# Bài thực hành 2: Hàm và tối ưu mã nguồn

Contents

[Bài thực hành 2: Hàm và tối ưu mã nguồn 1](#_Toc69828684)

[Phần 1. Thực hành về hàm 3](#_Toc69828685)

[1.1 Truyền tham trị, tham chiếu và tham số ngầm định 3](#_Toc69828686)

[Bài tập 1: Truyền tham trị: 3](#_Toc69828687)

[Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông. 3](#_Toc69828688)

[Bài tập 2: Truyền tham chiếu 4](#_Toc69828689)

[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. 4](#_Toc69828690)

[Bài tập 3: Tham số ngầm định 5](#_Toc69828691)

[Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax2+bx+cax2+bx+c với a, b, c định sẵn. 5](#_Toc69828692)

[1.2 Đa năng hóa hàm 7](#_Toc69828693)

[Bài tập 4: Đa năng hóa hàm 7](#_Toc69828694)

[Viết các hàm tính lập phương của số nguyên và số thực. 7](#_Toc69828695)

[Bài tập 5: Đa năng hóa toán tử 9](#_Toc69828696)

[Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức. 9](#_Toc69828697)

[1.3 Con trỏ hàm và tham số hóa hàm 11](#_Toc69828698)

[Bài tập 6: Con trỏ hàm 11](#_Toc69828699)

[Giả thuyết Collatz: bắt đầu từ số dương nn bất kỳ, nếu nn chẵn thì chia 2, nếu lẻ thì nhân 3 cộng 1, giả thuyết cho rằng ta luôn đi đến n=1n=1. 11](#_Toc69828700)

[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. 11](#_Toc69828701)

[Bài tập 7: Khái quát hóa hàm 13](#_Toc69828702)

[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. 13](#_Toc69828703)

[1.4 Biểu thức lamda và hàm nặc danh 15](#_Toc69828704)

[Bài tập 8: Sắp xếp 15](#_Toc69828705)

[Viết hàm so sánh cho thuật toán sắp xếp. 15](#_Toc69828706)

[Phần 2. Thực hành về tối ưu mã nguồn 17](#_Toc69828707)

[Bài tập 9: Tính hàm sigmoid 17](#_Toc69828708)

[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 10−610−6 và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản. Gợi ý: sử dụng kỹ thuật "chuẩn bị trước" như trong slide. 17](#_Toc69828709)

[Bài tập 10 (bonus): Tính tích hai ma trận vuông 22](#_Toc69828710)

[Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxNNxN 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)O(N3). 22](#_Toc69828711)

[Phần 3. Bài tập về nhà 26](#_Toc69828712)

[Bài tập 11: Tính tích hai đa thức 26](#_Toc69828713)

[Cho 2 đa thức A(x) và B(x) tương ứng có bậc NN và MM. Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc N+M−1N+M−1. 26](#_Toc69828714)

[Bài tập 12: Map Sort 28](#_Toc69828715)

[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. 28](#_Toc69828716)

[Hãy viết một chương trình sử dụng hàm nặc danh để giúp An làm bài tập. 28](#_Toc69828717)

[Bài tập 13: Big Integer 30](#_Toc69828718)

[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: 30](#_Toc69828719)

[struct bigNum{      char sign;      char num[101]; }; 30](#_Toc69828720)

[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. 30](#_Toc69828721)

# Phần 1. Thực hành về hàm

## 1.1 Truyền tham trị, tham chiếu và tham số ngầm định

### Bài tập 1: Truyền tham trị:

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

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap1\_Buoi2.cpp

#include <stdio.h>

#include <math.h>

float get\_hypotenuse(float x, float y) {

float z = sqrt(x\*x + y\*y);

return z;

}

int main(){

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

float x, y;

scanf("%f%f", &x, &y);

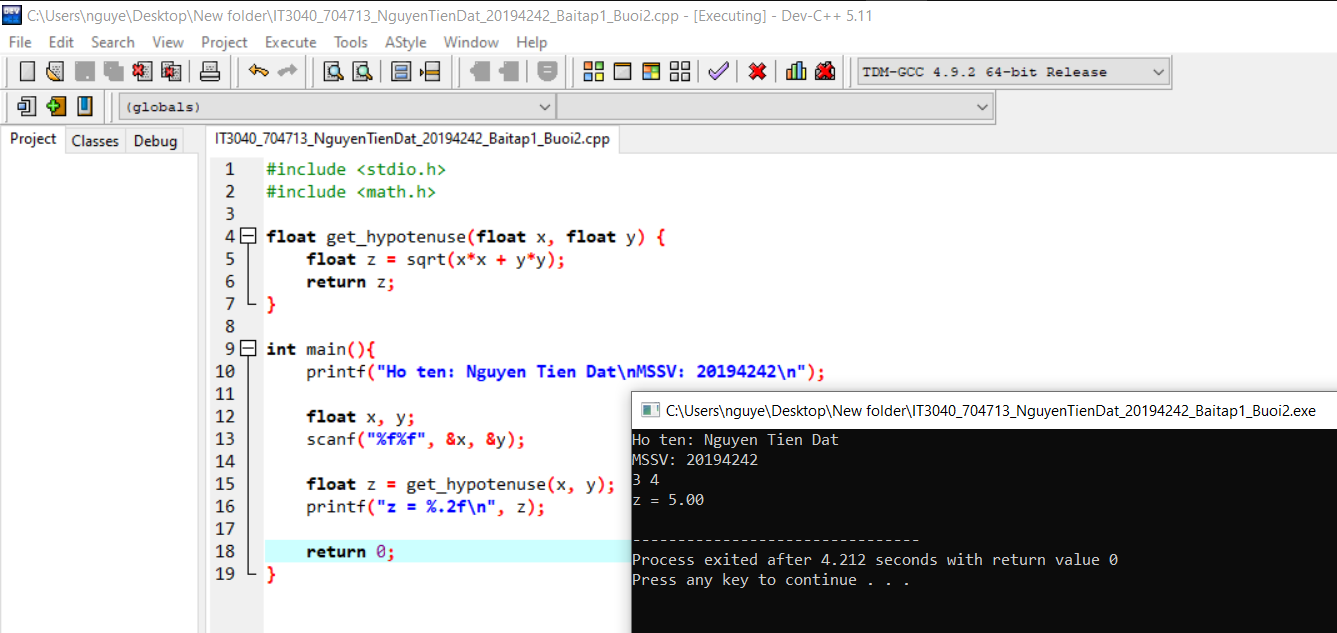
float z = get\_hypotenuse(x, y);

printf("z = %.2f\n", z);

return 0;

}

Kết quả:



### Bài tập 2: Truyền tham chiếu

### 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.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap2\_Buoi2.cpp

#include <stdio.h>

void rotate(int &x, int &y, int &z) {

int tmp = x;

x = y;

y = z;

z = tmp;

}

int main() {

int x, y, z;

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

scanf("%d%d%d", &x, &y, &z);

printf("Before: %d, %d, %d\n", x, y, z);

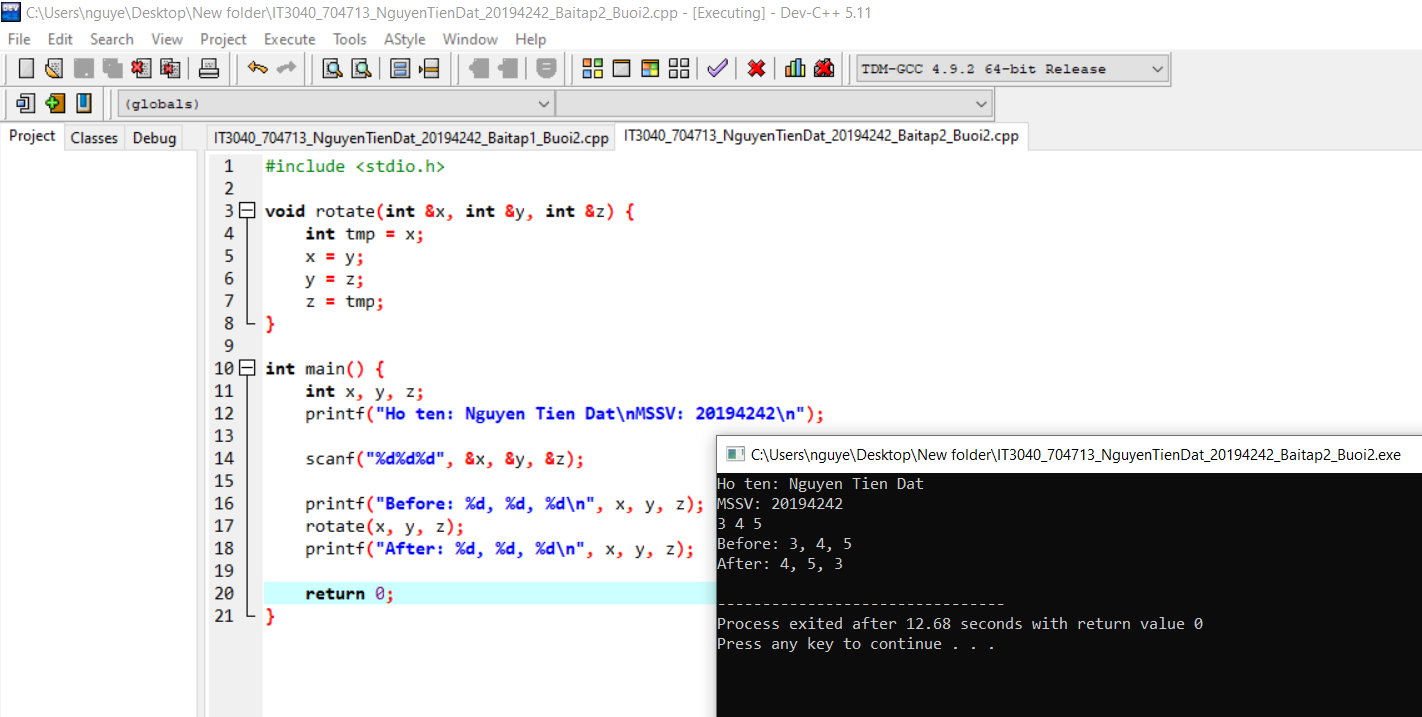
rotate(x, y, z);

printf("After: %d, %d, %d\n", x, y, z);

return 0;

}

Kết quả:



### Bài tập 3: Tham số ngầm định

### Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax2+bx+cax2+bx+c với a, b, c định sẵn.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap3\_Buoi2.cpp

#include <stdio.h>

int get\_value(int x) {

return 2\*x\*x + x;

}

int get\_value(int x, int a) {

return a\*x\*x + x;

}

int get\_value(int x, int a, int b) {

return a\*x\*x + b\*x;

}

int get\_value(int x, int a, int b, int c) {

return a\*x\*x + b\*x + c;

}

int main(){

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

int x;

scanf("%d", &x);

int a = 2; //# giÃ¡ trá»‹ máº·c Ä‘á»‹nh cá»§a a

int b = 1; //# giÃ¡ trá»‹ máº·c Ä‘á»‹nh cá»§a b

int c = 0; //# giÃ¡ trá»‹ máº·c Ä‘á»‹nh cá»§a c

scanf("%d%d%d", &a, &b, &c);

printf("a=2, b=1, c=0: %d\n", get\_value(x));

printf("a=%d, b=1, c=0: %d\n", a, get\_value(x, a));

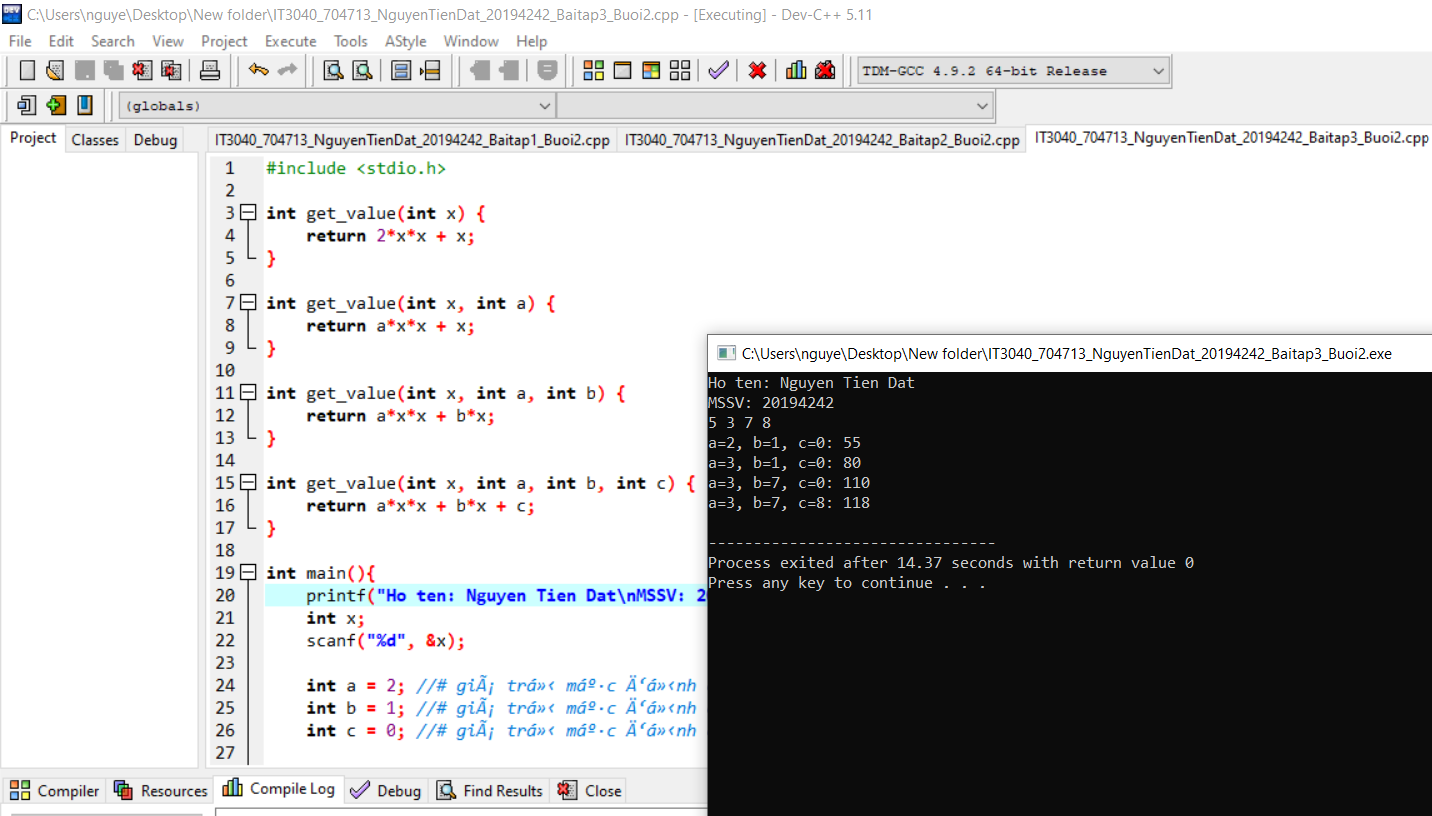
printf("a=%d, b=%d, c=0: %d\n", a, b, get\_value(x, a, b));

printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get\_value(x, a, b, c));

return 0;

}

Kết quả:



## 1.2 Đa năng hóa hàm

### Bài tập 4: Đa năng hóa hàm

### Viết các hàm tính lập phương của số nguyên và số thực.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap4\_Buoi2.cpp

#include <stdio.h>

int cube(int x) {

//# tráº£ vá» láº­p phÆ°Æ¡ng cá»§a x

return x\*x\*x;

}

//# viáº¿t hÃ m tÃ­nh láº­p phÆ°Æ¡ng cá»§a má»™t sá»‘ kiá»ƒu double

double cube(double x) {

return x\*x\*x;

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

int n;

double f;

scanf("%d %lf", &n, &f);

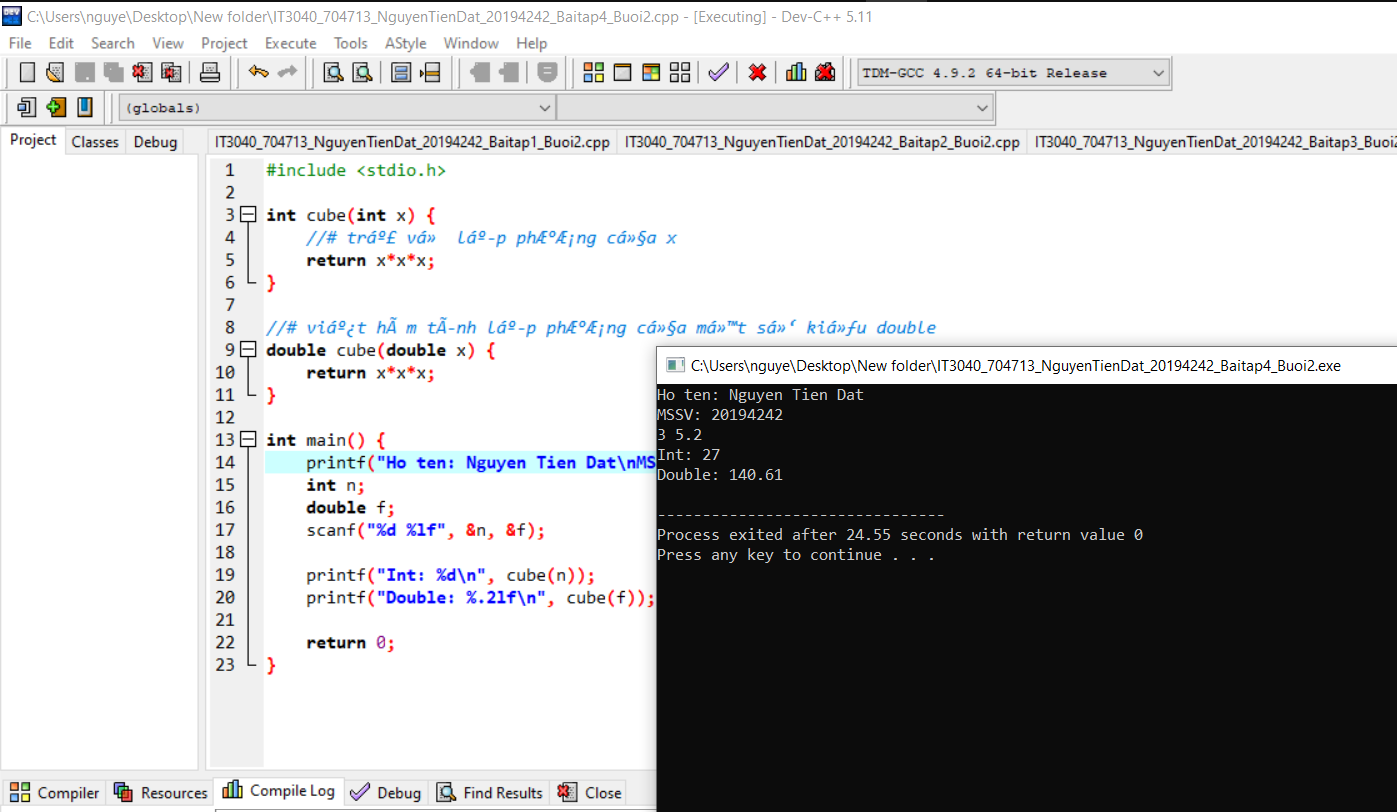
printf("Int: %d\n", cube(n));

printf("Double: %.2lf\n", cube(f));

return 0;

}

Kết quả:



### Bài tập 5: Đa năng hóa toán tử

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

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap5\_Buoi2.cpp

#include <iostream>

#include <ostream>

#include <math.h>

#include <iomanip>

using namespace std;

struct Complex {

double real;

double imag;

};

Complex operator + (Complex a, Complex b) {

Complex c;

c.imag = a.imag + b.imag;

c.real = a.real + b.real;

return c;

}

Complex operator - (Complex a, Complex b) {

Complex c;

c.imag = a.imag - b.imag;

c.real = a.real - b.real;

return c;

}

Complex operator \* (Complex a, Complex b) {

Complex c;

c.imag = a.imag \* b.real + b.imag \* a.real;

c.real = a.real \* b.real - b.imag \* a.imag;

return c;

}

Complex operator / (Complex a, Complex b) {

Complex c;

double A = b.real;

double B = b.imag;

double C = a.real;

double D = a.imag;

c.real = (A\*C + B\*D) / (A\*A + B\*B);

c.imag = (A\*D - B\*C) / (A\*A + B\*B);

return c;

}

ostream& operator << (ostream& out, const Complex &a) {

out << '(' << std::setprecision(2) << a.real << (a.imag >= 0 ? '+' : '-') << std::setprecision(2) << fabs(a.imag) << 'i' << ')';

return out;

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

double real\_a, real\_b, img\_a, img\_b;

cin >> real\_a >> img\_a;

cin >> real\_b >> img\_b;

Complex a{real\_a, img\_a};

Complex b{real\_b, img\_b};

cout << a << " + " << b << " = " << a + b << endl;

cout << a << " - " << b << " = " << a - b << endl;

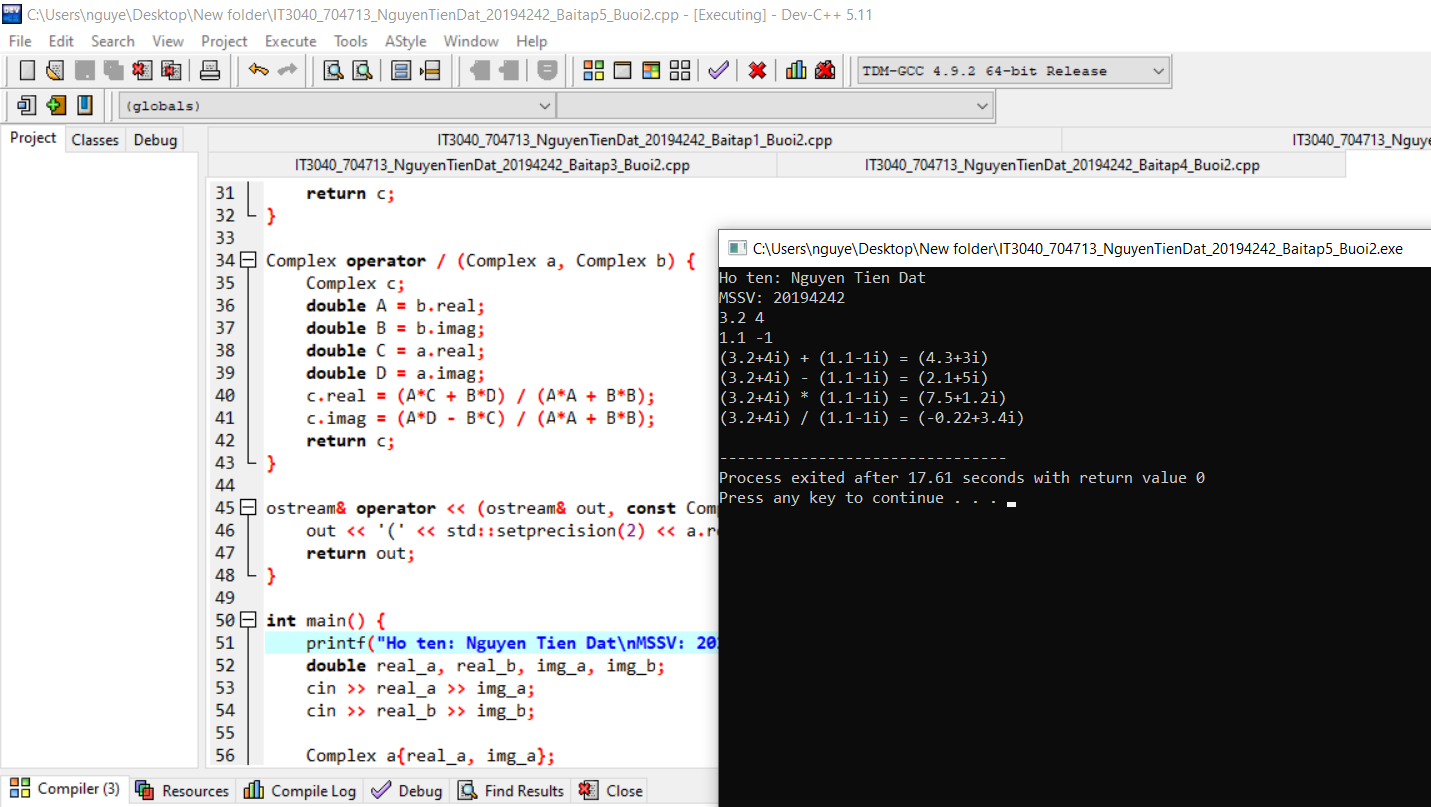
cout << a << " \* " << b << " = " << a \* b << endl;

cout << a << " / " << b << " = " << a / b << endl;

return 0;

}

Kết quả:



## 1.3 Con trỏ hàm và tham số hóa hàm

### Bài tập 6: Con trỏ hàm

### Giả thuyết Collatz: bắt đầu từ số dương nn bất kỳ, nếu nn chẵn thì chia 2, nếu lẻ thì nhân 3 cộng 1, giả thuyết cho rằng ta luôn đi đến n=1n=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.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap6\_Buoi2.cpp

#include <stdio.h>

void print(int n) {

printf("n=%d\n", n);

}

int mul3plus1(int n) {

return n \* 3 + 1;

}

int div2(int n) {

return n / 2;

}

// khai bÃ¡o cÃ¡c tham sá»‘ cho cÃ¡c con trá» hÃ m odd, even vÃ  output

void simulate(int n, int (\*odd)(int), int (\*even)(int), void (\*output)(int)) {

(\*output)(n);

if (n == 1) return;

if (n % 2 == 0) {

n = (\*even)(n);

} else {

n = (\*odd)(n);

}

simulate(n, odd, even, output);

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

int n;

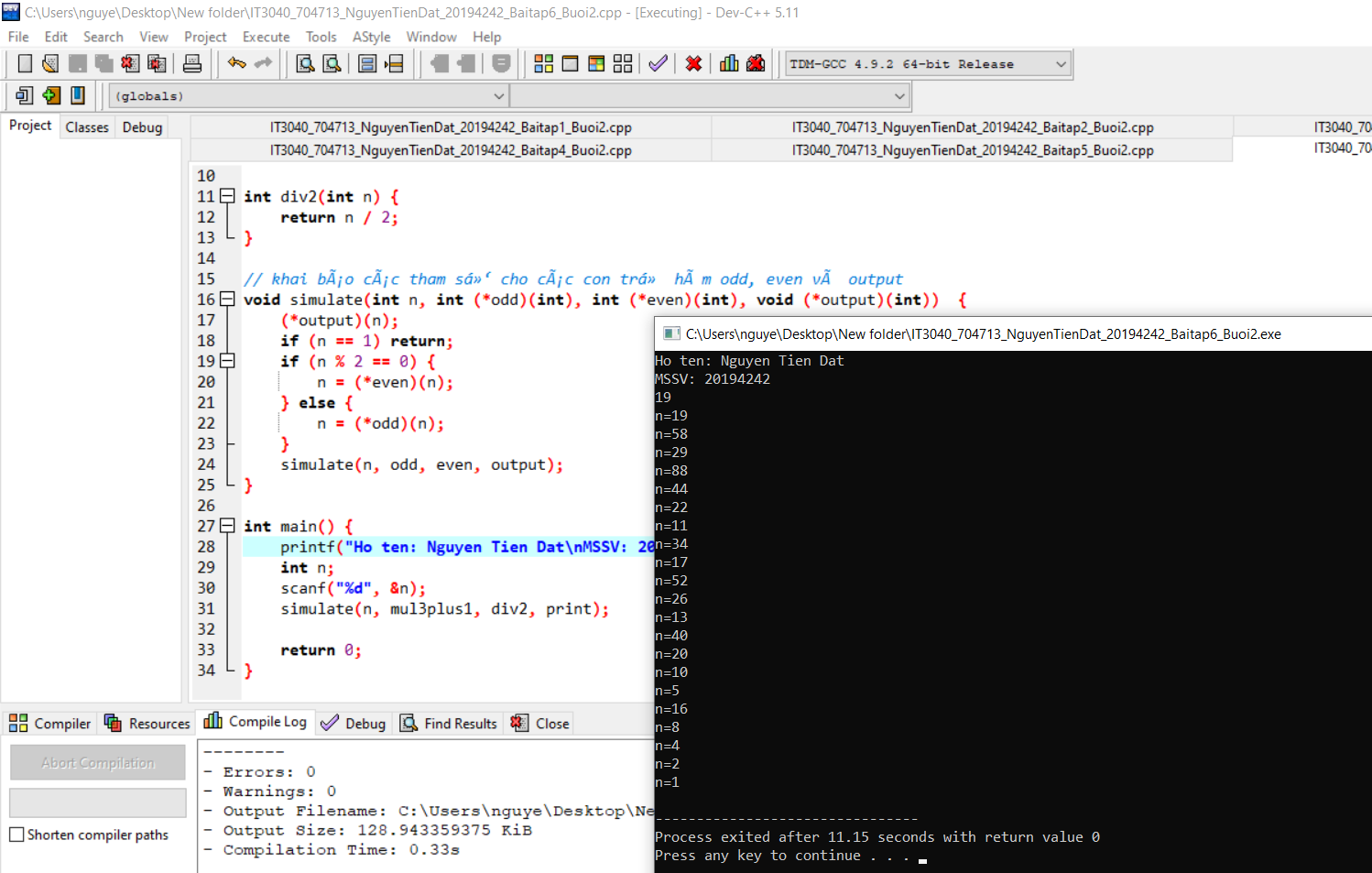
scanf("%d", &n);

simulate(n, mul3plus1, div2, print);

return 0;

}

Kết quả:



### Bài tập 7: Khái quát hóa hàm

### 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.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap7\_Buoi2.cpp

#include <iostream>

using namespace std;

template <typename T>

//# viáº¿t hÃ m arr\_sum

T arr\_sum (T \*arr1, int n1, T \*arr2, int n2) {

T sum1 = 0, sum2 = 0;

for (int i = 0; i < n1; i++) sum1 += arr1[i];

for (int i = 0; i < n2; i++) sum2 += arr2[i];

return sum1 + sum2;

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

int val;

cin >> val;

{

int a[] = {3, 2, 0, val};

int b[] = {5, 6, 1, 2, 7};

cout << arr\_sum(a, 4, b, 5) << endl;

}

{

double a[] = {3.0, 2, 0, val \* 1.0};

double b[] = {5, 6.1, 1, 2.3, 7};

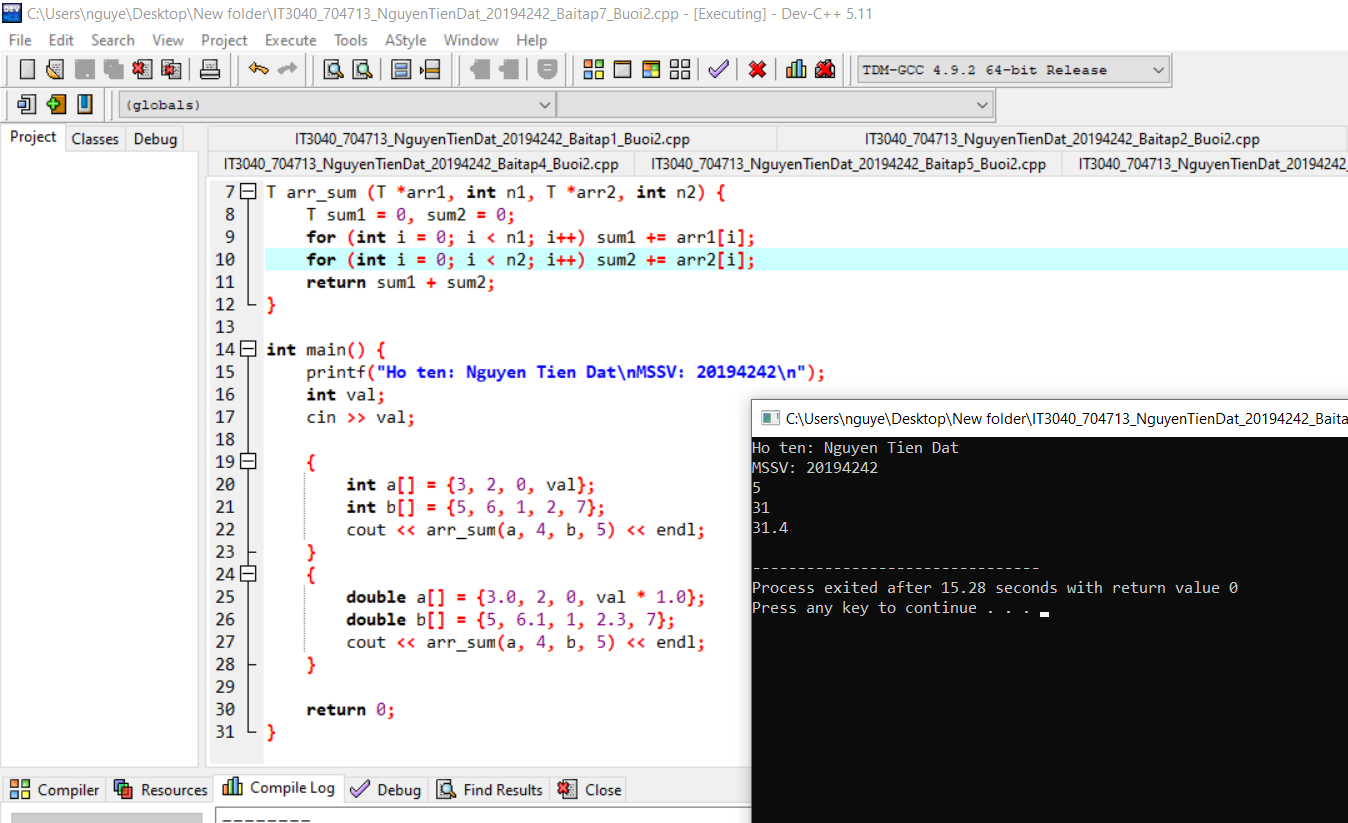
cout << arr\_sum(a, 4, b, 5) << endl;

}

return 0;

}

Kết quả:



## 1.4 Biểu thức lamda và hàm nặc danh

### Bài tập 8: Sắp xếp

### Viết hàm so sánh cho thuật toán sắp xếp.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap8\_Buoi2.cpp

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

int val1, val2;

cin >> val1 >> val2;

vector< vector<int> > a = {

{1, 3, 7},

{2, 3, 4, val1},

{9, 8, 15},

{10, val2},

};

//# sáº¯p xáº¿p cÃ¡c vector trong a theo tá»•ng cÃ¡c pháº§n tá»­ giáº£m dáº§n

sort(a.begin(), a.end(), [=] (vector<int> left, vector<int>right) {

int sum\_left = 0, sum\_right = 0;

for (unsigned int i = 0; i < left.size(); i++) sum\_left += left[i];

for (unsigned int i = 0; i < right.size(); i++) sum\_right += right[i];

return sum\_left > sum\_right;

});

for (const auto &v : a) {

for (int it : v) {

cout << it << ' ';

}

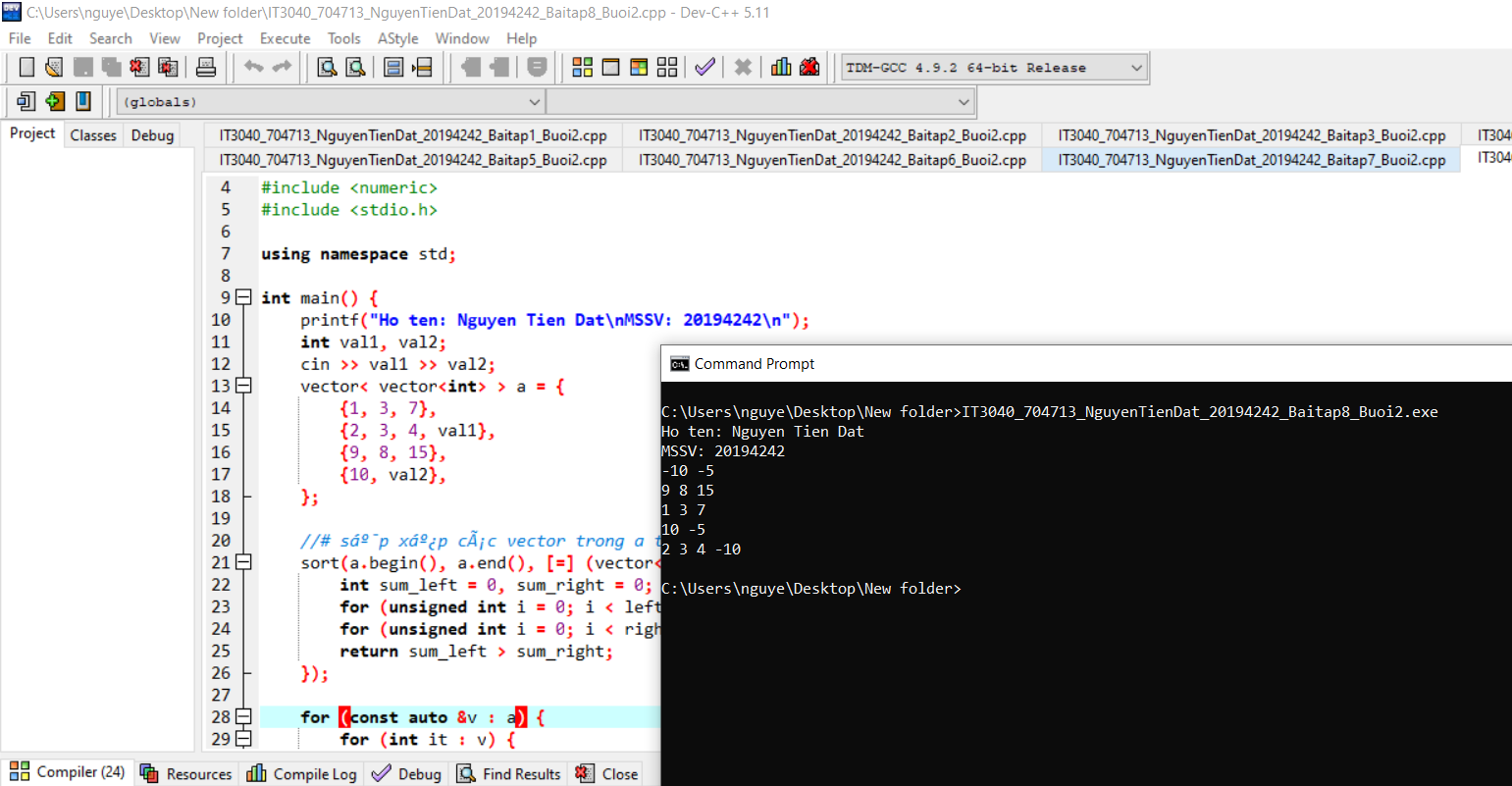
cout << endl;

}

return 0;

}

Kết quả:



# Phần 2. Thực hành về tối ưu mã nguồn

Hãy giải các bài toán sau đây một cách tối ưu nhất có thể, cố gắng sử dụng các kỹ thuật đã được học như inline, static, ...

### Bài tập 9: Tính hàm sigmoid

### 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 10−610−6 và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản. **Gợi ý:** sử dụng kỹ thuật "chuẩn bị trước" như trong slide.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap9\_Buoi2.cpp

#include <vector>

#include <algorithm>

#include <cmath>

#include <ctime>

#include <algorithm>

#include <cstdio>

using namespace std;

const int LIMIT = 100;

const int NUM\_ITER = 100000;

const int NUM\_INPUTS = NUM\_ITER \* 100;

double sigmoid\_slow(double x) {

return 1.0 / (1.0 + exp(-x));

}

double x[NUM\_INPUTS];

void prepare\_input() {

const int PRECISION = 1000000;

const double RANGE = LIMIT / 20.0;

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

x[i] = RANGE \* (rand() % PRECISION - rand() % PRECISION) / PRECISION;

}

}

//# BEGIN fast code

//# khai bÃ¡o cÃ¡c biáº¿n phá»¥ trá»£ cáº§n thiáº¿t

const double delta = 0.0001;

double sigmoid\_val[NUM\_ITER];

const double x0 = -5.0;

const double x\_end = 5.0;

//# hÃ m chuáº©n bá»‹ dá»¯ liá»‡u

void precalc() {

double tmp = x0;

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

sigmoid\_val[i] = sigmoid\_slow(tmp);

tmp += delta;

}

}

//# hÃ m tÃ­nh sigmoid(x) nhanh sigmoid\_fast(x)

inline double sigmoid\_fast(double x) {

if (x < x0) return (0.0);

if (x > x\_end) return (1.0);

int i = floor((x - x0) / delta);

return sigmoid\_val[i] + ((sigmoid\_val[i+1] - sigmoid\_val[i]) \* (x - x0 - i\*delta)) / delta;

}

//# END fast code

double benchmark(double (\*calc)(double), vector<double> &result) {

const int NUM\_TEST = 20;

double taken = 0;

result = vector<double>();

result.reserve(NUM\_ITER);

int input\_id = 0;

clock\_t start = clock();

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

double sum = 0;

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

double v = fabs(calc(x[input\_id]));

sum += v;

if (t == 0) result.push\_back(v);

if ((++input\_id) == NUM\_INPUTS) input\_id = 0;

}

}

clock\_t finish = clock();

taken = (double)(finish - start);

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

return taken;

}

bool is\_correct(const vector<double> &a, const vector<double> &b) {

const double EPS = 1e-6;

if (a.size() != b.size()) return false;

for (unsigned int i = 0; i < a.size(); ++i) {

if (fabs(a[i] - b[i]) > EPS) {

return false;

}

}

return true;

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

prepare\_input();

precalc();

vector<double> a, b;

double slow = benchmark(sigmoid\_slow, a);

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

double xval;

scanf("%lf", &xval);

printf("%.2f \n", sigmoid\_fast(xval));

if (is\_correct(a, b) && (slow/fast > 1.3)) {

printf("Correct answer! Your code is faster at least 30%%\n");

} else {

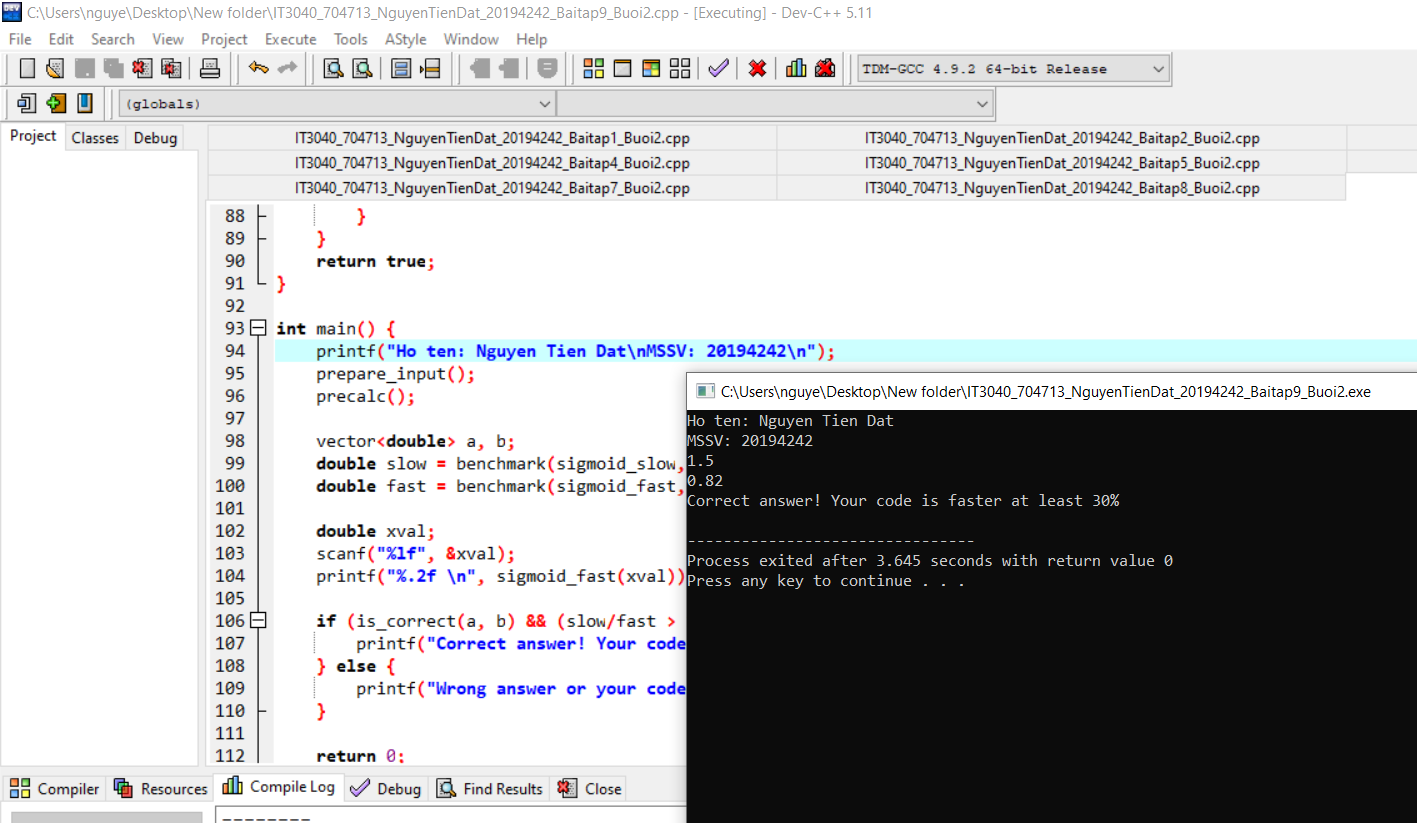
printf("Wrong answer or your code is not fast enough!\n");

}

return 0;

}

Kết quả:



### Bài tập 10 (bonus): Tính tích hai ma trận vuông

### Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxNNxN 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)O(N3).

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap10\_Buoi2.cpp

#include <iostream>

#include <cstring>

#include <ctime>

#include <cstdlib>

using namespace std;

const int N = 128;

struct Matrix {

unsigned int mat[N][N];

Matrix() {

memset(mat, 0, sizeof mat);

}

};

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;

}

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];

}

}

}

return c;

}

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;

}

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;

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;

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

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 {

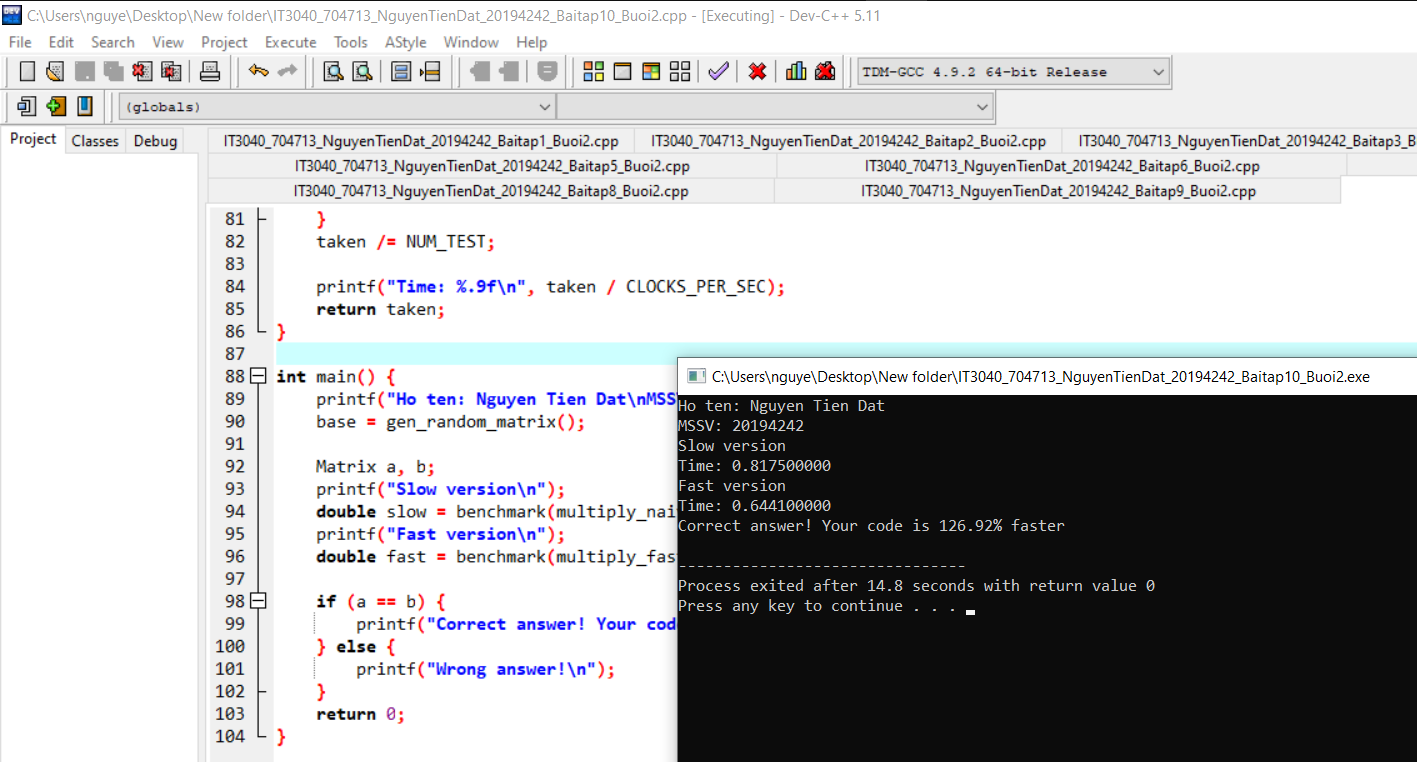
printf("Wrong answer!\n");

}

return 0;

}

Kết quả:



# Phần 3. Bài tập về nhà

### Bài tập 11: Tính tích hai đa thức

### Cho 2 đa thức A(x) và B(x) tương ứng có bậc NN và MM. Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc N+M−1N+M−1.

**Input:** Gồm 2 dòng biểu diễn các đa thức A(x) và B(x), mỗi dòng

* Số đầu tiên NN là bậc của đa thức;
* N+1N+1 số nguyên tiếp theo, số thứ i là hệ số của xi−1xi−1.

**Output:** Một số nguyên duy nhất là XOR của các hệ số của đa thức C(x).

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap11\_Buoi2.cpp

#include <iostream>

using namespace std;

struct Poly {

int bac;

int \*heso;

};

void input(Poly &poly) {

cin >> poly.bac;

poly.heso = new int[poly.bac + 1];

for (int i = 0; i <= poly.bac; i++) {

cin >> poly.heso[i];

}

}

void multi(Poly poly1, Poly poly2, Poly &res) {

res.bac = poly1.bac + poly2.bac;

res.heso = new int[res.bac + 1];

for (int i = 0; i <= res.bac; i++) {

res.heso[i] = 0;

for (int j = 0; j <= poly1.bac; j++) {

if (i-j >= 0 && i-j <= poly2.bac) res.heso[i] += poly1.heso[j] \* poly2.heso[i-j];

}

}

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

Poly p1, p2, result;

input(p1);

input(p2);

multi(p1,p2,result);

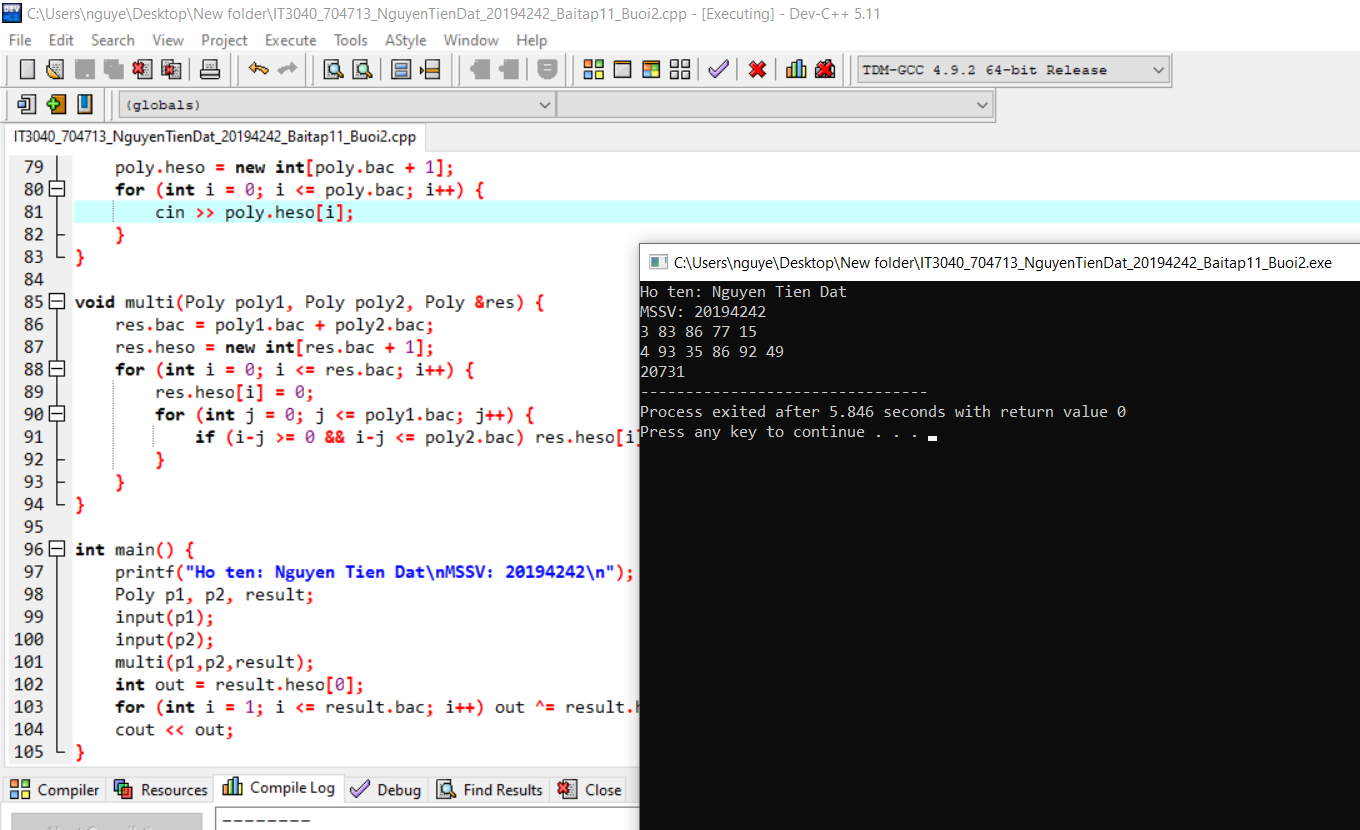
int out = result.heso[0];

for (int i = 1; i <= result.bac; i++) out ^= result.heso[i];

cout << out;

}

Kết quả:



### Bài tập 12: Map Sort

### 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≤109, |value| ≤109≤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.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap12\_Buoi2.cpp

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct datatype{

int value;

int key;

};

vector<datatype> a;

void print() {

for (const datatype &v : a) {

cout << v.key << " " << v.value << endl;

}

}

void input() {

int key\_tmp, value\_tmp;

while (cin >> key\_tmp && cin >> value\_tmp) {

datatype tmp;

tmp.key = key\_tmp;

tmp.value = value\_tmp;

a.push\_back(tmp);

}

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

input();

sort(a.begin(), a.end(), [] (datatype left, datatype right) {

if (left.value > right.value) return true;

else if (left.value < right.value) return false;

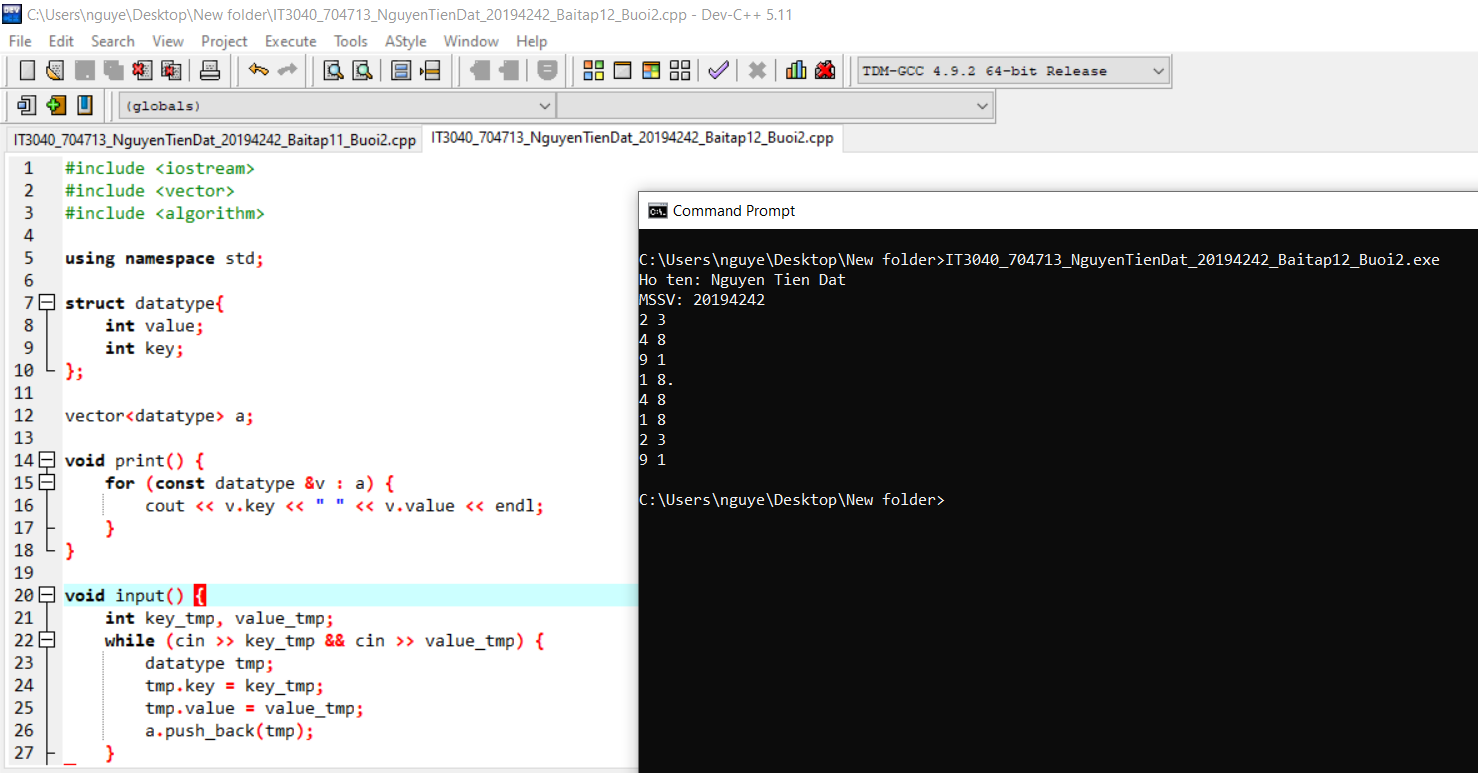
else return left.key >= right.key;

});

print();

}

Kết quả:



### Bài tập 13: Big Integer

### 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 aa và bb, mỗi dòng chứa 1 chuỗi ký tự mô tả 1 số nguyên lớn không vượt quá 1010010100. 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+4bab−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.

Tên file: IT3040\_704713\_NguyenTienDat\_20194242\_Baitap13\_Buoi2.cpp

#include <iostream>

using namespace std;

struct bigNum{

char sign;

char num[101];

};

void input(bigNum &number) {

string data;

cin >> data;

// 0 la am, 1 la duong

number.sign = data.at(0);

int length = data.length();

for (int i = 1; i < length; i++) number.num[100-length+i+1] = data.at(i);

for (int i = 0; i <= 100 - length+1; i++) number.num[i] = '0';

}

inline int charToInt(char c) { return (int)(c-48);}

inline char intToChar(int n) {return (char)(n+'0');}

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];

}

void add\_positive(char res[101], char \*a, char \*b) {

int carry = 0;

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

int tmp = charToInt(a[i]) + charToInt(b[i]) + carry;

res[i] = intToChar(tmp % 10);

carry = tmp / 10;

}

}

void sub\_positive(char res[101], char \*big, char \*small) {

int borrow = 0;

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

int tmp = charToInt(big[i]) - charToInt(small[i]) - borrow;

if (tmp < 0) {

tmp += 10;

borrow = 1;

}

else borrow = 0;

res[i] = intToChar(tmp);

}

}

// bigInt \* n

void multi\_positive\_const(char res[101], char \*a, unsigned int n) {

int carry = 0;

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

int tmp = charToInt(a[i]) \* n + carry;

res[i] = intToChar(tmp % 10);

carry = tmp / 10;

}

}

// so sanh abs(a) < abs(b) neu dung tra ve true

bool operator < (bigNum a, bigNum b) {

int i = 0;

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

if (a.num[i] != '0' && b.num[i] == '0') break;

if (a.num[i] == '0' && b.num[i] != '0') break;

if (a.num[i] != '0' && b.num[i] != '0' && b.num[i] != a.num[i]) break;

}

if (a.num[i] < b.num[i]) return true;

else return false;

}

bigNum operator + (bigNum a, bigNum b) {

bigNum result;

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

if (a.sign == '0') result.sign = '0';

else result.sign = '1';

add\_positive(result.num, a.num, b.num);

return result;

}

else {

// so duong nho a + so am lon b

if (a.sign == '1' && a < b) {

sub\_positive(result.num, b.num, a.num);

result.sign = '0';

return result;

}

// so duong lon a + so am nho b

else if (a.sign == '1' && !(a < b)) {

sub\_positive(result.num, a.num, b.num);

result.sign = '1';

return result;

}

// so am nho a + so duong lon b

else if (a.sign == '0' && a < b) {

sub\_positive(result.num, b.num, a.num);

result.sign = '1';

return result;

}

// so am lon a + so duong nho b

else {

sub\_positive(result.num, a.num, b.num);

result.sign = '0';

return result;

}

}

}

bigNum operator - (bigNum a, bigNum b) {

bigNum result;

if (a.sign != b.sign) {

if (a.sign == '0') result.sign = '0';

else result.sign = '1';

add\_positive(result.num, a.num, b.num);

return result;

}

else {

// so am nho a - so am lon b

if (a.sign == '0' && a < b) {

sub\_positive(result.num, b.num, a.num);

result.sign = '1';

return result;

}

// so am lon a - so am nho b

else if (a.sign == '0' && !(a < b)) {

sub\_positive(result.num, a.num, b.num);

result.sign = '0';

return result;

}

// so duong nho a - so duong lon b

else if (a.sign == '1' && a < b) {

sub\_positive(result.num, b.num, a.num);

result.sign = '0';

return result;

}

// so duong lon a - so duong nho b

else {

sub\_positive(result.num, a.num, b.num);

result.sign = '1';

return result;

}

}

}

bigNum operator \* (bigNum a, unsigned int n) {

bigNum result;

result.sign = a.sign;

multi\_positive\_const(result.num, a.num, n);

return result;

}

bigNum operator \* (bigNum a, bigNum b) {

bigNum result;

for (int i = 0; i <= 100; i++) result.num[i] = '0';

if (a.sign == b.sign) result.sign = '1';

else result.sign = '0';

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

char tmp[101];

// khoi tao tmp

for (int k = 0; k < 101; k++) tmp[k] = '0';

int carry = 0;

int k = i;

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

int tmp\_num = charToInt(a.num[j]) \* charToInt(b.num[i]) + carry;

tmp[k] = intToChar(tmp\_num % 10);

carry = tmp\_num / 10;

k--; if (k < 0) break;

}

add\_positive(result.num, result.num, tmp);

}

return result;

}

int main() {

printf("Ho ten: Nguyen Tien Dat\nMSSV: 20194242\n");

bigNum num1, num2, result;

input(num1);

input(num2);

result = num1\*num2 - num1\*3 + num2\*4;

printBigNumber(result);

return 0;

}

Kết quả:

