**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----🙞🙜🕮🙞🙜----**

****

BÁO CÁO ĐỒ ÁN I

BIỂU DIỄN VÀ TÍNH TOÁN SỐ HỌC TRÊN MÁY TÍNH

Nhóm sinh viên thực hiện:

Nguyễn Phượng Vỹ (\*) 1712929

Vũ Thành Đạt 1612087

Lê Đức Thuận 1712805

Đặng Thái Gia Thuận 1712173

(\*): Nhóm trưởng.

TP.Hồ Chí Minh, ngày 26 tháng 03 năm 2019

1. **Nội dung các hàm:**
2. **Phần số nguyên lớn:**
3. ***Báo cáo của 1712929 - Nguyễn Phượng Vỹ***

* **Phạm vi biểu diễn của kiểu dữ liệu QInt hệ thập phân là từ -2127-1**

**đến 2127-1**

* **Các hàm chính và miêu tả từng chức năng của hàm chính:**

1. Hàm Nhập**:** void ScanQInt(QInt &x,int &heso)

- Cho phép nhập dữ liệu vào chương trình ứng với các hệ số tương ứng là 2, 10, 16. Nếu hệ số khác 2, 10, 16 thì chương trình sẽ thông báo hệ số không hợp lệ và thoát khỏi chương trình.

- Hàm mặc định dữ liệu nhập luôn luôn đúng với từng hệ số. Nghĩa là:

+ Đối với hệ số bằng 2 thì trong chuỗi dữ liệu chỉ chứa số 1 và 0.

+ Đối với hệ số bằng 10 thì trong chuỗi dữ liệu chỉ chứa các chữ số từ 0 -> 9 không có chứa chữ cái.

+ Đối với hệ số bằng 16 thì trong chuỗi dữ liệu chỉ chứa các chữ số từ 0 -> 9 và có chứa các chữ cái từ A -> F.

1. Hàm Xuất**:** void PrintQInt(QInt x,int heso)

Cho phép xuất dữ liệu của chương trình ứng với các hệ số tương ứng là 2, 10, 16 ra màn hình. Nếu hệ số khác 2, 10, 16 thì chương trình sẽ thông báo hệ số không hợp lệ và thoát khỏi chương trình.

1. Hàm chuyển đổi số QInt thập phân sang nhị phân:string Q\_DecToBin(QInt x): Hàm này dùng để chuyển đổi số QInt hệ thập phân sang hệ nhị phân .
2. Hàm chuyển đổi số QInt nhị phân sang thập phân**:** QInt Q\_BinToDec(string a): Hàm này dùng để chuyển đổi số QInt hệ nhị phân sang hệ thập phân.
3. Hàm chuyển đổi số QInt nhị phân sang thập lục phân:string Q\_BinToHex(string a)

Hàm này chuyển đổi số QInt hệ nhị phân sang hệ thập lục phân.

1. Hàm chuyển đổi số QInt thập phân sang thập lục phân**:** string Q\_DecToHex(QInt x): Hàm này chuyển đổi số QInt hệ thập phân sang hệ thập lục phân.
2. QInt operator+(QInt &t): Hàm này sẽ trả về một QInt là tổng của 2 QInt.
3. QInt operator-(QInt &t): Hàm này sẽ trả về một QInt là hiệu của 2 QInt.

* **Trình bày các bước làm của từng hàm chính:**

1. Hàm Nhập**:**void ScanQInt(QInt &x,int &heso)

B1: Nhập hệ số và dữ liệu (kiểu chuỗi string) vào chương trình.

B2: Chuyển dữ liệu (kiểu chuỗi string) đã nhập thành kiểu QInt hệ nhị phân.

1. Hàm Xuất**:** void PrintQInt(QInt x,int heso)

B1: Nhập hệ số muốn biểu diễn của dữ liệu vào chương trình.

B2: Chuyển dữ liệu kiểu QInt hệ nhị phân thành kiểu QInt với hệ số muốn biểu diễn tương ứng ra màn hình của chương trình.

1. Hàm chuyển đổi số QInt thập phân sang nhị phân***:*** string Q\_DecToBin(QInt x)

B1: Chuyển đổi số QInt thập phân sang dữ liệu kiểu string hệ thập phân (hàm hỗ trợ).

B2: Chuyển đồi dữ liệu kiểu string hệ thập phân sang chuỗi string hệ nhị phân (hàm hỗ trợ).

B3 Chuyển đổi chuỗi string hệ nhị phân sang dữ liệu kiểu QInt hệ nhị phân (hàm hỗ trợ ).

1. Hàm chuyển đổi số QInt nhị phân sang thập phân**:** QInt Q\_BinToDec(string a)

B1: Khởi tạo dữ liệu kiểu QInt hệ thập phân, cho các giá trị đều bằng 0.

B2: Chuyển kiểu dữ liệu string hệ nhị phân sang kiểu dữ liệu QInt hệ nhị phân (hàm hỗ trợ).

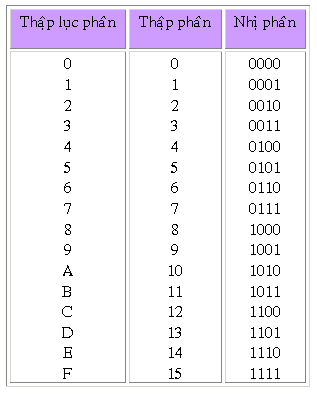
B3: Chuyển đổi số QInt nhị phân sang QInt thập phân.

1. Hàm chuyển đổi số QInt nhị phân sang thập lục phân: string Q\_BinToHex (string a)

B1: Nhóm từng bộ 4 bit trong QInt hệ nhị phần ( nhóm từ trái qua phải vì độ dài chuỗi đã được mặc định bằng 128)

B2: So sánh bộ 4 bit trong chuỗi với ký số tương ứng trong hệ thập lục phân

\* Sử dụng bảng sau đây:



B3: Chèn các kí tự tương ứng vào sau chuỗi hệ thập lục phân.

1. Hàm chuyển đổi số QInt thập phân sang thập lục phân**:** string Q\_DecToHex(QInt x)

B1: Chuyển đổi số QInt thập phân sang QInt nhị phân.

B2: Chuyển đổi số QInt nhị phân sang QInt thập lục phân.

1. Hàm toán tử QInt operator+(QInt &t)

(QInt) A + B =KQ

\*Lấy giá trị bit từ vị trí cuối đến vị trí đầu: i= 127 -> 0

- TH : nếu giá trị nhớ = 0:

+ TH1: Nếu giá trị bit A và B tại vị trí i đều bằng 0 thì giá trị bit KQ tại vị trí i bằng 0, bit nhớ bằng 0.

+ TH2: Nếu giá trị bit A và B tại vị trí i đều bằng 1 thì giá trị bit KQ tại vị trí i bằng 0, bit nhớ bằng 1

+ Các TH còn lại thì giá trị bit KQ tại vị trí i bằng 1, bit nhớ bằng 0

- TH : nếu giá trị nhớ = 1:

+ TH1: Nếu giá trị bit A và B tại vị trí i đều bằng 0 thì giá trị bit KQ tại vị trí i bằng 1, bit nhớ bằng 0.

+ TH2: Nếu giá trị bit A và B tại vị trí i đều bằng 1 thì giá trị bit KQ tại vị trí i bằng 1, bit nhớ bằng 1

+ Các TH còn lại thì giá trị bit KQ tại vị trí i bằng 0, bit nhớ bằng 1

1. Hàm toán tử QInt operator-(QInt &t)

(QInt) A - B =KQ

\*Lấy giá trị bit từ vị trí cuối đến vị trí đầu: i= 127 -> 0

B1: Tìm bù 2 của B

b1: Tìm bù 1 của B: duyệt B, nếu giá trị bit B tại vị trí i bằng 1 thì chuyển giá trị bit B tại vị trí i bằng 0, ngược lại nếu giá trị bit B tại vị trí i bằng 0 thì chuyển giá trị bit B tại vị trí i bằng 1.

b2: Cộng bù 1 của B với 1 (Kiểu QInt hệ nhị phân)

B2: Cộng bit A với bù 2 của bit B

* **Một số hàm hỗ trợ chính và miêu tả từng chức năng của hàm hỗ trợ:**

1. void BinToQInt(string &a, QInt &x): Chuyển chuỗi hệ nhị phân sang QInt
2. void QIntToBin(QInt &x,string &a): Chuyển QInt sang chuỗi hệ nhị phân
3. void DecToQInt(string &a, QInt &x): Chuyển chuỗi hệ thập phân sang QInt
4. void QIntToDec(QInt &x, string &a): Chuyển QInt sang chuỗi hệ thập phân
5. void HexToQInt(string &a, QInt &x): Chuyển chuỗi hệ số 16 phân sang QInt
6. void QIntToHex(QInt &x, string &a): Chuyển QInt sang chuỗi hệ số 16
7. string div\_2(string a): Hàm trả ra một chuỗi là kết quả của phép chia lấy nguyên cho 2
8. string Add(string a, string b): hàm cộng 2 chuỗi string
9. string mul\_2(string): Hàm nhân 2 cho chuỗi string
10. string mu\_2(int): x2
11. string DecToBin(string a): Chuyển số hệ 10 được đưa vào dạng chuỗi trả ra một chuỗi là chuỗi số nhị phân tương ứng
12. string BinToDec(string a): Chuyển số hệ 2 được đưa vào dạng chuỗi trả ra một chuỗi là chuỗi số thập phân tương ứng
13. string HexToBin(string a): Chuyển số hệ 16 được đưa vào dạng chuỗi trả ra một chuỗi là chuỗi số nhị phân tương ứng

* **Trình bày các bước làm của một số hàm hỗ trợ:**

1. void BinToQInt(string &a, QInt &x):

B1: Chuẩn hóa độ dài chuỗi nhị phân a

B2: Sao chép giá trị bit tại từng vị trí của chuỗi số nhị phân a vào QInt hệ nhị phân

1. void QIntToBin(QInt &x,string &a): Sao chép giá trị bit tại từng vị trí của QInt hệ nhị phân vào chuỗi số nhị phân a
2. void DecToQInt(string &a, QInt &x):

B1: Chuyển hệ 10 qua hệ nhị phân.

B2: Chuyển chuỗi hệ nhị phân sang QInt

1. void QIntToDec(QInt &x, string &a):

B1: Chuyển QInt sang chuỗi hệ nhị phân

B2: Chuyển chuỗi hệ nhị phân qua hệ 10

1. void HexToQInt(string &a, QInt &x):

B1: Chuyển hệ 16 qua hệ nhị phân.

B2: Chuyển chuỗi hệ nhị phân sang QInt

1. void QIntToHex(QInt &x, string &a):

B1: Chuyển QInt sang chuỗi hệ nhị phân

B2: Chuyển chuỗi hệ nhị phân qua hệ 16

1. string div\_2(string a):

B1: Chuyển chuỗi string sang kiểu int

B2: Chia chuỗi cho 2

B3: Xóa các số 0 ở trước kết quả

1. string Add(string a, string b):

B1: Chuẩn hóa độ dài 2 chuỗi a và b sao cho độ dài 2 chuỗi bằng nhau bằng cách chèn thêm kí tự 0 vào chuỗi có độ dài nhỏ hơn

B2: Duyệt và cộng

\* Tính tổng hàng đơn vị

-Tính tổng từng đôi một tại vị trí cuối cùng

-Gán phần đơn vị vào chuỗi kết quả

-Sau đó ta lấy lại phần chục bằng cách chia chuỗi cho 10 và lấy phần nguyên

-Lặp lại cho đến khi độ dài chuỗi bằng 0

1. string mul\_2(string a): vì a\*2=a+a nên ta gọi lại hàm add(a,a)
2. string mu\_2(int):

B1: Kiểm tra mũ số n có bằng 0 hay không, nếu bằng 0 trả về giá trị 1

B2: Lặp lại n-1 lần phép nhân 2 cho kết quả

1. string DecToBin(string a):

B1: Kiểm tra số âm hay dương

B2:

\* Nếu là số dương: Chuyển số nguyên dương qua nhị phân

\* Nếu là số âm:

B1: Chuyển số nguyên dương qua nhị phân

B2: Lấy bù 2 của dãy nhị phân

\*\* Cách chuyển số nguyên dương qua nhị phân:

Lấy cơ số 10 chia cho 2

- Số dư đưa vào kết quả

- Số nguyên đem chia tiếp cho 2

- Lặp lại cho đến khi số nguyên bằng 0

1. string BinToDec(string a):

Ý tưởng: Bắt đầu từ bên trái, nhân đôi kết quả, rồi cộng con số bên cạnh cho đến khi không còn con số nào nữa.  
+ B1: Kiểm tra bit đầu của số nhị phân. Nếu là 0 thì là số dương, 1 là số âm.  
+ B2: Chuyển sang hệ thập phân.

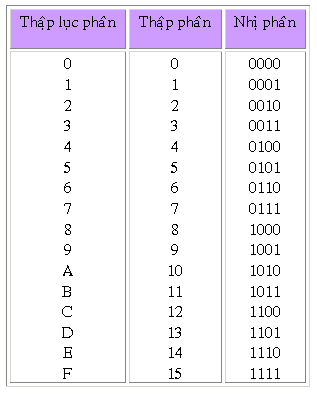
\* Nếu là số dương:  
Số thập phân kết quả sẽ là tổng các tích của kí tự nhị phân x 2 lũy thừa vị trí.  
\* Nếu là số âm:

B2: Ta lấy bù 2 của số nhị phân và tính toán như số dương  
B3: Thêm dấu “-” vào trước kết quả

1. string HexToBin(string a):

B1: So sánh từng kí tự trong chuỗi thập lục phân với ký số tương ứng trong hệ nhị phân

\* Sử dụng bảng sau đây:



B2: Nối chuỗi các kí tự tương ứng vào sau chuỗi hệ nhị phân.

1. ***Báo cáo của 1712805 – Lê Đức Thuận***

* **Các hàm chính và miêu tả từng chức năng của hàm chính:**

QInt operator >>(int): A dịch phải n bit

QInt operator <<(int): A dịch trái n bit

QInt rotateLeft(int): A xoay trái n bit

QInt rotateRight(int): A xoay phải N bit

QInt operator &(QInt): trả 1 biến QInt là kết quả của A&B

QInt operator |(QInt): trả 1 biến QInt là kết quả của A|B

QInt operator ^(QInt): trả 1 biến QInt là kết quả của A^B

QInt operator~(): trả 1 biến QInt là kết quả của ~A

bool operator>(QInt): so sánh xem A có lớn hơn B không, nếu có trả về 1, còn lại trả về 0

bool operator<(QInt): so sánh xem A có nhỏ hơn B không, nếu có trả về 1, còn lại trả về 0

bool operator>=(QInt): so sánh xem A có lớn hơn hoặc bằng B không, nếu có trả về 1, còn lại trả về 0

bool operator<=(QInt): so sánh xem A có nhỏ hơn hoặc bằng B không, nếu có trả về 1, còn lại trả về 0

bool operator ==(QInt): so sánh xem A có bằng B không,nếu có trả về 1, còn lại trả về 0

QInt & operator =(const QInt &): gán A bằng B

bool checkIfZero(): kiểm tra số trên có bằng 0 hay không

int findBitN(QInt): tìm kiếm vị trí bit 1 đầu tiên của 1 số dương

QInt operator \*(QInt ): tính toán kết quả A\*B trên bit và trả về QInt

QInt operator /(QInt ): tính toán kết quả A\*B trên bit và trả về QInt

* **Giải thích hướng làm các hàm quan trọng**

1. Các hàm logic &,|,~ ^

Các phép toán logic trên có 1 đặc điểm là có thể thao tác trên bit ở các biến int,unsigned int,... Vậy nên không nhất thiết phải thao tác trên bit, chỉ cần thao tác trực tiếp trên từng phần tử unsigned int trong mảng bằng cách chạy vòng lặp là được

2. Hàm dịch trái và phải (operator <<(int) và operator >>(int))

Áp dụng kỹ thuật như xử lý chuỗi tạo 1 biến QInt với tất cả các bit bằng 0 bằng constructor. Sau đó gán dần các bit từ (0->128 -số bít dịch)với độ lệch n bit .

\* Khó khăn, do kiểu QInt đc lưu theo kiểu từ trái sang phải ( VD: giá trị của của số 1 sẽ được lưu dưới dạng 1000,... trong QInt, 2 sẽ là 0100,...) nên khi dịch bit sẽ ngược với cách dịch bình thường. Dịch phải sẽ phải gán số về bên trái và ngược lại

3. Hàm Xoay trái và phải( rotateLeft(int) và rotateRight(int))

Cách làm như hàm dịch trái và phải bình thường, tuy nhiên sẽ gán thêm các bit số (128 - số bit dịch -> 128) vào chuỗi tạm đó

4.2 Hàm so sánh lớn và bé (operator>(QInt) và operator<(QInt))

Các bit được lưu có 1 số quy luật như sau

Trừ trường hợp số dương luôn lớn hơn số âm

Nếu là số dương thì số nào có bit 1 xuất hiện trước là số lớn hơn và ngược lại

A>B => -A < -B=> bù 2 của A < bù 2 của B

5. Hàm lớn hơn hoặc bằng và bé hơn hoặc bằng(operator>=(QInt) và operator<=(QInt))

Áp dụng quy tắc A< B => B >=A

1. Hàm xác định bằng

Cho chạy từ 0 tới 4, nếu các phần tử int của A mà khác B thì trả về 0, còn nếu tất cả bằng nhau thì trả về 1

Toán tử gán

Gán như các hàm gán bình thường (bool operator ==(QInt))

1. Hàm nhân QInt operator \*(QInt )

Áp dụng quy tắc như trong slide bài giảng

Giả sử ta có A = M\*Q

Bước 1: kiểm tra xem 1 trong 2 số có bằng 0 hay không, nếu có trả về 0.

Bước 2: kiểm tra độ dài từng chuỗi, lấy độ dài lớn nhất là độ dài chung cho cả 2 chuỗi

Bước 3: tạo 2 biến pos và neg để lưu giá trị âm và dương của hàm, có thể bỏ qua bước này nếu sử dụng phép cộng trừ

Bước 4: dịch trái M sang n+1 bit và dịch trái A sang 1 bit ( cho 1 biến QInt trờ thành dạng AQQ-1)

Bước 5: cho K bằng độ dài số lớn nhất và xét vòng lặp

Bước 5.1: xét trường hợp

Nếu Q0Q-1 = 01 thì A = A+M

Nếu Q0Q-1 = 10 thì A = A-M

Bước 5.2: dịch phải số học A

Bước 5.3: gán k =k-1, Nếu k>0 thì quay lại bước 5

Bước 6: gán giá trị A thuộc AQQ-1 cho 1 biến khác ( do trong lúc A+- B có khả năng tràn số trên các bit trước đó nên không thể lấy biến cũ được

Bước 7: dịch phải biến trên ( do giá trị nhận về là thuộc khoảng AQ)

Bước 8: kiểm tra số trên có âm hay không ( nếu biến trên âm phải cho tất cả các biến trước đó về 1) rồi trả về kết quả.

\*Nhược điểm

Lãng phí bit ( do phải dịch cả 2 biến trên sang 2\*n+1 vị trí và cho độ dài 2 số bằng nhau nên chỉ có thể tính toán được những số nhỏ hơn 2^63 bit 126 bit lưu kết quả + 1 bit nhớ, do đó những phép nhân số khá lớn và số nhỏ sẽ không thực hiện được.

1. Hàm chia operator /(QInt )

Áp dụng thuật toán trong slide ta có

Bước 1: kiểm tra tử và mẫu có bằng 0 hay không, nếu tử =0 thì trả về 0 còn mẫu bằng 0 thì ngừng xử lý

Bước 2: gán 1 biến temp = 0;

Bước 3: kiểm tra xem 2 số trên có nhỏ hon 0 hay không, nếu có thì thay đổi gí trị temp và chuyển số đó về dạng bù 2.

Bước 4: kiểm tra độ dài 2 bit và lấy độ dài lớn hơn và đặt k = số bit lớn hơn

Bước 5: cho biến trên thành dạng AQ ( cách làm giống thuật toán nhân)

Bước 6: khi K > 0

Bước 6.1: dịch trái A

Bước 6.1: cho A = A-M.

Bước 6.2: xét A

+Nếu A âm thì A = A+M và gán Q0 = 0

+Nếu A dương thì gán Q0=1

Bước 6.3: giảm k, nếu k>0 thì quay lại bước 6

Bước 7: kiểm tra biến tạm, nếu có 1 trong 2 số là 0 thì trả về giá trị bù 2 của kết quả, nếu không thì trả về kết quả bình thường

\* Khó khăn: Đọc hiểu sai ý của slide dẫn đến nhận ra 1 số điểm sai của thuật toán liên quan đến dấu, do hạn hẹp thời gian nên chỉ xét trường hợp dương chia dương và sử dụng tới bù 2 khi có số âm, việc làm này rất tốn bộ nhớ và thời gian xử lý nếu số trên là 1 số lớn .

1. **Phần số chấm động chính xác cao:**

***Báo cáo của 1612087 - Vũ Thành Đạt***

* **Các hàm chính và miêu tả từng chức năng của hàm chính:**

1. Hàm nhập: void ScanQfloat();

* Đọc dữ liệu từ file INPUT.txt
* File INPUT.txt gồm 2 phần type và value.

+ Type là kiểu dữ liệu nhập vào (chấm động hoặc số thực)

+ Value là giá trị của số nhập vào (string)

1. Hàm xuất: void PrintQfloat(fstream &fs);

* Ghi kết quả vào file OUTPUT.txt

1. void XuLy(): nếu value là số nhị phân thì cập nhật lại thành thập phân.
2. Qfloat BinToDec(): hàm chuyển đổi số Qfloat dạng chấm đ sang thập phân.
3. Qfloat DecToBinFloat(): hàm chuyển đổi số Qfloat thập phân sang nhị phân.
4. void SetBit(int bit, int i): đưa bit vào vị trí i
5. bool GetBit(int pos): lấy bit thứ i
6. string GetBits(int, int): lấy bit từ vị trí này tới vị trí kia trong data;
7. bool is0(): kiểm tra bằng 0
8. bool isINF(): hàm kiểm tra số vô cực
9. bool isNaN(): hàm kiểm tra số báo lỗi
10. bool isDenormalizedNumber(): hàm kiểm tra số không chuẩn
11. string GetValue(): hàm lấy giá trị của số thực
12. string mul(const string &num, int val): hàm nhân chuỗi số dạng thập phân cho một số val nào đó
13. string BinToDec\_fra(string num): hàm chuyển đổi phần thập phân của chuỗi số thực dạng nhị phân sang thập phân
14. string DecToBin\_fra(string num): hàm chuyển phần thập phân của số thực sang chấm động
15. bool isZero(const string &num): kiểm tra chuỗi bằng 0
16. string IntToBin\_EX(int): chuyển phần số mũ sang nhị phân
17. Qfloat BinToBinCD(int, string, string): chuyển chuỗi nhị phân về chuẩn hóa chấm động

* **Trình bày các bước làm của từng hàm chính:**

1. Hàm nhập: void ScanQfloat():

Bước 1: đọc file INPUT.txt

Bước 2: từ file lấy được 2 dữ liệu : type và value của Qfloat.

Bước 3: xác định kiểu dữ liệu vừa được nhập vào ( 10 hoặc 2 ) rồi xử lý

* type =10, Chuyển đổi phần value sang chấm động.
* type=2, lúc này value có 3 phần riêng biệt (s,e,m) được phân biệt bằng khoảng trắng. Cắt 3 phần rồi chuyển đổi chuỗi nhị phân sang dạng chuẩn chấm động.

1. Hàm xuất: void PrintQfloat(fstream &fs):

Xác định kiểu dữ liệu của Qfloat

* Type=10 : ghi phần chấm động vào file.
* Type=2 : ghi phần value vào file.

1. void XuLy():

Khi Qloat có type =2 thì chuyển value từ dạng nhị phân về thập phân.

1. Qfloat BinToDec():

* Bước 1: Kiểm tra các trường hợp lỗi, số vô cùng.
* Bước 2:Thực hiện tính toán tính phần thực và phần ảo
* Bước 3: Hoàn thiện số thực
* Bước 4: trả về giá trị Qfloat;

1. Qfloat DecToBinFloat():

* Bước 1: Tách phần số thực ra làm 2 phần nguyên và thập phân
* Bước 2: biến đổi phần thực và thập phân sang dạng biểu diễn nhị phân
* Bước 3: gọi hàm xử lí số nhị phân sang dạng chuẩn chấm động
* Bước 4: trả về giá trị Qfloat;

1. void SetBit(int bit, int i): đưa bit vào vị trí i
2. bool GetBit(int pos): lấy bit thứ i
3. string GetBits(int, int): lấy bit từ vị trí này tới vị trí kia trong data
4. bool is0(): kiểm tra bằng 0

Kiểm tra tất cả các bit đều = 0 thì số bằng 0

1. bool isINF(): hàm kiểm tra số vô cực

Nếu phần mũ tất cả =1 thì là số vô cực

1. bool isNaN(): hàm kiểm tra số báo lỗi

Nếu phần mũ tất cả =1 và không phải là số vô cực thì lỗi

1. bool isDenormalizedNumber():

Nếu phần mũ tất cả =0 và giá trị của nó không phải 0 thì là số sai định dạng.

1. string GetValue():

Hàm trả về giá trị value của Qfloat.

1. string mul(const string &num, int val):

Thực hiện nhân 1 số nguyên lớn cho một số int.

* Bước 1: tạo 1 biến kiểu string để lưu kết quả nhân
* Bước 2: nhân lần lượt từ hàng đơn vị đến hàng cuối cùng.
* Bước 3: đảo lại chuỗi. Vì khi thực hiện nhân lưu từ cuối lên đầu.
* Bước 4: trả về giá trị string.

1. string BinToDec\_fra(string num):

Thực hiện lần lượt gán các số nhị phân của phần thập phân sau đó nhân chúng cho 5. Sẽ được số thập phân đúng.

1. string DecToBin\_fra(string num):

* Bước 1: sử dụng vòng lặp do while điều kiện lặp số ban đầu =0
* Bước 2: nhân 2 num
* Bước 3: xác định xem sau khi nhân phần thập phân có lớn hơn 1 hay không. Nếu có thì gán vào bit =1, không thì bit =0
* Bước 4: trả về chuỗi nhị phân đã tính được.

1. bool isZero(const string &num):

Kiểm tra 1 chuỗi có phải =0 hay không.

1. string IntToBin\_EX(int):

Biến đổi số int về đạng nhị phân trả lại dưới dạng chuỗi.

1. Qfloat BinToBinCD(int, string, string):

Chuẩn hóa chấm động. bit đầu tiên là dấu.

Bit 1-15 là phần mũ.

Bit 16-127 là phần giá trị.

1. **Xử lý file:**

***Báo cáo của 1712173 – Đặng Thái Gia Thuận***

* **Các hàm chính và miêu tả chức năng của hàm chính**

1. void DocVaXuLyFileQInt(string &in, string &out): đọc và xử lý file số nguyên
2. void DocVaXuLyFileQfloat(Qfloat \*FL): đọc và xử lý file số thực

* **Một số hàm hỗ trợ chính và miêu tả từng chức năng của hàm hỗ trợ:**

1. void Convert(ifstream &In, ofstream &Out, string heso1, string heso2, string data):

Chuyển đổi số QInt thập phân sang nhị phân: bool \* DecToBin (QInt x)

Chuyển đổi số QInt nhị phân sang thập phân: QInt BinToDec(bool \*bit)  
Chuyển đổi số QInt nhị phân sang thập lục phân: char \*BinToHex(bool \*bit)  
Chuyển đổi số QInt thập phân sang thập lục phân: char \*DecToHex(QInt x)

1. bool KiemTraDau(string a):

Cho phép kiểm tra các phép tính “+”, “-” , “\*”, “/”, các phép so sánh và gán: “<”, “>”, “==”, “<=”, “>=”, “=”.

Các toán tử: AND “&”, OR “|”, XOR “^”, NOT “~”  
Các toán tử: dịch trái “<<”, dịch phải “>>”, xoay trái: “rol”, xoay phải: “ror”

1. bool KiemTraKyTu(char a):

Kiểm tra xem kí tự là 0->9 hay A->F

1. void ToanTuHaiNgoi(ifstream &In, ofstream &Out, string heso, string data1, string dau, string data2):

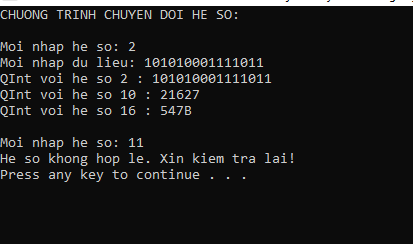
Cho phép thực hiện các phép tính và ghi kết quả ra file

1. void ToanTuMotNgoi():

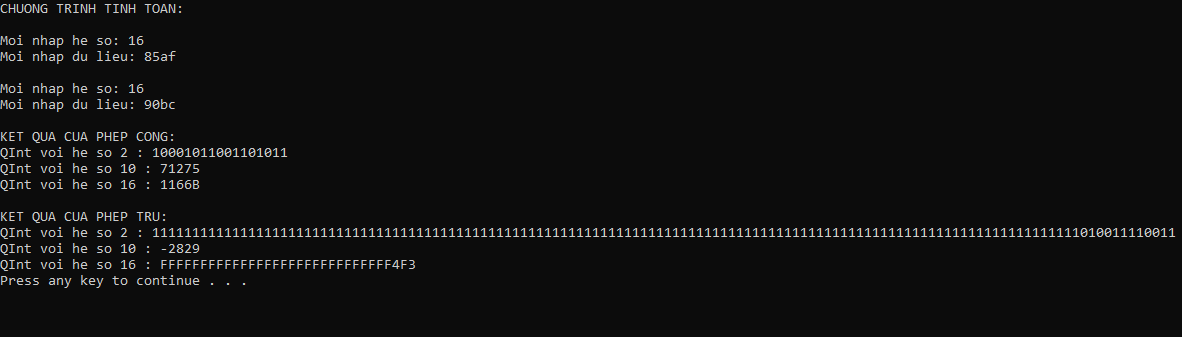
Cho phép thực hiện các chuyển đổi và ghi kết quả ra file

1. **Kết quả bài làm:**
2. **Số nguyên lớn**
3. **Kết quả của 1712929 - Nguyễn Phượng Vỹ**

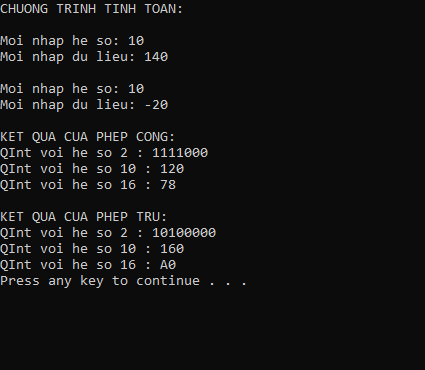
Test l:

****

Test 2:

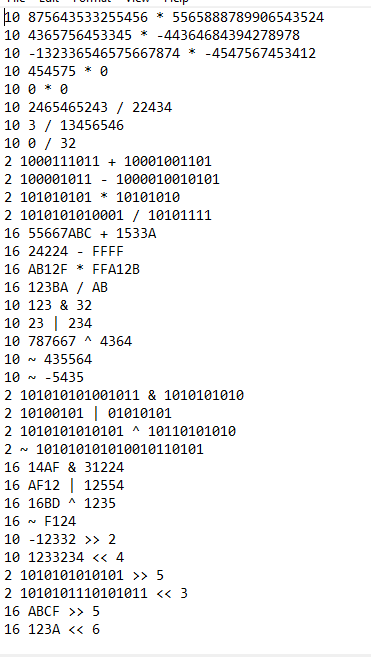


Test 3:

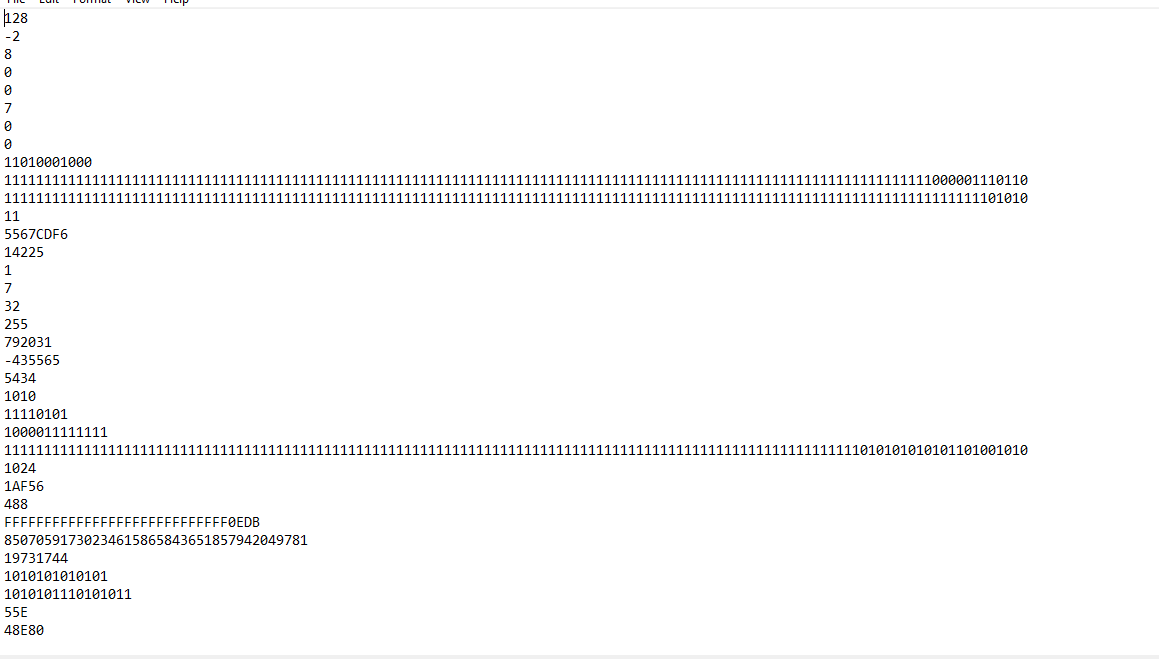


1. **Kết quả của 1712805 - Lê Đức Thuận**

**File input:**



**File output:**



1. **Phần số chấm động chính xác cao:**

**Kết quả của 1612087 - Vũ Thành Đạt**

|  |  |
| --- | --- |
| INPUT.txt | OUTPUT.txt |
| 2 0 10001000 01101100001000000000000 | 728.250 |
| 2 0 111111111111111 10000000000000000000000 | isNAN |
| 2 0 111111111111111 0000000000000000000000 | isINF |
| 2 1 111111111111111 0000000000000000000000 | -isINF |
| 2 1 00 001100000000000000000000 | is Denormalized Number |
| 2 1 10000010 10010100000000000000000 | -12.6250 |
| 10 728.25 | 0 000000010001000 01101100001000000000000000000000000000.. |
| 10 -12.625 | 1 000000010000010 10010100000000000000000000000000000000.. |

1. **Đánh giá các thành viên:**

***1. Nguyễn Phượng Vỹ (nhóm trưởng):***

🟑 Công việc:

* Chuyển đổi số QInt thập phân sang nhị phân: bool \* DecToBin(QInt x)
* Chuyển đổi số QInt nhị phân sang thập phân: QInt BinToDec(bool \*bit)
* Chuyển đổi số QInt nhị phân sang thập lục phân: char \*BinToHex(bool \*bit)
* Chuyển đổi số QInt thập phân sang thập lục phân: char \*DecToHex(QInt x)
* Các operator toán tử : “+”, “-”
* Ghép code phần số nguyên lớn

🟑 Mức độ hoàn thành: 10/10

***2. Lê Đức Thuận:***

🟑 Công việc:

* Nhập: void ScanQInt(QInt &x)
* Xuất: void PrintQInt(QInt x)
* Các operator toán tử : “\*”, “/”
* Các toán tử so sánh và gán: “<”, “>”, “==”, “<=”, “>=”, “=”
* Các toán tử: AND “&”, OR “|”, XOR “^”, NOT “~”
* Các toán tử: dịch trái “<<”, dịch phải “>>”, xoay trái: “rol”, xoay phải: “ror”
* Ghép code phần số nguyên lớn

🟑 Mức độ hoàn thành: 10/10

***3. Vũ Thành Đạt:***

🟑 Công việc:

* Hàm Nhập: void ScanQfloat(Qfloat &x)
* Hàm xuất: void PrintQfloat(Qfloat x)
* Hàm chuyển đổi số Qfloat nhị phân sang thập phân: Qfloat BinToDec(bool \*bit)
* Hàm chuyển đổi số Qfloat thập phân sang nhị phân bool \*DecToBin(Qfloat x)

🟑 Mức độ hoàn thành: 10/10

*4. Đặng Thái Gia Thuận:*

🟑 Công việc:

* Viết báo cáo
* Tạo testcase
* Viết hàm xử lí file

🟑 Mức độ hoàn thành: 8.5/10

1. **Các nguồn tài liệu tham khảo:**

<https://github.com/phuongnghi1704/cancer_project/blob/6c74e9886609027987a72a877e5df10c9a06c7f7/test/test/QInt.cpp>

<https://github.com/conglethanh/Bignum_QuadInt_QuadFloat>