{1.0, 2.0, 1.0/4, 1.0, 1.0, 3,

1.0/2, 1.0, 1.0/4, 3.0, 1.0/2, 1.0,

4.0, 4.0, 1.0, 1.0/5, 2.0, 1.0/3,

1.0, 1.0/3, 5.0, 1.0, 1.0/2, 4.0,

1.0, 2.0, 1.0/2, 2.0, 1.0, 1.0/3,

1.0/3, 1.0, 3.0, 1.0/4, 3.0, 1.0},

{1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0},

{1.0, 1.0/5, 4.0, 1.0/3, 1.0, 1.0/2,

5.0, 1.0, 2.0, 4.0, 1.0/2, 1.0/2,

1.0/4, 1.0/2, 1.0, 1.0/7, 1.0, 3.0,

3.0, 1.0/4, 7.0, 1.0, 1.0, 1.0/5,

1.0, 2.0, 1.0, 1.0, 1.0, 2.0,

1.0/2, 2.0, 1.0/3, 5.0, 1.0/2, 1.0},

{1.0, 3.0, 1.0/2, 1.0, 5.0, 7.0,

1.0/2, 1.0, 1.0, 3.0, 1.0/4, 1.0/2,

2.0, 1.0, 1.0, 1.0/7, 1.0/3, 2.0,

1.0, 1.0/3, 7.0, 1.0, 5.0, 1.0,

1.0/5, 4.0, 3.0, 1.0/5, 1.0, 1.0/4,

1.0/7, 2.0, 1.0/2, 1.0, 4.0, 1.0}

#include<iostream>

using namespace std;

const int nCriteria = 4;

const int nLanguage = 6;

float A1[nCriteria][nCriteria] =

{

{1.0, 1.0 / 5, 2, 2},

{5, 1.0, 3, 3},

{1.0 / 2, 1.0 / 3, 1.0, 1.0 / 6},

{1.0 / 2, 1.0 / 3, 6, 1.0}

};

float A1Copy[nCriteria][nCriteria] =

{

{1.0, 1.0 / 5, 2, 2},

{5, 1.0, 3, 3},

{1.0 / 2, 1.0 / 3, 1.0, 1.0 / 6},

{1.0 / 2, 1.0 / 3, 6, 1.0}

};

float w[nCriteria];

float wDash[nCriteria];

float lambda = 0;

float coherence = 0;

float A2[nCriteria][nLanguage\*nLanguage] =

{

{1.0, 2.0, 1.0 / 4, 1.0, 1.0, 3,

1.0 / 2, 1.0, 1.0 / 4, 3.0, 1.0 / 2, 1.0,

4.0, 4.0, 1.0, 1.0 / 5, 2.0, 1.0 / 3,

1.0, 1.0 / 3, 5.0, 1.0, 1.0 / 2, 4.0,

1.0, 2.0, 1.0 / 2, 2.0, 1.0, 1.0 / 3,

1.0 / 3, 1.0, 3.0, 1.0 / 4, 3.0, 1.0},

{1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0},

{1.0, 1.0 / 5, 4.0, 1.0 / 3, 1.0, 1.0 / 2,

5.0, 1.0, 2.0, 4.0, 1.0 / 2, 1.0 / 2,

1.0 / 4, 1.0 / 2, 1.0, 1.0 / 7, 1.0, 3.0,

3.0, 1.0 / 4, 7.0, 1.0, 1.0, 1.0 / 5,

1.0, 2.0, 1.0, 1.0, 1.0, 2.0,

1.0 / 2, 2.0, 1.0 / 3, 5.0, 1.0 / 2, 1.0},

{1.0, 3.0, 1.0 / 2, 1.0, 5.0, 7.0,

1.0 / 2, 1.0, 1.0, 3.0, 1.0 / 4, 1.0 / 2,

2.0, 1.0, 1.0, 1.0 / 7, 1.0 / 3, 2.0,

1.0, 1.0 / 3, 7.0, 1.0, 5.0, 1.0,

1.0 / 5, 4.0, 3.0, 1.0 / 5, 1.0, 1.0 / 4,

1.0 / 7, 2.0, 1.0 / 2, 1.0, 4.0, 1.0}

};

float A2Copy[nCriteria][nLanguage\*nLanguage] =

{

{1.0, 2.0, 1.0 / 4, 1.0, 1.0, 3,

1.0 / 2, 1.0, 1.0 / 4, 3.0, 1.0 / 2, 1.0,

4.0, 4.0, 1.0, 1.0 / 5, 2.0, 1.0 / 3,

1.0, 1.0 / 3, 5.0, 1.0, 1.0 / 2, 4.0,

1.0, 2.0, 1.0 / 2, 2.0, 1.0, 1.0 / 3,

1.0 / 3, 1.0, 3.0, 1.0 / 4, 3.0, 1.0},

{1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

1.0, 1.0, 1.0, 1.0, 1.0, 1.0},

{1.0, 1.0 / 5, 4.0, 1.0 / 3, 1.0, 1.0 / 2,

5.0, 1.0, 2.0, 4.0, 1.0 / 2, 1.0 / 2,

1.0 / 4, 1.0 / 2, 1.0, 1.0 / 7, 1.0, 3.0,

3.0, 1.0 / 4, 7.0, 1.0, 1.0, 1.0 / 5,

1.0, 2.0, 1.0, 1.0, 1.0, 2.0,

1.0 / 2, 2.0, 1.0 / 3, 5.0, 1.0 / 2, 1.0},

{1.0, 3.0, 1.0 / 2, 1.0, 5.0, 7.0,

1.0 / 2, 1.0, 1.0, 3.0, 1.0 / 4, 1.0 / 2,

2.0, 1.0, 1.0, 1.0 / 7, 1.0 / 3, 2.0,

1.0, 1.0 / 3, 7.0, 1.0, 5.0, 1.0,

1.0 / 5, 4.0, 3.0, 1.0 / 5, 1.0, 1.0 / 4,

1.0 / 7, 2.0, 1.0 / 2, 1.0, 4.0, 1.0}

};

float w2[nCriteria][nLanguage];

float w2Dash[nCriteria][nLanguage];

float lambdaLen[nLanguage] = { 0 };

float coherenceLen[nLanguage] = { 0 };

float d[nLanguage] = { 0 };

int indMax;

void print(int n, float \*matrix);

void normStep1(int n, float \*matrix);

float normStep23(int n, float \*matrix);

float productAw(int n, float \*matrix, float \*w);

int max(int n, float \*matrix);

int main() {

setlocale(LC\_ALL, "RUS");

// вычисление лямбд и ис

normStep1(nCriteria, &A1Copy[0][0]);

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

w[i] = normStep23(nCriteria, &A1Copy[i][0]);

}

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

wDash[i] = productAw(nCriteria, &A1[i][0], &w[0]);

}

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

wDash[i] /= w[i];

}

cout << endl;

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

lambda += wDash[i];

}

lambda /= nCriteria;

coherence = (lambda - nCriteria) / (nCriteria - 1);

cout << " Собственное значение марицы А1 = " << lambda << ", ИС = " << coherence << endl << endl;

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

normStep1(nLanguage, &A2Copy[i][0]);

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

for (int j = 0; j < nLanguage; j++)

w2[i][j] = normStep23(nLanguage, &A2Copy[i][j \* nLanguage]);

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

for (int j = 0; j < nLanguage; j++)

w2Dash[i][j] = productAw(nLanguage, &A2[i][j \* nLanguage], &w2[i][0]);

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

for (int j = 0; j < nLanguage; j++)

w2Dash[i][j] /= w2[i][j];

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

for (int j = 0; j < nLanguage; j++)

lambdaLen[i] += w2Dash[i][j];

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

lambdaLen[i] /= nLanguage;

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

coherenceLen[i] = (lambdaLen[i] - nLanguage) / (nLanguage - 1);

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

cout << " Собственное значение марицыА2" << (i + 1) << " = " << lambdaLen[i] << ", ИС = " << coherenceLen[i] << endl << endl;

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

for (int j = 0; j < nCriteria; j++)

d[i] += w[j] \* w2[i][j];

indMax = max(nLanguage, &d[0]);

cout << " Наилучшее решение " << (indMax + 1) << ", с D = " << d[indMax] << endl << endl;

system("pause");

return 0;

}

void print(int n, float \*matrix) {

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

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

cout <<(\* matrix) << " ";

}

cout << endl;

}

}

void normStep1(int n, float \*matrix) {

float \*sum = new float[n];

int i, j;

float \*k;

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

sum[i] = 0;

for (j = 0; j < n; j++)

sum[i] += \*(matrix + i \* n + j);

//cout << "Сумма в строке " << i+1<< sum[i] << endl;

}

float s = 0.0;

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

s += sum[i];

//cout << s << endl;

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

for (j = 0; j < n; j++) {

k = (matrix + i + j \* n);

(\*k) = (\*k) / s;

cout << \*k;

}

cout << endl;

}

float normStep23(int n, float \*matrix) {

int i;

float w = 0;

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

w += \*(matrix + i);

//cout << w / n<< endl;

return w / n;

}

float productAw(int n, float \*matrix, float \*w) {

int i;

float wD = 0;

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

wD += (\*(w + i)) \* (\*(matrix + i));

//cout << wD << endl;

return wD;

}

int max(int n, float \*matrix) {

float max = \*matrix;

int index = 0;

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

if (max < \*(matrix + i)) {

max = \*(matrix + i);

index = i;

}

return index;

}

0.16;0.33;0.45;0.77;0.25;0.69

0.32;0.14;0.03;0.13;0.24;0.14