ĐỒ ÁN 1

BIỂU DIỄN VÀ TÍNH TOÁN SỐ NGUYÊN LỚN

Quy định chung

1) Thời gian làm bài: 4 tuần

2) Làm nhóm: tối đa 3sv/nhóm, sv có thể làm cá nhân nếu muốn (nhưng không khuyến kích)

3) Thư mục nộp bài là MSSV1_MSSV2_MSSV3 có chứa các thư mục sau:

- Source: chứa source code chương trình
- Report: chứa báo cáo
- Relase: binary file

4) Các bài chép source lẫn nhau: 0 điểm thực hành

5) Email liên hệ GVHDTH: Nguyễn Thanh Quân (ntquan@fit.hcmus.edu.vn)

6) Deadline: 22h55 ngày 16/04/2017 (không giải quyết nộp trễ với bất kỳ lý do)

7) Vấn đáp: 22 hoặc 23/04/2017

Cách thức nộp bài

- Nộp bài trực tiếp trên moodle và đúng hạn. Không giải quyết dời deadline trong mọi tình huống (trừ trường hợp khách quan).

- Tên file: **MSSV1_MSSV2_MSSV3.rar** (Với MSSV1 < MSSV2 < MSSV3)

- Tổ chức thư mục nộp bài gồm:

1. **Report**: chứa báo cáo **MSSV1_MSSV2_MSSV3.pdf** trình bày:

- Thông tin thành viên nhóm, bảng phân công công việc.
- Nêu rõ môi trường lập trình.
- Nêu rõ ý tưởng để thiết kế và thực hiện đồ án. Cho biết phạm vi biểu diễn của các kiểu dữ liệu đã thiết kế.
- Chạy kiểm tra và chụp hình, chú thích từng chức năng của chương trình minh họa.
- Nêu rõ chức năng làm được, và chưa được.

- Đánh giá mức độ hoàn thành theo tỉ lệ phần trăm (%) của toàn bộ đồ án.
- Các nguồn tài liệu tham khảo.
- Lưu ý: không chép source code vào báo cáo.

2. **Release**: chứa file thực thi của chương trình (*.exe) biên dịch ở chế độ Release.
3. **Source**: chứa source code của chương trình, yêu cầu nộp cả project và bỏ thư mục Debug và các file database phát sinh (*.db). *Nhóm nào chỉ nộp file *.cpp và *.h và không biên dịch được thì bị 0 điểm.* Lưu ý: trước mỗi hàm, sinh viên ghi chú mô tả rõ chức năng của hàm trong source code.

- Nếu làm không đúng những yêu cầu trên, bài làm sẽ không được chấm.

Yêu cầu

Mô tả về Số nguyên lớn

Kiểu dữ liệu số nguyên lớn có dấu gọi là QInt có độ lớn 16 byte gồm một số chức năng:

(a) Chuyển đổi số QInt từ hệ thập phân sang hệ nhị phân và ngược lại

(b) Chuyển đổi số QInt từ hệ nhị phân sang hệ thập lục phân và ngược lại

(c) Các operator=, operator+, operator-, operator*, operator/

(d) Các toán tử AND "&", OR "|", XOR "^", NOT "~"

(e) Các toán tử: dịch trái "<<", dịch phải ">>"

(f) Các phép xoay trái "rol", xoay phải "ror" mỗi lần xoay chỉ xử lý cho đúng 1 bit, không xử lý cho trường hợp tổng quát xoay k bit

3. Chương trình kiểm tra

- Chương trình được chấm tự động nên sinh viên cần tuân theo đúng cấu trúc được mô tả như bên dưới. Nên tham khảo các chức năng ở chế độ Programmer của Caculator để hiểu rõ hơn về chương trình phải xử lý những chức năng nào.

- Chương trình thực thi đọc tham số dòng lệnh ở dạng command line: <MSSV1_MSSV2_MSSV3.exe> <input.txt> <output.txt>.

- *Quy định cấu trúc tập tin Input:*

  - Gồm n dòng: không biết trước giá trị n

- Trong mỗi dòng, sẽ có chỉ thị p có các giá trị sau đây (các chỉ thị này được phân biệt với các toán hạng phía sau bằng đúng 1 ký tự khoảng trắng): ○ p = 2: thực hiện tính toán, xử lý ở chế độ Binary
    - ○ p = 10: thực hiện tính toán, xử lý ở chế độ Decimal
    - ○ p = 16: thực hiện tính toán, xử lý ở chế độ Hexa

- Nếu trong 1 dòng có 2 chỉ thị p1 và p2 (p1 và p2 cách nhau bởi đúng 1 khoảng trắng): có nghĩa là chuyển toán hạng ở chế độ p1 sang chế độ p2.
- Trong một dòng, các toán tử hai ngôi và toán hạng được cách nhau bởi đúng 1 khoảng trắng.
- Độ dài dãy bit nhị phân không cố định và không vượt quá 128 bit.
- Độ dài dãy hexa không cố định và không vượt quá 32 ký tự.
- Mặc định dữ liệu trên từng dòng đã có tính đúng đắn, không cần xét tính hợp lệ của dữ liệu đầu vào. Mỗi thành phần trên một dòng cách nhau bởi đúng 1 khoảng trắng.

- *Quy định cấu trúc tập tin Output:*

- Mỗi dòng tương ứng của tập tin Output là kết quả của phép tính toán hoặc chuyển đổi.
- Với trường hợp Input là dãy số nhị phân (không đủ 128 bit) hoặc dãy hexa (không đủ 32 ký tự) thì ở output xuất đủ và đúng số bit hoặc ký tự hexa cần thiết của kết quả (không cần thêm các bit 0 hoặc hex 0 vào trước để cho đủ) (xem giống như caculator trong Windows hiển thị kết quả).

Ví dụ trường hợp đọc vào số nguyên QInt:

Dòng	INPUT.TXT	OUTPUT.TXT
1	2 1111100011101010111 + 01101110110111	1111110001100001110
2	2 11011011 * 010101111	1001010110110101
3	2 10 01101010111110111111	438207
4	10 2 8793278316383117319	11110100000100000000000000001010001 11011001010010010000000000111
5	16 85AF + 90BC	1166B
6	10 5678 >> 2	1419
7	2 ror 0111000110101110	11100011010111
8	10 ~ -2000	1999

HƯỚNG DẪN ĐỒ ÁN 1

1. Xây dựng cấu trúc dữ liệu như thế nào?

Số nguyên lớn có độ lớn 16 bytes có nhiều cách để biểu diễn. Đoạn code bên dưới minh họa 6 khả năng (có thể còn nhiều hơn nữa) biểu diễn số QInt. Trong đó, trường hợp (1), (2) và (3) tốn rất nhiều byte để có thể biểu diễn được số nguyên QInt 16 byte. Vậy nên, ta hãy suy nghĩ đến khả năng (4), (5), (6) và đây sẽ là cách chúng ta nên tiếp cận vì đã sử dụng đúng số lượng byte theo yêu cầu của đề bài. Vấn đề cần phải suy nghĩ là ứng với từng cấu trúc dữ liệu cụ thể, ta sẽ phải đi xây dựng các giải thuật tương ứng để xử lý các yêu cầu thực tế.

```
class QInt
{
	private:
		char arrBits[128]; // (1) dùng mảng tĩnh 128 byte
		char *arrBits;
		// (2) dùng mảng động 128 byte
		bool *arrBits;
		// (3) tương tự 2.
		int arrayBits[4];
		// (4) dùng mảng 4 phần tử int (16 byte)
		long long arrayBits[2]; // (5) dùng mảng 2 phần tử long long
		__int64 arrayBits[2]; // (6) tương tự 5.
	public:
};
```

2. Một số kỹ thuật cơ bản bổ trợ liên quan đến đồ án

```
void main()
{
   //0. toán tử sizeof
   cout << sizeof(QInt) << endl;
   //1. miền giá trị của kiểu dữ liệu
   // số nguyên 1 byte: -128 -> +127
   // min = 1000 0000 = -128
   // max = 0111 1111 = +127
   //+1 = 1000 0000 = -128
   char x = 128;
   cout << (int)x << endl;
```

```
    //2. phép dịch bit (luận lý hay số học)?
    char x3 = -24;
    x3 = x3 >>1;
    cout << (int)x3 << endl;

    unsigned char x3 = -24; // dịch số học
    x3 = x3 >> 1;
    cout << (int)x3 << endl;

    //3. biểu diễn Binary, Hexa của kiểu dữ liệu
    // ta biết min = 1000 0000 0000....
    __int64 x1 = 0x8000000000000000;
    cout << x1 << endl;
    // ta biết max = 0111 1111 1111...
    __int64 x2 = 0x7FFFFFFFFFFFFFFF;
    cout << x2 << endl;

    //4. đổi từ hệ 10 --> hệ 2
    char x4 = -24;
    char a[8] = { 0 };
    unsigned char mask = 0x80;
    for (int i = 0; i < 8; i++)
    {
    if ((x4 & mask) != 0)
    a[i] = 1;
    mask = mask >> 1;
    }
    cout << endl;
    for (int i = 0; i < 8; i++)
    cout << a[i];

    //5. đổi ngược từ hệ 2 --> hệ 10
    string Bits = "10101100";
    char k = 0; //nếu khai báo unsigned char thì điều gì xảy ra?
    for (int i = 0; i < 8; i++)
    k = (k << 1) | (Bits[i] – 48);
    cout << endl << (int)k << endl;
}
```

3. <u>Giải thuật xây dựng nên như thế nào?</u>

Ta cần giải quyết lần lượt các vấn đề sau:

- Do số nguyên lớn có độ dài lớn hơn kiểu dữ liệu mặc định được hỗ trợ bởi NNLTr (NNLTr hỗ trợ số nguyên tối đa là 8 byte) nên không thể dùng hàm nhập/ xuất thông thường cho số nguyên lớn được. Gợi ý: để đọc giá trị thập phân chỉ có thể đọc dãy số bằng cách đọc theo chuỗi.
- Làm sao để chuyển đổi dãy số ở dạng chuỗi về kiểu QInt? Ta có thể giải quyết bằng thuật toán đã học: tính các bit bằng cách đem dãy số chia 2 rồi lấy phần dư, sau đó bật/ tắt bit thực sự của kiểu số nguyên QInt.
- Hàm SetBit bên dưới sẽ do các bạn tự định nghĩa giải thuật tương ứng với CTDL của nó.

Giải thuật chuyển đổi từ chuỗi số thập phân sang giá trị với kiểu dữ liệu QInt:

```
i = 128;
X: là chuỗi số thập phân;
a: là kiểu QInt; len = X.Length - 1;
while (X != "")
{
        bit = (X[len] - 48) % 2;
        SetBit(a, i, bit);      Div2(X);
        i--;
}
```

4. Phải nhìn rõ hơn về dãy bit được nối dài là gì?

__INTT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
mảng 2 phần tử	A[0]								A[1]							
vị trí bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Để dễ hình dung, ta xét một kiểu dữ liệu được định nghĩa mới là số nguyên dài 2 byte (có tên là __INTT) được định nghĩa là mảng char A[2] (với 2 phần tử là A[0] và A[1]). Khi nói đến bit thứ 8 của __INTT nghĩa là đang phải xử lý với bit thứ 0 của A[0].

Khi nói đến bit thứ 7 của __INTT nghĩa là đang phải xử lý với bit thứ 7 của A[1]. Do đó, khi định nghĩa __INTT là char A[2], thì phần tử A[1] được hiểu là số nguyên 1 byte có dấu, nghĩa là bit thứ 7 của A[1] sẽ là bit dấu, các thao tác dịch bit sẽ là thao tác dịch bit số học chứ không còn là thao tác dịch bit luận lý. Vậy ta có cần phải thay đổi một chút việc định nghĩa kiểu số __INTT như trên để đảm bảo những bit của A[1] được xử lý đúng theo ý ta mong muốn? Từ đó, bạn suy ngược lại cho đồ án này.

5. Có được tận dụng sự hỗ trợ của NNLTr?

Câu trả lời hoàn toàn được và khuyến khích tận dụng sức mạnh của ngôn ngữ lập trình (gồm CTDL và hàm có sẵn) để làm đồ án đỡ mất sức hơn, nghĩa là được sử dụng kiểu dữ liệu, các toán tử, các phép toán thao tác trên bit mà NNLTr có hỗ trợ.

6. Câu hỏi thường gặp

Câu 1: Xét số hexa x = A8 thì có giá trị là âm hay dương?

Trả lời: không có định nghĩa số hexa có dấu hay không có dấu mà hãy chuyển dãy hexa về biểu diễn nhị phân của nó, xét trong ví dụ này thì sẽ là dãy nhị phân 8 bit (1010 1000), xét lại đề bài là biểu diễn số lớn có dấu 128 bit, nghĩa là dãy bit này chỉ dài 8 bit, còn 120 bit phía trước là bit 0. Vậy, giá trị đúng của x = 168_{10}.