**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** |  |  |  |
| **2** |  |  |  |
| **3** |  |  |  |
| **4** |  |  |  |
| **5** |  |  |  |
| **6** |  |  |  |
| **7** |  |  |  |

**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 1 – Tuần 31 3](#_Toc135811406)

[**Bài tập 1.1.**Viết một chương trình C nhập vào 3 số nguyên. Thiết lập một con trỏ để lần lượt trỏ tới từng số nguyên và hiển thị kết quả giá trị tham chiếu ngược của con trỏ. 3](#_Toc135811407)

[**Bài tập 1.2**. Viết chương trình in ra địa chỉ của 5 phần tử đầu tiên trong mảng được định nghĩa sau đây: int a[7]= {13, -355, 235, 47, 67, 943, 1222}; 4](#_Toc135811408)

[**Bài tập 1.3.** Viết chương trình yêu cầu nhập giá trị cho 3 biến số nguyên x, y, z kiểu int. Sau đó sử dụng duy nhất một con trỏ để cộng giá trị của mỗi biến thêm 100. 6](#_Toc135811409)

[**Bài tập 1.4.** Viết hàm countEven(int\*, int) nhận một mảng số nguyên và kích thước của mảng, trả về số lượng số chẵn trong mảng??? 7](#_Toc135811410)

[**Bài tập 1.5.** Viết hàm trả về con trỏ trỏ tới giá trị lớn nhất của một mảng các số double. Nếu mảng rỗng hãy trả về NULL. 8](#_Toc135811411)

[**Bài tập 1.6.** Viết hàm đảo ngược một mảng các số nguyên theo hai cách: dùng chỉ số và dùng con trỏ. Ví dụ mảng đầu vào là [9, -1, 4, 5, 7] thì kết quả là [7, 5, 4, -1, 9]. 9](#_Toc135811412)

[**Bài tập 1.7.** Viết chương trình nhập vào một mảng các số nguyên với số lượng các phần tử nhập từ bàn phím. Sau đó sắp xếp mảng theo thứ tự tăng dần. Hiển thị danh sách mảng trước và sau khi sắp xếp. Yêu cầu chỉ sử dụng con trỏ để truy cập mảng, không truy cập theo index mảng. 12](#_Toc135811413)

[**Bài tập 1.8.** Viết chương trình nhập vào một ma trận 2 chiều kích thước m\*n với m và n nhập từ bàn phím. Sau đó đưa ra tổng các phần tử chẵn của ma trận đó. 14](#_Toc135811414)

[**Bài tập 1.9.** Viết chương trình in ra tất cả các dãy con của một dãy cho trước. 17](#_Toc135811415)

[**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. 20](#_Toc135811416)

# Bài thực hành số 2

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

## **Bài tập 2.1.**Viết hàm tính độ dài cạnh huyền của tam giác từ 2 cạnh góc vuông

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#include <stdio.h>

#include <math.h>

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

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

float z;

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

return z;

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

}

int main(){

float x, y;

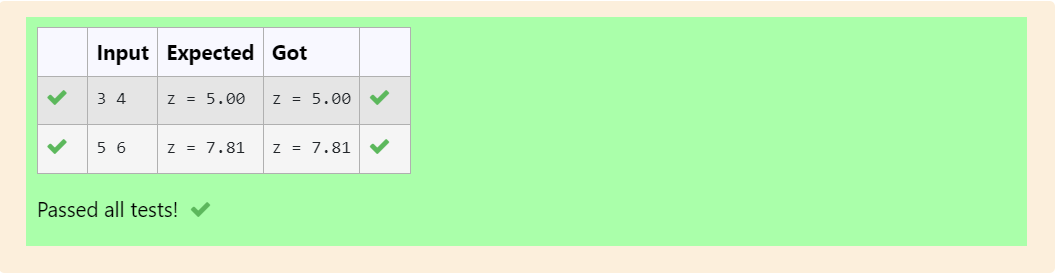
scanf("%f%f", &x, &y);

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

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

return 0;

}



## **Bài tập 2.2**. 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.

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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;

//# 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;

}



## **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.

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#include <stdio.h>

//# Viết hàm get\_value

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

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));

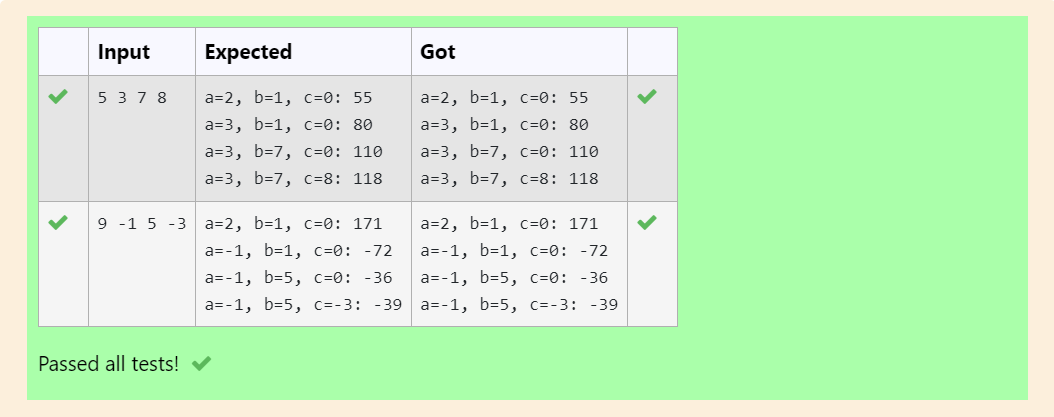
printf("a=%d, b=1, c=0: %d\n", a, get\_value(x, a));

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;

}



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

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#include <stdio.h>

int cube(int x) {

//#Trả về lập phương của x

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

return x\*x\*x;

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

}

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

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

double cube(double x) {

return x\*x\*x;

}

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

int main() {

int n;

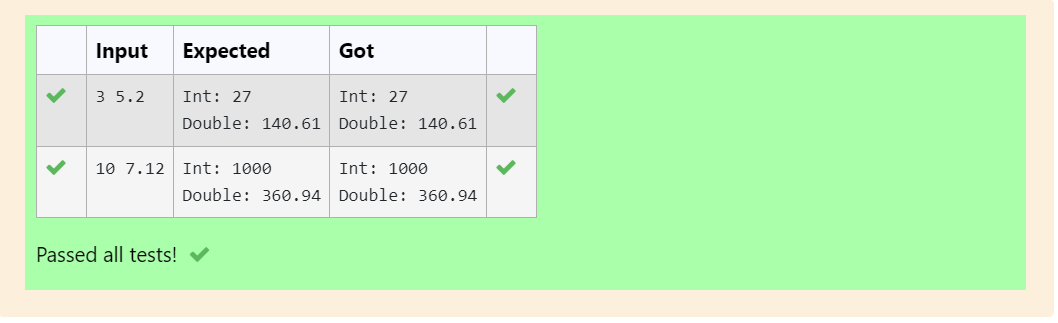
double f;

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

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

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

return 0;

}

## **Bài tập 2.5.** Viết toán tử tính tổng, hiệu, tích, thương của 2 số phức

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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 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;

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

tmpC.imag = (a.imag \* b.real - a.real \* b.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() {

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;

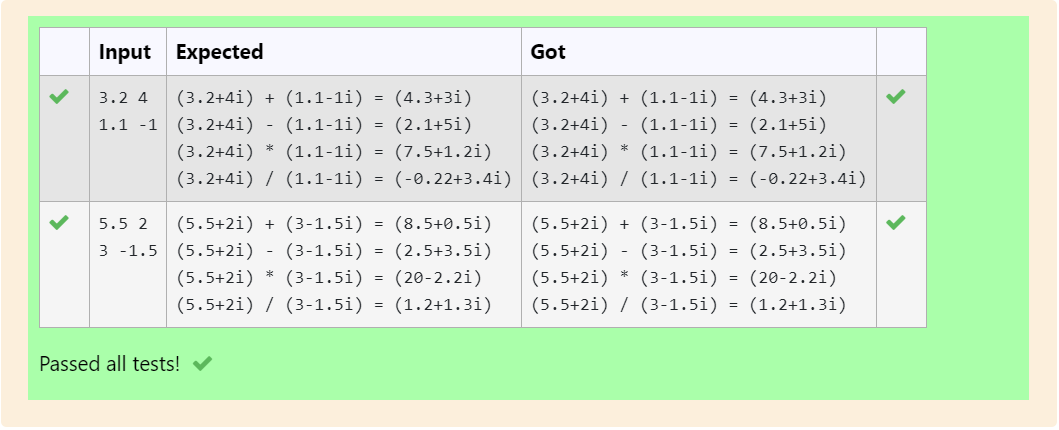
cout << a << " - " << b << " = " << a - b << endl;

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

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

return 0;

}



## **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.

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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() {

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

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

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

odd = mul3plus1;

even = div2;

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

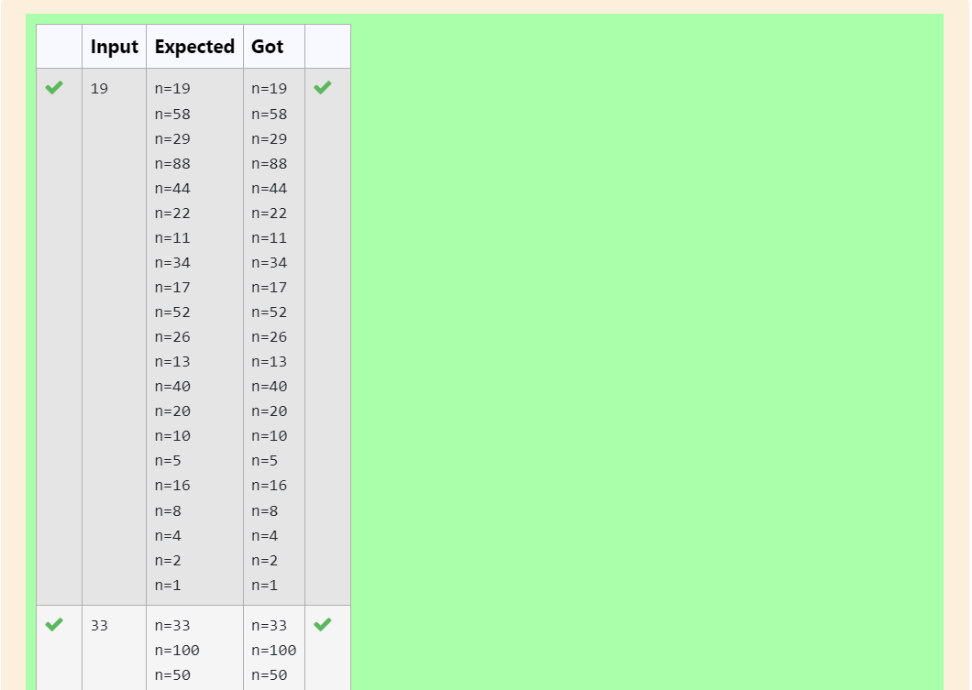
int n;

scanf("%d", &n);

simulate(n, odd, even, print);

return 0;

}



## **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.

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#include <iostream>

using namespace std;

//# viết hàm arr\_sum

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

template <typename T>

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

T sum = 0;

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

sum += a[i];

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

sum += b[i];

return sum;

}

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

int main() {

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};

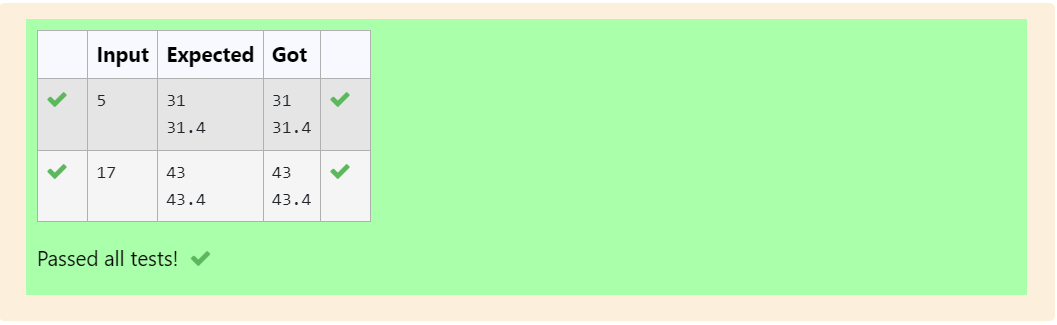
double b[] = {5, 6.1, 1, 2.3, 7};

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

}

return 0;

}



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

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

int main() {

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

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

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 << ' ';

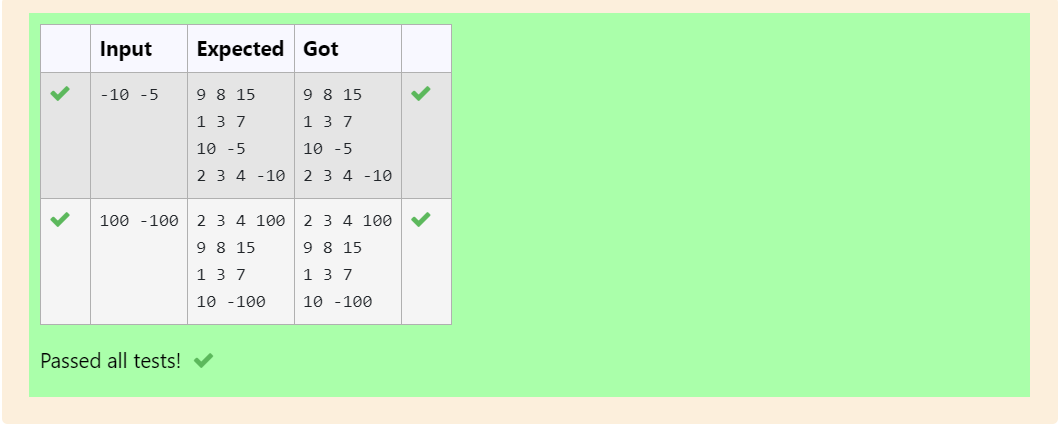
}

cout << endl;

}

return 0;

}



## **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.

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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() {

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

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() {

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 {

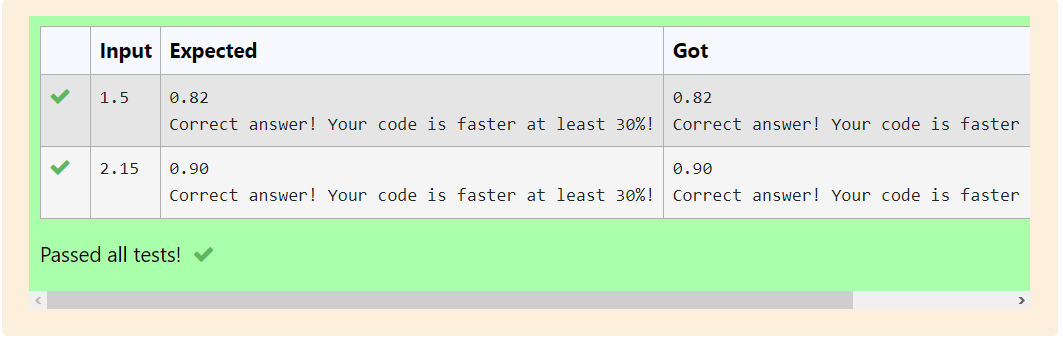
//printf("Wrong answer or your code is not fast enough!\n");

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

}

return 0;

}



## **Bài tập 2.10.** Tính tích hai ma trận vuông

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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) {

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

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;

}

A picture containing text, screenshot, font

Description automatically generated

## **Bài tập 2.11.** Tính tích hai đa thức

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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);

}

A picture containing text, screenshot, font

Description automatically generated

## **Bài tập 2.12.** Map Sort

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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();

}

## **Bài tập 2.13.** Big Integer

//Họ tên: Trần Tiến Đức

//MSSV: 20204643

#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);

}

A screen shot of a computer program

Description automatically generated with low confidence