

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA: CÔNG NGHỆ THÔNG TIN
BỘ MÔN: CÔNG NGHỆ PHẦN MỀM

ĐỀ THI GIỮA KỲ

Tên học phần: Toán ứng dụng CNTT

Mã học phần: 10223220.2510.23.10 Số tín chỉ: **03**

Phương pháp đánh giá (*): Tự luận có giám sát Thời gian làm bài: **90** phút

Đề số: **T000.01**

☐ Sinh viên được sử dụng tài liệu khi làm bài.

Họ tên: Lê Bá Nguyên Long

Lớp: 23T_DT4

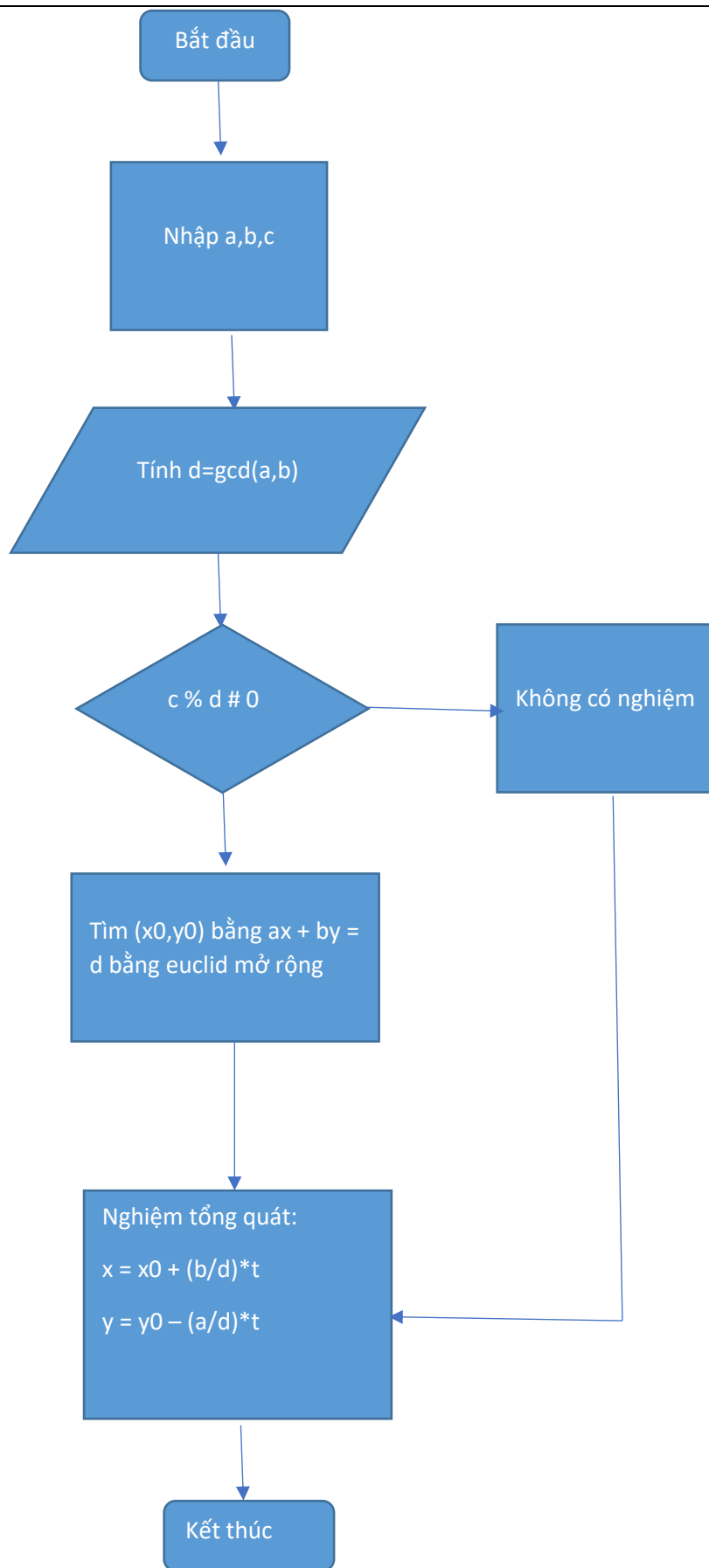
MSSV: 102230357

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV_HọTên.pdf và nộp bài thông qua MSTeam

Câu 1 (3 điểm):

- a) (1 điểm) Cho phương trình $Ax + By = C$ với A, B, C là các số nguyên dương và x, y là biến. Hãy trình thuật toán để giải phương trình trên.

Trả lời: Trình bày thuật toán bằng sơ đồ khối



b) (1 điểm) Hãy viết chương trình sử dụng thuật toán để giải phương trình $Ax + By = C$

Trả lời: Dán code vào bên dưới

```
#include <bits/stdc++.h>
using namespace std;

int extendedGCD(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1; y = 0;
        return a;
    }
    int x1, y1;
    int d = extendedGCD(b, a % b, x1, y1);
    x = y1;
    y = x1 - (a / b) * y1;
    return d;
}

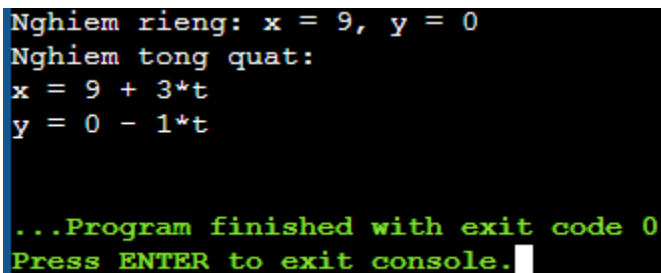
int main() {
    int A = 3, B = 9, C = 27;
    int x, y;
    int d = extendedGCD(A, B, x, y);

    if (C % d != 0) {
        cout << "Phuong trinh vo nghiem\n";
    } else {
        x *= C / d;
        y *= C / d;
        cout << "Nghiem rieng: x = " << x << ", y = " << y << "\n";

        cout << "Nghiem tong quat:\n";
        cout << "x = " << x << " + " << B/d << "*t\n";
        cout << "y = " << y << " - " << A/d << "*t\n";
    }
    return 0;
}
```

c) (1 điểm) Hãy minh hoạ kết quả chương trình

Trả lời: Dán kết quả gồm nghiệm riêng và nghiệm tổng quát của phương trình $3x + 9y = 27$.



```
Nghiem rieng: x = 9, y = 0
Nghiem tong quat:
x = 9 + 3*t
y = 0 - 1*t

...Program finished with exit code 0
Press ENTER to exit console.
```

Câu 2 (4 điểm): Cho ma trận A

a) (1 điểm) Trình bày thuật toán để phân rã ma trận A theo Cholesky

Trả lời: Trình bày thuật toán bằng sơ đồ khối hoặc ngôn ngữ tự nhiên ở đây:

* Khởi tạo ma trận L với cùng kích thước với A, tất cả phần tử ban đầu bằng 0.

* Duyệt qua các dòng và cột:

Với mỗi i từ 0 đến n-1

Với mỗi j từ 0 đến i

- Tính tổng $sum = \sum_{k=0}^{j-1} L[i][k] * L[j][k]$

- Nếu $i == j$:

→ $L[i][j] = \sqrt{A[i][i] - sum}$

- Nếu $i > j$:

→ $L[i][j] = (A[i][j] - sum) / L[j][j]$

* Trả về ma trận L.

b) (2 điểm) Viết chương trình để thực hiện phân rã Cholesky

Trả lời: Dán code vào bên dưới:

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

```
using namespace std;
```

```
const int N = 3;
```

```
void choleskyDecomposition(double A[N][N]) {  
    double L[N][N] = {0};
```

```
    for (int i = 0; i < N; i++) {  
        for (int j = 0; j <= i; j++) {  
            double sum = 0;  
  
            for (int k = 0; k < j; k++)  
                sum += L[i][k] * L[j][k];  
  
            if (i == j)  
                L[i][j] = sqrt(A[i][i] - sum);  
            else  
                L[i][j] = (A[i][j] - sum) / L[j][j];  
        }  
    }
```

```
    cout << "Ma tran L (tam giac duoi):\n";  
    for (int i = 0; i < N; i++) {  
        for (int j = 0; j < N; j++)  
            cout << fixed << setprecision(4) << L[i][j] << "t";  
        cout << "\n";  
    }
```

```
    cout << "\nMa tran L^T (chuyen vi):\n";  
    for (int i = 0; i < N; i++) {  
        for (int j = 0; j < N; j++)  
            cout << fixed << setprecision(4) << L[j][i] << "t";  
        cout << "\n";  
    }
```

```

}

int main() {
    double A[N][N] = {
        {4, 12, -16},
        {12, 37, -43},
        {-16, -43, 98}
    };

    choleskyDecomposition(A);
    return 0;
}

```

c) (2 điểm) Minh họa kết quả của thuật toán Cholesky

Trả lời: Dán kết quả minh họa với ma trận

4	12	-16
12	37	-43
-16	-43	98

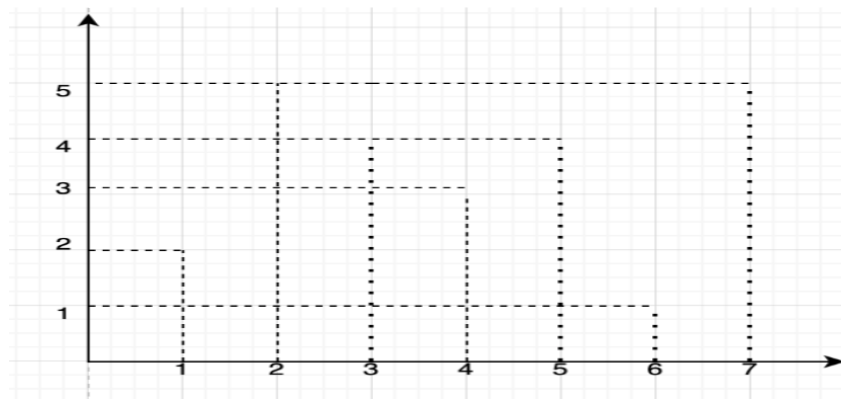
```

Ma tran L (tam giác dưới):
2.0000  0.0000  0.0000
6.0000  1.0000  0.0000
-8.0000  5.0000  3.0000

Ma tran L^T (chuyen vi):
2.0000  6.0000 -8.0000
0.0000  1.0000  5.0000
0.0000  0.0000  3.0000

```

Câu 3 (3 điểm): Cho không gian Oxy và 7 điểm tương ứng như hình vẽ dưới:



a) (1 điểm) Trình bày thuật toán xác định bao lồi

Trả lời: dán sơ đồ khối hoặc ngôn ngữ tự nhiên vào bên dưới:

- Sắp xếp các điểm theo hoành độ tăng dần (nếu bằng nhau thì xét tung độ).
- Tạo bao dưới:
 - Duyệt từ trái sang phải, thêm điểm vào stack.
 - Với mỗi điểm mới, kiểm tra 3 điểm cuối có tạo thành khúc rẽ phải không (dựa vào cross product).
 - Nếu không, loại điểm giữa.
- Tạo bao trên:
 - Tương tự, nhưng duyệt từ phải sang trái.
- Gộp bao dưới + bao trên:
 - Kết quả là danh sách điểm theo chiều kim đồng hồ hoặc ngược chiều, tạo nên bao lồi.

b) (1.5 điểm) viết chương trình xác định bao lồi

Trả lời: viết câu trả lời vào bên dưới:

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

```
using namespace std;
```

```
struct Point {  
    int x, y;  
    bool operator < (const Point& p) const {  
        return x < p.x || (x == p.x && y < p.y);  
    }  
};
```

```
int cross(Point O, Point A, Point B) {  
    return (A.x - O.x)*(B.y - O.y) - (A.y - O.y)*(B.x - O.x);  
}
```

```
vector<Point> convexHull(vector<Point>& P) {  
    int n = P.size(), k = 0;  
    if (n <= 3) return P;  
    sort(P.begin(), P.end());  
    vector<Point> H(2 * n);  
  
    for (int i = 0; i < n; i++) {  
        while (k >= 2 && cross(H[k-2], H[k-1], P[i]) <= 0) k--;  
        H[k++] = P[i];  
    }  
  
    for (int i = n - 2, t = k + 1; i >= 0; i--) {  
        while (k >= t && cross(H[k-2], H[k-1], P[i]) <= 0) k--;  
        H[k++] = P[i];  
    }  
}
```

```
H.resize(k - 1);  
return H;
```

```

}

int main() {
    vector<Point> points = {
        {1, 1}, {2, 2}, {3, 3},
        {4, 3}, {5, 4}, {6, 1}, {7, 5}
    };

    vector<Point> hull = convexHull(points);

    cout << "Cac diem thuoc bao loi:\n";
    for (auto p : hull)
        cout << "(" << p.x << ", " << p.y << ")\n";

    return 0;
}

```

- c) (0.5 điểm) Tính diện tích bao lồi xây dựng được và chỉ ra hai điểm trên bao lồi có kết khoảng cách ngắn nhất

Trả lời: viết câu trả lời vào bên dưới:

1. Tính diện tích bao lồi (Convex Hull Area)

Công thức Green (Green's Theorem), còn gọi là công thức Shoelace:

$$\text{Diện tích} = \frac{1}{2} \left| \sum_{i=0}^{n-1} (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \right|$$

Trong đó các điểm $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ là các đỉnh của bao lồi, được sắp theo thứ tự ngược chiều kim đồng hồ hoặc cùng chiều.

Các bước thực hiện:

- Dùng thuật toán Graham Scan hoặc Monotone Chain để xác định bao lồi
- Lấy danh sách điểm trên bao lồi, lần lượt tính tổng các tích chéo giữa hoành độ và tung độ của các cặp điểm liên tiếp

- Áp dụng công thức để tính diện tích

2. Hai điểm trên bao lồi có khoảng cách ngắn nhất

Để tìm hai điểm trên bao lồi có khoảng cách ngắn nhất, ta thực hiện như sau:

- Duyệt qua mọi cặp điểm trên bao lồi (do số lượng điểm bao lồi thường nhỏ nên dùng brute-force hiệu quả)

- Tính khoảng cách Euclidean giữa các cặp điểm:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- Chọn ra cặp có khoảng cách nhỏ nhất

CODE:

```

double polygonArea(const vector<Point>& poly) {
    double area = 0;
    int n = poly.size();
    for (int i = 0; i < n; i++) {
        int j = (i+1)%n;
        area += (poly[i].x * poly[j].y) - (poly[j].x * poly[i].y);
    }
    return abs(area) / 2.0;
}

```

```
double dist(Point A, Point B) {  
    return sqrt((A.x - B.x)*(A.x - B.x) + (A.y - B.y)*(A.y - B.y));  
}  
  
pair<Point, Point> closestPair(vector<Point>& poly) {  
    double minD = 1e18;  
    pair<Point, Point> res;  
    int n = poly.size();  
    for (int i = 0; i < n; i++) {  
        for (int j = i+1; j < n; j++) {  
            double d = dist(poly[i], poly[j]);  
            if (d < minD) {  
                minD = d;  
                res = {poly[i], poly[j]};  
            }  
        }  
    }  
    return res;  
}
```

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

Đà Nẵng, ngày 18 tháng 09 năm 2025
TRƯỞNG BỘ MÔN
(đã duyệt)