuỗi 10 thành dãy bit lưu ở QInt)

- Nếu là số **dương** : Chia lấy dư cho 2 ->Bit -> Setbit từ bit 127 -> Chia chuỗi cho 2 -> Tiếp tục lặp lại

-Nếu là số **âm** : Bỏ dấu âm -> Biến kiểm tra số âm = true -> Làm giống như số dương -> Dùng toán tử NOT để được bù 1-> rồi cộng bit 1 để được số bù 2

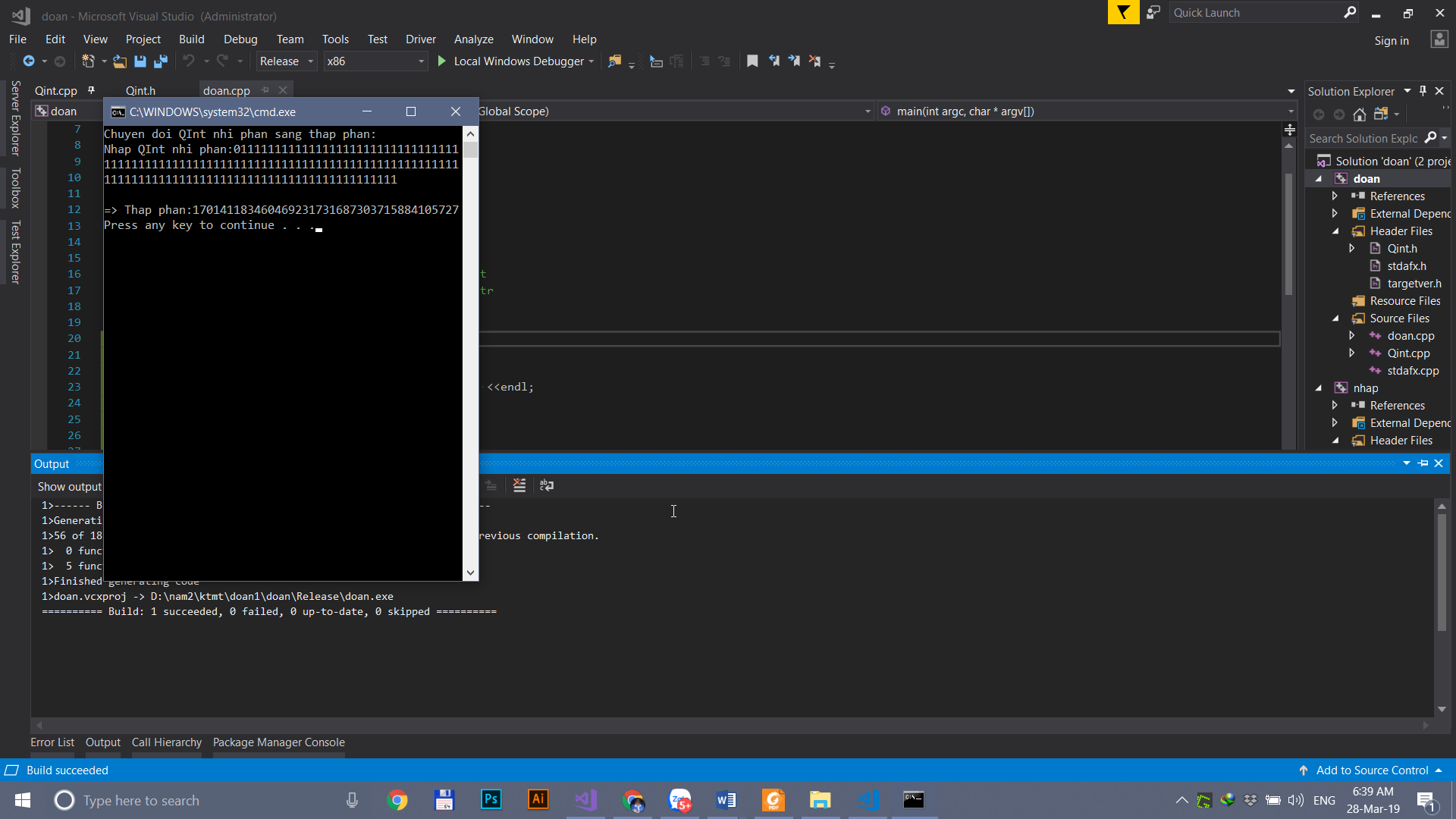
Hàm **QintToBinaryString**: (chuyển dãy bit ở Qint thành chuỗi nhị phân)

- Nếu bit 1 thì chuỗi thêm kí tự ‘1’, nếu bit 0 chuỗi thêm kí tự ‘0’ ⇨ Xóa bớt số 0 ở đầu ⇨Chuỗi nhị phân.

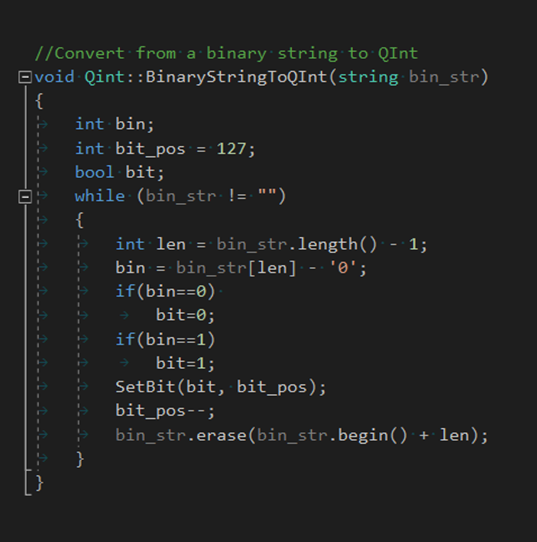


1. **Hàm chuyển đổi số QInt nhị phân sang thập phân:**

**Test:**

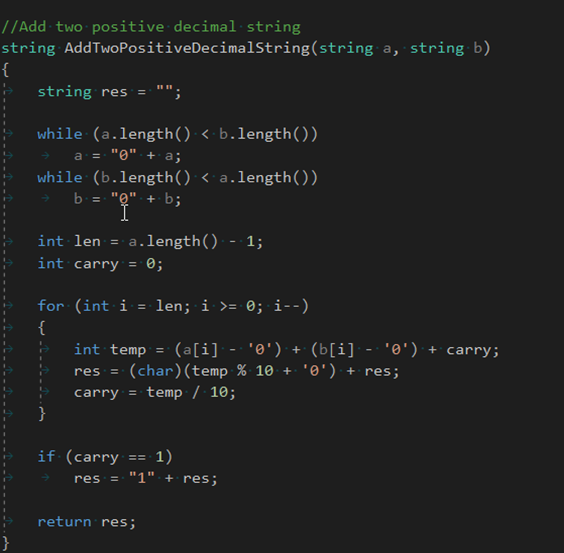
****

**Code:**

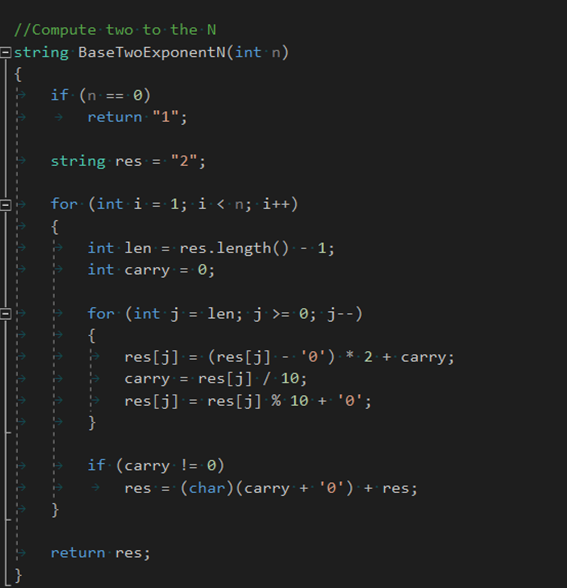
****

**BinaryStringToQInt() :** Chuyển chuỗi nhị phân sang dãy bit lưu ở QInt

1. Hàm **AddTwoPositiveDecinmalString**(): Cộng hai chuỗi số nguyên dương



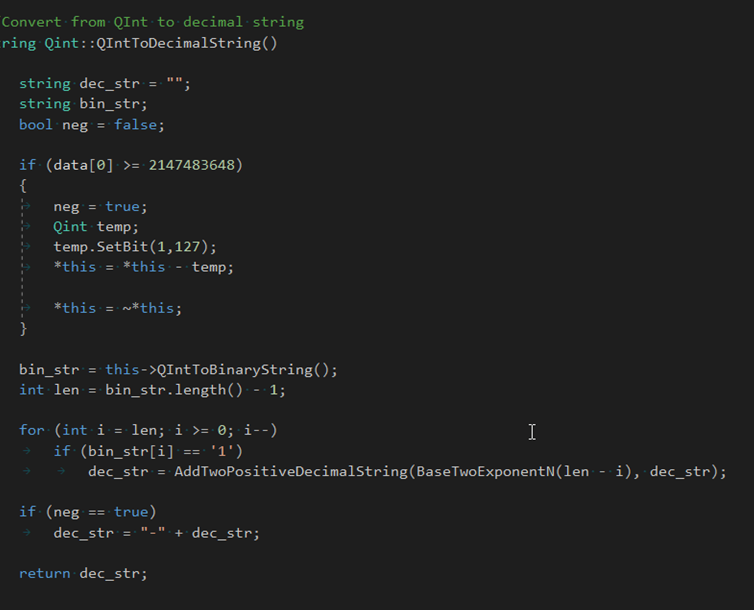
1. Hàm **BaseTwoExponentN(int n):** Trả vê chuỗi số kết quả 2 mũ N



1. Hàm **QIntToDecimalString**(): Chuyển dãy bit thành số thập phân

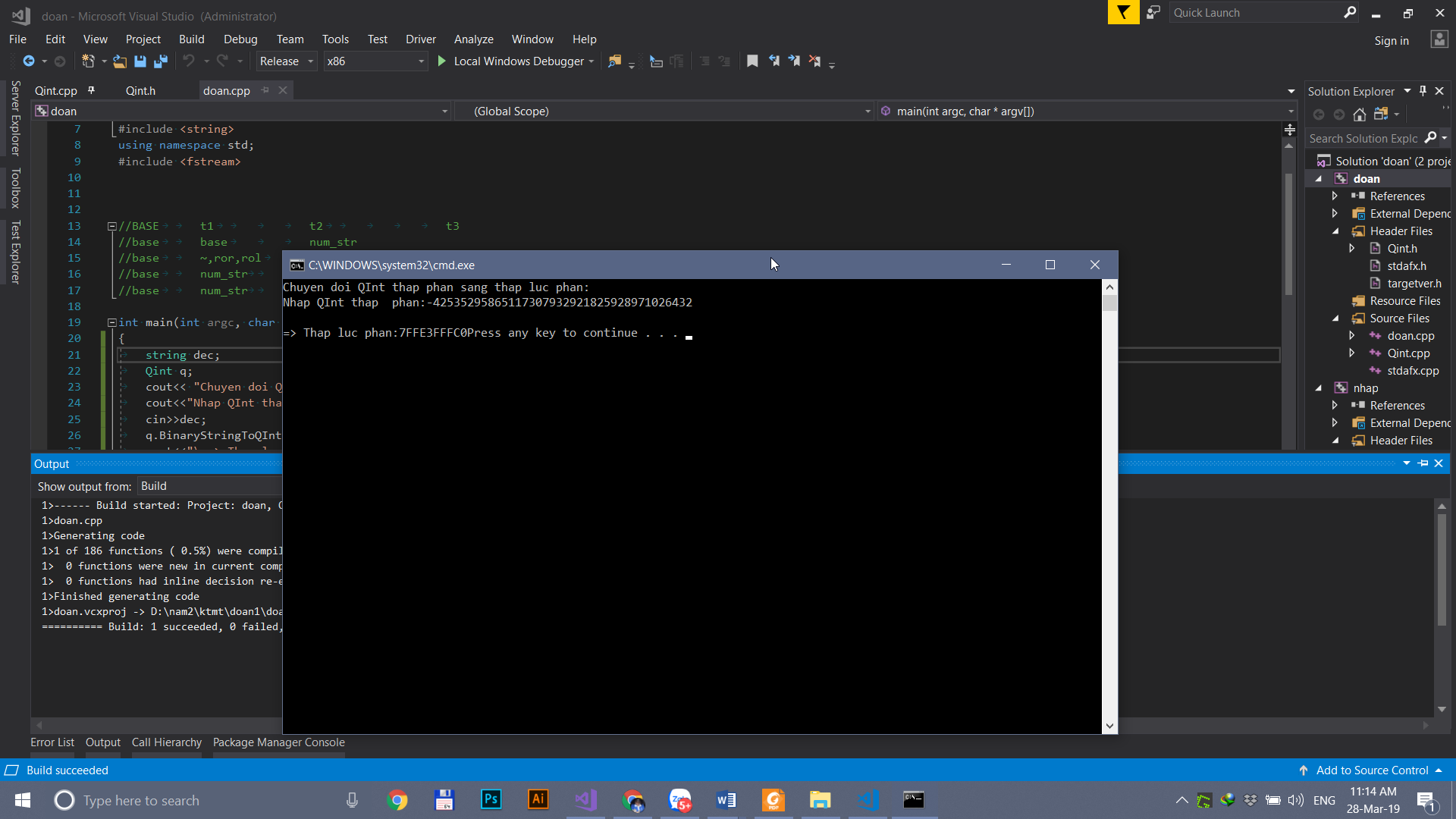
- Nếu data[0] của Qint >= 2^31 => Bù 2 => số âm -> trừ bit 1, và toán tử AND để được dãy bit của sô nguyen dương

- Lấy dãy bit -> Chuyển thành hệ 10 -> nếu số âm thì thêm dấu –

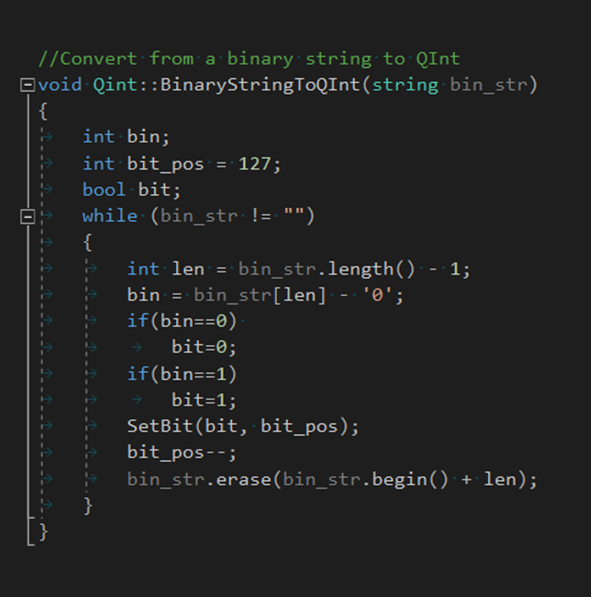


1. **Hàm chuyển đổi số QInt nhị phân sang thập lục phân:**

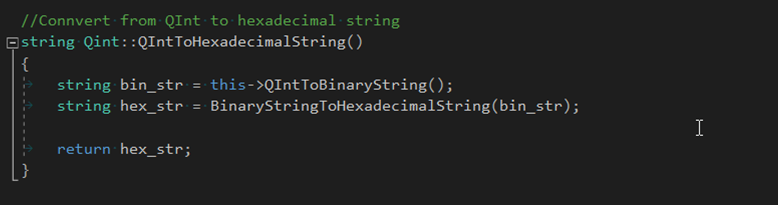
**Test:**



**Code:**

****

**BinaryStringToQInt**() : Chuyển chuỗi nhị phân sang dãy bit lưu ở Qint



**QIntToHexadecimalString**(): Chuyển dãy bit thành số thập lục phân

Chuyển bit về nhị phân -> chuyển nhị phân về thập phân

1. Hàm **decimalstringtohexadecimalstring** là hàm chuyển từ hệ 10 sang 16:

Dùng những hàm trước đó chuyển chuỗi hệ 10 sang QInt (dùng hàm **DecimalStringtoQInt**), chuyển QInt sang chuỗi hệ 2 (dùng hàm **QIntToBinaryString**), chuyển chuỗi hệ 2 sang hệ 16 (dùng hàm **BinaryStringToHexaDecimalString**)

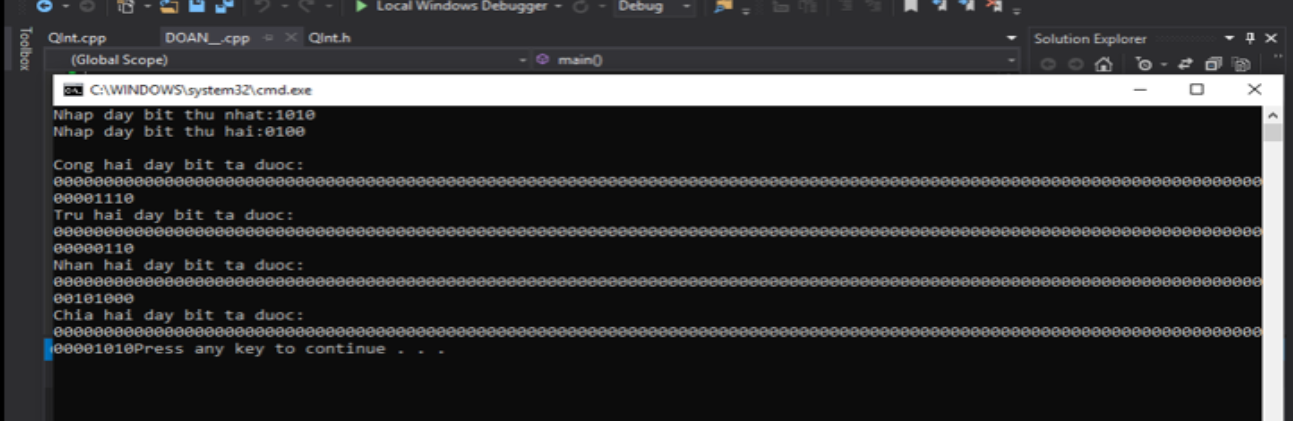


1. **Các hàm tính toán:**

* **Hàm cộng:** khai báo một bit nhớ, lấy từng bit của dãy bit , dùng toán tử ^ (xor)

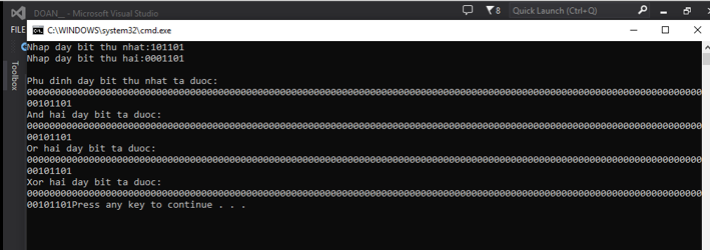
2 bit của dãy bit và bit nhớ rồi đưa vào biến kết quả, bit nhớ sẽ bằng 1 khi 2 bit đều bằng 1 hoặc 1 trong 2 bit và bit nhớ đều bằng 1.

* **Hàm trừ:** dựa trên hàm cộng, phép trừ 1 số cho 1 số là cộng 1 số với số đối của nó, nên ta phải tìm số đối của số thứ hai, bằng cách tìm bù 2 của nó, chính là lấy phủ định rồi cộng 1.
* **Hàm nhân**: Khởi tạo: A = 0; k = n; Q-1 = 0 (thêm 1 bit = 0 vào cuối Q) Lặp khi k > 0 . Nếu 2 bit cuối của Q0Q-1 = 10 thì A – M -> A = 01 thì A + M -> A = 00, 11 thì A không thay đổi Shift right [A, Q, Q-1]
* **Hàm chia**: Khởi tạo: A = n bit 0 nếu Q > 0; A = n bit 1 nếu Q < 0; k = n. Lặp khi k > 0 { Shift left (SHL) [A, Q] A – M -> A   
  **# Nếu A < 0**: Q0 = 0 và A + M -> A   
  **# Ngược lại**: Q0 = 1,k = k – 1.   
  🡺Kết quả: Q là thương, A là số dư.



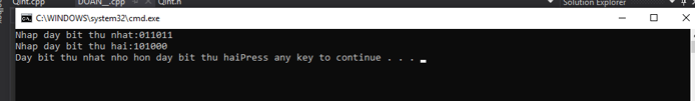
1. **Các phép toán AND, OR, XOR, NOT :**

* Phép toán **and**: kêt quả sẽ là 1 nếu cả 2 bit đều bằng 1, các trường hợp còn lại bằng 0.
* Phép toán **or**: kết quả sẽ là 0 nếu cả 2 bit đều bằng 0, các trường hợp còn lại bằng 1.
* Phép toán **not**: kết quả sẽ là 0 nếu bit đó bằng 1 và ngược lại.
* Phép toán **xor**: kết quả sẽ bằng 0 nếu cả hai bit có giá trị bằng nhau, ngược lại bằng 1.



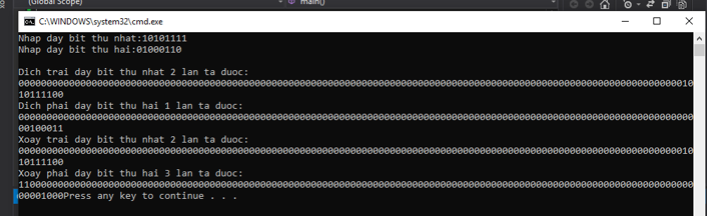
1. **Các phép so sánh <, <=, >, >=,==,=:**

* Phép so sánh **<:** xét dấu trước, nếu cả hai dãy bit cùng dấu thì so sánh từ trái sang phải, dãy nào nhỏ hơn thì nhỏ hơn (nếu dấu +), dãy nào lớn hơn thì nhỏ hơn (nếu dấu -).
* Phép so sánh **>:** ngược lại phép <.
* Phép so sánh **==:** nếu kiểm tra nó không bé hơn và không lớn hơn.
* Phép so sánh **<=,>=:** kết hợp hai điều kiện nhỏ hơn và bằng.

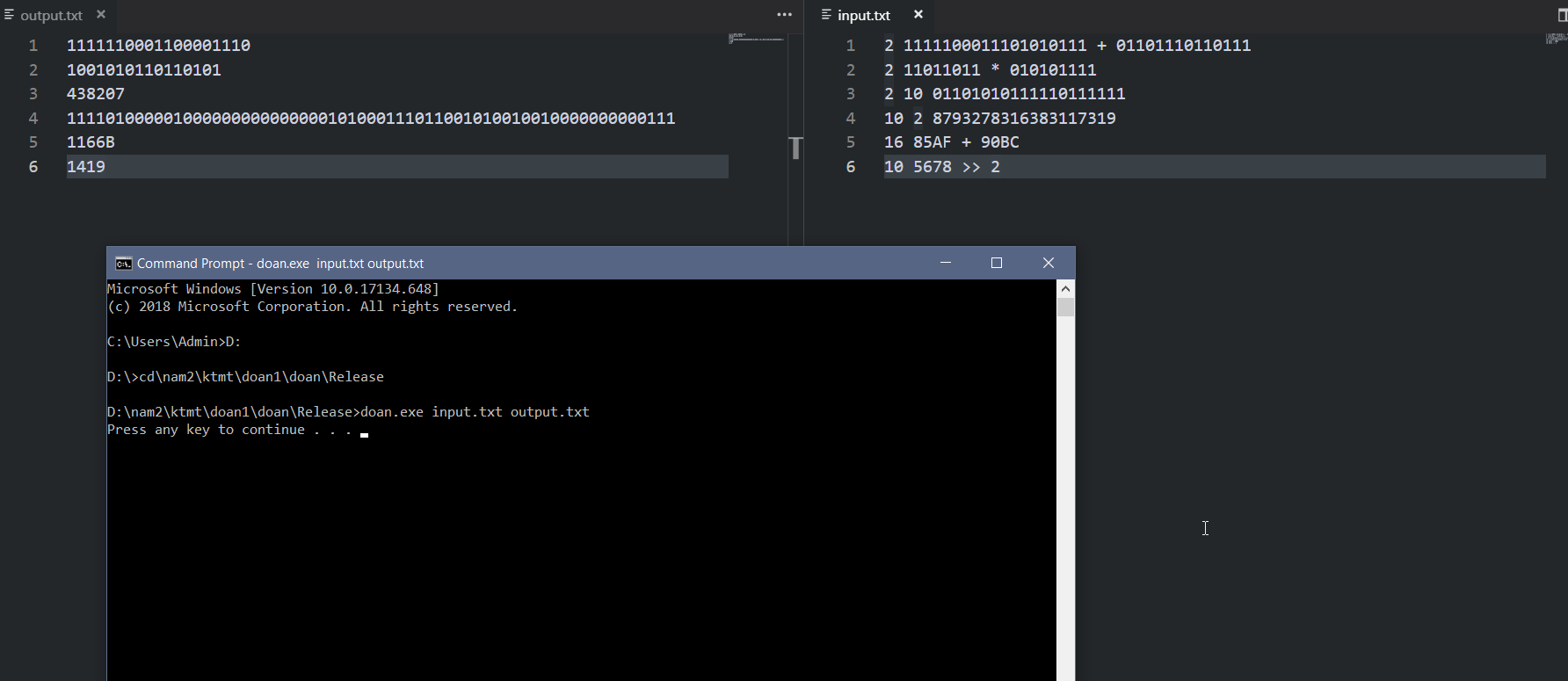


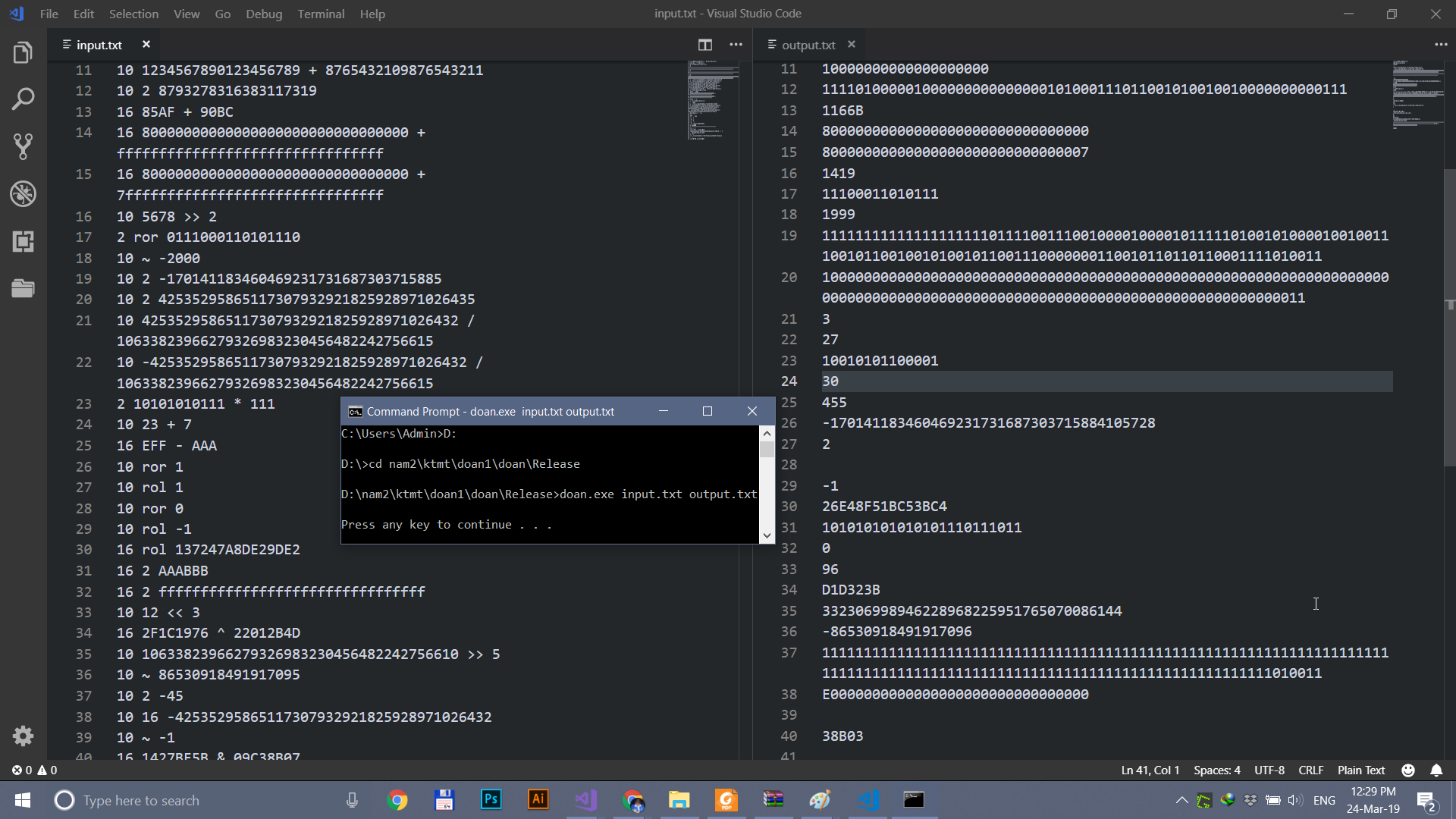
1. **Các toán tử: dịch trái (<<), dịch phải (>>), hàm xoay trái (rol), hàm xoay phải (ror):**

* Phép **dịch trái**: Chuyển tất cả các bit sang trái, bỏ bit trái nhất, thêm 0 ở bit phải nhất.
* Phép **dịch phải**: Chuyển tất cả các bit sang phải, bỏ bit phải nhất, thêm 0 ở bit trái nhất.
* Phép **xoay trái**: Chuyển tất cả các bit sang trái, bit trái nhất thành bit phải nhất.
* Phép **xoay phải**: Chuyển tất cả các bit sang phải, bit phải nhất thành bit trái nhất.

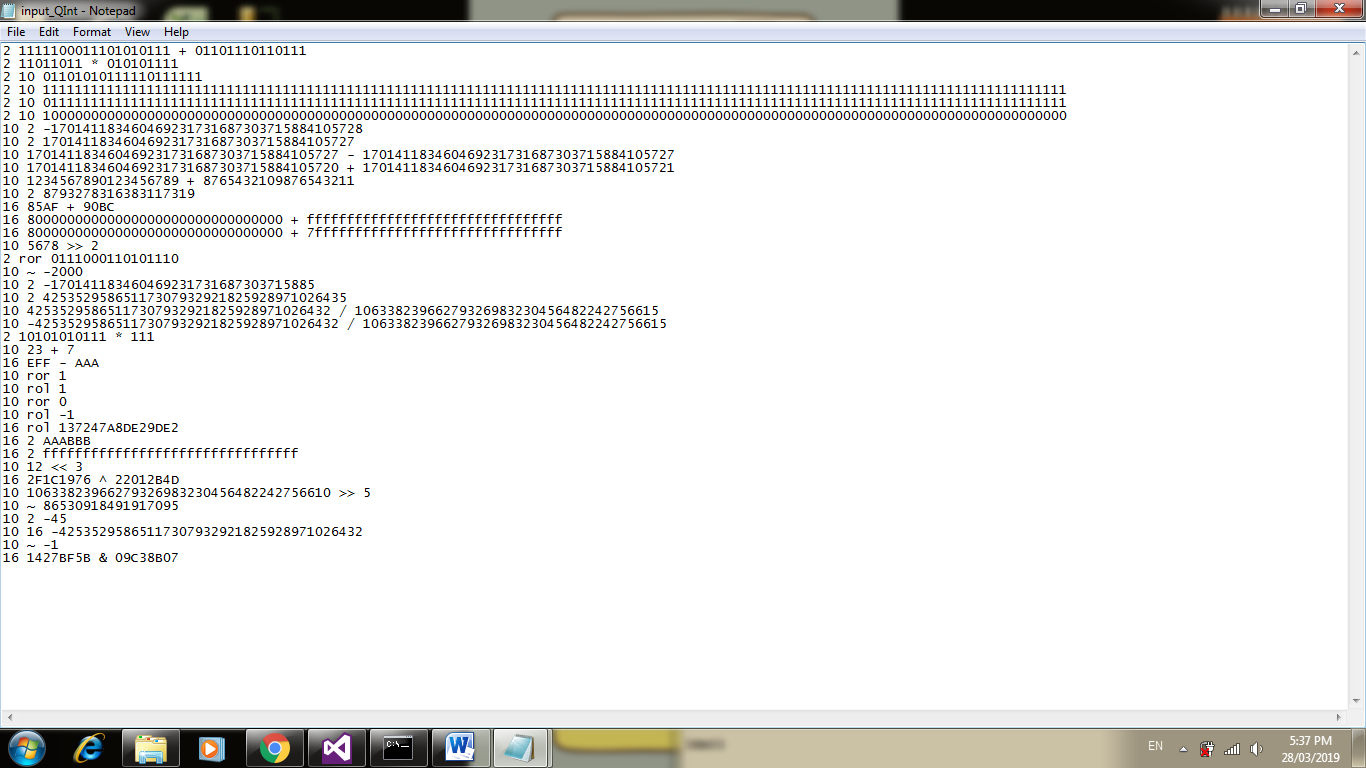


1. **Testcase của các hàm ở câu 1:**

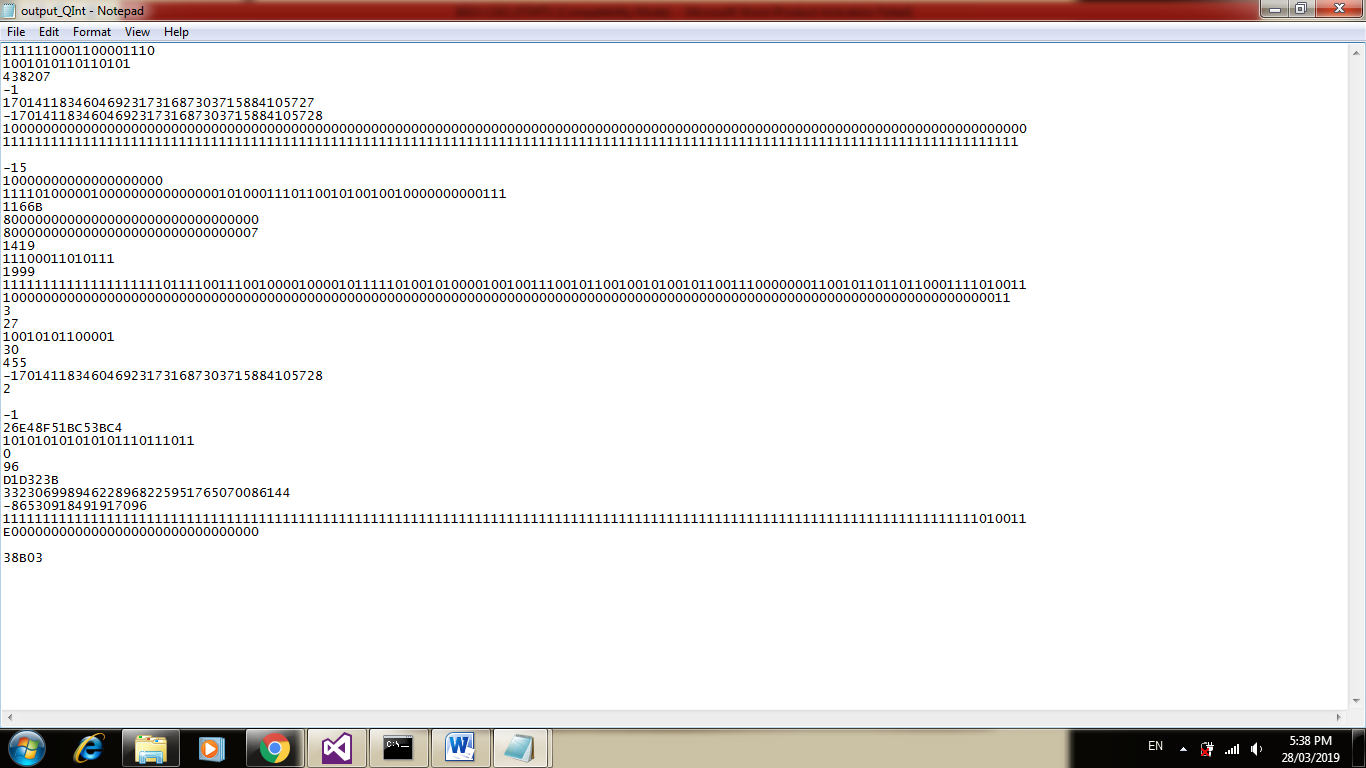




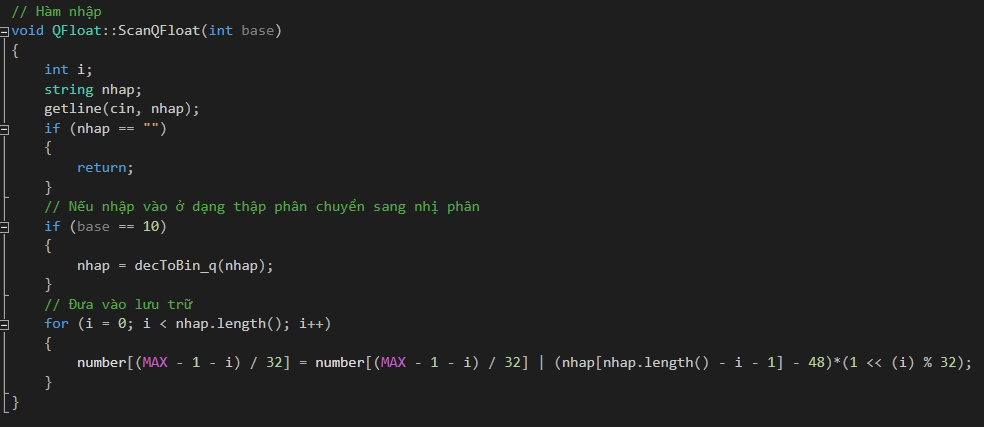
**File input\_QInt.txt :**



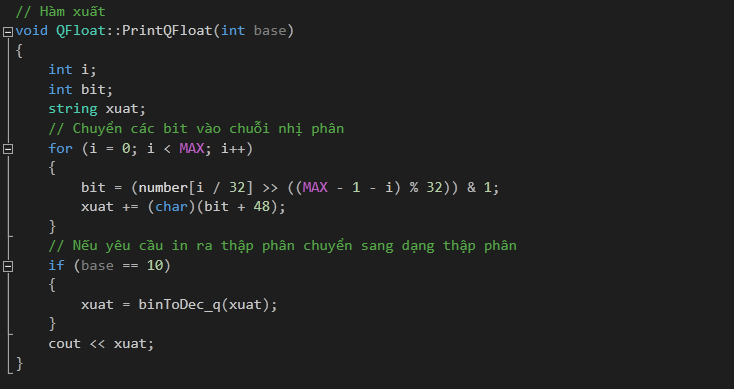
**File output\_QInt.txt :**



1. **Số chấm động chính xác cao:**
2. **Hàm Nhập:**

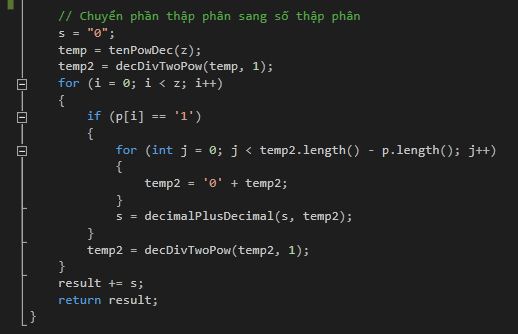
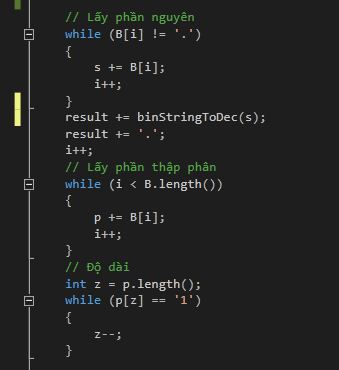
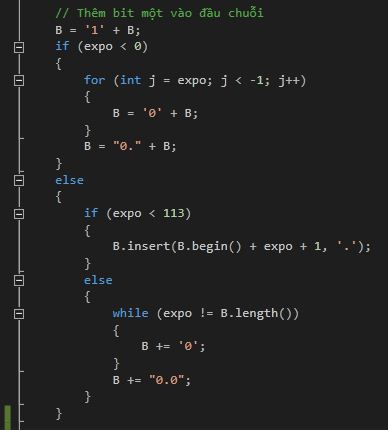
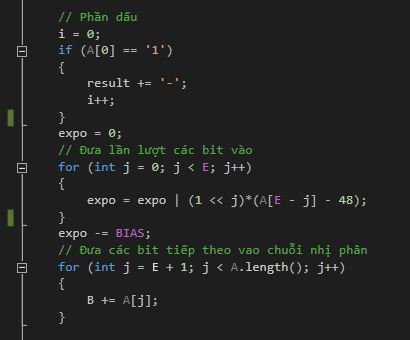
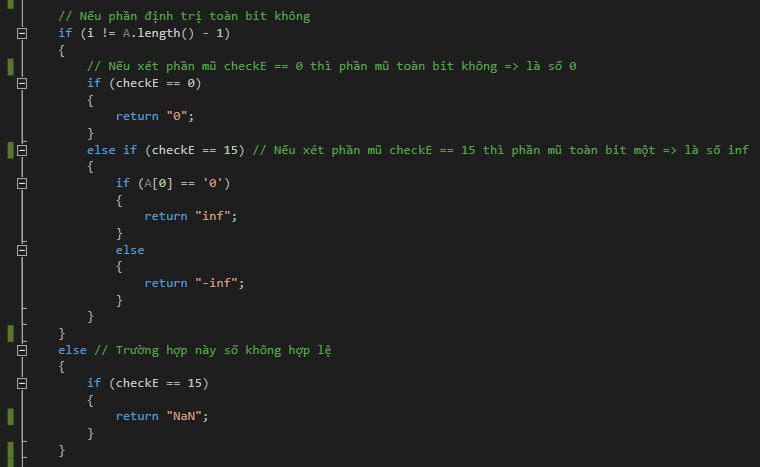
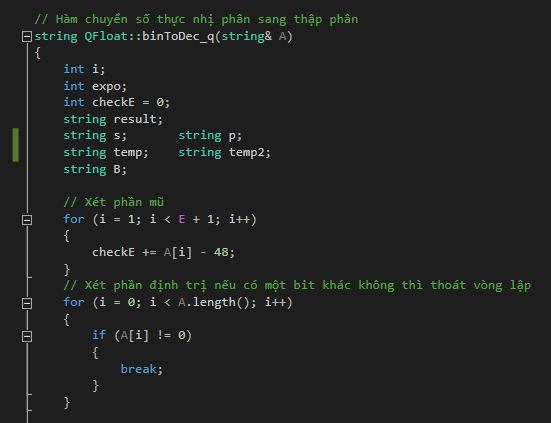
****

1. **Hàm Xuất:**

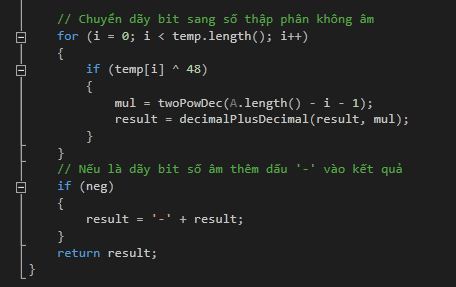
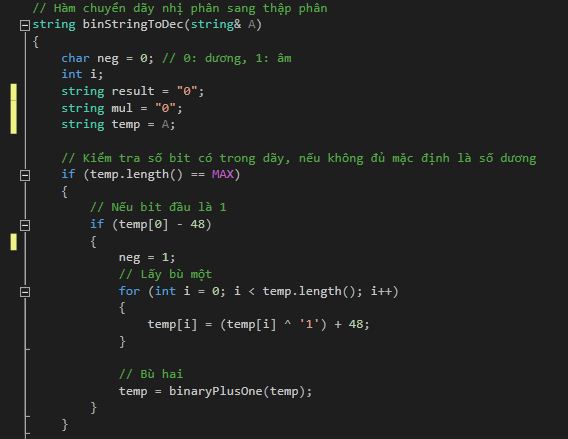
****

1. **Hàm chuyển đổi số Qfloat nhị phân sang thập phân:**

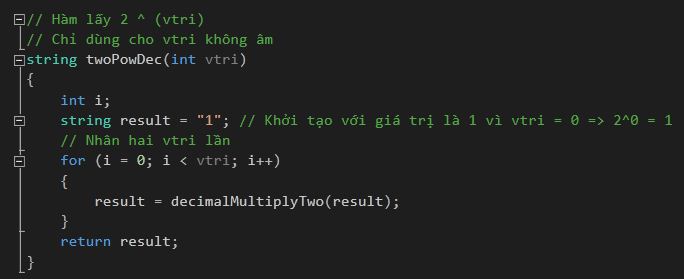
Hàm **binToDec\_q(string& A):** chuyển số thực nhị phân sang thập phân



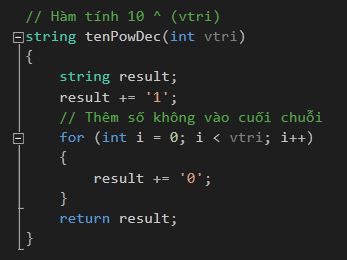
Hàm **binStringToDec(string& A):** chuyển dãy nhị phân sang thập phân



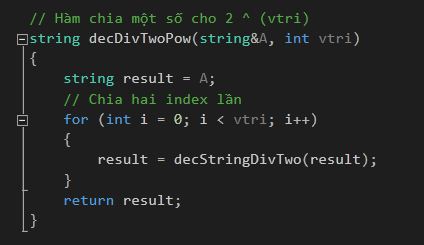
Hàm **twoPowDec(int vtri):** hàm trả về kết quả của công thức 2 ^ (vtri)



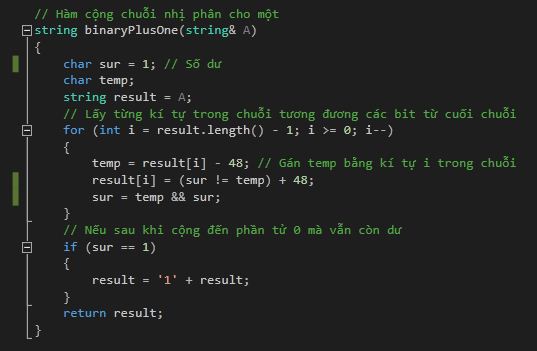
Hàm **tenPowDec(int vtri):** hàm tính 10 ^ (vtri)



Hàm **decDivTwoPow(string&A, int vtri):** hàm chia một số cho 2 ^ (vtri)

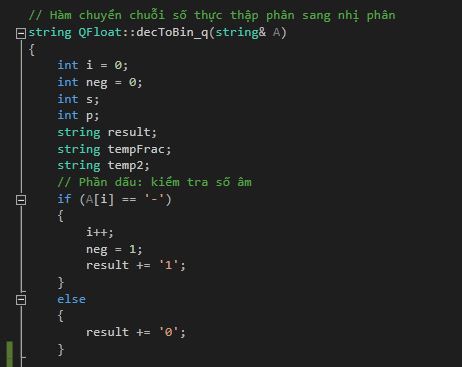


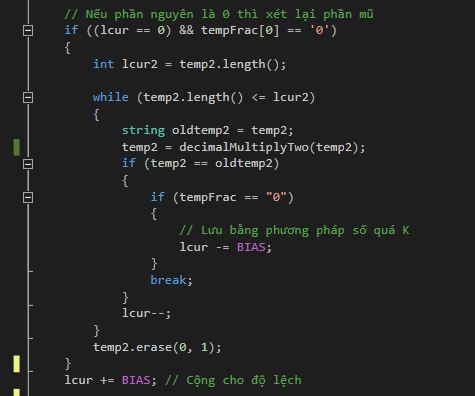
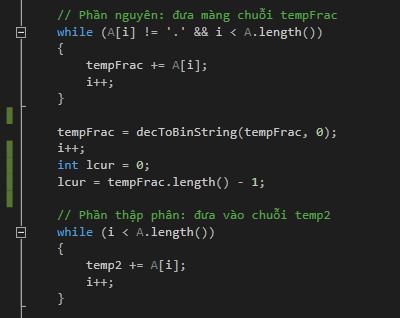
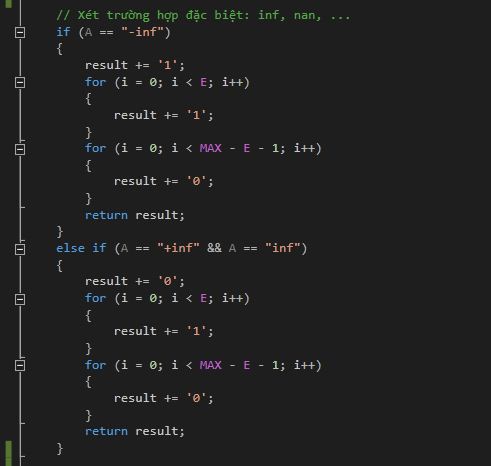
Hàm **binaryPlusOne(string&A):** hàm cộng chuỗi nhị phân cho 1

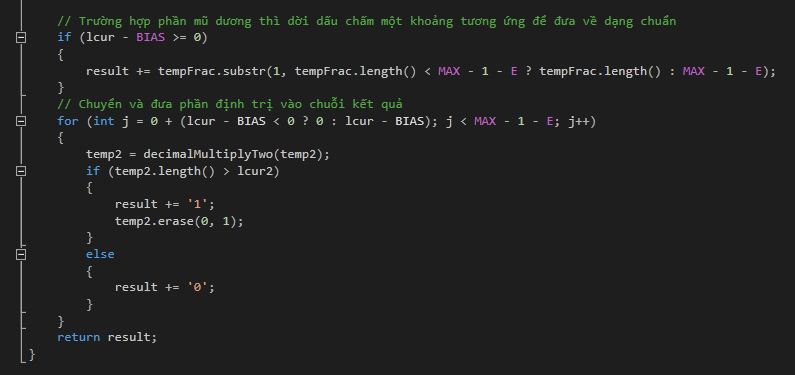


1. **Hàm chuyển đổi số Qfloat thập phân sang nhị phân:**

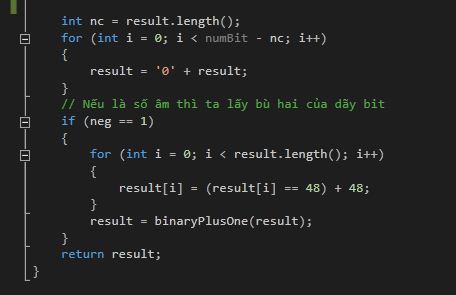
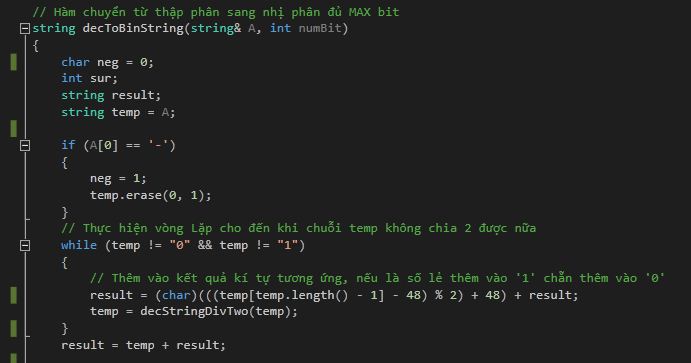
Hàm **decToBin\_q(string& A):** chuyển chuỗi số thực thập phân sang nhị phân



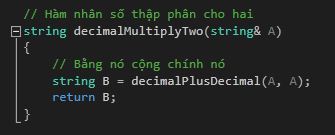




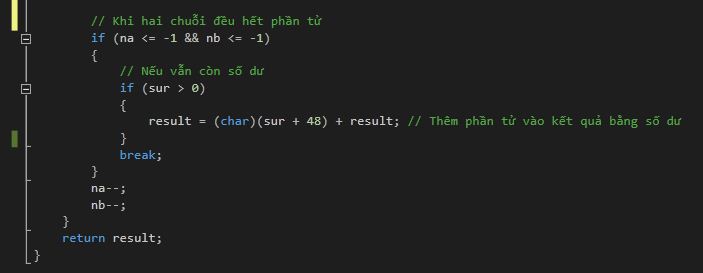
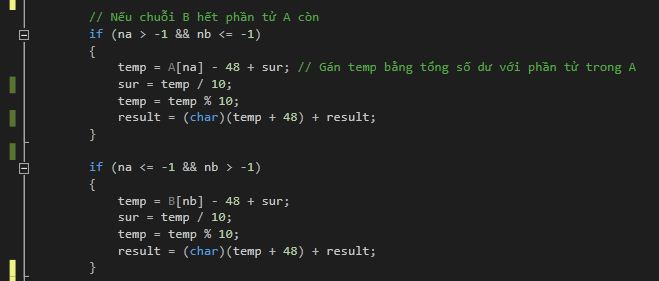
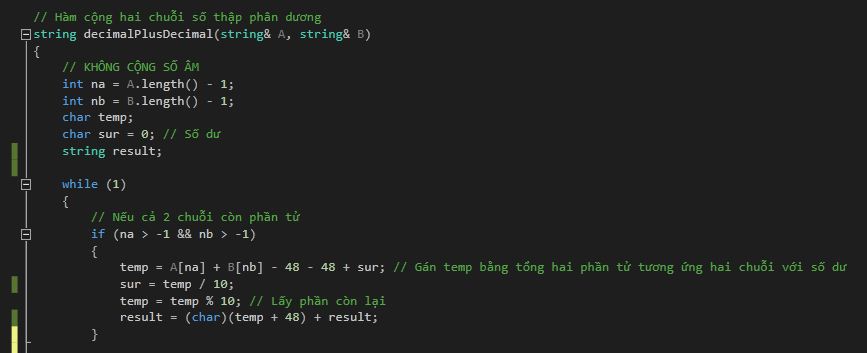
Hàm **decToBinString(string& A, int numBit)**: chuyển từ thập phân sang nhị phân được lưu ở dạng chuỗi



Hàm **decimalMultiplyTwo(string& A):** nhân số thập phân cho 2

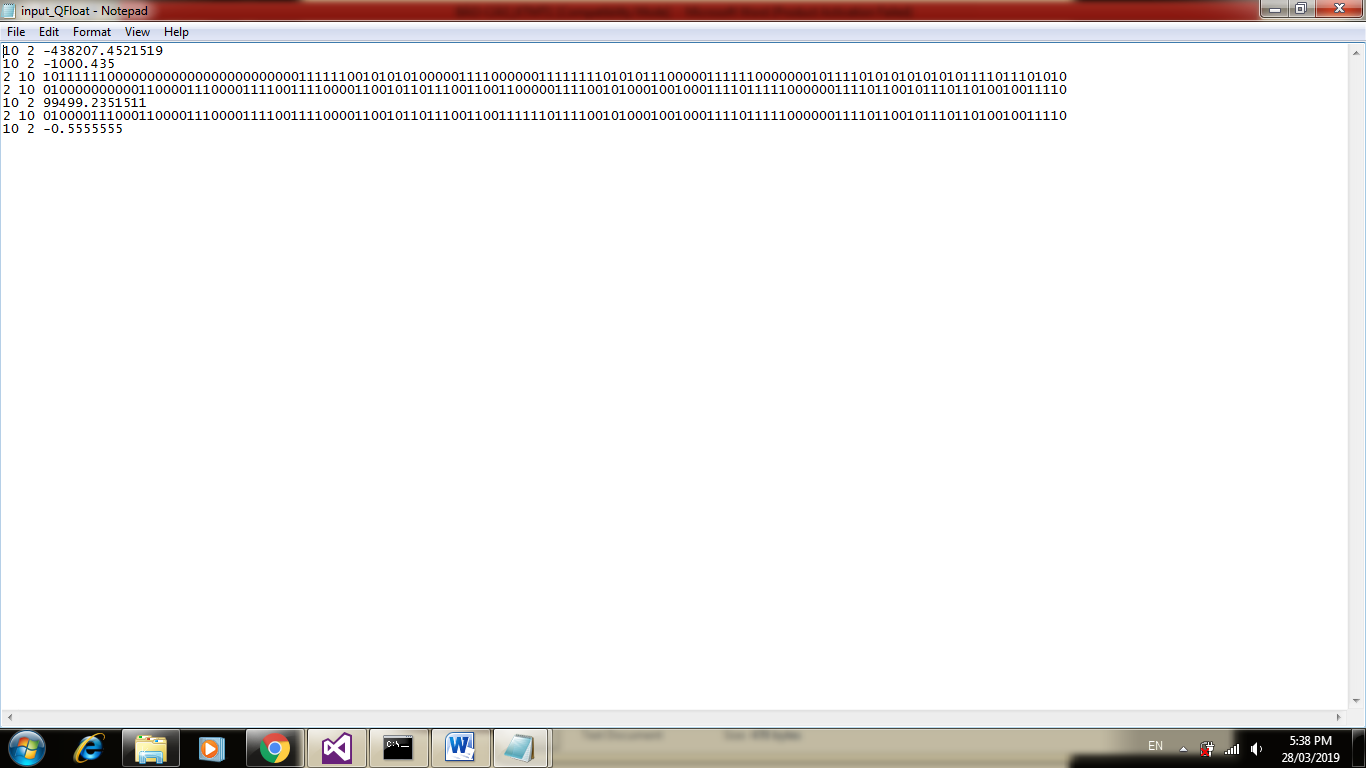


Hàm **decimalPlusDecimal(string& A, string& B):** cộng lại 2 chuỗi số thập phân dương

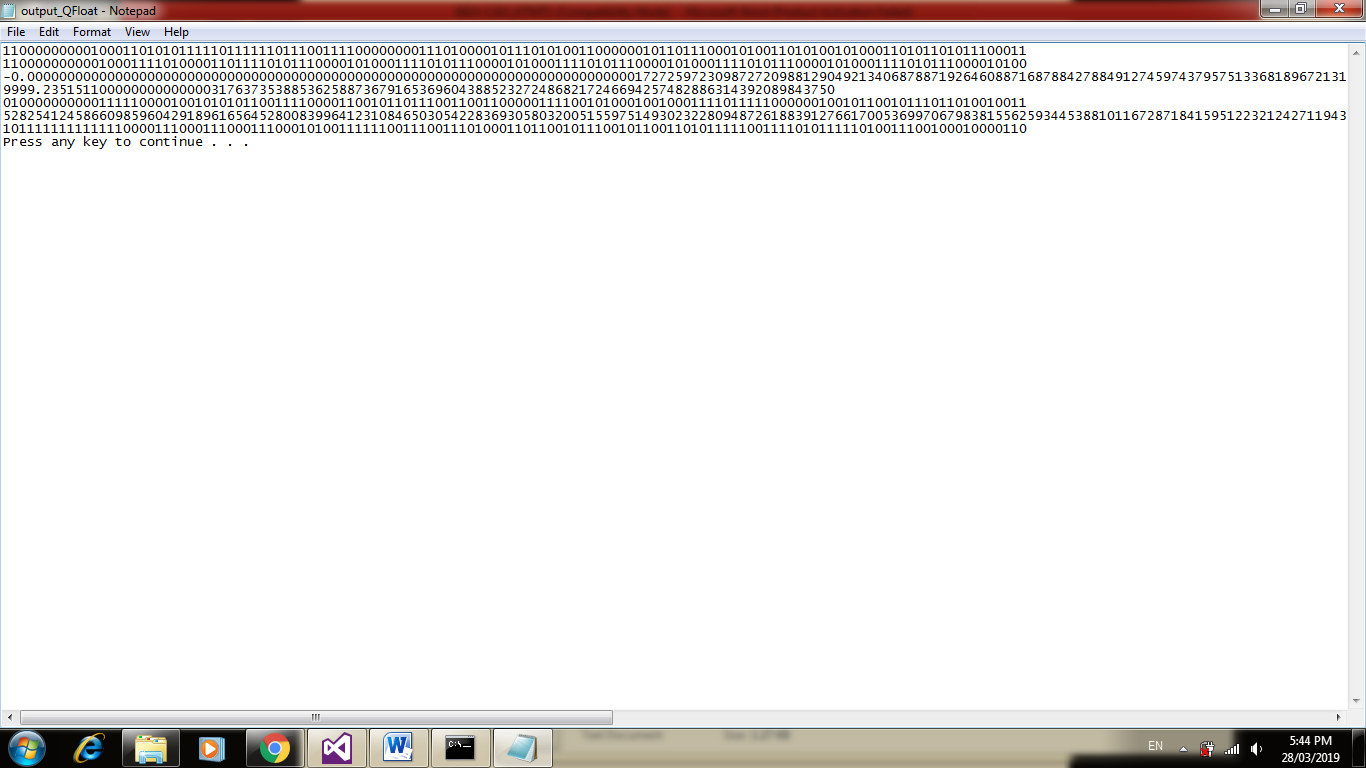


1. **Testcase của các hàm ở câu 2:**

**File input\_QFloat.txt :**



**File output\_QFloat.txt**



III / Tài liệu tham khảo

<https://www.h-schmidt.net/FloatConverter/IEEE754.html>

<https://www.geeksforgeeks.org/divide-large-number-represented-string/?fbclid=IwAR2KsO_OnY_4u-ACui_v4LRLVOBwTD59D4N34rfQ1jeRaEWxm1XQlEcOrsY>

<https://stackoverflow.com/questions/25459568/c-convert-a-real-number-to-64-bit-floating-point-binary?fbclid=IwAR1WzEngoca56ei1s8aWReLU-PS5ojU0wdvtgIlV-aWVJi_ABRfRrCy4e7g>

<https://www.exploringbinary.com/converting-floating-point-numbers-to-binary-strings-in-c/?fbclid=IwAR16T8o2ByVeYt8eAPb_gCeQ_zTGn-ek0Ie9E0JdopFpL-AwMdaujB0JCxY>

<https://bugs.vn/anxe0?fbclid=IwAR0MP6LIv6wCBlvCT5boJGMw9LkG9v8Vm9eb1Zd6OY9N5JxIFtZWJwm4dco>

<https://github.com/lamductan/QInt_QFloat?fbclid=IwAR0-TXKh02TNYAu_Nnh5jFngVqm7kchIa9krkhudKt1ecBtARBtj2fkPtZA>