**CÁCH ĐÁNH GIÁ ĐIỂM THỰC HÀNH**

**HỌC PHẦN: IT3040 - KỸ THUẬT LẬP TRÌNH – 2022.2**

1. **Quy định, yêu cầu:** 
   * Tài liệu và nội dung thực hành chấm điểm trên hệ thống:

https://lab.soict.hust.edu.vn/

* + Bài tập trên lớp chấm điểm tự động (các bài không chấm trên hệ thống làm vào máy tính 🡺 làm báo cáo thực hành – Theo mẫu).
  + Hạn nộp báo cáo trên Teams (Bài tập trên lớp + Bài tập về nhà): 1 tuần.

1. **Đánh giá điểm thực hành**

1. Chuyên cần (đúng giờ, nghiêm túc trong giờ học) - Điểm danh trên Teams: 10%

2. Báo cáo thực hành (bài tập trên lớp + Về nhà) theo mẫu nộp trên Teams: 40%

3. Trắc nghiệm – Form trên Teams: 10%

4. Kiểm tra thực hành: 40%. (Tiết 2,3 buổi thực hành thứ 5).

**Điểm thưởng: 5% 🡪 10% (Cho Mục 1,2 điểm TB từ 9-10).**

Tham gia thực hành đúng giờ đầy đủ theo thời khóa biểu (nếu có lý do không đi thực hành đúng kíp được thì gửi mail xin phép thực hành bù trước 1 ngày qua mail [hoalt@soict.hust.edu.vn](mailto:hoalt@soict.hust.edu.vn), Tiêu đề: đăng ký học bù – IT3040 – MaLopTH.

Các kíp có thể bù:

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Thời gian, địa điểm, Tuần học** | **Mã nhóm** | **Mã lớp** |
| **1** | Sáng T4: Tiết 645-915 , Địa điểm: B1-203, Tuần: 32,34,36,38,40 |  | 727638 |
| **2** | Sáng T4: Tiết 645-915 , Địa điểm: B1-203, Tuần: 31,35,37,39,41 |  | 727634 |
| **3** | Sáng T4: Tiết 915-1145 , Địa điểm: B1-203, Tuần: 31,35,37,39,41 |  | 727635 |
| **4** | Sáng T4: Tiết 915-1145 , Địa điểm: B1-203, Tuần: 32,34,36,38,40 |  | 727639 |

**Nếu nghỉ không có lý do 3 buổi, không thực hành bù thì điểm chuyên cần, báo cáo và BTVN coi như 0 điểm thực hành.**

Contents

[Bài thực hành số 2 – Tuần 34 3](#_Toc136002697)

[**Bài tập 2.1.** Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông. 3](#_Toc136002698)

[**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](#_Toc136002699)

[**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. 8](#_Toc136002700)

[**Bài tập 2.4**. Viết các hàm tính lập phương của số nguyên và số thực. 10](#_Toc136002701)

[**Bài tập 2.5**. Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức 12](#_Toc136002702)

[**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. 17](#_Toc136002703)

[**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. 20](#_Toc136002704)

[**Bài tập 2.8**. Viết hàm so sánh cho thuật toán sắp xếp. 22](#_Toc136002705)

[**Bài tập 2.9**. 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 10−6 và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản. 25](#_Toc136002706)

[**Bài tập 1.10**. Viết chương trình nhập vào 2 ma trận vuông cùng kích thước n\*n, trong đó n nhập từ bàn phím. Sau đó tính tổng và tích của hai ma trận đó và đưa kết quả ra màn hình. Yêu cầu sử dụng cấp phát động để cấp phát bộ nhớ cho các ma trận. 32](#_Toc136002707)

# Bài thực hành số 2 – Tuần 34

## **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 with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

//Nguyen Huu Dung - 20215545

/\* Bài 2.1. Viết hàm tính độ dài cạnh huyền của

tam giác theo độ hai cạnh góc vuông.

\*/

#include <stdio.h>

#include <math.h>

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

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

//# Nguyen Huu Dung - 20215545 #

return sqrt(x\*x+y\*y); //Tinh canh huyen

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

}

int main(){

float x, y;

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

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

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

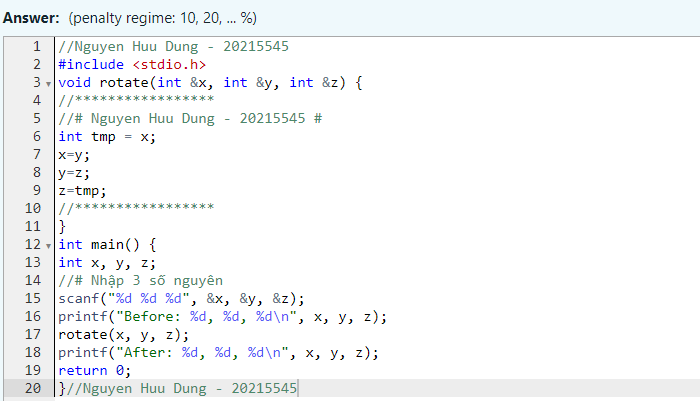
return 0;

} //Nguyen Huu Dung – 20215545

## **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



A screenshot of a computer

Description automatically generated

//Nguyen Huu Dung - 20215545

#include <stdio.h>

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

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

//# Nguyen Huu Dung - 20215545 #

int tmp = x;

x=y;

y=z;

z=tmp;

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

}

int main() {

int x, y, z;

//# Nhập 3 số nguyên

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

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

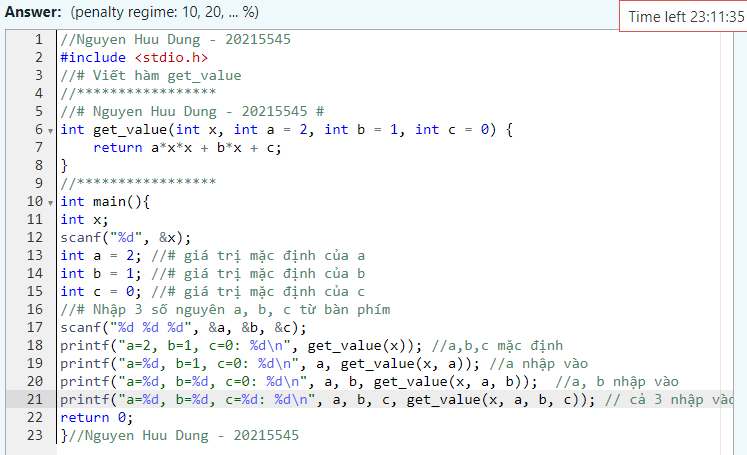
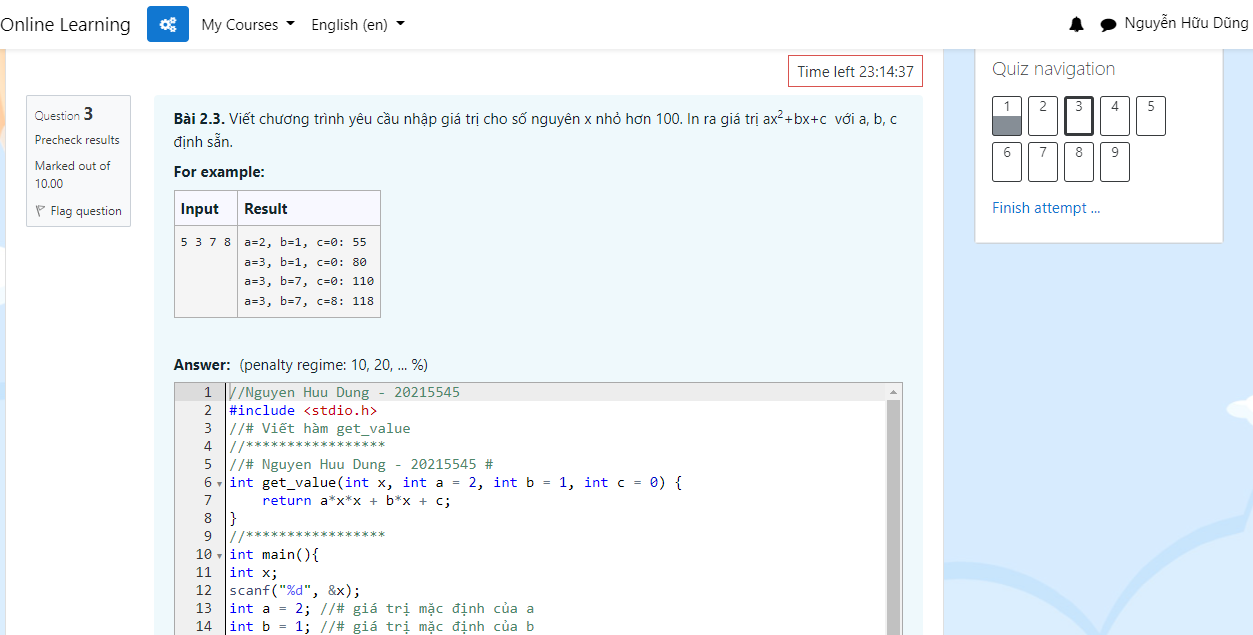
rotate(x, y, z);

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

return 0;

}//Nguyen Huu Dung - 20215545

## **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

Description automatically generated

//Nguyen Huu Dung - 20215545

#include <stdio.h>

//# Viết hàm get\_value

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

//# Nguyen Huu Dung - 20215545 #

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

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

}

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

int main(){

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

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

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

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

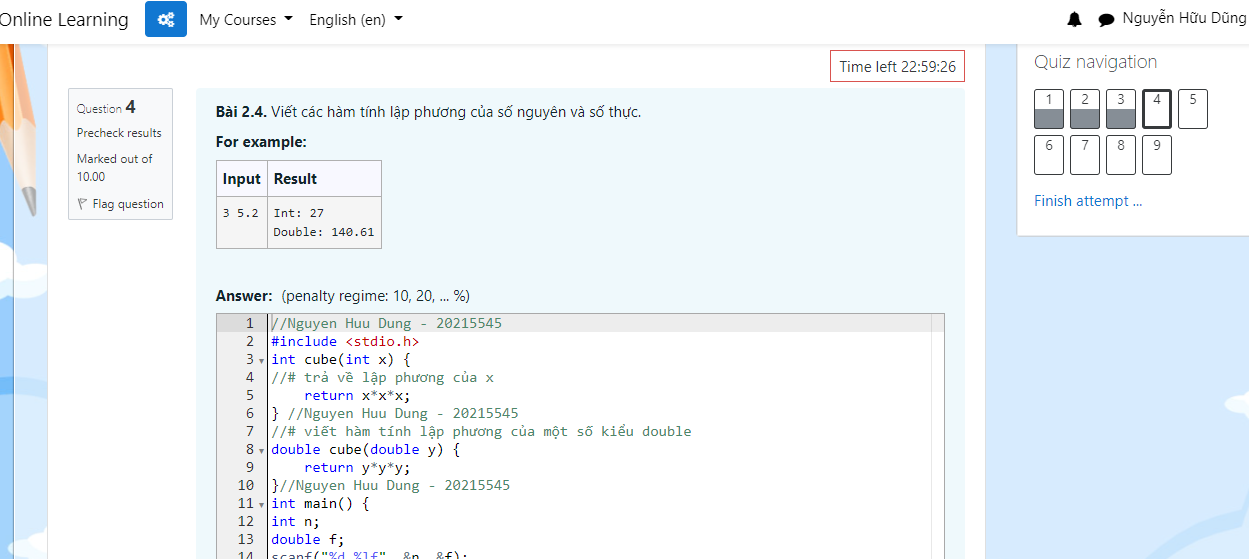
printf("a=%d, b=%d, c=0: %d\n", a, b, get\_value(x, a, b)); //a, b nhập vào

printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get\_value(x, a, b, c)); // cả 3 nhập vào

return 0;

}//Nguyen Huu Dung – 20215545

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



A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

//Nguyen Huu Dung - 20215545

#include <stdio.h>

int cube(int x) {

//# trả về lập phương của x

return x\*x\*x;

} //Nguyen Huu Dung - 20215545

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

double cube(double y) {

return y\*y\*y;

}//Nguyen Huu Dung - 20215545

int main() {

int n;

double f;

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

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

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

return 0;

}//Nguyen Huu Dung – 20215545

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

A screenshot of a computer

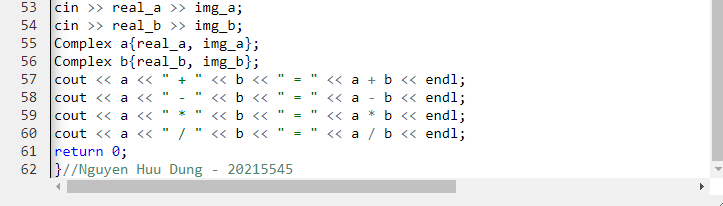
Description automatically generated

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence



A screenshot of a computer

Description automatically generated with medium confidence

//Nguyen Huu Dung - 20215545

#include <iostream>

#include <ostream>

#include <math.h>

#include <iomanip>

using namespace std;

struct Complex {

double real;

double imag;

};

Complex operator + (Complex a, Complex b) {

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Tinh tong hai so phuc

//# Nguyen Huu Dung - 20215545 #

Complex tmp;

tmp.real = a.real + b.real;

tmp.imag = a.imag + b.imag;

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

return tmp;

}

Complex operator - (Complex a, Complex b) {

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Tinh hieu hai so phuc

//# Nguyen Huu Dung - 20215545 #

Complex tmp;

tmp.real = a.real - b.real;

tmp.imag = a.imag - b.imag;

return tmp;

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

}

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Tinh tich hai so phuc

//# Nguyen Huu Dung - 20215545 #

Complex tmp;

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

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

return tmp;

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

}

Complex operator / (Complex a, Complex b) {

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Tinh thuong hai so phuc

//# Nguyen Huu Dung - 20215545 #

Complex tmp;

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

tmp.imag = (a.imag\*b.real - a.real\*b.imag) / (b.real\*b.real + b.imag\*b.imag);

return tmp;

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

}

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() {

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;

}//Nguyen Huu Dung – 20215545

## **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 with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a cell phone

Description automatically generated with low confidence

//Nguyen Huu Dung - 20215545

#include <stdio.h>

void print(int n) {

printf("n=%d\n", n); //Hàm in ra n

}

int mul3plus1(int n) {

return n \* 3 + 1; //Hàm trả về 3n+1

}

int div2(int n) { //Hàm trả về n/2

return n / 2;

}

// khai báo các tham số cho các con trỏ hàm odd, even và output

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

(\*output)(n);

if (n == 1) return; //Nếu n=1 thì in thoát hàm

if (n % 2 == 0) { //Nếu n chẵn

n = (\*even)(n);

} else {

n = (\*odd)(n); //Nếu n lẻ

}

simulate(n, odd, even, output); //Tiếp tục đệ quy

}

int main() {

int (\*odd)(int) = mul3plus1; //Gán con trỏ cho hàm tương ứng

int (\*even)(int) = div2;

/\*Nguyen Huu Dung - 20215545\*/

int n;

scanf("%d", &n);

simulate(n, odd, even, print);

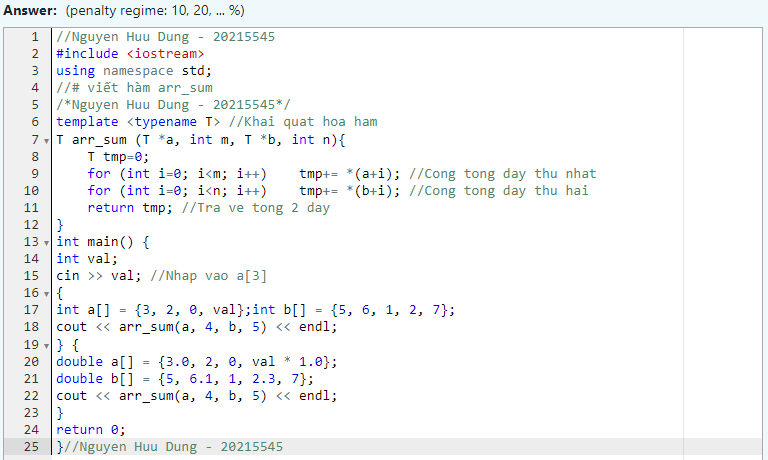
return 0;

} //Nguyen Huu Dung – 20215545

## **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 with medium confidence



A screenshot of a computer

Description automatically generated

//Nguyen Huu Dung - 20215545

#include <iostream>

using namespace std;

//# viết hàm arr\_sum

/\*Nguyen Huu Dung - 20215545\*/

template <typename T> //Khai quat hoa ham

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

T tmp=0;

for (int i=0; i<m; i++) tmp+= \*(a+i); //Cong tong day thu nhat

for (int i=0; i<n; i++) tmp+= \*(b+i); //Cong tong day thu hai

return tmp; //Tra ve tong 2 day

}

int main() {

int val;

cin >> val; //Nhap vao a[3]

{

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;

}//Nguyen Huu Dung – 20215545

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

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

//Nguyen Huu Dung - 20215545

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

int main() {

int val1, val2;

cin >> val1 >> val2;

vector< vector<int> > a = { //Tao Vector chua Vector trong phan tu

{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

/\*Nguyen Huu Dung - 20215545\*/

for (int i=0; i<(int)a.size()-1; i++ ) {

for (int j=i; j<(int)a.size(); j++)

if (accumulate(a[i].begin(), a[i].end(), 0) < accumulate(a[j].begin(), a[j].end(), 0)) {

vector <int> tmp = a[i];

a[i] = a[j];

a[j] = tmp;

} /\*Neu tong cua Vector phia truoc ma

be hon tong cua Vector phia sau thi hoan doi cho nhau.

accumulate (vt.begin(),vt.end(),0): tinh tong Vector\*/

}

for (const auto &v : a) { //duyet tung vector trong a

for (int it : v) { //duyet tung phan tu trong vector

cout << it << ' ';

}

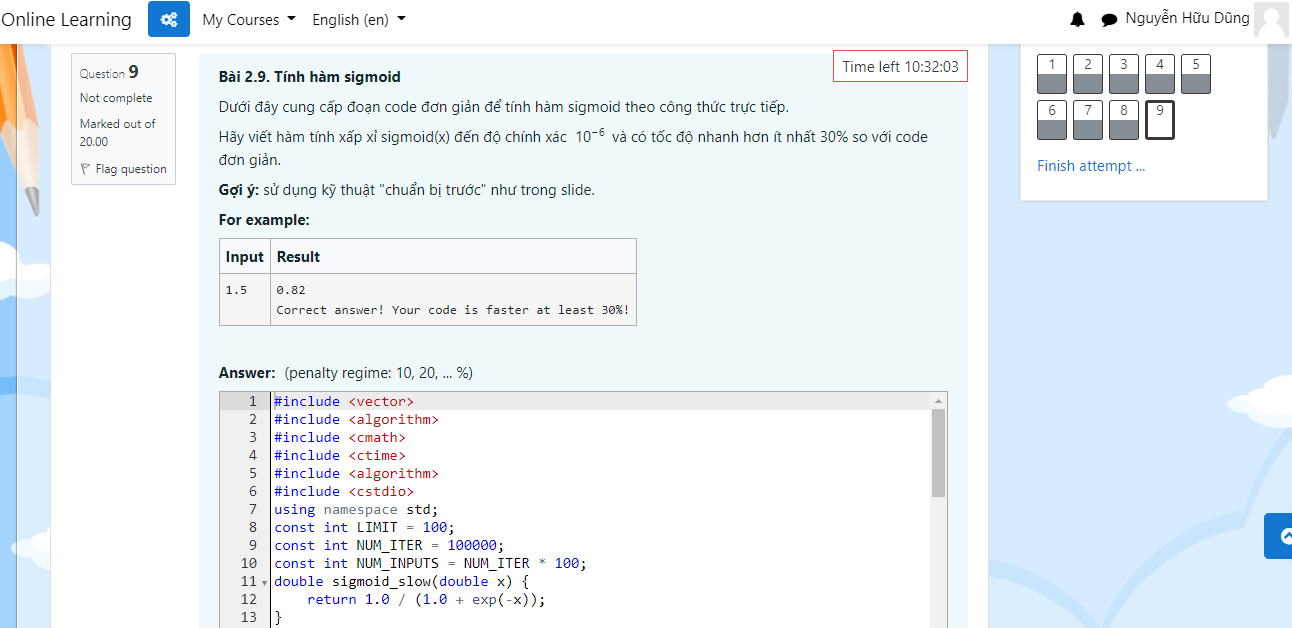
cout << endl;

}

return 0;

}//Nguyen Huu Dung – 20215545

## **Bài tập 2.9**. 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 10-6 và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản.



A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer code

Description automatically generated with medium confidence

A screenshot of a computer error

Description automatically generated with medium confidence

#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 báo các biến phụ trợ cần thiết

/\*Nguyen Huu Dung - 20215545\*/

int N=20000;

double a[20000], Delta = (double)28.0/N;

//# hàm chuẩn bị dữ liệu

void precalc() {

/\*Nguyen Huu Dung - 20215545\*/

for (int i=1; i<N-1; i++) {

a[i]=1.0/(1.0+exp(14-i\*Delta));

}

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

inline double sigmoid\_fast(double x) {

/\*Nguyen Huu Dung - 20215545\*/

if (x>=14) return 1;

else if (x<=-14) return 0;

else {

int i = (x+14)/Delta;

return a[i]+a[i]\*(1-a[i])\*(x+14-Delta\*i);

}

}

//# 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 (int i = 0; i < (int)a.size(); ++i) {

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

return false;

}

}

return true;

}

int main() {

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;

} //Nguyen Huu Dung - 20215545

## **Bài tập 2.10**. 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.

A picture containing text, screenshot, software, multimedia software

Description automatically generatedA picture containing text, screenshot, software, multimedia software

Description automatically generatedA screenshot of a computer program

Description automatically generated with medium confidence

//Nguyen Huu Dung - 20215545

#include <iostream>

#include <cstring>

#include <ctime>

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) { //Ham so sanh hai ma tran

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) { //Ham tinh tich 2 ma tran cach thong thuong

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) { //Ham tinh nhanh tich 2 ma tran

/\*Nguyen Huu Dung - 20215545\*/

Matrix c;

for (int i = 0; i < 128; ++i) {

for (int j = 0; j < 128; ++j) {

int sum = 0;

for (int k = 0; k < 128; ++k) {

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

}

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

}

}

return c;

}

Matrix gen\_random\_matrix() { //Tao ngau nhien ma tran

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) { //Ham do toc do tinh toan

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;

}//Nguyen Huu Dung – 20215545

## **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.

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).A screen shot of a computer program

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

//Nguyen Huu Dung - 20215545

#include <iostream>

#include <cmath>

using namespace std; //Khai bao thu vien

int main(){

int N, M;

cin>>N;

int a[N+1]; //Mang a chua he so cua A(x) co N+1 phan tu

for (int i = 0; i <= N; i++) cin>>a[i]; //a[i] la he so cua x^i

cin>>M;

int b[M+1]; //Mang b chua he so cua B(x) co M+1 phan tu

for (int i = 0; i <= M; i++) cin>>b[i]; //b[i] la he so cua x^i

int K = N + M;

long c[K+1]={0}; //Mang c chua he so cua tich C(x) = A(x)\*B(x)

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

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

c[i+j] += a[i] \* b[j]; //Tinh C(x) và gan vao mang c

}

}

int S = 0; //S la ket qua cua XOR cac he so

for (int i = 0; i <= K; i++) S = S^c[i];

cout<<S;

return 0;

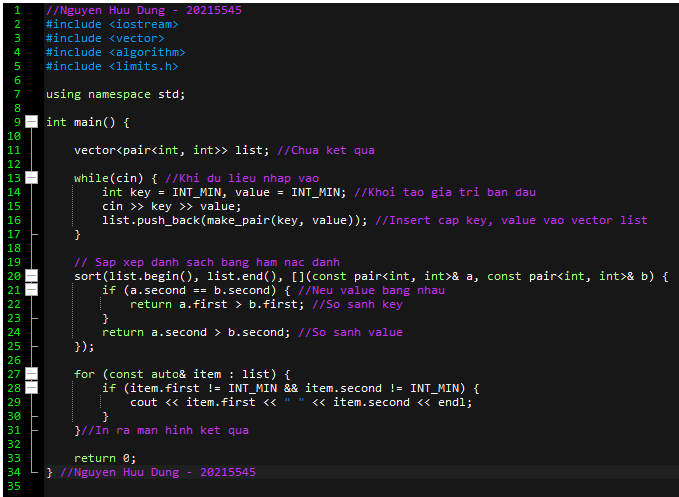
} //Nguyen Huu Dung – 20215545

## **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.



A screenshot of a computer

Description automatically generated with medium confidence

//Nguyen Huu Dung - 20215545

#include <iostream>

#include <vector>

#include <algorithm>

#include <limits.h>

using namespace std;

int main() {

vector<pair<int, int>> list; //Chua ket qua

while(cin) { //Khi du lieu nhap vao

int key = INT\_MIN, value = INT\_MIN; //Khoi tao gia tri ban dau

cin >> key >> value;

list.push\_back(make\_pair(key, value)); //Insert cap key, value vao vector list

}

// Sap xep danh sach bang ham nac danh

sort(list.begin(), list.end(), [](const pair<int, int>& a, const pair<int, int>& b) {

if (a.second == b.second) { //Neu value bang nhau

return a.first > b.first; //So sanh key

}

return a.second > b.second; //So sanh value

});

for (const auto& item : list) {

if (item.first != INT\_MIN && item.second != INT\_MIN) {

cout << item.first << " " << item.second << endl;

}

}//In ra man hinh ket qua

return 0;

} //Nguyen Huu Dung – 20215545

## **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 10^100. 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.

A screen shot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, screenshot, display, software

Description automatically generated

A picture containing text, screenshot, software, font

Description automatically generated

A picture containing text, screenshot, software, display

Description automatically generated

A picture containing text, screenshot, software, font

Description automatically generated

A screenshot of a computer program

Description automatically generated with medium confidence

A picture containing text, screenshot, software, multimedia software

Description automatically generated

//Nguyen Huu Dung - 20215545

#include<bits/stdc++.h>

using namespace std;

struct bigNum {

char sign;

char num[101];

};

// Nhap va xu ly xau

void input(bigNum &x, bigNum &y){

string tmp;

cin >> tmp;

x.sign = tmp[0];

int lens1 = tmp.length() - 1;

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

x.num[101-lens1+i] = tmp[i+1];

}

for(int i=0; i<101-lens1; i++) x.num[i] = '0';

cin >> tmp;

y.sign = tmp[0];

int lens2 = tmp.length() - 1;

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

y.num[101-lens2+i] = tmp[i+1];

}

for(int i=0; i<101-lens2; i++) y.num[i] = '0';

}

// Cong hai so nguyen cung dau

void add(char result[], char \*x, char \*y){

int c = 0;

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

int tmp = (int)x[i] - 48 + (int)y[i] - 48 + c;

c = tmp / 10;

result[i] = tmp % 10 + 48;

}

}

// Tru hai so nguyen cung dau (|x| > |y|)

void sub(char result[], char \*x, char\* y){

int c = 0;

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

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

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

if(tmp1 >= tmp2 + c){

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

c = 0;

} else {

tmp1 = tmp1 + 10;

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

c = 1;

}

}

}

//Nhan hai so nguyen

void multi(char result[], char \*x, char \*y){

// clear array result

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

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

// int 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)x[i] - 48) \* ((int)y[j] - 48) + c;

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

c = sum / 10;

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

}

add(result,tmp,result);

}

}

// Kiem tra x > y ?

bool check(char \*x, char \*y){

int i, k;

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

if(x[i] != '0') break;

}

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

if(y[k] != '0') break;

}

if(i > k) return false;

else if(i < k) return true;

else { // i == k

int t = i;

while(t < 101){

if(x[t] < y[t]) return false;

else if (x[t] > y[t]) return true;

else {

t++;

}

}

}

return true;

}

// Toan tu +

bigNum operator + (bigNum x, bigNum y){

bigNum result;

if(x.sign == '1' && y.sign == '1'){

result.sign = '1';

add(result.num,x.num,y.num);

return result;

} else if(x.sign == '1' && y.sign == '0'){

if(check(x.num,y.num)){

result.sign = '1';

sub(result.num,x.num,y.num);

return result;

} else {

result.sign = '0';

sub(result.num,y.num,x.num);

return result;

}

} else if(x.sign == '0' && y.sign == '1'){

if(check(x.num,y.num)){

result.sign = '0';

sub(result.num,x.num,y.num);

return result;

} else {

result.sign = '1';

sub(result.num,y.num,x.num);

return result;

}

} else {

result.sign = '0';

add(result.num,x.num,y.num);

return result;

}

}

//Toan tu -

bigNum operator - (bigNum x, bigNum y){

bigNum result;

if(x.sign == '1' && y.sign == '0'){

y.sign = '1';

result = x + y;

return result;

} else if(x.sign == '1' && y.sign == '1'){

y.sign = '0';

result = x + y;

return result;

} else if(x.sign == '0' && y.sign == '1'){

y.sign = '0';

result = x + y;

return result;

} else {

y.sign = '1';

result = x + y;

return result;

}

}

//Toan tu \*

bigNum operator \* (bigNum x, bigNum y){

bigNum result;

if(x.sign == '1' && y.sign == '1'){

result.sign = '1';

multi(result.num,x.num,y.num);

return result;

} else if(x.sign == '1' && y.sign == '0'){

result.sign = '0';

multi(result.num,x.num,y.num);

return result;

} else if(x.sign == '0' && y.sign == '1'){

result.sign = '0';

multi(result.num,x.num,y.num);

return result;

} else {

result.sign = '1';

multi(result.num,x.num,y.num);

return result;

}

}

// In ra

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 a, b;

input(a,b);

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 result = a\*b - so3 \* a + so4 \* b;

printBigNumber(result);

} //Nguyen Huu Dung - 20215545