Đại học Quốc gia TPHCM

Trường Đại học Khoa học tự nhiên

Khoa Công nghệ thông tin  
--oOo--

**BÁO CÁO ĐỒ ÁN 1**

Môn: Kiến Trúc Máy Tính Và Hợp Ngữ

Năm học: 2018- 2019 Học kỳ 1

Chủ đề: \_BIỂU DIỄN VÀ TÍNH TOÁN SỐ HỌC\_

|  |  |
| --- | --- |
| **Nhóm:**   * 1753113 - Nguyễn Trọng Triết * 1753139 - Trương Trần Hải Yến | **Giáo viên phụ trách:**  Thầy: Lê Quốc Hòa  Thầy: Lê Viết Long |

Năm 2018

1. Thành Viên

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ và Tên** | **Công việc** |
| **1753113** | Nguyễn Trọng Triết | Convert: 2 to 10, 2 to 16, 10 to 2, 10 to 16, 16 to 2, 16 to 10.  “&”, “|”, “<<”, “>>”, “ror”, “rol” |
| **1753139** | Trương Trần Hải Yến | Calculate: “+”, “-”, “\*”, “/”.  Xử lí File. |

1. Môi trường lập trình

* Microsoft Visual Studio 2017 bằng C++.
* Xcode bằng C++.

1. Ý tưởng thiết kế

3.1 QInt

* Sử dụng vector gồm 4 biến kiểu uint32\_t để biểu diễn QInt và QFloat, trong đó vector [0] chứa 32bit đầu tiên (từ 0 ->31), vector[1] chưa 32bit tiếp theo (32 ->63), vector[2] chứa 32 kế (64-95), và vector[3] chứa 32bit còn lại.
* Xây dựng các hàm convert và tính toán của QInt: chuyển đổi qua lại giữa các dạng cơ số, các toán tử “+”, “-”, “\*”, “/”, các phép xử lý bit “|”, “&”, “^”, “~”, “<<”, “>>”, “ror”, “rol”.
* Các hàm chuyển đổi cơ số, khởi tạo QInt: Các số kiểu QInt được nhập vào ở dạng chuỗi string, cho phép người dùng nhập vào các hệ số 2, 8, 10, 16 ở dạng chuỗi string. Chuỗi string sau khi nhập vào sẽ được convert về chuỗi string ở dạng nhị phân. Sau đó khởi tạo QInt, dùng 4 biến uint32\_t 128 bit để lưu các bit nhị phân của chuỗi string truyền vào.
* Các hàm In QInt ra các hệ cơ số 2, 10, 16: Do QInt được chứa trong 4 biến uint32\_t, ta sử dụng hàm getBit để lấy các bit ở từng vị trí rồi chuyển đổi từ hệ 2(bit chứa trong QInt) sang hệ cơ số người dùng yêu cầu.
* Toán tử AND &: dùng phép AND của uinit32\_t cho từng data từ data[0] đến data[3].
* Toán tử OR |: dùng phép OR của uinit32\_t cho từng data từ data[0] đến data[3].
* Toán tử XOR ^: dùng phép XOR của uinit32\_t cho từng data từ data[0] đến data[3].
* Toán tử NOT ~: dùng phép not của uint32\_t cho từng data từ data[0] đến data[3].
* Phép dịch trái <<: Phép dịch trái dịch data[3] sang 1 bit, sau đó gán data[2] ở bit thứ 31 vào data[3] bit thứ 0, cứ như thê đến data[0].
* Phép dịch phải >>: Phép dịch phải dịch từ data[0] sang 1 bit, sau đó xét bit thứ 0 ở data[1] gán vào bit thứ 0 của data[0], lặp lại đến data[3]
* Phép ROL: sử dụng phép dịch để dịch đi k lần lưu lại những bit bị dịch ra ngoài gán lại vào dãy bit từ bên phải.
* Phép ROR: sử dụng phép dịch để dịch đi k lần lưu lại những bit bị dịch ra ngoài gán lại vào dãy bit từ bên phải.
* Operator +: Tạo 1 kiểu QInt Result = \*this, ta dùng phép & 2 số và gán vào QInt Carry, là những bit 1 giữa 2 số. Sau đó ta sử dụng phép ^ 2 số và gán vào Result. Cuối cùng ta dùng phép << để đưa các bit 1 về bên trái 1bit và tiếp tục thực hiện phép cộng.
* Operator -: Ta xem như a – b = a + (-b). Chuyển b về dạng bù 2 (b -> bù 1 + 1 -> bù 2). a + bù 2 của b = a – b.
* Operator \*: Tạo biến bool nega = 0, flag số âm. Đầu tiên kiểm tra các số xem là âm hay dương, nếu 1 trong 2 âm thì nega = 1, nếu âm thì chuyển về dương. Sau đó tạo biến QInt Result, nếu QInt lẻ thì cộng kết quả với số thứ 1. Rồi << số thứ 1 về 1 bit (nhân số thứ 1 với 2). Rồi >> số thứ 2 về 1 bit (chia số thứ 2 với 2). Cuối cùng nếu nega = 1, thì đưa kết quả về dạng bù 2.
* Operator /: Tạo biến bool nega nhứ phép nhân. Tạo biến QInt result, dịch result << 1. Sau đó phần tử thứ 0 của result = phần tử cuối của biến thứ 1 & giá trị 1 dịch trái 31bit, rồi | với phần tử thứ 0 của result, rồi dịch phải 31bit. Tiếp đó dịch trái số thứ 1 về 1bit. Biến result = chính nó trừ cho số thứ 2. Nếu result âm thì result = result + số thứ 2, ngược lại thì phần tử thứ 0 của số thứ 1 = chính nó | 1. Kết quả sẽ = số thứ 1 đã được biến đổi sau khi thực hiên các phép toán trên (result = số thứ 1). Nếu nega = 1, thì đưa kết quả về dạng bù 2.

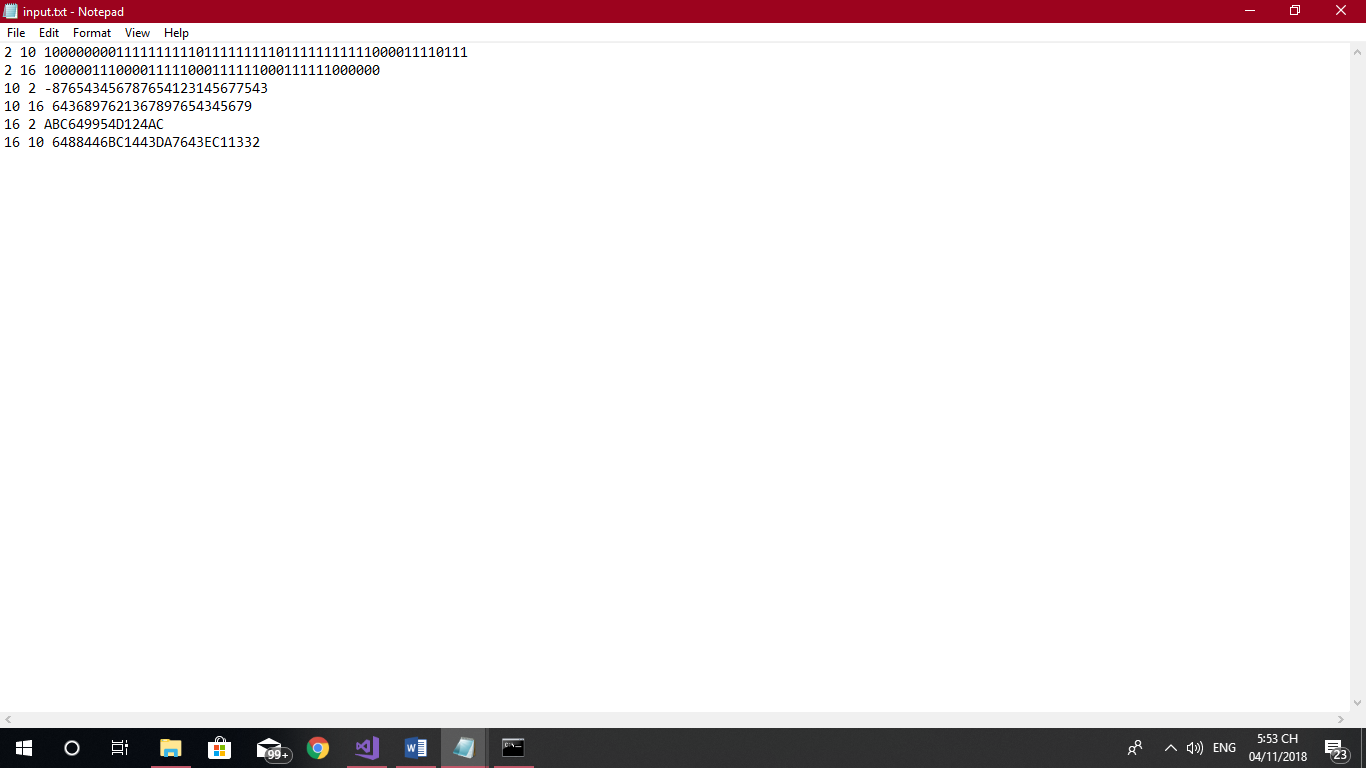
3.2 QFloat

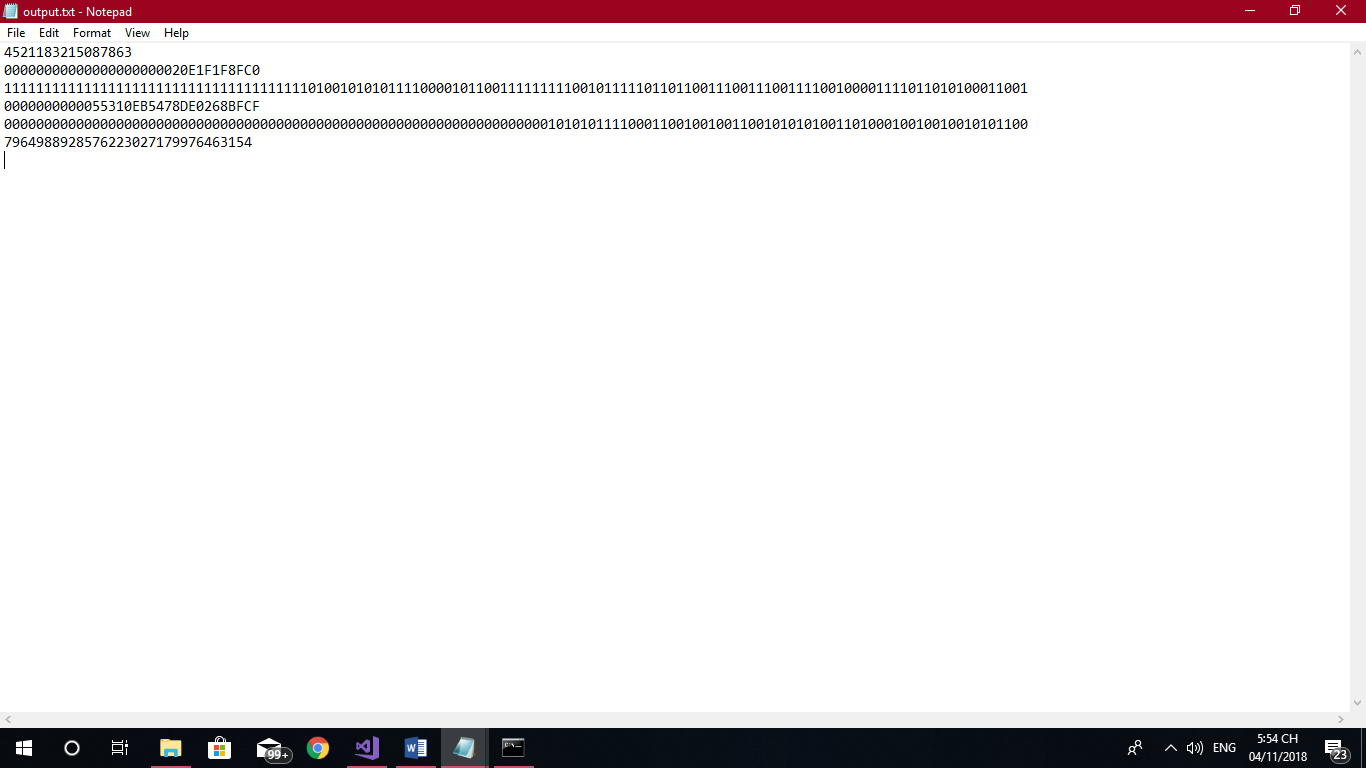
* Kiểu dữ liệu QFloat được lưu bằng vector gồm 4 biến kiểu uint32\_t, với bit đầu tiên thể hiện dấu, 15bit tiếp theo thể hiện E, 112bit còn lại thể hiện phần Significant.
* Xây dựng các hàm convert QFloat: chuyển đổi qua lại giữa các dạng cơ số.
* Hàm nhập, khởi tạo QFloat: Hàm khởi tạo QFloat được yêu cầu người dùng nhập vào 2 tham số(chuỗi string, và hệ của chuỗi vừa nhập 2 hoặc 10), hàm khởi tạo sẽ chuyển hệ nhập vào thành hệ nhị phân dạng chuỗi 128 kí tự được sắp xếp theo: vị trí 0 là bit dấu, 15 vị trí tiếp theo là Exponent, 112 vị trí còn lại thể hiện significant. Sau khi có chuỗi được sắp xếp sẵn ta cho nạp vào QFloat để lưu ở 4 biến uint32\_t.
* Hàm xuất(In QFloat): Dùng hàm getBit để lấy bit dấu xét QFloat là âm hay dương, xét biến exp(exponent) dựa vào exp tìm ra vị trí dấu ‘.’ Sau đó cho nó về đúng vị trí.
* Phạm vi biểu diễn: -2127 -> 2127 – 1.

1. Chạy và Test Chương Trình

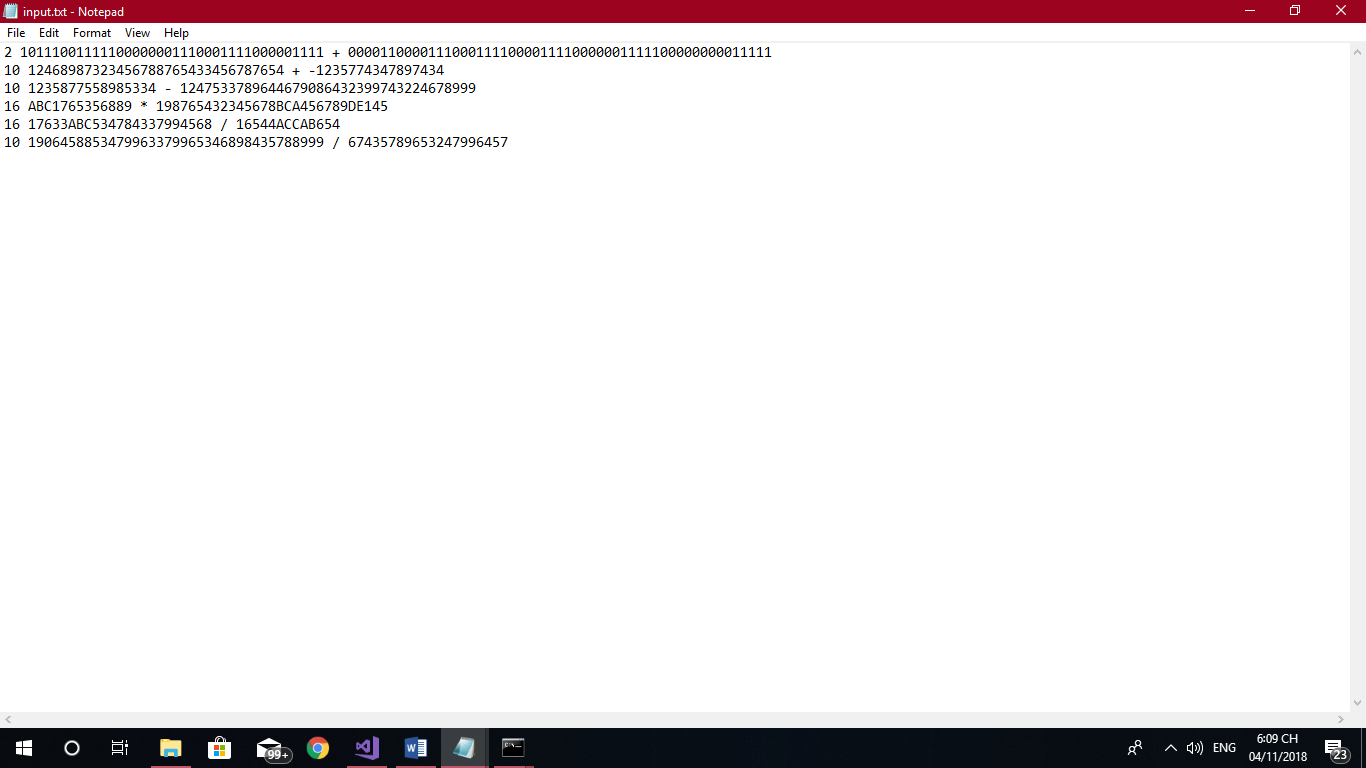
4.1 QInt

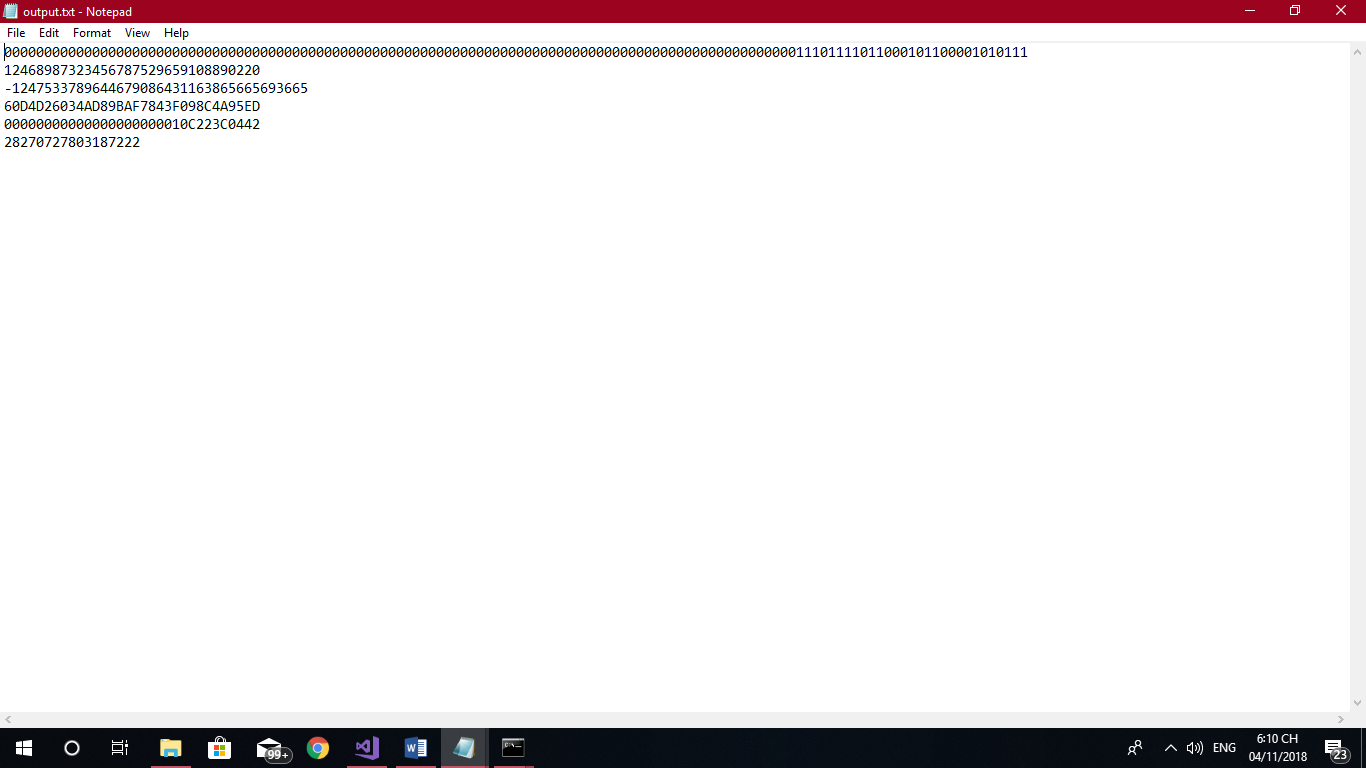
* Chuyển đổi cơ số



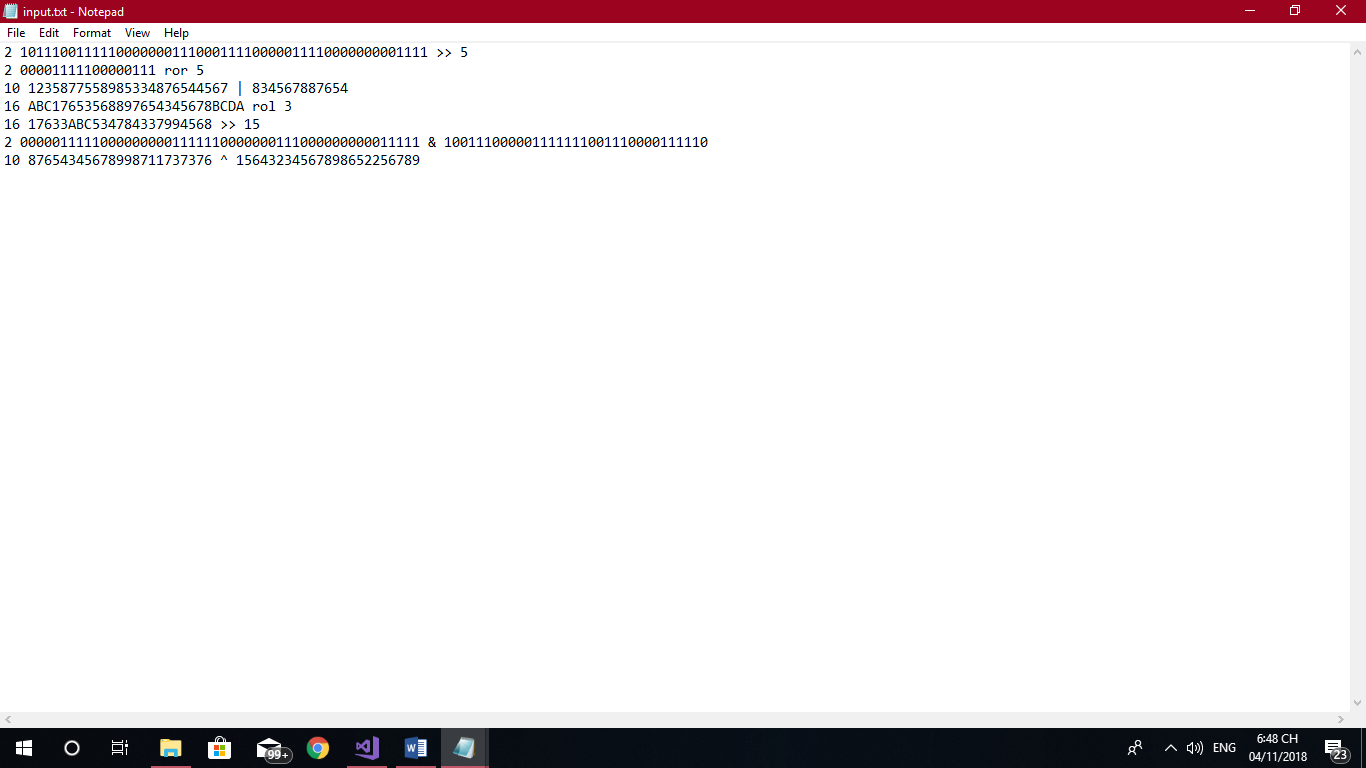


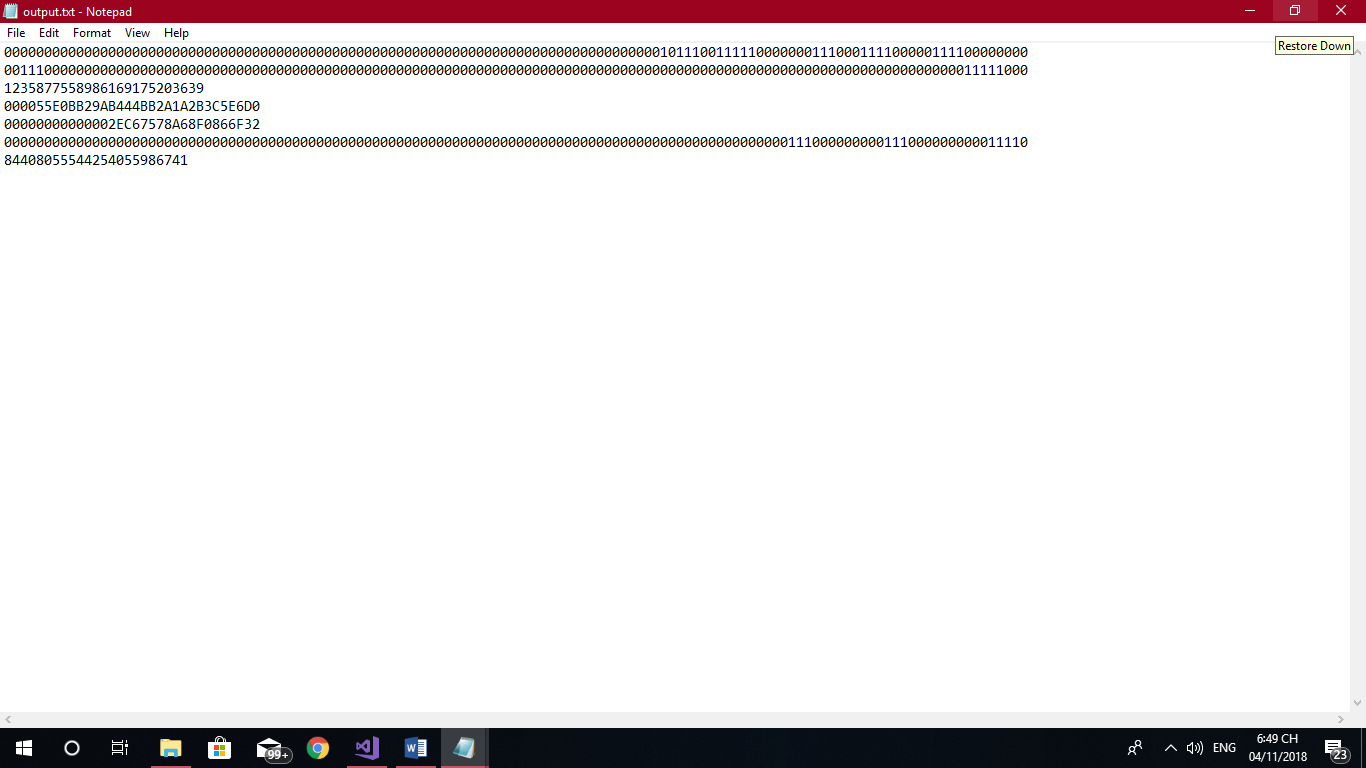
* Toán tử +, -, \*, /





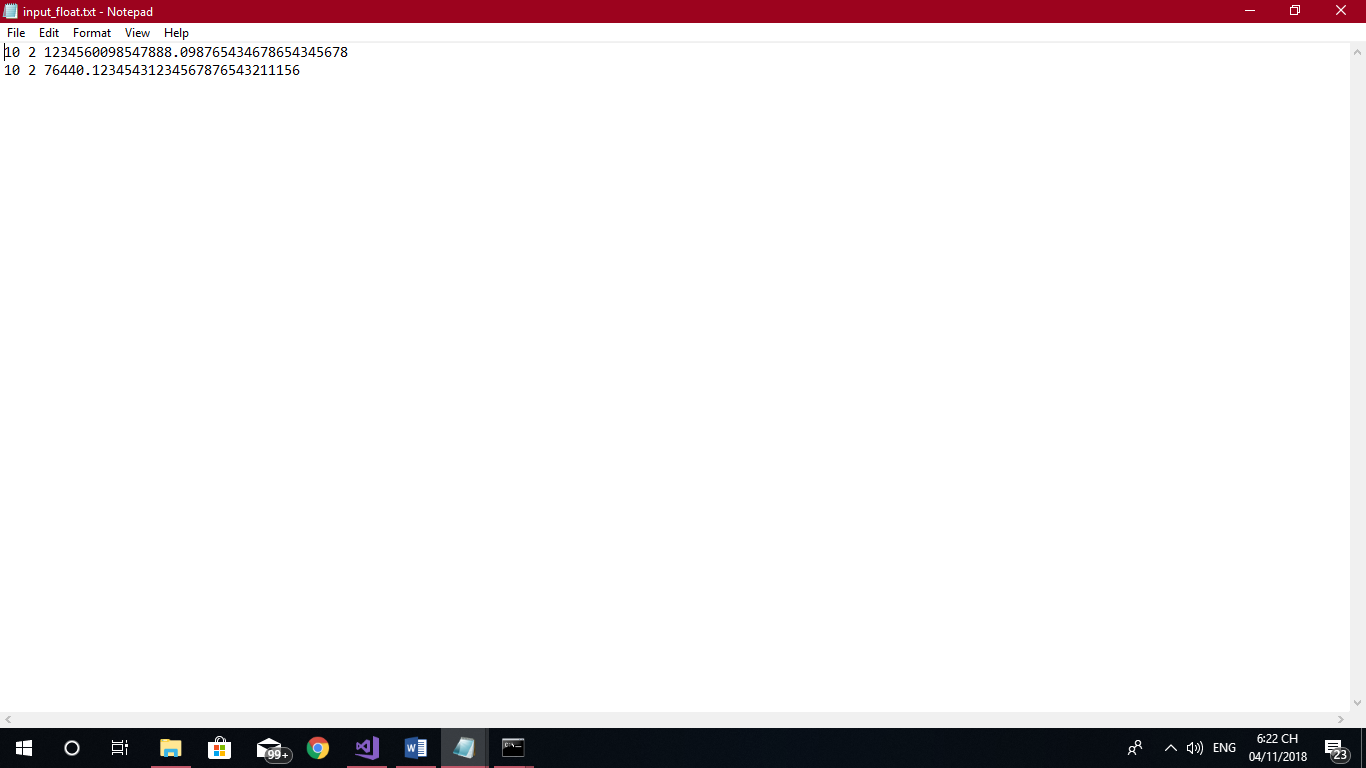
* Toán tử |, &, ^, <<, >>, ror, rol

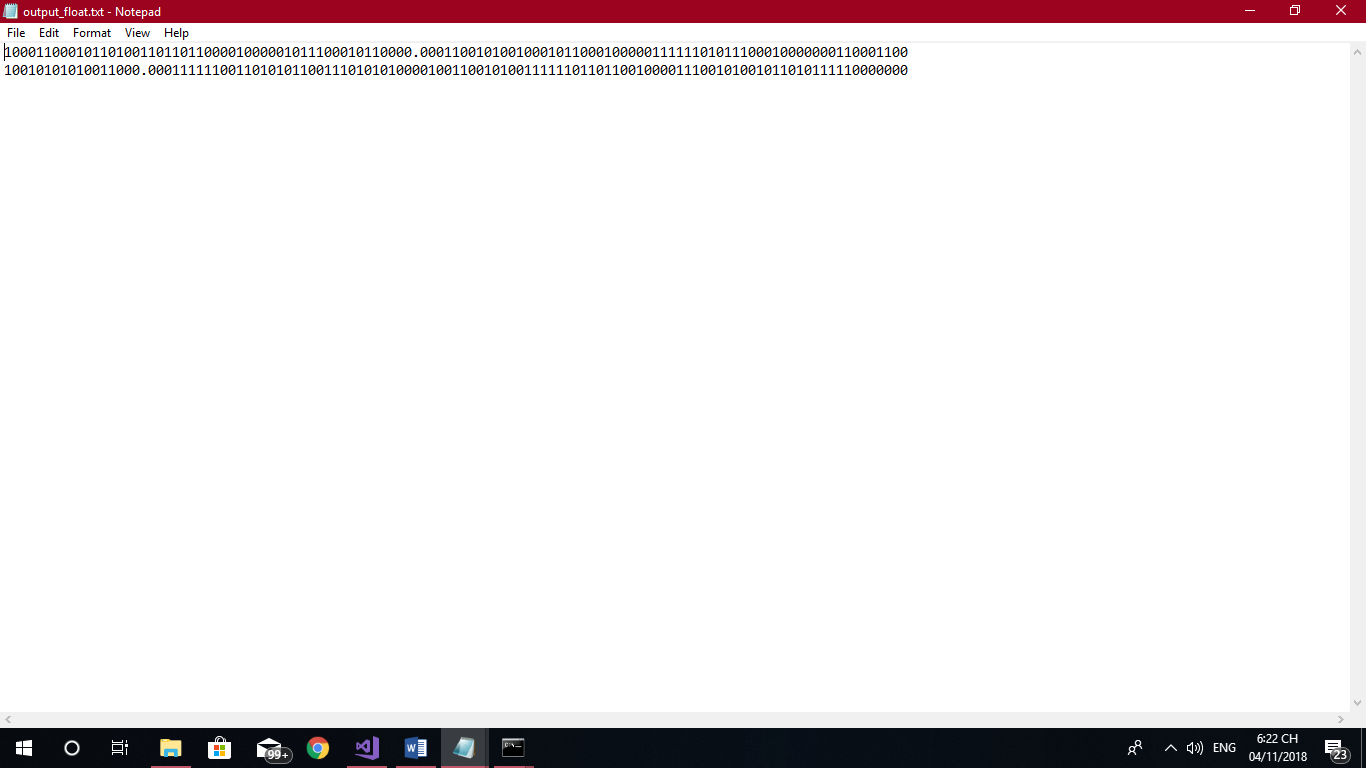




4.2 QFloat

* Chuyển đổi cơ số





1. Đánh giá mức độ hoàn thành:

* Hoàn thành tương đối đầy đủ các chức năng được yêu cầu.
* Còn chuyển đổi 2 to 10 của QFloat chưa làm được.
* Hoàn thành QInt:
* Convert:100%.
* Operator + - \* /: 100%.
* Toán tử logic “&”, “|”, “^”, “~”,” rol”, “ror”: 100%.
* Hoàn thành QFloat:
* Convert:
* 2->10: 50%.
* 10 -> 2: 100%.
* Hoàn thành toàn đồ án 95%.

1. Nguồn tài liệu tham khảo

* Bải slide bài giảng của các thầy trên lớp.
* <https://stackoverflow.com>
* <https://cplusplus.com>