Міністерство освіти і науки України

Національний університет "Львівська політехніка"

Кафедра ЕОМ

****

**Звіт**

з лабораторної роботи № 1

з дисципліни: “ Паралельні та розподілені обчислення”

на тему: “ ВИКОРИСТАННЯ ФУНКЦІОНАЛЬНОЇ ДЕКОМПОЗИЦІЇ ДЛЯ РОЗВ’ЯЗКУ ОБЧИСЛЮВАЛЬНИХ ЗАДАЧ.”

Варіант 6

Виконав: студент .гр. КІ-33

Котик В.В.

Прийняв: асистент каф. ЕОМ

Козак Н.Б.

Львів 2020

**Мета:** Вивчити методи декомпозицій задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції.

**ЗАВДАННЯ.**

Використовуючи метод функціональної декомпозиції, розробити алгоритм обчислення запропонованого матрично-векторного виразу, який би враховував можливість паралельного виконання і був оптимальним з точки зору часових затрат.

На основі створеного алгоритму написати програму яка дозволяє обчислити вираз та ілюструє проведену декомпозицію.

**Вагомі зауваження.**

**а)** Окрім безпосередніх обчислень, програма повинна мати інтерфейс користувача, який забезпечує:

1. ввід(з клавіатури) розмірності даних (n);
2. можливість вибору–ввід даних(тобто елементів матриці та векторів) з клавіатури чи генерування їх випадковим чином;
3. вивід на екран (або у файл) проміжних результатів за потребою користувача;
4. обов’язковий вивід остаточних результатів на екран і у файл у зрозумілому вигляді.

**б)** Всі вхідні дані є цілими числами, більшими за нуль.

**в)** Необхідно знайти такі коефіцієнт нормалізації результатів(тобто пониження чи підвищення їх порядку).

Правила знаходження елементів виразу.

1).Задати\* квадратну матрицю А порядку n. Отримати вектор(стовпець) , де b – вектор-стовпець, елементи якого обраховуються за формулою, згідно варіанту.

2).Задати квадратну матрицю А1 порядку n та вектори-стовпці b1 та c1 з n елементами кожен. Отримати вектор  згідно формули, що задається варіантом.

3).Задати квадратні матриці А2 та B2 порядку n. Отримати матрицю , яка залежить від А2, B2 та додатково визначеної матриці С2, елементи якої знаходяться за формулою, вказаною варіантом.

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | матриця | | |
| bi=6/i2 | A1(6b1-c1) | A2(10B2+C2)  Cij=1/(i+j)3 |

**ПОРЯДОК ВИКОНАННЯ РОБОТИ**

Вираз, який слід обрахувати:



При чому елементи  визначаються згідно правил:

, де  , і=1,2,...n



, де 

**Розрахунок:**

* Вектора-стовпця b:

;

;

;

* Матриці С2



















**Схема декомпозиції:**



**Код програми:**

#include <iostream>

#include <vector>

#include <ctime>

#include <iomanip>

#include <cmath>

using namespace std;

vector<vector<float>> mat\_Add\_mat(vector<vector<float>> a, vector<vector<float>> b, int size) {

vector<vector<float>> res;

res.resize(size);

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

res[i].resize(size);}

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

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

res[i][j] = (a[i][j] + b[i][j]);}}

return res;}

vector<vector<float>> mat\_Mult\_mat(vector<vector<float>> a, vector<vector<float>> b, int size) {

vector<vector<float>> res;

res.resize(size);

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

res[i].resize(size);}

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

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

res[i][j] = 0;

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

res[i][j] += a[i][k] \* b[k][j];}}}

return res;}

vector<vector<float>> val\_Mult\_mat(float a, vector<vector<float>> b, int size) {

vector<vector<float>> res;

res.resize(size);

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

res[i].resize(size);}

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

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

res[i][j] = a \* b[i][j];}}

return res;}

vector<float> mat\_Mult\_vect(vector<vector<float>> a, vector<float> b, int size) {

vector<float> res;

res.resize(size);

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

res[i] = 0;

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

res[i] += (a[i][j] \* b[j]);}}

return res;}

vector<float> vec\_Sub\_vec(vector<float> a, vector<float> b, int size) {

vector<float> res;

res.resize(size);

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

res[i] = a[i] - b[i];}

return res;}

vector<float> val\_Mult\_vect(int a, vector<float> b, int size) {

vector<float> res;

res.resize(size);

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

res[i] = a \* b[i];}

return res;}

vector<float> vectStr\_Mult\_mat(vector<float> a, vector<vector<float>> b, int size) {

vector<float> res;

res.resize(size);

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

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

res[i] += (a[j] \* b[j][i]);}}

return res;}

float vectCol\_Mult\_vectStr(vector<float> a, vector<float> b, int size) {

float res = 0;

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

res += a[i] \* b[i];}

return res;}

int main()

{

srand(time(NULL));

int n, i, j, choice = 0;

vector<float> b1, c1;

vector<float> b, y1, y2;

vector<vector<float>> A, A1, A2, B2, C2, Y3, x1 , x2;

float \_y2\_y1, \_y1\_y2, \_y2\_Y3\_y1;

vector<float> \_6b1, \_6b1\_c1, \_y2\_Y3;

vector<vector<float>> \_10C2, \_B2\_10C2, \_Y3\_2, add1, add2, add3, add1\_add2, add3\_Y3;

cout << "x = y2`\* y1 \* Y3^2 + y1`\* y2 \* Y3^2 + y2` \* Y3 \* y1 \* Y3 + Y3" << endl << endl;

start:

cout << "Enter size(n):";

cin >> n;

if (n < 3)

{cout << "n must be >= 3" << endl;

goto start;}

{b.resize(n); b1.resize(n); c1.resize(n);

y1.resize(n); y2.resize(n); \_6b1.resize(n);

\_6b1\_c1.resize(n); \_y2\_Y3.resize(n);

A.resize(n);

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

A[i].resize(n);}

A1.resize(n);

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

A1[i].resize(n);}

A2.resize(n);

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

A2[i].resize(n);}

B2.resize(n);

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

B2[i].resize(n);}

C2.resize(n);

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

C2[i].resize(n);}

Y3.resize(n);

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

Y3[i].resize(n);}

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

{ b[i] = 6 / (float)((i + 1) \* (i + 1)); }}

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

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

C2[i][j] = 1 / (pow((i + j + 2), 3));}}

x1.resize(n);

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

x1[i].resize(n);}

x2.resize(n);

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

x2[i].resize(n);}

\_10C2.resize(n);

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

\_10C2[i].resize(n);}

\_B2\_10C2.resize(n);

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

\_B2\_10C2[i].resize(n);}

\_Y3\_2.resize(n);

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

\_Y3\_2[i].resize(n);}

add1.resize(n);

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

add1[i].resize(n);}

add2.resize(n);

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

add2[i].resize(n);}

add3.resize(n);

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

add3[i].resize(n);}

add1\_add2.resize(n);

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

add1\_add2[i].resize(n);}

add3\_Y3.resize(n);

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

add3\_Y3[i].resize(n);}}

cout << endl << "1 Input Matrix" << endl << "2 Randomize" << endl << "Choice:";

cin >> choice;

if (choice == 1) {

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

cout << "b1[" << i + 1 << "]= ";

cin >> b1[i];}

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

cout << "c1[" << i + 1 << "]= ";

cin >> c1[i];}

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

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

cout << "A[" << i + 1 << "][" << j + 1 << "]= ";

cin >> A[i][j];}}

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

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

cout << "A1[" << i + 1 << "][" << j + 1 << "]= ";

cin >> A1[i][j];}}

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

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

cout << "A2[" << i + 1 << "][" << j + 1 << "]= ";

cin >> A2[i][j];}}

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

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

cout << "B2[" << i + 1 << "][" << j + 1 << "]= ";

cin >> B2[i][j];}}}

else if (choice == 2) {

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

b1[i] = 1 + (rand() % 10);}

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

c1[i] = 1 + (rand() % 10);}

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

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

A[i][j] = 1+ rand() % 10;}}

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

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

A1[i][j] =1 + rand() % 10;}}

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

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

A2[i][j] = 1+rand() % 10;}}

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

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

B2[i][j] = 1+rand() % 10;}}}

cout << "--------------------------------------------------" << endl;

{cout << "\t" << " " << "b" << endl;

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

cout << b[i] << "\t";}

cout << endl << endl;

cout << "\t" << "b1" << endl;

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

cout << b1[i] << "\t";}

cout << endl << endl;

cout << "\t" << "c1" << endl;

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

cout << c1[i] << "\t";}

cout << endl << endl;

cout <<"\t"<< "A" << endl;

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

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

cout << A[i][j] << "\t";}

cout << endl;}

cout << endl;

cout << "\t" << "A1" << endl;

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

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

cout << A1[i][j] << "\t";}

cout << endl;}

cout << endl;

cout << "\t" << "A2" << endl;

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

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

cout << A2[i][j] << "\t";}

cout << endl;}

cout << endl;

cout << "\t" << "B2" << endl;

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

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

cout << B2[i][j] << "\t";}

cout << endl;}

cout << endl;

cout << "\t" << "C2" << endl;

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

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

cout<<fixed << setprecision(3) << C2[i][j] << setw(3) << "\t";}

cout << endl;}

cout << endl;}

y1 = mat\_Mult\_vect(A, b, n); // y1 =A\*b (matrix\*vect\_colum = vect\_colum)

\_6b1 = val\_Mult\_vect (6,b1,n); // \_6b1 = 6 \* b1 (vect\_colum)

\_6b1\_c1 = vec\_Sub\_vec(\_6b1, c1, n); // \_6b1\_c1 = \_6b1 - c1 (vect\_colum)

y2 = mat\_Mult\_vect(A1, \_6b1\_c1, n); // y2 = A1 \* \_6b1\_c1 (matrix\*vect\_colum = vect\_colum)

\_10C2 = val\_Mult\_mat (10, C2, n); // \_10C2 = 10 \* C2 (matrix)

\_B2\_10C2 = mat\_Add\_mat(B2, \_10C2,n); // \_B2\_10C2 = B2 + \_10C2 (matrix)

Y3 = mat\_Mult\_mat(A2, \_B2\_10C2, n); // Y3 = A2 \* \_B2\_10C2 (matrix)

\_y2\_y1 = vectCol\_Mult\_vectStr(y2, y1, n); // \_y2\_y1 = y2` \* y1 (value)

\_Y3\_2 = mat\_Mult\_mat(Y3, Y3, n); // \_Y3\_2 = Y3 \* Y3 (matrix)

add1 = val\_Mult\_mat(\_y2\_y1, \_Y3\_2, n); // y2`y1Y3^2 = \_y2\_y1 \* \_Y3\_2 (matrix)

\_y1\_y2 = vectCol\_Mult\_vectStr(y1, y2, n); // \_y1\_y2 = y1` \* y2 (value)

add2 = val\_Mult\_mat(\_y1\_y2, \_Y3\_2, n); // y1`y2Y3^2 = \_y1\_y2 \* \_Y3\_2 (matrix)

\_y2\_Y3 = vectStr\_Mult\_mat(y2, Y3, n); // \_y2\_Y3 = y2` \* Y3 (string )

\_y2\_Y3\_y1 = vectCol\_Mult\_vectStr(\_y2\_Y3, y1, n); // \_y2\_Y3\_y1 = \_y2\_Y3 \* y1 (value)

add3 =val\_Mult\_mat (\_y2\_Y3\_y1, Y3, n); // y2`Y3y1Y3 = \_y2\_Y3\_y1 \* Y3 (matrix)

add1\_add2 = mat\_Add\_mat(add1, add2 , n); // add1\_add2 = add1 + add2 (matrix)

add3\_Y3 =mat\_Add\_mat (add3, Y3, n); // add3\_Y3 = add3 + Y3 (matrix)

x1 = mat\_Add\_mat (add1\_add2, add3\_Y3, n); // x (matrix)

//y1=A\*b

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

y1[i] = 0;

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

y1[i] += (A[i][j] \* b[j]);}}}

//y2=A1(6b1 - c1)

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

\_6b1[i] = 6 \* b1[i];}

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

\_6b1\_c1[i] = \_6b1[i] - c1[i];}

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

y2[i] = 0;

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

y2[i] += (A1[i][j] \* \_6b1\_c1[j]);}}}

//Y3 = A2\*(B2 + 10\*C2)

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

//\_10C2[i][j] = 0;

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

\_10C2[i][j] = 10 \* C2[i][j];}}

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

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

\_B2\_10C2[i][j] = (B2[i][j] + \_10C2[i][j]);}}

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

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

Y3[i][j] = 0;

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

Y3[i][j] += (A2[i][k] \* \_B2\_10C2[k][j]);}}}}

//\_Y3\_2 = Y3^2

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

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

\_Y3\_2[i][j] = 0;

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

\_Y3\_2[i][j] += Y3[i][k] \* Y3[k][j];}}}}

//\_y2\_y1 = y2`\*y1

{\_y2\_y1 = 0;

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

\_y2\_y1 += y2[i] \* y1[i];}}

// y2`y1Y3^2 = \_y2\_y1 \* \_Y3\_2

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

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

add1[i][j] = \_y2\_y1 \* \_Y3\_2[i][j];}}}

// \_y1\_y2 = y1` \* y2 (value)

{\_y1\_y2 = 0;

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

\_y1\_y2 += y1[i] \* y2[i];}}

// y1`y2Y3^2 = \_y1\_y2 \* \_Y3\_2 (matrix)

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

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

add2[i][j] = \_y1\_y2 \* \_Y3\_2[i][j];}}}

// \_y2\_Y3 = y2` \* Y3 (string )

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

\_y2\_Y3[i] = 0;

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

\_y2\_Y3[i] += (y2[j] \* Y3[j][i]);}}}

// \_y2\_Y3\_y1 = \_y2\_Y3 \* y1 (value)

{\_y2\_Y3\_y1 = 0;

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

\_y2\_Y3\_y1 += \_y2\_Y3[i] \* y1[i];}}

// y2`Y3y1Y3 = \_y2\_Y3\_y1 \* Y3 (matrix)

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

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

add3[i][j] = \_y2\_Y3\_y1 \* Y3[i][j];}}}

// add1\_add2 = add1 + add2 (matrix)

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

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

add1\_add2[i][j] = (add1[i][j] + add2[i][j]);}}}

// add3\_Y3 = add3 + Y3 (matrix)

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

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

add3\_Y3[i][j] = (add3[i][j] + Y3[i][j]);}}}

// x (matrix)

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

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

x2[i][j] = (add1\_add2[i][j] + add3\_Y3[i][j]);}}}

cout << "X1: " << endl;

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

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

cout << fixed << setprecision(3) << x1[i][j] << "\t";}

cout << endl;}

cout << endl << endl;

cout << "X2: " << endl;

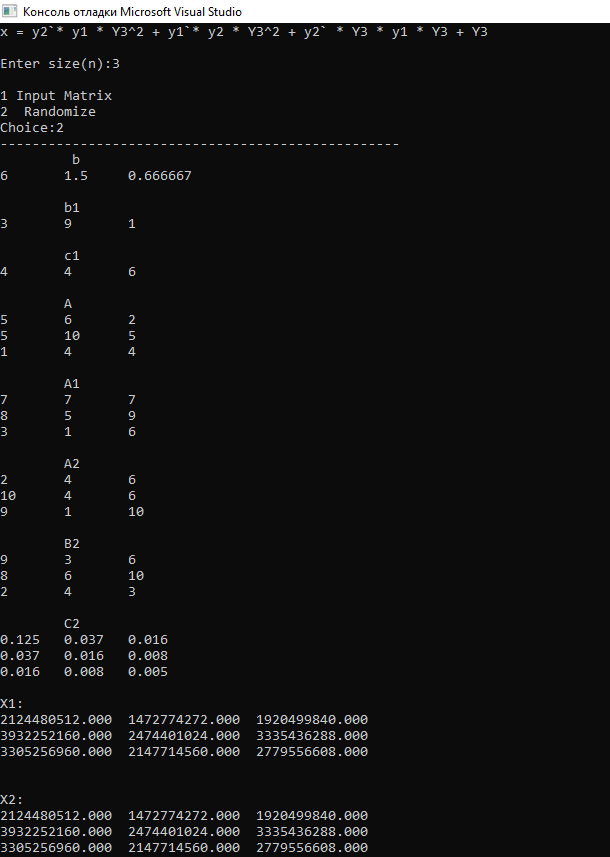
for (i = 0; i < n; i++) {

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

cout <<fixed << setprecision(3) << x2[i][j] << "\t";}

cout << endl;}

return 0;}

****

**Рис.1. Результат виконання програми**

**Висновок:** На цій лабораторній роботі я вивчив методи декомпозицій задач. Набув навиків розв’язування задач з використанням функціональної декомпозиції.