Лабораторна робота №2. Використання функціональної декомпозиції для розв'язку обчислювальних задач

Варіант № 8.

Мета роботи: вивчити методи декомпозицій задач. Набути навичок розв'язування задач з використанням функціональної декомпозиції.

Завдання.

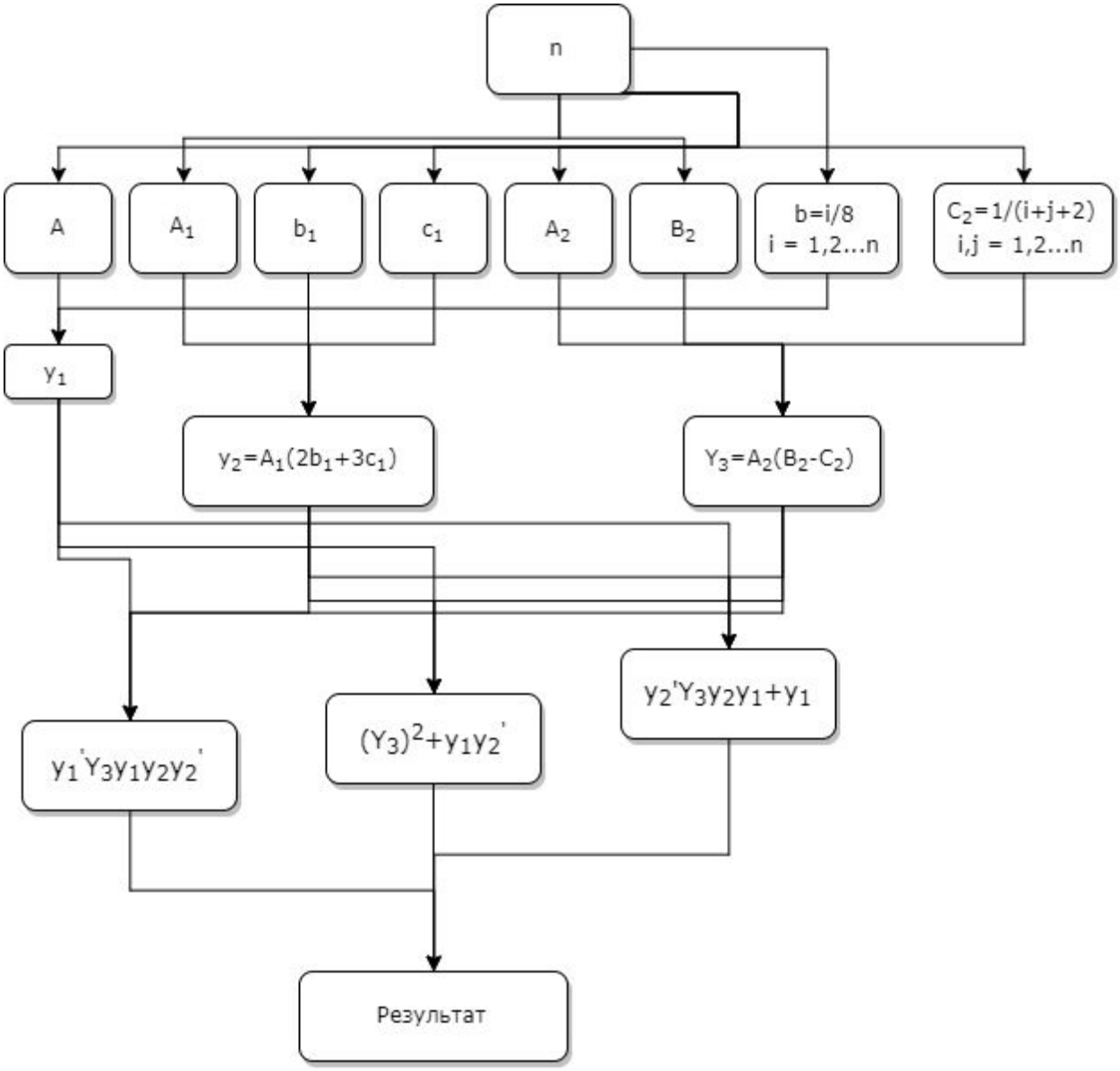
8	$x=(K_1 y_1' Y_3 y_1 y_2 y_2' + K_2 Y_3^2 + y_1 y_2')K_1(y_2' Y_3 y_2 y_1 + y_1)$		
	$b_i=8/i$ $i=1,2,...n$	$A_1(2b_1+3c_1)$	$A_2(B_2-C_2)$ $C_{ij}=1/(i+j+2)$

Схема декомпозиції задачі:
Для кожного окремого елемента, який потрібно буде обчислити - я створив getter, setter, а також lock object. Наприклад:

```
private Object bl = new Object();
Vector<double> _b = null;
public Vector<double> b {
   get {
       lock(bl) {
           if (_b == null) {
               _b = Vector<double>.Build.Dense(n, i => 8.0 / (1+i));
               if (debug) Console.WriteLine("Generated b: {0}", _b);
           }
       }
       return _b;
    }
   set { this._b = value;}
}
```

Тут реалізовано lazy calculation - тобто данні не обраховуються до тих пір, поки вони явно не знадобляться.
bl - це lock object для вектора b, він забезпечить відсутність зайвих перерахунків, якщо наприклад до цього вектора спробують одночасно доступитись два різних потоки.
_b - це прихована змінна, яка й містить сам вектор, зроблено це для того щоб до вектора можна було мати доступ лише через відповідно визначені функції.
b - це реалізація функцій для доступу та запису, як бачимо, тут використовується блокування, для того щоб не допустити зайвих обчислень у разі використання декількох потоків.

Декомпозиція прикладу



Як ми всі знаємо - створення потоку це не безкоштовна операція, іноді при занадто малих об’ємах обчислення створення ще одного потоку не тільки не дає пришвидшення, а й сповільнює роботу програми. Отже, я вирішив розбити обчислення прикладу на 3 великих частини, які містять достатню кількість обчислень, щоб виправдати створення для них окремого потоку.

Текст програми

```
using System;
using System.Diagnostics;
using System.Threading;
using MathNet.Numerics;
using MathNet.Numerics.LinearAlgebra;

namespace lab2
{
   class ExpressionNo8 {

       static void Main(string[] args) {
           ExpressionNo8 expr = new ExpressionNo8();
           int k1, k2;
           Stopwatch sw = new Stopwatch();
           while(true) {
               k1 = CLI.GetInt("Enter K1: ", 1);
               k2 = CLI.GetInt("Enter k2: ", 1);
               Console.WriteLine("Calculating results...");
               sw.Start();
               double result = expr.Calculate(k1, k2);
               sw.Stop();
               Console.WriteLine("Done!\nResult: {0}\nTook: {1} ms", result,
sw.ElapsedMilliseconds);
           }
       }

       bool debug;
       int n;

       Random _r = null;
       public Random r {
           get { if (_r == null) _r = new Random(); return _r;}
           set { _r = value; }
       }

       private Object bl = new Object();
       Vector<double> _b = null;
       public Vector<double> b {
           get {
               lock(bl) {
                   if (_b == null) {
                       _b = Vector<double>.Build.Dense(n, i => 8.0 / (1+i));

                       if (debug) Console.WriteLine("Generated b: {0}", _b);
                   }
               }
```

```
            return _b;
        }
        set { this._b = value;}
    }

    Vector<double> b1;
    Vector<double> c1;

    private Object y1l = new Object();
    Vector<double> _y1 = null;
    Vector<double> y1 {
        get {
            lock(y1l) {
                if (_y1 == null) {
                    _y1 = A * b;

                    if (debug) Console.WriteLine("Generated y1: {0}", _y1);
                }
            }
            return _y1;
        }
        set { _y1 = value;  }
    }

    private Object y2l = new Object();
    Vector<double> _y2 = null;
    Vector<double> y2 {
        get {
            lock(y2l) {
                if (_y2 == null) {
                    _y2 = A1 * (b1 + c1);

                    if (debug) Console.WriteLine("Generated y2: {0}", _y2);
                }
            }

            return _y2;
        }
        set { _y2 = value; }
    }

    Matrix<double> A;
    Matrix<double> A1;
    Matrix<double> A2;
    Matrix<double> B2;


    private Object C2l = new Object();
    Matrix<double> _C2 = null;
    Matrix<double> C2 {
```

```
            get {
                lock(C2l) {
                    if (_C2 == null) {
                        _C2 = Matrix<double>.Build.Dense(n, n, (i,j)=> 1.0 /
((i+1)+(j+1)+2));

                        if (debug) Console.WriteLine("Generated C2: {0}", _C2);
                    }
                }
                return _C2;
            }
            set { _C2 = value; }
        }

        private Object Y3l = new Object();
        Matrix<double> _Y3 = null;
        Matrix<double> Y3 {
            get {
                lock(Y3l) {
                    if (_Y3 == null) {
                        _Y3 = A2 * (B2 - C2);

                        if (debug) Console.WriteLine("Generated Y3: {0}", _Y3);
                    }

                }

                return _Y3;
            }
            set { _Y3 = value; }
        }

        public ExpressionNo8(int n, Matrix<double> A, Matrix<double> A1,
                Vector<double> b1, Vector<double> c1,
                Matrix<double> A2, Matrix<double> B2) {

            this.n = n;
            this.A = A;

            this.A1 = A1;
            this.b1 = b1;
            this.c1 = c1;

            this.A2 = A2;
            this.B2 = B2;

        }

        public ExpressionNo8() {
```

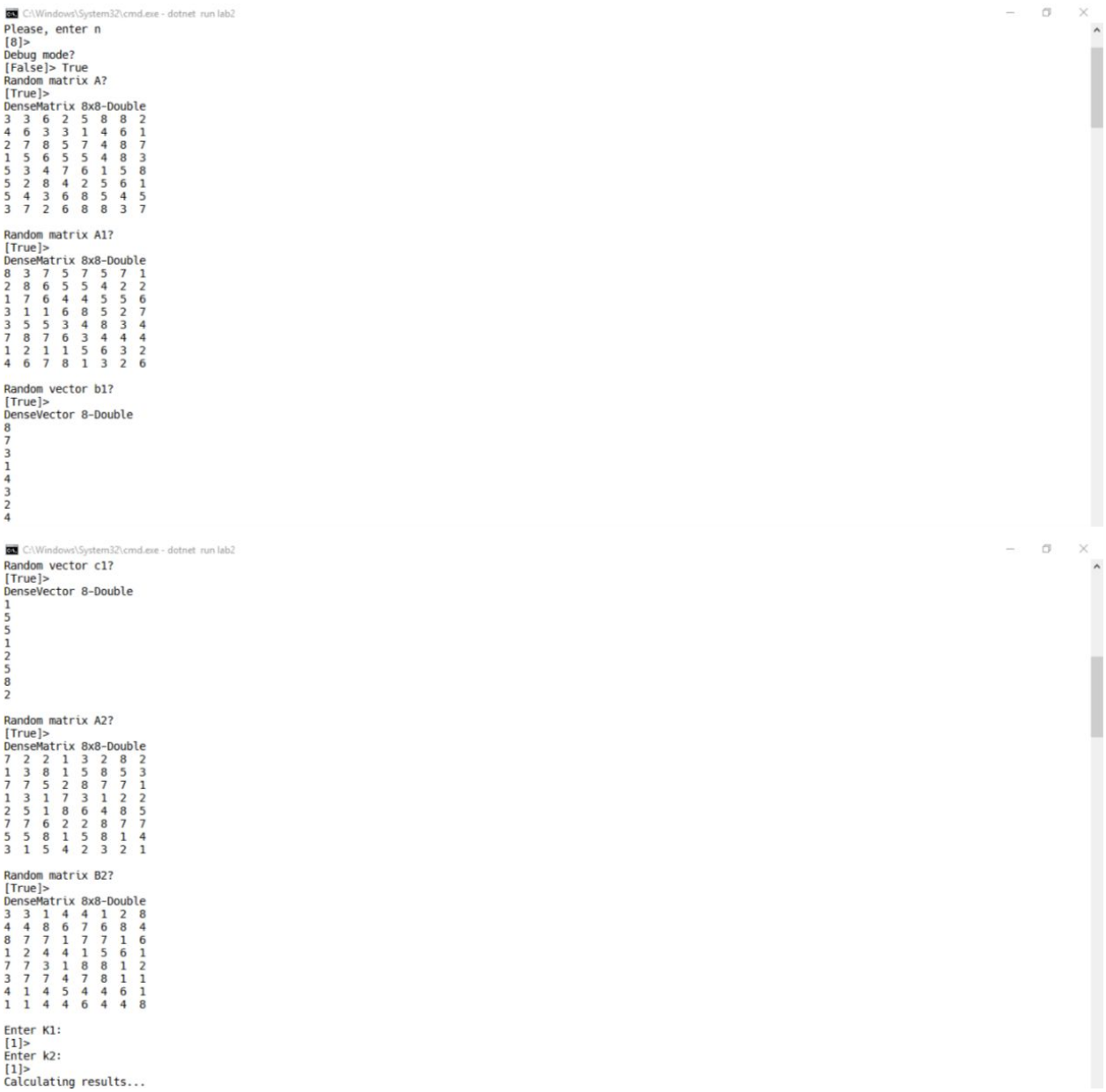
```
int n = CLI.GetInt("Please, enter n", 8);
bool debug = CLI.GetBool("Debug mode?", false);

Matrix<double> A = null;
bool ans = CLI.GetBool("Random matrix A?", true);
if (ans)
    A = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    A = CLI.GetMatrix("Enter A matrix.", n, n);
Console.WriteLine(A);

Matrix<double> A1 = null;
ans = CLI.GetBool("Random matrix A1?", true);
if (ans)
    A1 = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    A1 = CLI.GetMatrix("Enter A1 matrix.", n, n);
Console.WriteLine(A1);

Vector<double> b1 = null;
ans = CLI.GetBool("Random vector b1?", true);
if (ans)
    b1 = Vector<double>.Build.Dense(n, i => 1+r.Next(n));
else
    b1 = CLI.GetVector("Enter b1 vector.", n);
Console.WriteLine(b1);

Vector<double> c1 = null;
ans = CLI.GetBool("Random vector c1?", true);
if (ans)
    c1 = Vector<double>.Build.Dense(n, i => 1+r.Next(n));
else
    c1 = CLI.GetVector("Enter c1 vector.", n);
Console.WriteLine(c1);

Matrix<double> A2 = null;
ans = CLI.GetBool("Random matrix A2?", true);
if (ans)
    A2 = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    A2 = CLI.GetMatrix("Enter A2 matrix.", n, n);
Console.WriteLine(A2);

Matrix<double> B2 = null;
ans = CLI.GetBool("Random matrix B2?", true);
if (ans)
    B2 = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    B2 = CLI.GetMatrix("Enter B2 matrix.", n, n);
Console.WriteLine(B2);
```

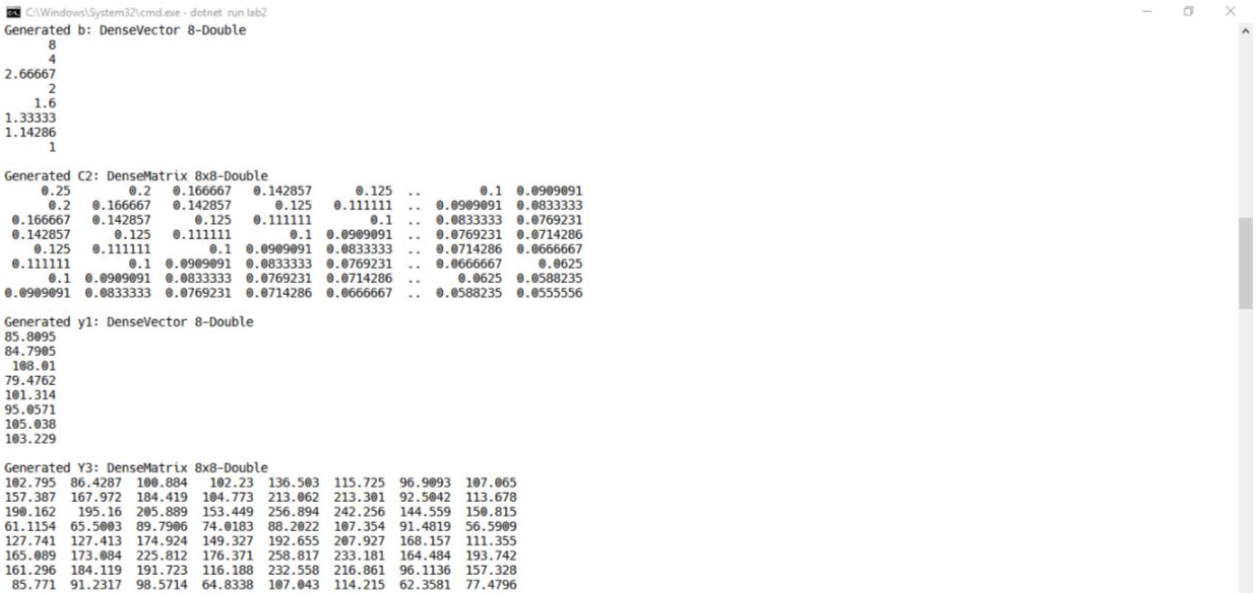
```
    calculate(n, A, A1, b1, c1, A2, B2, debug);
}


private void calculate(int n, Matrix<double> A, Matrix<double> A1,
        Vector<double> b1, Vector<double> c1,
        Matrix<double> A2, Matrix<double> B2, bool debug=false) {

    this.debug = debug;
    //
    this.n = n;
    this.A = A;

    this.A1 = A1;
    this.b1 = b1;
    this.c1 = c1;

    this.A2 = A2;
    this.B2 = B2;

}

public int K1;
public int K2;

private double firstPart = 0;
private void calculateFirstPart() {
    firstPart = K1 * y1 * Y3 * y1 * y2 * y2;
}

private Matrix<double> secondPart = null;
private void calculateSecondPart() {
    secondPart = K2 * Y3 * Y3 + y1 * y2;
}

private Vector<double> thirdPart = null;
private void calculateThirdPart() {
    thirdPart = y2 * Y3 * y2 * y1 + y1;
}
public double Calculate(int K1, int K2) {
    reset();
    this.K1 = K1;
    this.K2 = K2;

    Thread firstPartThread = new Thread(calculateFirstPart);
    Thread secondPartThread = new Thread(calculateSecondPart);
    Thread thirdPartThread = new Thread(calculateThirdPart);
    firstPartThread.Start();
    secondPartThread.Start();
```

```
            thirdPartThread.Start();

            firstPartThread.Join();
            secondPartThread.Join();
            thirdPartThread.Join();

            return ((firstPart + secondPart) * K1 * (thirdPart))[0];
        }

        private void reset() {
            b = null;
            y1 = null;
            y2 = null;
            Y3 = null;
            C2 = null;
        }

    }

    class CLI {

        static void showPrompt(string prompt, string current) {
            Console.WriteLine(prompt);
            Console.Write("[{0}]> ", current);
        }

        static string[] BOOL_YES = {"yes", "y", "true", "da", "sure"};
        static string[] BOOL_NO = {"no", "n", "false", "net", "not sure"};

        public static bool GetBool(string prompt, bool d) {
            showPrompt(prompt, d.ToString());
            string s = Console.ReadLine().Trim().ToLower();
            if (Array.Exists(BOOL_YES, e => s.Equals(e))) {
                return true;
            }
            else if (Array.Exists(BOOL_NO, e => s.Equals(e))) {
                return false;
            }
            return d;
        }

        public static int GetInt(string prompt, int d) {
            showPrompt(prompt, d.ToString());
            return ParseInt(d);
        }

        public static int ParseInt(int d) {
            int result = -1;
            bool success = Int32.TryParse(Console.ReadLine(), out result);
            if (success && result > 0)
```

```
            return result;
        return d;
    }

    public static Vector<double> GetVector(string prompt, int w) {
        int d = 1;
        Vector<double> result = Vector<double>.Build.Dense(w, d);
        Console.WriteLine(prompt);

        for(int j = 0; j < w; j++) {
            showPrompt("Enter element.", (j+1).ToString());
            result[j] = ParseInt(d);
        }
        Console.WriteLine(result);

        return result;
    }

    public static Matrix<double> GetMatrix(string prompt, int w, int h) {
        int d = 1;
        Matrix<double> result = Matrix<double>.Build.Dense(w, h, d);
        Console.WriteLine(prompt);

        for(int i = 0; i < h; i++) {
            for(int j = 0; j < w; j++) {
                showPrompt("Enter element.", String.Format("{0}, {1}", (j+1),
(i+1)));
                result[i,j] = ParseInt(d);
            }
            Console.WriteLine(result);
        }
        return result;
    }

  }
}
```

Приклад роботи програми





Висновок

В роботі був використаний паралелізм на рівні підзадач які виникли в результаті функціональної декомпозиції. Вирішення задачі представляє з себе консольну утиліту. В коді програми використано lock та Thread. Процес обчислення результатів продумано так - щоб не виникало колізій в пам'яті. Також є можливість вираховувати час виконання програми.