TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 1

Bộ môn: Kiến trúc máy tính và Hợp ngữ

Giảng viên bộ môn: Lê Viết Long

Nhóm thực hiện:

1612143: Trần Phan Phú Duy

1612222: Nguyễn Quang Huân

1612243: Nguyễn Thị Thanh Hương

1612285: Dương Văn Khang

Hồ Chí Minh, ngày 25 tháng 03 năm 2017

Nội dung

A. Phân chia công việc và đánh giá mức độ hoàn thành.

Tên sinh viên	Công việc được giao	Đánh giá mức độ hoàn thành
Trần Phan Phú Duy	- Chuyển từ QInt sang Chuỗi (chuỗi số nguyên lớn) - Chuyển Hexa -> Decimal	**Hoàn thành tốt**
Nguyễn Quang Huân	- Các thao tác đọc – ghi File - Đọc tham số dòng lệnh ở dạng Comandline	**Hoàn thành tốt**
Nguyễn Thị Thanh Hương	- Các Toán tử trên QInt	**Hoàn thành tốt**
Dương Văn Khang	- Chuyển từ chuỗi số nguyên lớn sang QInt - Chuyển đổi giữa các hệ cơ số	**Hoàn thành tốt**

B. Tổng quan Đồ án

I. Phân tích các chức năng

1. Đưa chuỗi số nguyên lớn vào QInt và Xuất ra một số kiểu QInt

Xây dựng cấu trúc QInt

- QInt có độ dài 16 byte (128 bit) nên ta có cấu trúc như sau:

```
struct QInt
{
    int data[4] = { 0 };
}
```

Thao tác Đưa chuỗi số nguyên lớn vào QInt và Xuất ra một số kiểu QInt

	Void ScanQInt(QInt &x)
Chức năng	Đưa chuỗi số nguyên lớn vào QInt
Ý tưởng	+ Dùng biến sign để xác định chuỗi nhập vào là số âm hay số dương (1: âm, 0: dương) + Xét thứ tự data để bắt đầu đưa bit vào là a=3, nhét vào bit thứ k=0. + Chia chuỗi S cho 2 và lấy phần dư -> bit, đưa bit vào vị trí thứ k của data thứ a: x.data[a] = x.data[a] | (bit<<k) + Chuỗi S gán lại bằng thương của chuỗi S với 2. + Nếu k=32 (đã đưa đủ bit vào data thứ a). Thực hiện gán lại k=0 và a = a – 1 + Lặp lại các công việc trên cho đến khi phần tử trong chuỗi S chỉ còn '0' + Xét biến sign. Nếu biến sign mang giá trị 1, nghĩa là số ban đầu là số âm thì tiến hành đảo tất cả bit trong Qint rồi cộng với bit 1.

Mã nguồn	`void ScanQInt(QInt &x)`
	`{`
	`  string a;`
	`  cout << "Nhap chuoi so nguyen lon: ";`
	`  getline(cin, a);`
	`  StringOfQIntToQInt(x, a);`
	`}`
	Ý tưởng trên được thể hiện ở trong hàm StringOfQIntToQInt
	(Xem chi tiết trong Source code)

	Void PrintQInt(QInt x)
Chức năng	In ra số QInt (dãy 128bit) tương ứng từ chuỗi số nguyên lớn
Ý tưởng	Xét từng phần tử data thứ j (0<=j<4) Mỗi phần tử data thứ j đều có 32 bit nên công việc cần làm là: dịch từng bit qua phải (31-i) vị trí (0<=i<32), sau đó AND(&) 1 để xuất bit đó ra `cout << ((a.data[j] >> (31 - i)) & 1);`

Mã nguồn	`void PrintQInt(QInt a)`
	`  int dem = 0;`
	`  for (int j = 0; j < 4; j++) {`
	`  for (int i = 0; i < 32; i++) {`
	`  cout << ((a.data[j] >> (31 - i)) & 1);`

VD: Trên số nguyên lớn có dấu và không dấu

- ***Số nguyên lớn không dấu:***

Nhap chuoi so nguyen lon: 3245687955
Day 128 bit trong QInt:
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 11000001011101010100010010010011

- ***Số nguyên lớn không dấu:***

Nhap chuoi so nguyen lon: -3245687955
Day 128 bit trong QInt:

11111111111111111111111111111111 11111111111111111111111111111111
11111111111111111111111111111111 00111110100010101011101101101101

2. Chuyển đổi giữa các hệ cơ số

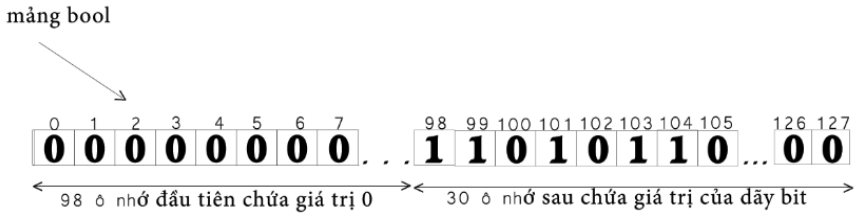
a. bool* DecToBin(QInt x):

Chức năng: Hàm chuyển đổi QInt thập phân sang nhị phân
Mô tả:

+ Đây là hàm chuyển từ kiểu dữ liệu QInt sang một mảng bool. Mỗi phần tử trong mảng bool là một bit có giá trị là 0 hoặc là 1.
+ Đối số truyền vào sẽ là một biến QInt và hàm trả về một mảng bool chứa các giá trị bit của biến QInt tương ứng.

Ví dụ: Ta có giá trị nguyên của QInt x là 900000000, đưa biến x vào hàm DecToBin, ta sẽ nhận được một mảng bool có giá trị là: 11010110100100111010010000000.

❖ ***Lưu ý****: mảng bool này có kích thước mặc định là 128 ô nhớ, phần dãy bit lưu trong mảng bool sẽ được lưu từ cuối mảng về đầu. Những ô nhớ đầu mảng nếu không có giá trị sẽ mặt định là 0.*



b. QInt BinToDec(bool *bit):

Chức năng: Hàm chuyển đổi số QInt nhị phân sang thập phân
Mô tả:

+ Đây là hàm chuyển từ một mảng bool chứa các giá trị bit sang kiểu dữ liệu QInt thập phân.
+ Đối số truyền vào là một mảng bool chứa các giá trị bit và hàm trả ra một biến QInt chứa giá trị tương ứng.

Ví dụ: Ta có một mảng bit có giá trị: 10010010010010010010111 1 . Sau khi đưa mảng bit vào hàm BinToDec(), ta sẽ nhận được biến QInt có giá trị nguyên là 9586991.

c. char *BinToHex(bool *bit):

Chức năng: Hàm chuyển đổi số QInt nhị phân sang thập lục phân

Mô tả:

+ Đây là hàm chuyển từ một mảng bool chứa các giá trị bit sang một chuỗi chứa giá trị của hệ số 16 tương ứng với mảng bit truyền vào.
+ Đối số truyền vào là một mảng bool chứa các giá trị bit và hàm trả về một chuỗi chứa giá trị của hệ số 16 tương ứng với giá trị của mảng bool truyền vào.

Ví dụ: Ta có một mảng bit có giá trị: 10010010010010010010111 1. Sau khi đưa mảng bit vào hàm BinToHex(), ta nhận được chuỗi có giá trị "92492F"

d. char *DecToHex(QInt x):

Chức năng: Hàm chuyển đổi số QInt thập phân sang thập lục phân

Mô tả:

+ Đây là hàm chuyển từ một kiểu dữ liệu QInt sang chuỗi chuỗi có giá trị của hệ 16 tương ứng với biến QInt truyền vào.
+ Đối số truyền vào là một biến QInt và hàm trả về một chuỗi chứa giá trị của hệ 16 tương ứng với giá trị của QInt.

Ví dụ: Ta có biến x thuộc kiểu QInt có giá trị nguyên là: 11112850. Sau khi đưa biến x vào hàm DecToHex(), ta nhận được chuỗi có giá trị "A99192".

❖ **Lưu ý**: Tất cả những hàm trên đều sử dụng được cho trường hợp số nguyên có dấu.

3. Các phép toán trên QInt

a. Toán tử AND(&), OR(|), XOR(^), NOT(~), SHIL(<<), SHIR(>>)

<table>
<tr><th>Operator</th><th>Thực hiện</th></tr>
<tr><td>AND(&)</td><td>
- Input: QInt a, QInt b
- Output: QInt c (với c = a&b)
- Xét lần lượt data thứ i của hai số QInt a, b gọi là a.data[i], b.data[i]. Sau đó thực hiện phép AND(&) giữa a.data[i], b.data[i] với (0<=i<=3).
- Tuân thủ quy luật: AND dùng tắt bit.

<pre>QInt operator &(QInt a, QInt b)
{
    QInt c;
    for (int i = 3; i >= 0; i--)
    {
        c.data[i] = a.data[i] & b.data[i];
    }
    return c;
}</pre>
</td></tr>
<tr><td>OR(|)</td><td>
- Input: QInt a, QInt b
- Output: QInt c (với c = a|b)
- Xét lần lượt data thứ i của hai số QInt a, b gọi là a.data[i], b.data[i]. Sau đó thực hiện phép OR(|) giữa a.data[i], b.data[i] với (0<=i<=3).
- Tuân thủ quy luật: OR dùng bật bit.

<pre>QInt operator |(QInt a, QInt b)
{
    QInt c;
    for (int i = 3; i >= 0; i--)
    {
        c.data[i] = a.data[i] | b.data[i];
    }
    return c;
}</pre>
</td></tr>
<tr><td>XOR(^)</td><td>
- Input: QInt a, QInt b
- Output: QInt c (với c = a^b)
- Xét lần lượt data thứ i của hai số QInt a, b gọi là a.data[i], b.data[i]. Sau đó thực hiện phép XOR(^) giữa a.data[i], b.data[i] với (0<=i<=3).
- Tuân thủ quy luật: 2 bit giống nhau, khi XOR nhau đều ra bit 0.

<pre>QInt operator^(QInt a, QInt b)
{
    QInt c;
    for (int i = 3; i >= 0; i--) {
        c.data[i] = a.data[i] ^ b.data[i];
    }
    return c;
}</pre>
</td></tr>
</table>

<table>
<tr>
<td>NOT(~)</td>
<td>
- Input: QInt a
- Output: QInt c (với c = ~a)
- Xét lần lượt data thứ i của hai số QInt a, b gọi là a.data[i]. Sau đó thực hiện phép NOT(~) giữa a.data[i] với (0<=i<=3).
- Tuân thủ quy luật: NOT đảo bit.
<pre>
QInt operator~(QInt a)
{
    QInt c;
    for (int i = 3; i >= 0; i--)
    {
        c.data[i] = ~a.data[i];
    }
    return c;
}
</pre>
</td>
</tr>
<tr>
<td>SHIL(<<)</td>
<td>
- Input: QInt a, int k (k>0, k là số bit dịch trái)
- Output: QInt c (với c = a << k)
<pre>
QInt operator <<(QInt a, int k)
{
    //Thực hiện dịch trái k bit
    for (int i = 1; i <= k; i++)
    {
        //Do dịch trái nên phải chạy từ data[0]->data[3]
        for (int j = 0; j <= 3; j++)
        {
            //Lần lượt dịch trái 1 bit Data thứ j
            a.data[j] <<= 1;
            // Xét Data nằm bên phải Data[j], gọi là: rData
            if (j + 1 <= 3) {
                //Lấy ra bit đầu tiên (bit trái nhất) của rData đó
                int temp = a.data[j + 1] >> 31;
                bool bit = temp & 1;
    //Sau đó OR với data <=> dịch sang trái trong khuôn khổ Data[j]
                a.data[j] = a.data[j] | bit;
            }
        }
    }
    return a;
}
</pre>
</td>
</tr>
</table>

SHIR(>>)

- Input: QInt a, int k (k>0, k là số bit dịch phải)
- Output: QInt c (với c = a >>k)

```
for (int i = 1; i <= k; i++)
    {
        //Do dịch phải nên phải chạy từ Data[3] lùi về Data[0]
        for (int j = 3; j >= 0; j--)
        {
            //Lần lượt dịch phải 1 bit trong Data thứ j
            a.data[j] >>= 1;
            //Thao tác bật bit nếu QInt có dấu
            //Xét Data nằm bên trái Data thứ j, gọi là:  lData
            if (j - 1 >= 0)
            {
                // Lấy ra bit cuối cùng (phải nhất) của lData đó.
                bit = a.data[j - 1] & 1;
                //Cho bit đó đè lên vị trí bit đầu tiên của data thứ j
                //[....]
            }
        }
    }
```

❖ **Lưu ý**: Phép dịch phải có tính luận lý nên cần xét thêm:

```
//Thao tác bật bit đầu tiên lên 1 nếu QInt có dấu
            bool bit;
            bit = a.data[0] >> 127 & 1;
            if (j == 0)
            {
                if (bit == 1) {
                    int temp = pow(2, 31);
                    a.data[0] = a.data[0] | temp;
                }
            }
```

Ví dụ 1: Cho 2 số QInt A, B:

A = 00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000011010011

B = 00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000001111

i) C = A & B =

00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000000000011

ii) C = A | B =

00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000011011111

iii)C = A ^ B =
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000011011100
iv)C = ~A =
11111111111111111111111111111111 11111111111111111111111111111111
11111111111111111111111111111111 11111111111111111111111100101100

Ví dụ 2: Cho QInt A (là số không dấu)

A = 00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000011010011
i) C = A << 1 =
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000110100110
ii) C = A >> 1 =
00000000000000000000000000000000 00000000000000000000000000000000
00000000000000000000000000000000 00000000000000000000000001101001

Ví dụ 3: Cho QInt A (là số có dấu)

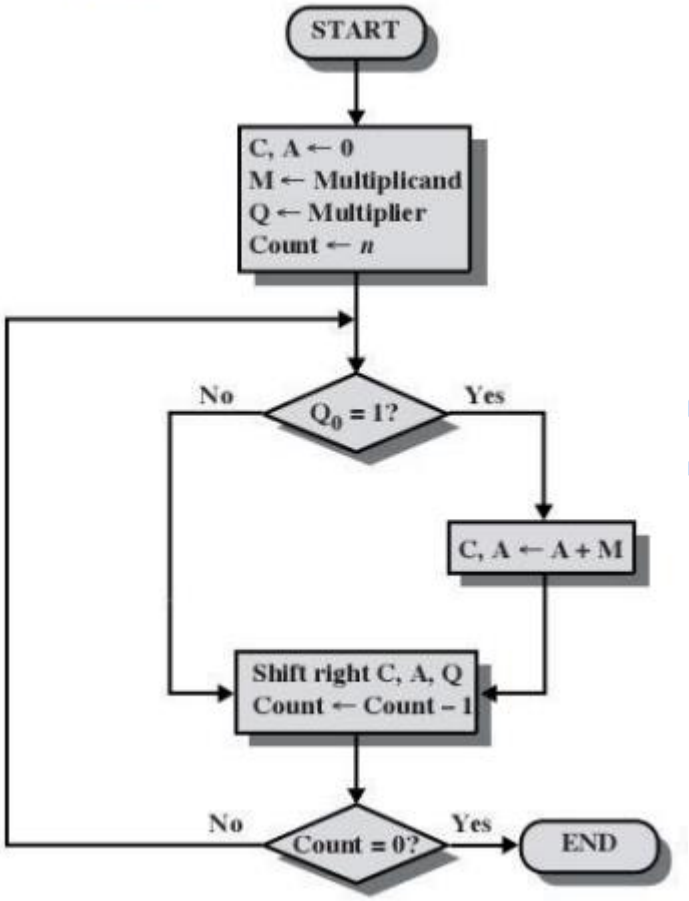
A = 11111111111111111111111111111111 11111111111111111111111111111111
11111111111111111111111111111111 11111111111111111111111100101101
i) C = A << 1 =
11111111111111111111111111111111 11111111111111111111111111111111
11111111111111111111111111111111 11111111111111111111111001011010
ii) C = A >> 1 =
11111111111111111111111111111111 11111111111111111111111111111111
11111111111111111111111111111111 11111111111111111111111110010110

b. Toán tử CỘNG(+), TRỪ(-), NHÂN(*), CHIA(/, %)

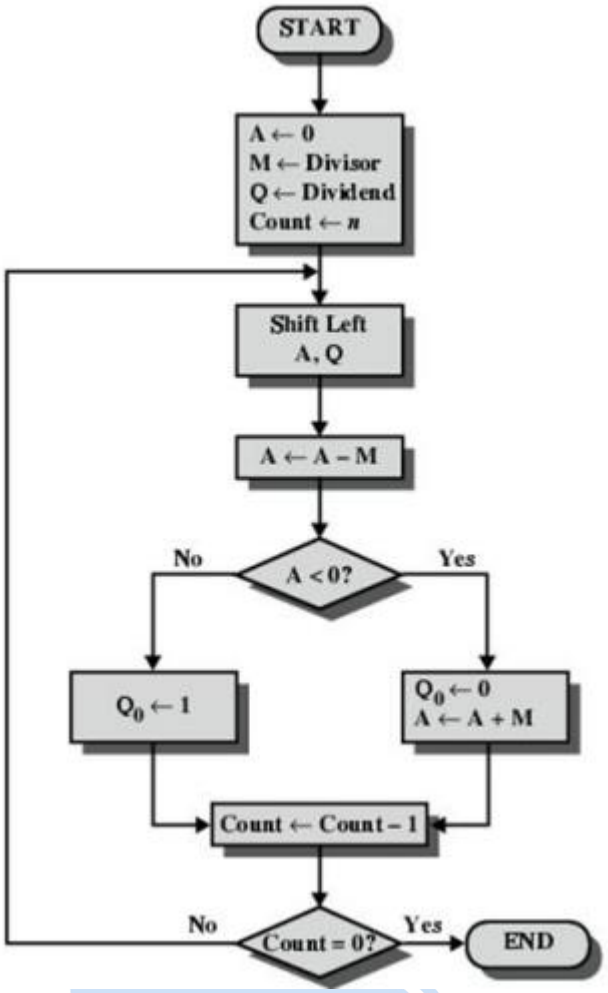
Operator	Thực hiện
+	- Input: QInt a, QInt b - Output: QInt c (với c = a + b) - Thực hiện làm phép toán trên chuỗi bit + Chuyển a, b sang chuỗi bit (string s1, string s2) + Cộng trực tiếp từng phần tử trên hai chuỗi s1 và s2, lưu vào chuỗi tổng s + Đổi chuỗi s sang QInt => Kết quả
-	- Input: QInt a, QInt b - Output: QInt c (với c = a - b) a – b = a + (bù2 của b) - Thực hiện tương tự phép cộng QInt operator-(QInt a, QInt b) return a + TwoSComplement(b);

Thực hiện theo **thuật toán BOOTH**:

*



Thuật toán Booth (nguồn: Slide bài giảng của thầy Phạm Tuấn Sơn)

/, %	Thực hiện theo **thuật toán chia:**  Thuật toán Booth (nguồn: Slide bài giảng của thầy Phạm Tuấn Sơn)

❖ **Lưu ý**: Tất cả các Operator(+, - , *, /) đều áp dụng được lên số nguyên có dấu.

II. Đánh giá mức độ hoàn thành của từng chức năng:

	Chức năng	Đánh giá mức độ - Ghi chú
1	**Đưa chuỗi số nguyên lớn vào QInt và Xuất ra một số kiểu QInt:** ▪ void ScanQInt(QInt &x) void PrintQInt(QInt x)	• Mức độ: 100% • Xử lý được trường hợp số có dấu và số không dấu.
2	**Chuyển đổi giữa các hệ cơ số**: ▪ bool * DecToBin (QInt x) ▪ QInt BinToDec(bool *bit) ▪ char *BinToHex(bool *bit) ▪ char *DecToHex(QInt x)	• Mức độ: 100% • Chuyển đổi qua lại giữa các hệ cơ số chính xác như App. Calculator.
3	**Tính toán trên QInt thông qua các toán tử:** ▪ AND "&", OR "|", XOR "^", NOT "~", SHIL(<<), SHIR(>>) ▪ "+", "-" , "*", "/"	• Mức độ: 95% • Tính toán được trên số nguyên không dấu và có dấu. • Những trường hợp dữ liệu nhập không hợp lệ, dẫn đến overflow khi thực hiện tính toán => Chưa khắc phục được.

III. Hướng dẫn sử dụng chương trình

Bước 1: Chuẩn bị tập tin input

i. Định dạng : .txt
ii. Tên: input
iii. Quy ước:

➢ Phép tính toán trên các hệ số :

*<hệ số i>*khoảng cách*<số a>*khoảng cách*<kí hiệu toán tử>*khoảng cách*<số b>*

Riêng phép NOT (kí hiệu toán tử : ~) sẽ khuyết khoảng cách<*số b*>

Trong đó:

- <*hệ số i*> là hệ số biểu diễn của <*số a*>,<*số b*> và kết quả của phép tính, nhận các giá trị sau:

Hệ số	Nhị phân	Thập phân	Thập lục phân
i	2	10	16

- <*số a*>,<*số b*> là 2 toán hạng tham gia vào phép tính
  Hai số phải được biểu diễn theo <*hệ số i*>
- <*kí hiệu toán tử*> là kí hiệu đại diện cho phép tính cần thực hiện, nhận các giá trị sau:

Toán tử	Cộng	Trừ	Nhân	Chia	Chia lấy dư	AND	OR	XOR	NOT
Kí hiệu	+	-	*	/	%	&	|	^	~

Kết quả: Trả về kết quả phép tính biểu diễn theo <*hệ số i*>
Ví dụ:

```
2 1111100011101010111 + 01101110110111
```

(Phép cộng 2 số nhị phân 1111100011101010111 và 01101110110111)

➢ Phép dịch chuyển bit

<*hệ số i*>khoảng cách<*số x*>khoảng cách<*kí hiệu*>khoảng cách<*số bits dịch chuyển*>

Trong đó:

- <*hệ số i*> là hệ số biểu diễn của <*số x*> và kết quả của phép dịch chuyển

Hệ số	Nhị phân	Thập phân	Thập lục phân
i	2	10	16

- <*số x*> là số cần dịch chuyển bit, được biểu diễn theo <*hệ số i*>

Phép dịch bit	Sang trái	Sang phải
Kí hiệu	<<	>>

- <*số bits dịch chuyển*> nhận giá trị nguyên đại diện cho số bits cần dịch chuyển

Kết quả: Trả về kết quả phép dịch bit biểu diễn theo <*hệ số i*>

Ví dụ:

```
10 5678 >> 2
```

(Thực hiện dịch sang phải 2 bits số hệ 10 có giá trị 5678)

➢ Phép chuyển đổi giữa các hệ số

*<hệ số i>*khoảng cách*<hệ số j>*khoảng cách*<số x>*

Trong đó :

- *<hệ số i>* là hệ số ban đầu
- *<hệ số j>* là hệ số kết quả

Hệ số	Nhị phân	Thập phân	Thập lục phân
i,j	2	10	16

- *<số i>* là số cần chuyển đổi hệ số, được biểu diễn theo *<hệ số i>*

Kết quả : Trả về kết quả là dạng biểu diễn của *<số x>* theo *<hệ số j>*

Ví dụ:

```
10 2 8793278316383117319
```

(Chuyển số hệ thập phân có giá trị 8793278316383117319 sang hệ nhị phân)

❖ **File mẫu:**

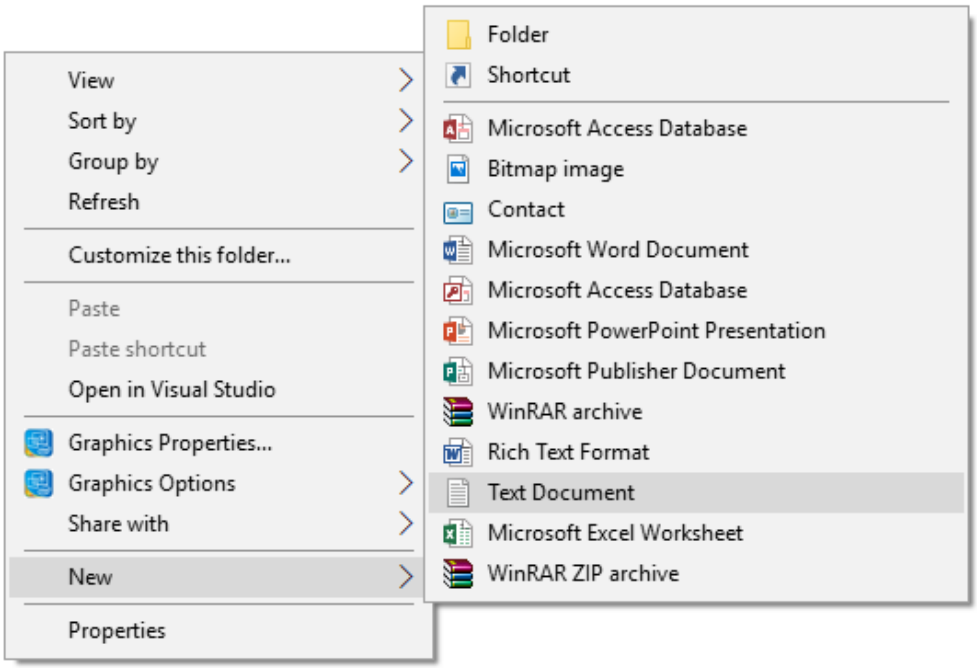
input.txt - Notepad

File Edit Format View Help

```
2 1001010100111 + 11110111010111
10 45611 - 77548
16 156EF * 59CF
2 1010111111 % 111101111
10 12345 & 56789
16 3D | 4E
2 11011 ^ 00100
10 6632 ~
16 13D >> 5
2 10001111 << 9
2 10 10101111111
2 16 10101111111
10 2 510
10 16 510
16 10 12A
16 2 12A
```

❖ **Thực hiện cài đặt tập tin input:**

+ Truy cập vào thư mục chứa tập tin thực thi
+ (Địa chỉ thư mục có dạng ...*\1612285_1612143_1612243_1612222\ Release*)
+ Nhấp chuột phải vào cửa sổ thư mục Release -> New -> Text Document



+ Nhập tên tập tin là *input* -> nhấn Enter. Ta thu được tập tin *input.txt*
+ Nhấp đúp vào tập tin *input.txt*, nhập dữ liệu cho tập tin theo quy ước đã nêu trên
+ Khi đã nhập dữ liệu xong, nhấn tổ hợp phím Ctrl + S để lưu dữ liệu.
  => Công việc cài đặt tập tin input hoàn thành.

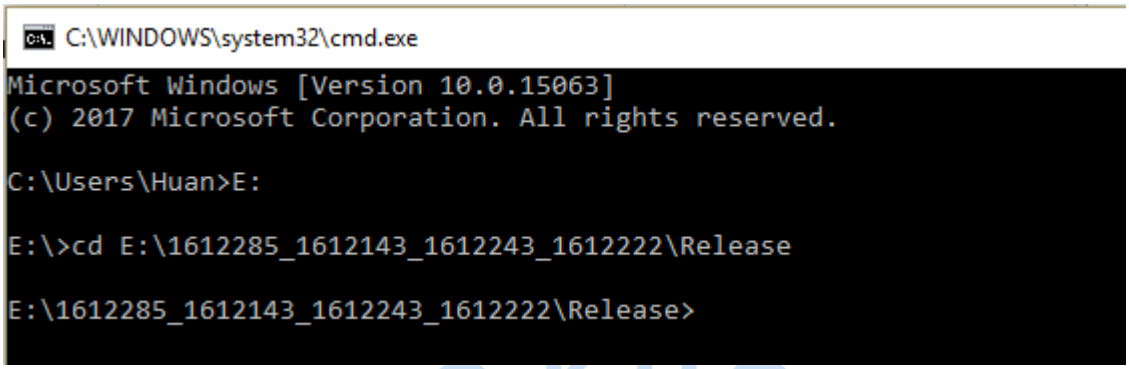
Bước 2: Khởi động và chạy chương trình

+ Bật giao diện Command Line của Window
  ⇨ Nhấn tổ hợp phím **Window + R** -> nhập **cmd.exe** -> nhấn **Enter**
+ Dùng lệnh **cd** để di chuyển đến thư mục chứa tập tin thực thi theo cú pháp :
  ⇨ **cd**khoảng cách*<đường dẫn đến tập tin exe>*

(Sau khi giải nén, tập tin exe nằm trong thư mục tên Release, thuộc thư mục cha tên 1612285_1612143_1612243_1612222. Do đó đường dẫn sẽ có dạng: *...\1612285_1612143_1612243_1612222\Release*)

❖ Trong trường hợp, tập tin exe không nằm ở ổ đĩa mặc định của hệ thống, ta gõ
*<tên ổ đĩa chứa tập tin exe>***:**
rồi nhấn Enter và thực hiện lệnh **cd** như trên

Ví dụ:

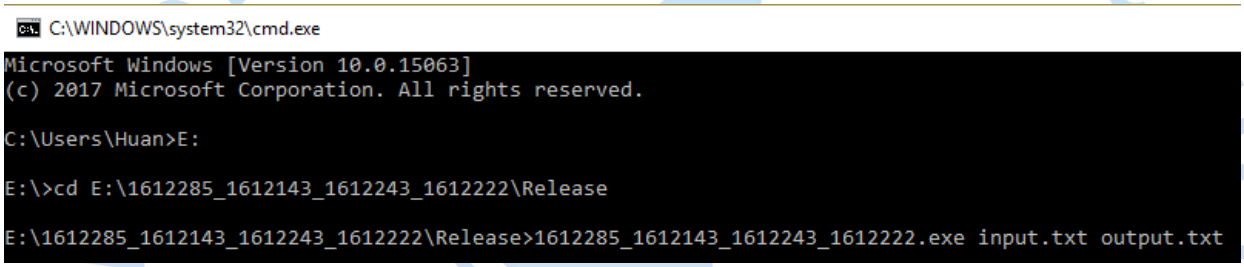


Gõ lệnh theo cú pháp sau

*<tên tập tin thực thi>*khoảng cách*<tên tập tin input>*khoảng cách*<tên tập tin output>*

Trong trường hợp này, lệnh trên sẽ như sau

*1612285_1612143_1612243_1612222.exe*khoảng cách*input.txt*khoảng cách*output.txt*



Nhấn Enter và chương trình sẽ được thực thi

- Thông báo lỗi:

- Loi doc FILE! : Không tìm thấy tập tin input -> Khắc phục:
  - + Kiểm tra thư mục Release đã tồn tại tập tin input.txt chưa
  - + Kiểm tra tên file input.txt đã đúng chưa
- '…exe' is not recognized as an internal or external command, operable program or batch file : Không tìm thấy tập tin thực thi .exe trong thư mục hiện hành

-> Khắc phục: Kiểm tra lại tên của tập tin thực thi phải là 1612285_1612143_1612243_1612222.exe

Bước 3: Nhận kết quả trả về từ chương trình thông qua tập tin output

- Truy cập vào thư mục chưa tập tin thực thi 1612285_1612143_1612243_1612222.exe
- Địa chỉ thư mục có dạng *...\1612285_1612143_1612243_1612222\Release*
- Nhấp đúp vào tập tin output.txt

- Nội dung tập tin output.txt:

Kết quả của từng thao tác ở tập tin input.txt sẽ được ghi vào tập tin output.txt theo thứ tự tương ứng