Bài tập lớn

Phiên bản 2, ngày 21/5/2012

Ra đề: Hoàng Thị Điệp (diep.thi.hoang@gmail.com)

Số nguyên dương lớn nhất có thể biểu diễn bởi kiểu dữ liệu có sẵn trong C++ là ULLONG_MAX = 18446744073709551615 (kiểu unsigned long long dài 8 byte), là số có 20 chữ số. Trong thực tế, ta có thể cần làm việc với các số nguyên lớn hơn, ví dụ: để lưu chính xác kết quả 21!. Hãy cài đặt lớp BigNumber biểu diễn số nguyên dương lớn.

Ý tưởng biểu diễn số nguyên dương lớn là dùng một vector<unsigned char> , mỗi phần tử biểu diễn một cặp chữ số.
Ví dụ 1: số n = 2345 thì vector tương ứng v có 2 phần tử là
v[0] = 45
v[1] = 23
Từ vector v, ta có thể suy ra được giá trị nó biểu diễn là v[1] * 100^1 + v[0] * 100^0
Ví dụ 2: số m = 21! = 51090942171709440000 thì vector tương ứng là

v[0]	v[1]	v[2]	v[3]	v[4]	v[5]	v[6]	v[7]	v[8]	v[9]
0	0	44	9	17	17	42	9	9	51

Để cài đặt cách lưu trữ nói trên, ta khai báo 1 biến thành viên private của lớp BigNumber là vector<unsigned char> digitpairs;

Dưới đây là giao diện yêu cầu của lớp BigNumber

class	BigNumber	
	BigNumber();	*// Hàm kiến tạo mặc định, vector rỗng*
	BigNumber(string value);	*// Hàm kiến tạo vector digitpairs từ xâu value lưu số lớn*
	BigNumber(const BigNumber& n1);	*// Hàm kiến tạo sao chép, tạo số lớn mới bằng số lớn n1 // cho trước. Có xét phép tự gán.*
	~BigNumber();	*// Hàm hủy, xóa tất cả các phần tử trong digitpairs*
unsigned int	length() const;	*// Trả về số chữ số của số lớn*
string	toString() const;	*// Trả về xâu giá trị của số lớn*
friend ostream&	operator<<(ostream& out, const BigNumber& number);	*// Ghi giá trị của số lớn ra luồng xuất*
friend istream&	operator>>(istream& in, BigNumber& number);	*// Đọc giá trị từ luồng nhập vào số lớn*
friend const BigNumber	operator+(const BigNumber& n1, const BigNumber& n2);	*// Nạp chồng phép cộng 2 số lớn*
friend const BigNumber	operator-(const BigNumber& n1, const BigNumber& n2);	*// Nạp chồng phép trừ 2 số lớn*
friend const BigNumber	operator*(const BigNumber& n1, const BigNumber& n2);	*// Nạp chồng phép nhân 2 số lớn*
friend const BigNumber	operator/(const BigNumber& n1, const BigNumber& n2);	*// Nạp chồng phép chia 2 số lớn*
friend const BigNumber	operator%(const BigNumber& n1, const BigNumber& n2);	*// Nạp chồng phép tìm dư khi chia n1 cho n2*
friend BigNumber&	operator++(BigNumber& n1);	*// Nạp chồng phép tự tăng trước*
friend const BigNumber	operator++(BigNumber& n1, int);	*// Nạp chồng phép tự tăng sau*

```
friend BigNumber&        operator--(BigNumber& n1);          // Nạp chồng phép tự giảm trước
friend const BigNumber   operator--(BigNumber& n1, int);     // Nạp chồng phép tự giảm sau
BigNumber&               operator=(const BigNumber& n1);     // Gán số lớn hiện tại bằng n1,
                                                             // đồng thời trả về giá trị này
unsigned char&           operator[](unsigned int i);         // Trả về chữ số thứ i của số lớn hiện tại (i >= 0)
```

Giải thích thay đổi trong phiên bản 2

- Phép tự tăng/giảm trước cho phép biểu thức dạng ++x = BigNumber("123"); // tương đương ++x = 123 với kiểu int
- Phép tự tăng/giảm sau **không** cho phép biểu thức dạng x++ = BigNumber("123"); // tương đương x++ = 123 với kiểu int
- Phép gán cho phép biểu thức dạng chuỗi x = y = BigNumber("123"); // tương đương x = y = 123; với kiểu int
- Phép truy cập đến chữ số thứ i cho phép biến đổi chữ số này, ví dụ: x[0] = 9; gán hàng đơn vị của số x bằng 9

Yêu cầu nộp:

- Nộp các file bài làm vào repository đặt tên là approject trên account bitbucket của bạn. Cần nộp ít nhất 3 files sau:
  - BigNumber.h và BigNumber.cpp
  - report.pdf mô tả các thuật toán đã dùng để cài đặt từng phép toán trên số lớn
- Thời gian làm: 4 tuần (từ 02/5 đến 30/5).
- Sinh viên đăng ký làm bài tập lớn (tối đa 2 người/nhóm) trước ngày 09/5 bằng cách điền tên vào form đã được gửi vào email cá nhân.