TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT 2025

**Bài tập 01 - Ôn tập**

Sinh viên thực hiện: Nguyễn Trí Cường 3124411049

Thành phố Hồ Chí Minh, tháng 1 năm 2025

**BT1**

Phân tích bài toán:

Chúng ta cần giải phương trình bậc 2 có dạng: ax^2 + bx + c = 0

Có 3 hệ số a, b, c nhập vào từ bàn phím

Các trường hợp xảy ra:

- Nếu a = 0 và b = 0 thì kiểm tra c: nếu c = 0, phương trình có vô số nghiệm; ngược lại vô nghiệm

- Nếu a = 0 nhưng b khác 0, phương trình trở thành bậc 1, có 1 nghiệm duy nhất x = -c/b

- Nếu a khác 0, tính delta = b^2 - 4ac:

+ Nếu delta < 0, phương trình vô nghiệm

+ Nếu delta = 0, phương trình có nghiệm kép x = -b/(2a)

+ Nếu delta > 0, phương trình có 2 nghiệm phân biệt

Thiết kế giải thuật:

1. Nhập 3 số a, b, c

2. Xử lý theo các trường hợp như trên

3. In ra số nghiệm và các nghiệm (nếu có), làm tròn đến 2 chữ số thập phân

Testcase:

|  |  |
| --- | --- |
| Input | Output |
| 0 0 0 | Phuong trinh co vo so nghiem |
| 0 0 1 | Phuong trinh vo nghiem |
| 1 2 1 | Phuong trinh co 1 nghiem -1.00 |
| 1 -3 2 | Phuong trinh co 2 nghiem 1.00 và 2.00 |
| 1 0 -4 | Phuong trinh co 2 nghiem -2.00 và 2.00 |

//GiaiPTBac2.cpp

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

void GiaiPTBac2(double a, double b, double c)

{

if (a == 0)

{

if (b == 0)

{

if (c == 0)

{

cout << "Phuong trinh co vo so nghiem" << endl;

}

else

{

cout << "Phuong trinh vo nghiem" << endl;

}

}

else

{

cout << "Phuong trinh co 1 nghiem" << endl;

cout << fixed << setprecision(2) << -c/b << endl;

}

}

else {

double Delta = b\*b-4\*a\*c;

if (Delta < 0)

{

cout << "Phuong trinh vo nghiem" << endl;

}

else if (Delta == 0)

{

cout << "Phuong trinh co 1 nghiem" << endl;

cout << fixed << setprecision(2) << -b/(2\*a) << endl;

}

else

{

cout << "Phuong trinh co 2 nghiem" << endl;

double x1 = (-b-sqrt(Delta)) / (2\*a);

double x2 = (-b+sqrt(Delta)) / (2\*a);

cout << fixed << setprecision(2) << x1 << " " << x2 << endl;

}

}

}

int main()

{

double a, b, c;

cin >> a >> b >> c;

GiaiPTBac2(a, b, c);

return 0;

}

11111111111111111111111111111111111111111111111111111111111111111111111111111111

**BT2**

Phân tích bài toán:

Chúng ta cần giải phương trình trùng phương có dạng: ax^4 + bx^2 + c = 0

Đặt y = x^2, phương trình trở thành: ay^2 + by + c = 0 (phương trình bậc 2 theo y)

Sau khi giải y, với mỗi nghiệm y >= 0, tính x = ±sqrt(y)

Nếu nghiệm y < 0, loại bỏ vì không hợp lệ

Thiết kế giải thuật:

1. Nhập 3 số a, b, c

2. Nếu a = 0, giải phương trình bậc 1 bx^2 + c = 0

3. Nếu a != 0, tính delta và giải phương trình bậc 2 với y

4. Tìm các giá trị x từ y và sắp xếp chúng tăng dần

5. In ra số nghiệm và các nghiệm (nếu có)

Testcase:

|  |  |
| --- | --- |
| Input | Output |
| 0 0 0 | Phuong trinh co vo so nghiem |
| 1 0 1 | Phuong trinh vo nghiem |
| 1 -2 1 | Phuong trinh co 2 nghiem -100 và 1.00 |
| 1 0 -1 | Phuong trinh co 4 nghiem -1.00, -0.00, 0.00 1.00 |
| 1 2 -3 | Phuong trinh o 2 nghiem -1.21 và 1.21 |

//GiaiPTTrungPhuong.cpp

#include <iostream>

#include <cmath>

#include <iomanip>

#include <vector>

#include <algorithm>

using namespace std;

void GiaiPTTrungPhuong(double a, double b, double c)

{

if (a == 0)

{

if (b == 0)

{

if (c == 0)

{

cout << "Phuong trinh co vo so nghiem" << endl;

}

else

{

cout << "Phuong trinh vo nghiem" << endl;

}

}

else

{

double y = -c / b;

if (y < 0)

{

cout << "Phuong trinh vo nghiem" << endl;

}

else

{

cout << "Phuong trinh co 2 nghiem" << endl;

double x1 = -sqrt(y), x2 = sqrt(y);

cout << fixed << setprecision(2) << x1 << " " << x2 << endl;

}

}

}

else

{

double Delta = b \* b - 4 \* a \* c;

if (Delta < 0)

{

cout << "Phuong trinh vo nghiem" << endl;

}

else

{

vector<double> nghiem;

double y1 = (-b - sqrt(Delta)) / (2 \* a);

double y2 = (-b + sqrt(Delta)) / (2 \* a);

if (y1 >= 0)

{

nghiem.push\_back(-sqrt(y1));

nghiem.push\_back(sqrt(y1));

}

if (y2 >= 0)

{

nghiem.push\_back(-sqrt(y2));

nghiem.push\_back(sqrt(y2));

}

if (nghiem.empty())

{

cout << "Phuong trinh vo nghiem" << endl;

}

else

{

sort(nghiem.begin(), nghiem.end());

nghiem.erase(unique(nghiem.begin(), nghiem.end()), nghiem.end());

cout << "Phuong trinh co " << nghiem.size() << " nghiem" << endl;

for (double x : nghiem) {

cout << fixed << setprecision(2) << x << " ";

}

cout << endl;

}

}

}

}

int main()

{

double a, b, c;

cin >> a >> b >> c;

GiaiPTTrungPhuong(a, b, c);

return 0;

}

222222222222222222222222222222222222222222222222222222222222222222222222222222

**BT3**

Phân tích bài toán:

Chúng ta cần tìm các phần tử chung của 3 dãy số nguyên dương x, y, z.

Kết quả cần trả về là số lượng phần tử chung và danh sách các phần tử này (không trùng lặp, tăng dần).

Thiết kế giải thuật:

1. Nhập số phần tử và nội dung của 3 dãy x, y, z

2. Loại bỏ các phần tử trùng lặp trong mỗi dãy bằng cách sử dụng set

3. Tìm giao của 3 dãy bằng cách sử dụng hàm set\_intersection.

4. In ra số lượng và các phần tử chung

Testcase:

|  |  |
| --- | --- |
| Input | Output |
| 31 2 334 5 637 8 9 | 0 |
| 51 2 3 4 553 4 5 6 752 3 5 7 9 | 23 5 |
| 410 20 30 40530 40 50 60 70620 30 40 50 60 70 | 230 40 |
| 61 2 3 4 5 661 3 5 7 9 1163 5 7 9 11 13 | 23 5 |
| 5100 200 300 400 5004100 200 300 4003300 400 500 | 2300 400 |

//PhanTuChung.cpp

#include <iostream>

#include <set>

#include <vector>

#include <algorithm>

using namespace std;

void TimPhanTuChung(vector<int>& x, vector<int>& y, vector<int>& z)

{

set<int> setX(x.begin(), x.end());

set<int> setY(y.begin(), y.end());

set<int> setZ(z.begin(), z.end());

vector<int> giaoXY;

set\_intersection(setX.begin(), setX.end(), setY.begin(), setY.end(), back\_inserter(giaoXY));

vector<int> giaoXYZ;

set\_intersection(giaoXY.begin(), giaoXY.end(), setZ.begin(), setZ.end(), back\_inserter(giaoXYZ));

cout << giaoXYZ.size() << endl;

for (int val : giaoXYZ)

{

cout << val << " ";

}

cout << endl;

}

int main()

{

int nx, ny, nz;

cin >> nx;

vector<int> x(nx);

for (int& xi : x) cin >> xi;

cin >> ny;

vector<int> y(ny);

for (int& yi : y) cin >> yi;

cin >> nz;

vector<int> z(nz);

for (int& zi : z) cin >> zi;

TimPhanTuChung(x, y, z);

return 0;

}