ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

**BÁO CÁO BÀI THỰC HÀNH SỐ 2**

**MÔN KỸ THUẬT LẬP TRÌNH**

Sinh viên: Lê Đình Trí Tuệ

MSSV: 20210909

Mã lớp: 732833

Contents

[Bài thực hành số 2 – Tuần 11 4](#_Toc152267603)

[**Bài tập 2.1.**Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông. 4](#_Toc152267604)

[**Bài tập 2.2**. Viết hàm hoán vị vòng tròn 3 biến a, b, c. Sau khi thực hiện hàm, các biến a, b, c tương ứng nhận các giá trị mới b, c, a. 5](#_Toc152267605)

[**Bài tập 2.3.** Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax2+bx+c với a, b, c định sẵn. 7](#_Toc152267606)

[**Bài tập 2.4.** Viết các hàm tính lập phương của số nguyên và số thực. 9](#_Toc152267607)

[**Bài tập 2.5.** Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức. 11](#_Toc152267608)

[**Bài tập 2.6.** Giả thuyết Collatz: bắt đầu từ số dương n bất kỳ, nếu n chẵn thì chia 2, nếu lẻ thì nhân 3 cộng 1, giả thuyết cho rằng ta luôn đi đến n=1 . Hãy viết chương trình mô phỏng lại quá trình biến đổi để kiếm chứng giả thuyết với giá trị của n nhập từ bàn phím. 14](#_Toc152267609)

[**Bài tập 2.7.** Viết hàm tính tổng các phần tử trong hai mảng. Yêu cầu sử dụng function template để cho phép hàm làm việc với các mảng số nguyên lẫn số thực. 18](#_Toc152267610)

[**Bài tập 2.8.** Viết hàm so sánh cho thuật toán sắp xếp. 20](#_Toc152267611)

[**Bài tập 2.9.** Dưới đây cung cấp đoạn code đơn giản để tính hàm sigmoid theo công thức trực tiếp. Hãy viết hàm tính xấp xỉ sigmoid(x) đến độ chính xác 10−6 và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản. 22](#_Toc152267612)

[**Bài tập 2.10 (BONUS).** Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxN theo công thức trực tiếp. Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với code đơn giản. Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc sử dụng thuật toán tốt hơn O(N3) . 25](#_Toc152267613)

[**Bài tập 2.11.** Cho 2 đa thức A(x) và B(x) tương ứng có bậc N và M . Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc N+M−1 . Input: Gồm 2 dòng biểu diễn các đa thức A(x) và B(x), mỗi dòng Số đầu tiên N là bậc của đa thức; N+1 số nguyên tiếp theo, số thứ i là hệ số của xi−1 . Output: Một số nguyên duy nhất là XOR của các hệ số của đa thức C(x). 25](#_Toc152267614)

[**Bài tập 2.12.** Hôm nay, cô giáo giao cho An một câu hỏi hóc búa. Cô cho một danh sách với mỗi phần tử có dạng <key, value> và yêu cầu An sắp xếp danh sách đó giảm dần theo giá trị value. Nếu 2 phần tử có value giống nhau thì sắp xếp giảm dần theo key. Hãy viết một chương trình sử dụng hàm nặc danh để giúp An làm bài tập. Input: Danh sách đầu vào. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách (|key| ≤109 , |value| ≤109 ). Output: In danh sách đã được sắp xếp theo yêu cầu. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách. 28](#_Toc152267615)

[**Bài tập 2.13.** Số nguyên lớn là các số nguyên có giá trị rất lớn và không thể biểu diễn bằng các kiểu dữ liệu nguyên cơ bản. Để biểu diễn số nguyên lớn, ta có thể dùng kiểu struct như sau: struct bigNum{ char sign; char num[101]; }; Nhiệm vụ các bạn là đa năng hóa các toán tử để thực hiện các phép toán số học với kiểu dữ liệu số nguyên lớn vừa định nghĩa ở trên. Input: Dữ liệu vào gồm hai dòng mô tả hai số nguyên lớn a và b , mỗi dòng chứa 1 chuỗi ký tự mô tả 1 số nguyên lớn không vượt quá 10100 . Chữ số đầu của mỗi chuỗi ký tự sẽ thể hiện dấu của số đó: 0 là âm, 1 là dương. Các chữ số sau thể hiện giá trị của số đó. Output: In ra giá trị của biểu thức ab−3a+4b . Kết quả in ra một số nguyên lớn dưới dạng chuỗi ký tự có định dạng như mô tả trong dữ liệu vào. 32](#_Toc152267616)

Figure 1 4

Figure 2 4

Figure 3 5

Figure 4 6

Figure 5 7

Figure 6 8

Figure 7 9

Figure 8 10

Figure 9 11

Figure 10 11

Figure 11 12

Figure 12 12

Figure 13 13

Figure 14 14

Figure 15 15

Figure 16 16

Figure 17 17

Figure 18 18

Figure 19 18

Figure 20 19

Figure 21 20

Figure 22 21

Figure 23 22

Figure 24 22

Figure 25 23

Figure 26 23

Figure 27 24

Figure 28 25

Figure 29 27

Figure 30 27

Figure 31 29

Figure 32 30

Figure 33 31

Figure 34 32

Figure 35 41

# Bài thực hành số 2 – Tuần 12

**Copy code (kèm ảnh code và kết quả vào )**

## **Bài tập 2.1.**Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông.

A screenshot of a computer

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

## **Bài tập 2.2**. Viết hàm hoán vị vòng tròn 3 biến a, b, c. Sau khi thực hiện hàm, các biến a, b, c tương ứng nhận các giá trị mới b, c, a.

A screenshot of a computer

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

## **Bài tập 2.3.** Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax2+bx+c với a, b, c định sẵn.

A screenshot of a computer code

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

## **Bài tập 2.4.** Viết các hàm tính lập phương của số nguyên và số thực.

A screenshot of a computer code

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

## **Bài tập 2.5.** Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức.

A screenshot of a computer

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

A screenshot of a computer code

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

## **Bài tập 2.6.** Giả thuyết Collatz: bắt đầu từ số dương n bất kỳ, nếu n chẵn thì chia 2, nếu lẻ thì nhân 3 cộng 1, giả thuyết cho rằng ta luôn đi đến n=1 . Hãy viết chương trình mô phỏng lại quá trình biến đổi để kiếm chứng giả thuyết với giá trị của n nhập từ bàn phím.

A screenshot of a computer

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

## **Bài tập 2.7.** Viết hàm tính tổng các phần tử trong hai mảng. Yêu cầu sử dụng function template để cho phép hàm làm việc với các mảng số nguyên lẫn số thực.

A screenshot of a computer

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

## **Bài tập 2.8.** Viết hàm so sánh cho thuật toán sắp xếp.

A screenshot of a computer program

Description automatically generated

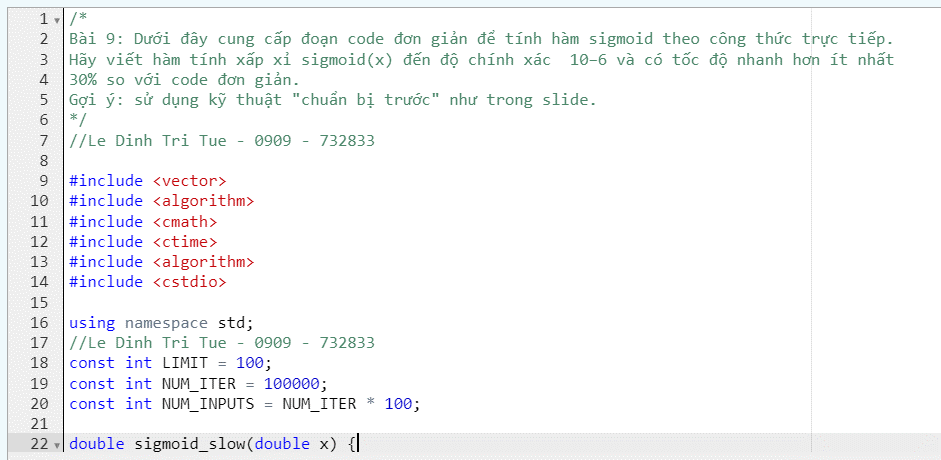
Figure

A screenshot of a computer program

Description automatically generated

Figure

## **Bài tập 2.9.** Dưới đây cung cấp đoạn code đơn giản để tính hàm sigmoid theo công thức trực tiếp. Hãy viết hàm tính xấp xỉ sigmoid(x) đến độ chính xác 10−6 và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản.



Figure

A screenshot of a computer program

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

A screenshot of a computer code

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

## **Bài tập 2.10 (BONUS).** Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxN theo công thức trực tiếp. Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với code đơn giản. Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc sử dụng thuật toán tốt hơn O(N3) .

**CODE:**

/\*

Bài 10: Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxN

theo công thức trực tiếp.

Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với

code đơn giản.

Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc

sử dụng thuật toán tốt hơn O(N3)

.

// Le Dinh Tri Tue - 0909 - 732833

\*/

#include <iostream>

#include <cstring>

#include <ctime>

using namespace std;

const int N = 128;

// Le Dinh Tri Tue - 0909 - 732833

struct Matrix {

unsigned int mat[N][N];

Matrix() {

memset(mat, 0, sizeof mat);

}

};

// Le Dinh Tri Tue - 0909 - 732833

bool operator == (const Matrix &a, const Matrix &b) {

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

if (a.mat[i][j] != b.mat[i][j]) return false;

}

}

return true;

}

// Le Dinh Tri Tue - 0909 - 732833

Matrix multiply\_naive(const Matrix &a, const Matrix &b) {

Matrix c;

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

for (int k = 0; k < N; ++k) {

c.mat[i][j] += a.mat[i][k] \* b.mat[k][j];

}

}

}

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

for (int k = 0; k < N; ++k) {

c.mat[i][j] += 0;

}

}

}

return c;

}

// Le Dinh Tri Tue - 0909 - 732833

Matrix operator + (const Matrix &a, const Matrix &b) {

Matrix result;

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

result.mat[i][j] = a.mat[i][j] + b.mat[i][j];

}

}

return result;

}

// Le Dinh Tri Tue - 0909 - 732833

Matrix operator - (const Matrix &a, const Matrix &b) {

Matrix result;

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

result.mat[i][j] = a.mat[i][j] - b.mat[i][j];

}

}

return result;

}

// Le Dinh Tri Tue - 0909 - 732833

Matrix multiply\_fast(const Matrix &a, const Matrix &b) {

Matrix c;

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

int sum = 0;

for (int k = 0; k < N; ++k) {

sum += a.mat[i][k] \* b.mat[k][j];

}

c.mat[i][j] = sum;

}

}

return c;

}

// Le Dinh Tri Tue - 0909 - 732833

Matrix gen\_random\_matrix() {

Matrix a;

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j) {

a.mat[i][j] = rand();

}

}

return a;

}

Matrix base;

// Le Dinh Tri Tue - 0909 - 732833

double benchmark(Matrix (\*multiply) (const Matrix&, const Matrix&), Matrix &result) {

const int NUM\_TEST = 10;

const int NUM\_ITER = 64;

Matrix a = base;

result = a;

double taken = 0;

for (int t = 0; t < NUM\_TEST; ++t) {

clock\_t start = clock();

for (int i = 0; i < NUM\_ITER; ++i) {

a = multiply(a, result);

result = multiply(result, a);

}

clock\_t finish = clock();

taken += (double)(finish - start);

}

taken /= NUM\_TEST;

printf("Time: %.9f\n", taken / CLOCKS\_PER\_SEC);

return taken;

}

// Le Dinh Tri Tue - 0909 - 732833

int main() {

base = gen\_random\_matrix();

Matrix a, b;

printf("Slow version\n");

double slow = benchmark(multiply\_naive, a);

printf("Fast version\n");

double fast = benchmark(multiply\_fast, b);

if (a == b) {

printf("Correct answer! Your code is %.2f%% faster\n", slow / fast \* 100.0);

} else {

printf("Wrong answer!\n");

}

return 0;

}

// Le Dinh Tri Tue - 0909 - 732833

A screen shot of a computer program

Description automatically generated

Hình ảnh 34 – Bài 1.10

A screenshot of a computer program

Description automatically generated

Hình ảnh 35 – Bài 1.10

A screenshot of a computer screen

Description automatically generated

Hình ảnh 36 – Bài 1.10

## **Bài tập 2.11.** Cho 2 đa thức A(x) và B(x) tương ứng có bậc N và M . Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc N+M−1 . Input: Gồm 2 dòng biểu diễn các đa thức A(x) và B(x), mỗi dòng Số đầu tiên N là bậc của đa thức; N+1 số nguyên tiếp theo, số thứ i là hệ số của xi−1 . Output: Một số nguyên duy nhất là XOR của các hệ số của đa thức C(x).

**CODE:**

/\*

Bài 11: Cho 2 đa thức A(x) và B(x) tương ứng có bậc N và M

Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc N+M−1

// Le Dinh Tri Tue - 0909 - 732833

\*/

#include <bits/stdc++.h>

void find\_index(int a[], int b[], int n, int m, int c[]) {

for (int i = 0; i < n + m + 1; i++) {

for (int j = 0; j < n + 1; j++) {

if (i - j >= 0 and i - j < m + 1) {

c[i] += a[j] \* b[i-j];

}

}

}

}

// Le Dinh Tri Tue - 0909 - 732833

int cal\_XOR(int c[], int n) {

if (n > 0) {

int result = c[0];

for (int i = 1; i < n + 1; i++) {

result = result ^ c[i];

}

return result;

}

return -1;

}

// Le Dinh Tri Tue - 0909 - 732833

int main() {

int n;

std::cin>>n;

int a[n + 1];

for (int i = 0; i < n + 1; i++) {

std::cin>>a[i];

}

int m;

std::cin>>m;

int b[m + 1];

for (int i = 0; i < m + 1; i++) {

std::cin>>b[i];

}

int c[m+n+1];

memset(c,0,sizeof c);

find\_index(a, b, n, m, c);

std::cout<<cal\_XOR(c, n + m);

}

// Le Dinh Tri Tue - 0909 - 732833

A blue background with many small letters

Description automatically generated with medium confidence

Figure

A blue and white pixelated background

Description automatically generated

Figure

## **Bài tập 2.12.** Hôm nay, cô giáo giao cho An một câu hỏi hóc búa. Cô cho một danh sách với mỗi phần tử có dạng <key, value> và yêu cầu An sắp xếp danh sách đó giảm dần theo giá trị value. Nếu 2 phần tử có value giống nhau thì sắp xếp giảm dần theo key. Hãy viết một chương trình sử dụng hàm nặc danh để giúp An làm bài tập. Input: Danh sách đầu vào. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách (|key| ≤109 , |value| ≤109 ). Output: In danh sách đã được sắp xếp theo yêu cầu. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách.

**CODE:**

/\*

Bài 12: Hôm nay, cô giáo giao cho An một câu hỏi hóc búa. Cô cho một danh

sách với mỗi phần tử có dạng <key, value> và yêu cầu An sắp xếp danh sách

đó giảm dần theo giá trị value. Nếu 2 phần tử có value giống nhau thì sắp

xếp giảm dần theo key.

Hãy viết một chương trình sử dụng hàm nặc danh để giúp An làm bài tập.

// Le Dinh Tri Tue - 0909 - 732833

\*/

#include <bits/stdc++.h>

bool customCompare(const std::pair<int, int> &a, const std::pair<int, int> &b) {

if (a.second == b.second) {

return a.first > b.first;

}

return a.second > b.second;

}

int main() {

std::vector<std::pair<int, int>> myList;

int n;

std::cin>>n;

for (int i = 0; i < n; i++) {

int a, b;

std::cin>>a>>b;

myList.push\_back(std::make\_pair(a, b));

}

std::sort(myList.begin(), myList.end(), customCompare);

for (const auto& pair : myList) {

std::cout<< pair.first <<" "<< pair.second <<"\n";

}

return 0;

}

// Le Dinh Tri Tue - 0909 – 732833

A blue background with white and blue lines

Description automatically generated

Figure

A screenshot of a computer

Description automatically generated

Figure

A screen shot of a computer code

Description automatically generated

Figure

A screenshot of a computer program

Description automatically generated

Figure

## **Bài tập 2.13.** Số nguyên lớn là các số nguyên có giá trị rất lớn và không thể biểu diễn bằng các kiểu dữ liệu nguyên cơ bản. Để biểu diễn số nguyên lớn, ta có thể dùng kiểu struct như sau: struct bigNum{ char sign; char num[101]; }; Nhiệm vụ các bạn là đa năng hóa các toán tử để thực hiện các phép toán số học với kiểu dữ liệu số nguyên lớn vừa định nghĩa ở trên. Input: Dữ liệu vào gồm hai dòng mô tả hai số nguyên lớn a và b , mỗi dòng chứa 1 chuỗi ký tự mô tả 1 số nguyên lớn không vượt quá 10100 . Chữ số đầu của mỗi chuỗi ký tự sẽ thể hiện dấu của số đó: 0 là âm, 1 là dương. Các chữ số sau thể hiện giá trị của số đó. Output: In ra giá trị của biểu thức ab−3a+4b . Kết quả in ra một số nguyên lớn dưới dạng chuỗi ký tự có định dạng như mô tả trong dữ liệu vào.

**CODE:**

/\*

Bài 13: Số nguyên lớn là các số nguyên có giá trị rất lớn và không thể biểu diễn

bằng các kiểu dữ liệu nguyên cơ bản. Để biểu diễn số nguyên lớn, ta có thể dùng

kiểu struct như sau:

struct bigNum{

char sign;

char num[101];

};

Nhiệm vụ các bạn là đa năng hóa các toán tử để thực hiện các phép toán số học với

kiểu dữ liệu số nguyên lớn vừa định nghĩa ở trên.

Le Dinh Tri Tue - 0909 - 732833

\*/

#include<bits/stdc++.h>

using namespace std;

struct bigNum {

char sign;

char num[101];

};

// Nhập vào hai số nguyên lớn

// Le Dinh Tri Tue - 0909 - 732833

void input(bigNum &num1, bigNum &num2){

string tmp;

cin >> tmp;

num1.sign = tmp[0];

int lens1 = tmp.length() - 1;

for (int i = 0; i < lens1; i++){

num1.num[100-lens1+i+1] = tmp[i+1];

}

for (int i = 0; i < 100-lens1+1; i++){

num1.num[i] = '0';

}

cin >> tmp;

num2.sign = tmp[0];

int lens2 = tmp.length() - 1;

for (int i = 0; i < lens2; i++){

num2.num[100-lens2+i+1] = tmp[i+1];

}

for (int i = 0; i < 100-lens2+1; i++){

num2.num[i] = '0';

}

}

// cộng hai số nguyên dương lớn

// Le Dinh Tri Tue - 0909 - 732833

void add(char res[], char \*num1, char \*num2){

int c = 0;

for(int i=100; i>=0; i--){

int tmp = (int)num1[i] - 48 + (int)num2[i] - 48 + c;

c = tmp / 10;

res[i] = tmp % 10 + 48;

}

}

// trừ hai số nguyên dương lớn, num1 > num2

// Le Dinh Tri Tue - 0909 - 732833

void sub(char res[], char \*num1, char\* num2){

int c = 0;

for (int i = 100; i >= 0; i--){

int tmp1 = (int)num1[i] - 48;

int tmp2 = (int)num2[i] - 48;

if (tmp1 >= tmp2 + c){

res[i] = tmp1 - tmp2 - c + 48;

c = 0;

} else {

tmp1 = tmp1 + 10;

res[i] = tmp1 - tmp2 - c + 48;

c = 1;

}

}

}

// nhân hai số nguyên lớn

// Le Dinh Tri Tue - 0909 - 732833

void multi(char res[], char \*num1, char \*num2){

for(int i = 0; i < 101; i++){

res[i] = '0';

}

for (int i = 100; i >= 0; i--){

char tmp[101];

int k;

for (k = 0; k < i; k++){

tmp[100-k] = '0';

}

int c = 0, sum = 0;

for (int j = 100; j >= 0; j--){

sum = ((int)num1[i] - 48) \* ((int)num2[j] - 48) + c;

tmp[k] = (sum % 10) + 48;

c = sum / 10;

k--;

if(k < 0) break;

}

add(res,tmp,res);

}

}

// kiểm tra xem số nguyên lớn num1 có lớn hơn num2 không

// Le Dinh Tri Tue - 0909 - 732833

bool check(char \*num1, char \*num2){

int foo1, foo2;

for (foo1 = 0; foo1 < 101; foo1++){

if (num1[foo1] != '0') break;

}

for (foo2 = 0; foo2 < 101; foo2++){

if (num2[foo2] != '0') break;

}

if (foo1 > foo2) return false;

else if (foo1 < foo2) return true;

else {

int foo = foo1;

while (foo < 101){

if (num1[foo] < num2[foo]) return false;

else if (num1[foo] > num2[foo]) return true;

else foo++;

}

}

return true;

}

// định nghĩa cho operator +

// Le Dinh Tri Tue - 0909 - 732833

bigNum operator + (bigNum num1, bigNum num2){

bigNum res;

if (num1.sign == '1' && num2.sign == '1'){

res.sign = '1';

add(res.num, num1.num, num2.num);

return res;

} else if (num1.sign == '1' && num2.sign == '0'){

if (check(num1.num, num2.num)){

res.sign = '1';

sub(res.num, num1.num, num2.num);

return res;

} else {

res.sign = '0';

sub(res.num, num2.num, num1.num);

return res;

}

} else if (num1.sign == '0' && num2.sign == '1'){

if (check(num1.num, num2.num)){

res.sign = '0';

sub(res.num, num1.num, num2.num);

return res;

} else {

res.sign = '1';

sub(res.num, num2.num, num1.num);

return res;

}

} else {

res.sign = '0';

add(res.num, num1.num, num2.num);

return res;

}

}

// định nghĩa cho operator -

// Le Dinh Tri Tue - 0909 - 732833

bigNum operator - (bigNum num1, bigNum num2){

bigNum res;

if (num1.sign == '1' && num2.sign == '0'){

num2.sign = '1';

res = num1 + num2;

return res;

} else if (num1.sign == '1' && num2.sign == '1'){

num2.sign = '0';

res = num1 + num2;

return res;

} else if (num1.sign == '0' && num2.sign == '1'){

num2.sign = '0';

res = num1 + num2;

return res;

} else {

num2.sign = '1';

res = num1 + num2;

return res;

}

}

// định nghĩa cho operator \*

// Le Dinh Tri Tue - 0909 - 732833

bigNum operator \* (bigNum num1, bigNum num2){

bigNum res;

if (num1.sign == '1' && num2.sign == '1'){

res.sign = '1';

multi(res.num, num1.num, num2.num);

return res;

} else if (num1.sign == '1' && num2.sign == '0'){

res.sign = '0';

multi(res.num, num1.num, num2.num);

return res;

} else if (num1.sign == '0' && num2.sign == '1'){

res.sign = '0';

multi(res.num, num1.num, num2.num);

return res;

} else {

res.sign = '1';

multi(res.num, num1.num, num2.num);

return res;

}

}

// print BigNum

// Le Dinh Tri Tue - 0909 - 732833

void printBigNumber(bigNum number){

cout << number.sign;

int start;

for (start = 0; start < 101; start++){

if (number.num[start] != '0') break;

}

for (int i = start; i < 101; i++){

cout << number.num[i];

}

}

// Le Dinh Tri Tue - 0909 - 732833

int main(){

bigNum num1, num2;

input(num1,num2);

bigNum so3, so4;

so3.sign = '1', so4.sign = '1';

for (int i = 0; i < 100; i++){

so3.num[i] = '0';

so4.num[i] = '0';

}

so3.num[100] = 3 + 48;

so4.num[100] = 4 + 48;

bigNum res = num1\*num2 - so3 \* num1 + so4 \* num2;

printBigNumber(res);

}

// Le Dinh Tri Tue - 0909 – 732833

A screenshot of a computer program

Description automatically generated

Figure