**Danh mục**

[**Phần 1. Truyền tham trị, tham chiếu và tham số ngầm định**](#_heading=h.1fob9te) 5

[**Bài tập 1**: Viết hàm tính độ dài cạnh huyền của tam giác từ 2 cạnh của góc vuông 5](#_heading=h.3znysh7)

[**Bài tập 2**: Viết 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. 6](#_heading=h.tyjcwt)

[**Bài tập 3:** Viết chương trình nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax^2 + bx + c với a, b, c định sẵn. 9](#_heading=h.3dy6vkm)

[**Phần 2. Đa năng hóa hàm**](#_heading=h.1t3h5sf) 11

[**Bài tập 4**: Viết hàm tính lập phương của số nguyên và số thực. 11](#_heading=h.4d34og8)

[**Bài tập 5:** Viết toán tử tính tổng, hiệu, tích, thương của 2 số phức 13](#_heading=h.2s8eyo1)

[**Phần 3. Con trỏ hàm và tham số hóa hàm**](#_heading=h.ch8bf4vta3hz) 17

[**Bài tập 6**: Giả thuyết Collatz : bắt đầu từ số dương n bất kì, nếu n chẵn thì chia 2, n 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 kiểm chứng giả thuyết với giá trị n nhập từ bàn phím. 17](#_heading=h.17dp8vu)

[**Bài tập 7**: Viết hàm tính tổng các phần tử trong 2 mảng. Yêu cầu sử dụng function template để cho phép hàm làm việc với số nguyên lẫn số thực. 21](#_heading=h.26in1rg)

[**Phần 4. Biểu thức lamda và hàm nặc danh**](#_heading=h.uhn7dfq7ivg5) 23

[**Bài tập 8**: Viết hàm so sánh cho thuật toán sắp xếp 23](#_heading=h.lnxbz9)

[**Phần 5. Thực hành về tối ưu mã nguồn**](#_heading=h.35nkun2) 26

[**Bài tập 9**: Dưới đây cung cấp đoạn code đơn giản để tính hàm sigmoid theo công thức trực tiếp. Hãy viết hàm sấp xỉ sigmoid(x) đến độ chính xác 10^(-6) và có tốc độ nhanh ít nhất 30% so với code đơn giản. 26](#_heading=h.1ksv4uv)

[**Bài tập 10**: Dưới đây cung cấp đoạn code đơn giản để tính tích của 2 ma trận. Hãy viết hàm tính tích 2 ma trận nhưng có tốc độ nhanh ít nhất hơn 10% so với code đơn giản. 32](#_heading=h.44sinio)

[**Phần 6. Bài tập về nhà**](#_heading=h.4vpg1xrhw0zp) 37

[**Bài tập 11**: Cho 2 đa thức A(x) và B(x) có bậc tương ứng là M và N. Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc là M+N-1. 37](#_heading=h.zfli054kapxz)

[**Bài tập 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. 42](#_heading=h.v2b2d6vfimsi)

[**Bài tập 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. 44](#_heading=h.libz8ngisv1h)

Danh mục hình ảnh

[Hình 1 Bài 2.1 In ra cạnh huyền của tam giác 4](#_Toc56171952)

[Hình 2 Bài 2.2 In ra hoán vị các chữ số sau khi thực hiện hàm 6](#_Toc56171953)

[Hình 3 Bài 2.3 In ra giá trị của hàm số 8](#_Toc56171954)

[Hình 4 Bài 2.4 Hàm tính lập phương của số nguyên và số thực 10](#_Toc56171955)

[Hình 5 Bài 2.5 Hàm tính tổng, hiệu, tích, thương của 2 số phức 14](#_Toc56171956)

[Hình 6 Bài 2.6 : Kiểm chứng định lý collatz 16](#_Toc56171957)

[Hình 7 Bài 2.7 In ra tổng của 2 mảng số nguyên và số thực 19](#_Toc56171958)

[Hình 8 Bài 2.8 Hàm so sánh cho thuật toán sắp xếp 21](#_Toc56171959)

[Hình 9 Bài 2.9 Hàm sigmoid 27](#_Toc56171960)

[Hình 10 Bài 2.10 In ra thời gian tính toán của 2 thuật toán tính tích 2 ma trận 32](#_Toc56171961)

[Hình 11 Bài 2.11 In ra tích của 2 đa thức 36](#_Toc56171962)

[Hình 12 Bài 2.12 In ra map đã sắp xếp 39](#_Toc56171963)

[Hình 13 Bài 2.13 Xử lý số nguyên lớn 49](#_Toc56171964)

# Bài thực hành số 2: Truyền tham trị, tham chiếu và tham số ngầm định

## Phần 1. Truyền tham trị, tham chiếu và tham số ngầm định

## Bài tập 1: Viết hàm tính độ dài cạnh huyền của tam giác từ 2 cạnh của góc vuông

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_1.cpp**

#include <stdio.h>

#include <math.h>

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

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

return z;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

int main(){

printf("Ho Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\n");

float x, y;

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

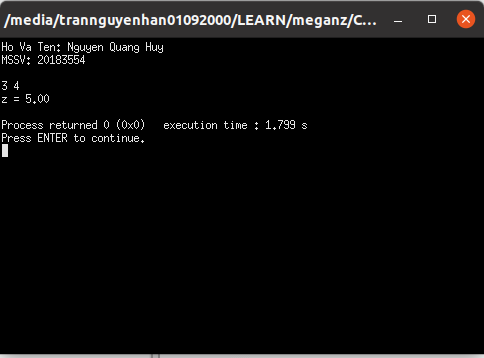
float z = get\_hypotenuse(x, y);

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

return 0;

}

Kết quả:



Hình 1 Bài 2.1 In ra cạnh huyền của tam giác

## Bài tập 2: Viết 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: 20183554-NguyenQuangHuy\_Bai 2\_2.cpp

#include <stdio.h>

#include<bits/stdc++.h>

using namespace std;

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

int tmp = x;

x = y;

y = z;

z = tmp;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

int main() {

printf("Ho Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\n");

int x, y, z;

//# Nhập 3 số nguyên

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cin >> x >> y >> z;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

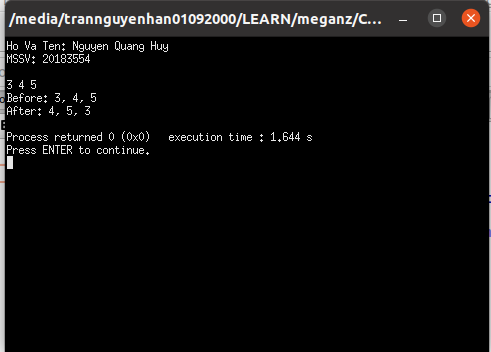
rotate(x, y, z);

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

return 0;

}

### Kết quả:



Hình 2 Bài 2.2 In ra hoán vị các chữ số sau khi thực hiện hàm

## Bài tập 3: Viết chương trình nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax^2 + bx + c với a, b, c định sẵn.

### Tên file: 20183554-NguyenQuangHuy\_Bai 2\_3.cpp

### #include <stdio.h>

### #include <bits/stdc++.h>

### using namespace std;

### 

### //# Viết hàm get\_value

### /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Ho va ten : Nguyen Quang Huy

### MSSV : 20183554

### \*/

### int get\_value(int x, int a=2, int b=1, int c=0){

### int S = a\*x\*x + b\*x + c;

### return S;

### }

### /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 

### int main(){

### printf("Ho Va Ten: Nguyen Quang Huy\n");

### printf("MSSV: 20183554\n\n");

### int x;

### scanf("%d", &x);

### 

### int a = 2; //# giá trị mặc định của a

### int b = 1; //# giá trị mặc định của b

### int c = 0; //# giá trị mặc định của c

### 

### //# Nhập 3 số nguyên a, b, c từ bàn phím

### /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### cin >> 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));

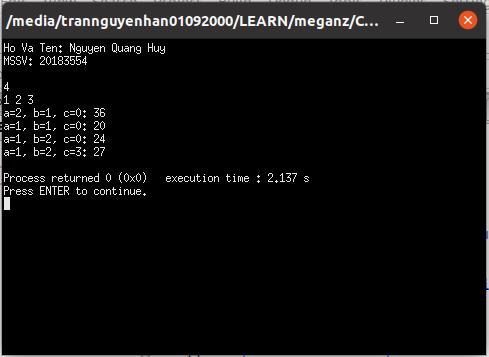
### printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get\_value(x, a, b, c));

### 

### return 0;

### }

### Kết quả:



Hình 3 Bài 2.3 In ra giá trị của hàm số

## Phần 2. Đa năng hóa hàm

## Bài tập 4: Viết hàm tính lập phương của số nguyên và số thực.

### Tên file: 20183554-NguyenQuangHuy\_Bai 2\_4.cpp

#include <stdio.h>

int cube(int x) {

//# trả về lập phương của x

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten :Nguyen Quang Huy

MSSV : 20183554

\*/

return x\*x\*x;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

//# viết hàm tính lập phương của một số kiểu double

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double cube(double x){

return x\*x\*x;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

printf("Ho Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\n");

int n;

double f;

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

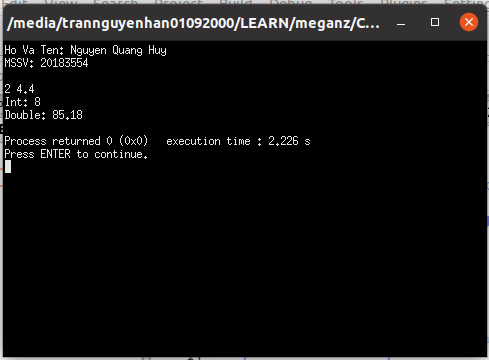
printf("Int: %d\n", cube(n));

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

return 0;

}

Kết quả :



Hình 4 Bài 2.4 Hàm tính lập phương của số nguyên và số thực

## Bài tập 5: Viết toán tử tính tổng, hiệu, tích, thương của 2 số phức

### Tên file: 20183554-NguyenQuangHuy\_Bai 2\_5.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) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

Complex tmpC;

tmpC.real = a.real + b.real;

tmpC.imag = a.imag + b.imag;

return tmpC;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

Complex operator - (Complex a, Complex b) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Complex tmpC;

tmpC.real = a.real - b.real;

tmpC.imag = a.imag - b.imag;

return tmpC;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Complex tmpC;

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

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

return tmpC;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

Complex operator / (Complex a, Complex b) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Complex tmpC;

Complex inverse;

inverse.real = b.real;

inverse.imag = -b.imag;

tmpC = a \* inverse;

tmpC.real = tmpC.real / (b.real \* b.real + b.imag \* b.imag);

tmpC.imag = tmpC.imag / (b.real \* b.real + b.imag \* b.imag);

return tmpC;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

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 Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\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;

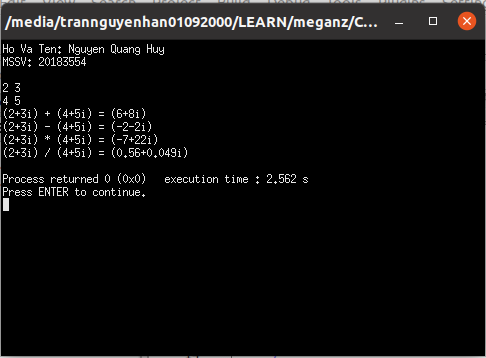
cout << a << " \* " << b << " = " << a \* b << endl;

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

return 0;

}

Kết quả:



Hình 5 Bài 2.5 Hàm tính tổng, hiệu, tích, thương của 2 số phức

## Phần 3. Con trỏ hàm và tham số hóa hàm

## Bài tập 6: Giả thuyết Collatz : bắt đầu từ số dương n bất kì, nếu n chẵn thì chia 2, n 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 kiểm chứng giả thuyết với giá trị n nhập từ bàn phím.

### Tên file: 20183554-NguyenQuangHuy\_Bai 2\_6.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 báo các tham số cho các con trỏ hàm odd, even và 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 Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\n");

int (\*odd)(int) = NULL;

int (\*even)(int) = NULL;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

odd = mul3plus1;

even = div2;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int n;

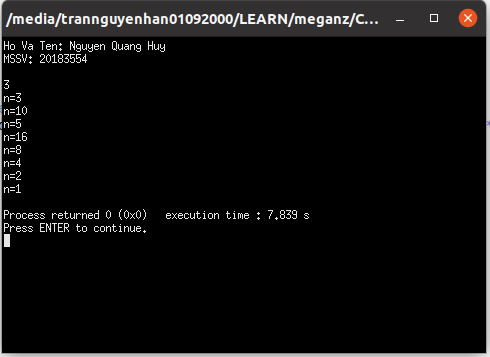
scanf("%d", &n);

simulate(n, odd, even, print);

return 0;

}

Đáp án:



Hình 6 Bài 2.6 : Kiểm chứng định lý collatz

## Bài tập 7: Viết hàm tính tổng các phần tử trong 2 mảng. Yêu cầu sử dụng function template để cho phép hàm làm việc với số nguyên lẫn số thực.

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_7.cpp**

#include <iostream>

using namespace std;

//# viết hàm arr\_sum

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

template <typename T>

T arr\_sum(T a[], int n, T b[], int m){

T sum;

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

sum += a[i];

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

sum += b[i];

return sum;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

printf("Ho Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\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};

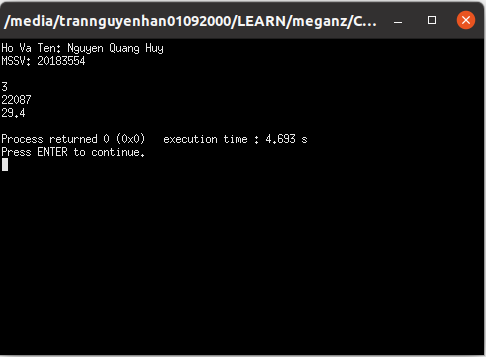
cout << arr\_sum(a, 4, b, 5) << endl;

}

return 0;

}

Đáp án:



Hình 7 Bài 2.7 In ra tổng của 2 mảng số nguyên và số thực

## Phần 4. Biểu thức lamda và hàm nặc danh

## Bài tập 8: Viết hàm so sánh cho thuật toán sắp xếp

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_8.cpp**

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

int main() {

printf("Ho Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\n");

int val1, val2;

cin >> val1 >> val2;

vector< vector<int> > a = {

{1, 3, 7},

{2, 3, 4, val1},

{9, 8, 15},

{10, val2},

};

//# sắp xếp các vector trong a theo tổng các phần tử giảm dần

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

sort(a.begin(), a.end(), [](vector<int> x, vector<int> y){

int sum1 = 0;

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

sum1 += x[i];

int sum2 = 0;

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

sum2 += y[i];

return sum1 > sum2;

});

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (const auto &v : a) {

for (int it : v) {

cout << it << ' ';

}

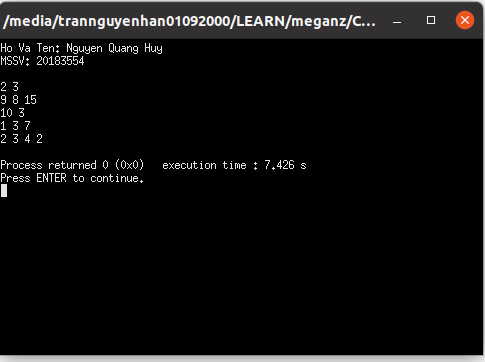
cout << endl;

}

return 0;

}

Kết quả:



Hình 8 Bài 2.8 Hàm so sánh cho thuật toán sắp xếp

## Phần 5. Thực hành về tối ưu mã nguồn

## Bài tập 9: Dưới đây cung cấp đoạn code đơn giản để tính hàm sigmoid theo công thức trực tiếp. Hãy viết hàm sấp xỉ sigmoid(x) đến độ chính xác 10^(-6) và có tốc độ nhanh ít nhất 30% so với code đơn giản.

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_9.cpp**

#include <vector>

#include <algorithm>

#include <cmath>

#include <ctime>

#include <algorithm>

#include <cstdio>

#include<iostream>

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 báo các biến phụ trợ cần thiết

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define MAX\_N 100000

#define denta 0.0001

double sigmoid[MAX\_N];

const double start = -5.0;

const double stop = 5.0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//# hàm chuẩn bị dữ liệu

void precalc() {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

double foo = start;

for(int i=0; i<MAX\_N; i++){

sigmoid[i] = sigmoid\_slow(foo);

foo += denta;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

//# hàm tính sigmoid(x) nhanh sigmoid\_fast(x)

inline double sigmoid\_fast(double x) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(x < start) return 0.0;

if(x > stop) return 1.0;

int i = floor((x - start) / denta);

return sigmoid[i] + ((sigmoid[i+1] - sigmoid[i]) \* (x - start - i\*denta)) / (denta);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

//# 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 Va Ten: Nguyen Quang Huy\n");

printf("MSSV: 20183554\n\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 {

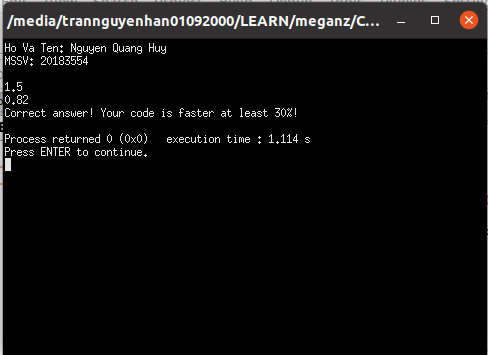
printf("Wrong answer or your code is not fast enough!\n");

}

return 0;

}

Kết quả:



Hình 9 Bài 2.9 Hàm sigmoid

## Bài tập 10: Dưới đây cung cấp đoạn code đơn giản để tính tích của 2 ma trận. Hãy viết hàm tính tích 2 ma trận nhưng có tốc độ nhanh ít nhất hơn 10% so với code đơn giản.

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_10.cpp**

#include <iostream>

#include <cstring>

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) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

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() {

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 {

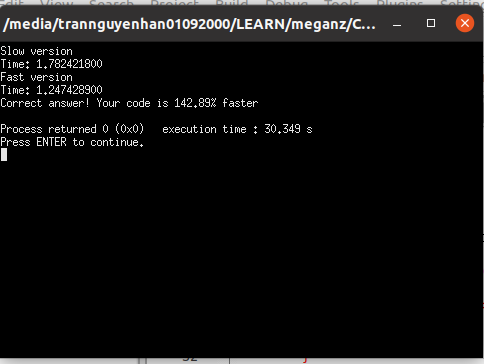
printf("Wrong answer!\n");

}

return 0;

}

Kết quả:



Hình 10 Bài 2.10 In ra thời gian tính toán của 2 thuật toán tính tích 2 ma trận

## Phần 6. Bài tập về nhà

## Bài tập 11: Cho 2 đa thức A(x) và B(x) có bậc tương ứng là M và N. Hãy tính ma trận tích C(x) = A(x) \* B(x) có bậc là M+N-1.

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_11.cpp**

#include<bits/stdc++.h>

using namespace std;

typedef complex<double> base;

typedef vector<base> vb;

void input(int &n, int &m, vb &x, vb &y){

cin >> n;

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

int tmp;

cin >> tmp;

base mycomplex(tmp,0);

x.push\_back(mycomplex);

}

cin >> m;

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

int tmp;

cin >> tmp;

base mycomplex(tmp,0);

y.push\_back(mycomplex);

}

}

void fft(vb & a, bool revert){

int n = (int)a.size();

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

int bit = n >> 1;

while(j>=bit){

j =j- bit;

bit=bit>>1;

}

j =j+ bit;

if (i < j)

swap (a[i], a[j]);

}

for(int len=2; len<=n; len<<=1){

double ang = 2\*M\_PI/len;

if(revert==1) ang=-ang;

complex<double> wlen (cos(ang), sin(ang));

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

complex<double> w (1);

for (int j=0; j<len/2; ++j) {

complex<double> u = a[i+j], v = a[i+j+len/2] \* w;

a[i+j] = u + v;

a[i+j+len/2] = u - v;

w =w \* wlen;

}

}

}

if(revert==1)

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

a[i] /=n;

}

int multiPly(int n, int m, vb x, vb y){

int p = 1;

while(p < max (n, m)) p=p<<1;

p=p<<1;

x.resize(p);

y.resize(p);

fft(x, false);

fft(y, false);

vector<base> h(p);

for (int i=0; i<p; i++)

h[i] =x[i]\*y[i];

fft (h, true);

int res = (int)(real(h[0])+0.5);

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

res = res ^ (int)(real(h[i])+0.5);

}

return res;

}

int main(){

int n, m;

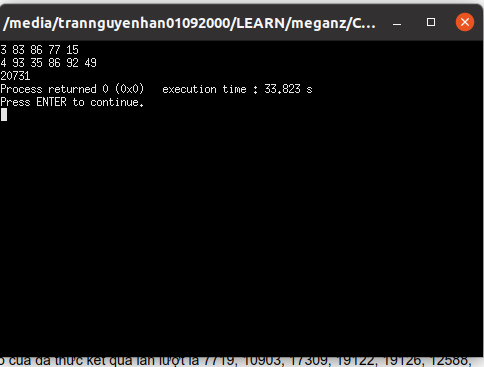
vector<base> x,y;

input(n,m,x,y);

cout << multiPly(n,m,x,y);

}

Kết quả :



Hình 11 Bài 2.11 In ra tích của 2 đa thức

## Bài tập 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.

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_12.cpp**

#include<bits/stdc++.h>

using namespace std;

struct element {

int key;

int value;

};

vector<element> lst;

void input(){

int tmp1, tmp2;

while(cin >> tmp1 && cin >> tmp2){

element tmp;

tmp.key = tmp1;

tmp.value = tmp2;

lst.push\_back(tmp);

}

}

void print(){

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

cout << lst[i].key << " " << lst[i].value << endl;

}

}

int main(){

input();

sort(lst.begin(),lst.end(),[] (element a, element b){

if(a.value > b.value) return true;

else if (a.value < b.value) return false;

else {

return a.key >= b.key;

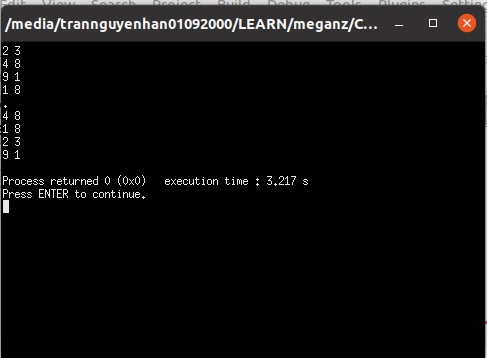
}

});

print();

}

Kết quả :



Hình 12 Bài 2.12 In ra map đã sắp xếp

## Bài tập 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.

**Tên file: 20183554-NguyenQuangHuy\_Bai 2\_13.cpp**

#include<bits/stdc++.h>

using namespace std;

struct bigNum {

char sign;

char num[101];

};

// input and preprocess data

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';

}

// add 2 positive big number

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;

}

}

// sub 2 positive big number, num1 > num2

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;

}

}

}

// multi 2 positive big number

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

// clear array res

for(int i=0; i<101; i++) res[i] = '0';

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

// init 1 array temp

char tmp[101];

// add i number 0 to last array

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);

}

}

// check number1 >= number2

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 { // foo1 == foo2

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;

}

// overloading operator "+"

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;

}

}

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;

}

}

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 bignumber

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];

}

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;

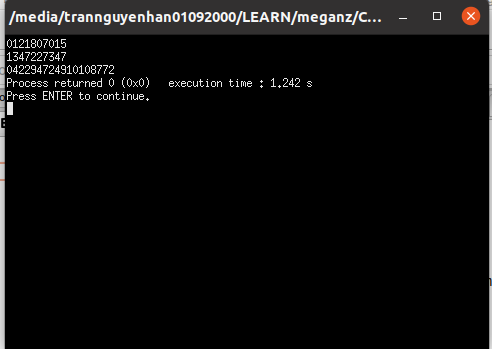
so4.num[100] = 4 + 48;

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

printBigNumber(res);

}

Kết quả:



Hình 13 Bài 2.13 Xử lý số nguyên lớn