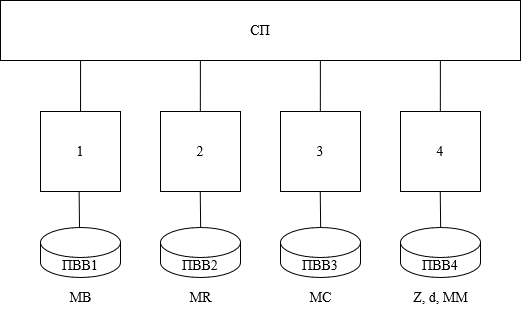
**Завдання**

Мова програмування: Java

Засоби організації взаємодії потоків: монітори мови Java, синхронізовані блоки

Математична задача: MR= MB\*(MC\*MM)\*d + min(Z)\*MC

Структура ПКС:



**Розробка програми**

Етап 1. Побудова паралельного алгоритму

1. ai = min(ZH), i=1..4

2. a = min(a, ai), i=1..4

3. MRH = MB\*(MC\*MMH)\*d + a\*MCH

Спільні ресурси: a, а, MB, MC, d, a.

Етап 2. Розробка алгоритмів

Точки синхронізації,

Задача Т1 критичні ділянки

1. Сигнал Т2, T3, T4 про введення даних. S2,3,4-1

2. Очікування завершення обчислень в Т2, T3, T4. W2,3,4-1

3. Копіювання a1 = a, MB1 = MB, MC1 = MC, d1 = d КД

4. Обчислення а1 = min(ZH).

5. Обчислення а = min(a, a1). КД

6. Сигнал Т2, Т3, Т4 про завершення обчислення а. S2,3,4-2

7. Очікування завершення обчислень в Т2, T3, T4. W2,3,4-2

8. MRH = MB1\*(MC1\*MMH)\*d1 + a1\*MCH

9. Сигнал Т2 про завершення обчислень. S2-3

Задача Т2

1. Сигнал Т1, T3, T4 про введення даних. S1,3,4-1

2. Очікування завершення обчислень в Т1, T3, T4. W1,3,4-1

3. Копіювання a2 = a, MB2 = MB, MC2 = MC, d2 = d КД

4. Обчислення а2 = min(ZH).

5. Обчислення а = min(a, a2). КД

6. Сигнал Т1, Т3, Т4 про завершення обчислення а. S1,3,4-2

7. Очікування завершення обчислень в Т1, T3, T4. W1,3,4-2

8. MRH = MB2\*(MC2\*MMH)\*d2 + a2\*MCH

9. Очікування завершення обчислень в Т1, Т3 Т4. W1,3,4-3

10. Вивід MRH.

Задача Т3

1. Сигнал Т1, T2, T4 про введення даних. S1,2,4-1

2. Очікування завершення обчислень в Т1, T2, T4. W1,2,4-1

3. Копіювання a3 = a, MВ3 = MB, MC3 = MC, d3 = d КД

4. Обчислення а3 = min(ZH).

5. Обчислення а = min(a, a3). КД

6. Сигнал Т1, Т2, Т4 про завершення обчислення а. S1,2,4-2

7. Очікування завершення обчислень в Т1, T2, T4. W1,2,4-2

8. MRH = MB3\*(MC3\*MMH)\*d3 + a3\*MCH

9. Сигнал Т2 про завершення обчислень. S2-3

Задача Т4

1. Сигнал Т1, T2, T3 про введення даних. S1,2,3-1

2. Очікування завершення обчислень в Т1, T2, T3. W1,2,3-1

3. Копіювання a4 = a, MВ4 = MB, MC4 = MC, d4 = d КД

4. Обчислення а4 = min(ZH).

5. Обчислення а = min(a, a4). КД

6. Сигнал Т1, Т2, Т3 про завершення обчислення а. S1,2,3-2

7. Очікування завершення обчислень в Т1, T2, T3. W1,2,3-2

8. MRH = MB4\*(MC4\*MMH)\*d4 + a4\*MCH

9. Сигнал Т2 про завершення обчислень. S2-3

Етап 3. Розробка структурної схеми взаємодії потоків

copyD - Копіювання спільного ресурсу d потоками Т1, Т2, Т3, Т4.

setD - Введення спільного ресурсу d потоком Т4.

copyMB - Копіювання спільного ресурсу МВ потоками Т1, Т2, Т3, Т4.

setMB - Введення спільного ресурсу MB потоком Т1.

copyMC - Копіювання спільного ресурсу МС потоками Т1, Т2, Т3, Т4.

setMC - Введення спільного ресурсу MC потоком Т3.

copyA - Копіювання спільного ресурсу а потоками Т1, Т2, Т3, Т4.

calcA - обчислення а = min(a, ai), i = 1..4 потоками Т1, Т2, Т3, Т4.

SignalCalcA - сигнал про завершення обчислення а в потоках Т1, Т2, Т3, Т4.

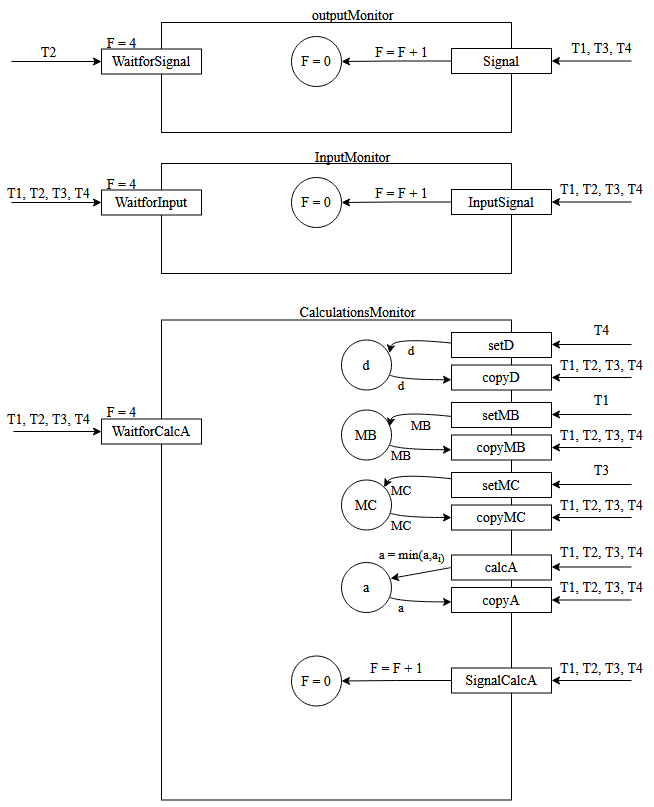
WaitforCalcA - очікування сигналів про завершення обчислення а в потоках Т1, Т2, Т3, Т4.

WaitforSignal - очікування сигналів про завершення обчислення MR в потоках Т1, Т3, Т4.

Signal - сигнал про завершення обчислення MR в потоках Т1, Т3, Т4.

WaitforInput - очікування сигналів про завершення вводу в потоках Т1, Т2, Т3, Т4.

InputSignal - сигнал про завершення вводу в потоках Т1, Т2, Т3, Т4.



Етап 4. Лістинг програми

1. /\*\* -- Lab #2. Java

2. \* -- MR= MB\*(MC\*MM)\*d + min(Z)\*MC

3. \* --

4. \* -- Date

5. \*/

6. public class Lab2{

7. public static int N = 8;

8. public static int P = 4;

9. public static int H = N/P;

10. static int a = Integer.MIN\_VALUE;

11. static int d;

12. static int[] Z = new int[N];

13. static int[][] MR = new int[N][N];

14. static int[][] MB = new int[N][N];

15. static int[][] MC = new int[N][N];

16. static int[][] MM = new int[N][N];

17. public static void main(String[] args) {

18. System.out.println("Lab2 started.");

19. InputMonitor inputMonitor = new InputMonitor();

20. CalculationMonitor calcMonitor = new CalculationMonitor();

21. OutputMonitor outputMonitor = new OutputMonitor();

22. T1 T1 = new T1(inputMonitor, calcMonitor, outputMonitor);

23. T2 T2 = new T2(inputMonitor, calcMonitor, outputMonitor);

24. T3 T3 = new T3(inputMonitor, calcMonitor, outputMonitor);

25. T4 T4 = new T4(inputMonitor, calcMonitor, outputMonitor);

26. T1.start();

27. T2.start();

28. T3.start();

29. T4.start();

30. try {

31. T1.join();

32. T2.join();

33. T3.join();

34. T4.join();

35. } catch (InterruptedException e) {

36. e.printStackTrace();

37. }

38. System.out.println("Lab2 finished.");

39. }

40. }

Calculations.java

1. import java.util.Arrays;

2. public class Calculations {

3. private static int H = Lab2.H;

4. public static void inputVector(int[] Vect, int val){

5. for (int i = 0; i < Vect.length; i++){

6. Vect[i] = val;

7. }

8. }

9. public static void inputMatrix(int[][] Matr, int val){

10. for (int i = 0; i < Matr.length; i++)

11. for (int j = 0; j < Matr[i].length; j++)

12. Matr[i][j] = val;

13. }

14. public static void outputMatrix(int[][] Mat){

15. for (int[] aVect : Mat) {

16. System.out.print(Arrays.toString(aVect) + " ");

17. }

18. }

19. public static int vectorMin(int[] vector, int id)

20. {

21. int min = vector[H \* id];

22. for (int i = H \* id; i < H \* (id + 1); i++) {

23. if (vector[i] < min)

24. min = vector[i];

25. }

26. return min;

27. }

28. }

CaclMonitor.java

1. public class CalculationMonitor {

2. private static int F = 0;

3. private int d = 1;

4. private int[][] MC = Lab2.MC;

5. private int[][] MB = Lab2.MB;

6. synchronized int copyA(int q) {

7. return q;

8. }

9. synchronized int calcA(int val1, int val2) { return val1 < val2 ? val1 : val2; }

10. synchronized int copyD() { return this.d; }

11. synchronized int[][] copyMC() { return this.MC; }

12. synchronized int[][] copyMB() { return this.MB; }

13. synchronized void SignalCalcA() {

14. F++;

15. if (F >= 4) notifyAll();

16. }

17. synchronized void WaitforCalcA(){

18. try {

19. if (F < 4) wait();

20. } catch(Exception e){

21. e.printStackTrace();

22. }

23. }

24. }

InputMonitor.java

1. public class InputMonitor {

2. private static int F = 0;

3. synchronized void InputSignal(){

4. F++;

5. if (F >= 4) notifyAll();

6. }

7. synchronized void WaitforInput(){

8. try {

9. if (F < 2) wait();

10. } catch (Exception e){

11. e.printStackTrace();

12. }

13. }

14. }

OutputMonitor.java

15. public class OutputMonitor {

16. private static int F = 0;

17. synchronized void Signal(){

18. F++;

19. if (F >= 4) notifyAll();

20. }

21. synchronized void WaitforSignal(){

22. try {

23. if (F < 3) wait();

24. } catch (Exception e){

25. e.printStackTrace();

26. }

27. }

28. }

T1.java

1. public class T1 extends Thread{

2. private InputMonitor inputMonitor;

3. private CalculationMonitor calcMonitor;

4. private OutputMonitor outputMonitor;

5. T1(InputMonitor inputMonitor, CalculationMonitor calcMonitor, OutputMonitor outputMonitor){

6. this.calcMonitor = calcMonitor;

7. this.inputMonitor = inputMonitor;

8. this.outputMonitor = outputMonitor;

9. }

10. public void run(){

11. int minRes;

12. int start = 0, end = Lab2.H;

13. System.out.println("T1 started");

14. Lab2.d = 1;

15. Calculations.inputMatrix(Lab2.MB, 1);

16. Calculations.inputMatrix(Lab2.MC, 1);

17. Calculations.inputVector(Lab2.Z,1);

18. Calculations.inputMatrix(Lab2.MR, 1);

19. Calculations.inputMatrix(Lab2.MM, 1);

20. // cигнал Т2, Т3, T4 про завершення введення даних

21. inputMonitor.InputSignal();

22. // oчікування завершення введення даних у інших потоках

23. inputMonitor.WaitforInput();

24. // копія MB, MC, d.

25. int d1 = calcMonitor.copyD();

26. int[][] MB1 = calcMonitor.copyMB();

27. int[][] MC1 = calcMonitor.copyMC();

28. int e = 0;

29. // обчислення a1 = min(BH)

30. minRes = Calculations.vectorMin(Lab2.Z, e);

31. // обчислення a

32. Lab2.a = calcMonitor.calcA(minRes, Lab2.a);

33. // сигнал потокам про завершення обчислення а

34. calcMonitor.SignalCalcA();

35. // очікування сигналу від потоків про завершення обчислення а.

36. calcMonitor.WaitforCalcA();

37. // копія а

38. int a1 = calcMonitor.copyA(Lab2.a);

39. // Обчислення MRН = MB1\*(MC1\*MMH)\*d1 + a1\*MCH

40. int[][] MC\_M = new int[Lab2.N][Lab2.N];

41. for (int j = start; j < end; j++){

42. for (int i = 0; i < Lab2.N; i++){

43. MC\_M[i][j] = 0;

44. for (int k = 0; k < Lab2.N; k++){

45. MC\_M[i][j] += MC1[i][k] \* Lab2.MM[j][k];

46. }

47. }

48. }

49. int[][] MB\_CM = new int[Lab2.N][Lab2.N];

50. for (int j = start; j < end; j++){

51. for (int i = 0; i < Lab2.N; i++){

52. MB\_CM[i][j] = 0;

53. for (int k = 0; k < Lab2.N; k++){

54. MB\_CM[i][j] += MB1[i][k] \* MC\_M[j][k];

55. }

56. }

57. }

58. for (int j = start; j < end; j++){

59. for (int i = 0; i < Lab2.N; i++){

60. Lab2.MR[i][j] = MB\_CM[i][j]\*d1 + a1\*Lab2.MC[i][j];

61. }

62. }

63. // cигнал про завершення обчислення MR.

64. outputMonitor.Signal();

65. System.out.println("T1 finished.");

66. }

67. }

T2.java

1. public class T2 extends Thread{

2. private InputMonitor inputMonitor;

3. private CalculationMonitor calcMonitor;

4. private OutputMonitor outputMonitor;

5. T2(InputMonitor inputMonitor, CalculationMonitor calcMonitor, OutputMonitor outputMonitor){

6. this.inputMonitor = inputMonitor;

7. this.calcMonitor = calcMonitor;

8. this.outputMonitor = outputMonitor;

9. }

10. public void run(){

11. int minrEs;

12. int start = Lab2.H, end = Lab2.H\*2;

13. System.out.println("T2 started");

14. Lab2.d = 1;

15. Calculations.inputMatrix(Lab2.MB, 1);

16. Calculations.inputMatrix(Lab2.MC, 1);

17. Calculations.inputVector(Lab2.Z,1);

18. Calculations.inputMatrix(Lab2.MR, 1);

19. Calculations.inputMatrix(Lab2.MM, 1);

20. // cигнал Т1, Т3, T4 про завершення введення даних

21. inputMonitor.InputSignal();

22. // oчікування завершення введення даних у інших потоках

23. inputMonitor.WaitforInput();

24. // копія MB, MC, d.

25. int d2 = calcMonitor.copyD();

26. int[][] MB2 = calcMonitor.copyMB();

27. int[][] MC2 = calcMonitor.copyMC();

28. int e = 1;

29. // обчислення a2 = min(BH)

30. minrEs = Calculations.vectorMin(Lab2.Z, e);

31. // обчислення a

32. Lab2.a = calcMonitor.calcA(minrEs, Lab2.a);

33. // сигнал потокам про завершення обчислення а

34. calcMonitor.SignalCalcA();

35. // очікування сигналу від потоків про завершення обчислення а.

36. calcMonitor.WaitforCalcA();

37. // копія а

38. int a2 = calcMonitor.copyA(Lab2.a);

39. // Обчислення MRН = MB2\*(MC2\*MMH)\*d2 + a2\*MCH

40. int[][] MC\_M = new int[Lab2.N][Lab2.N];

41. for (int j = start; j < end; j++){

42. for (int i = 0; i < Lab2.N; i++){

43. MC\_M[i][j] = 0;

44. for (int k = 0; k < Lab2.N; k++){

45. MC\_M[i][j] += MC2[i][k] \* Lab2.MM[j][k];

46. }

47. }

48. }

49. int[][] MB\_CM = new int[Lab2.N][Lab2.N];

50. for (int j = start; j < end; j++){

51. for (int i = 0; i < Lab2.N; i++){

52. MB\_CM[i][j] = 0;

53. for (int k = 0; k < Lab2.N; k++){

54. MB\_CM[i][j] += MB2[i][k] \* MC\_M[j][k];

55. }

56. }

57. }

58. for (int j = start; j < end; j++){

59. for (int i = 0; i < Lab2.N; i++){

60. Lab2.MR[i][j] = MB\_CM[i][j]\*d2 + a2\*Lab2.MC[i][j];

61. }

62. }

63. // очікування сигналу від Т1, Т3, Т4 про завершення обчислення MR.

64. outputMonitor.WaitforSignal();

65. // вивід MR.

66. if (Lab2.N <= 15){

67. System.out.print("[ ");

68. Calculations.outputMatrix(Lab2.MR);

69. System.out.println("]");

70. }

71. System.out.println("T2 finished.");

72. }

73. }

T3.java

1. public class T3 extends Thread{

2. private InputMonitor inputMonitor;

3. private CalculationMonitor calcMonitor;

4. private OutputMonitor outputMonitor;

5. T3(InputMonitor inputMonitor, CalculationMonitor calcMonitor, OutputMonitor outputMonitor){

6. this.calcMonitor = calcMonitor;

7. this.inputMonitor = inputMonitor;

8. this.outputMonitor = outputMonitor;

9. }

10. public void run(){

11. int minreS;

12. int start = Lab2.H\*2, end = Lab2.H\*3;

13. System.out.println("T3 started");

14. Lab2.d = 1;

15. Calculations.inputMatrix(Lab2.MB, 1);

16. Calculations.inputMatrix(Lab2.MC, 1);

17. Calculations.inputVector(Lab2.Z,1);

18. Calculations.inputMatrix(Lab2.MR, 1);

19. Calculations.inputMatrix(Lab2.MM, 1);

20. // cигнал Т2, Т3, T4 про завершення введення даних

21. inputMonitor.InputSignal();

22. // oчікування завершення введення даних у інших потоках

23. inputMonitor.WaitforInput();

24. // копія MB, MC, d.

25. int d3 = calcMonitor.copyD();

26. int[][] MB3 = calcMonitor.copyMB();

27. int[][] MC3 = calcMonitor.copyMC();

28. int e = 2;

29. // обчислення a3 = min(BH)

30. minreS = Calculations.vectorMin(Lab2.Z, e);

31. // обчислення a

32. Lab2.a = calcMonitor.calcA(minreS, Lab2.a);

33. // сигнал потокам про завершення обчислення а

34. calcMonitor.SignalCalcA();

35. // очікування сигналу від потоків про завершення обчислення а.

36. calcMonitor.WaitforCalcA();

37. // копія а

38. int a3 = calcMonitor.copyA(Lab2.a);

39. // Обчислення MRН = MB3\*(MC3\*MMH)\*d3 + a3\*MCH

40. int[][] MC\_M = new int[Lab2.N][Lab2.N];

41. for (int j = start; j < end; j++){

42. for (int i = 0; i < Lab2.N; i++){

43. MC\_M[i][j] = 0;

44. for (int k = 0; k < Lab2.N; k++){

45. MC\_M[i][j] += MC3[i][k] \* Lab2.MM[j][k];

46. }

47. }

48. }

49. int[][] MB\_CM = new int[Lab2.N][Lab2.N];

50. for (int j = start; j < end; j++){

51. for (int i = 0; i < Lab2.N; i++){

52. MB\_CM[i][j] = 0;

53. for (int k = 0; k < Lab2.N; k++){

54. MB\_CM[i][j] += MB3[i][k] \* MC\_M[j][k];

55. }

56. }

57. }

58. for (int j = start; j < end; j++){

59. for (int i = 0; i < Lab2.N; i++){

60. Lab2.MR[i][j] = MB\_CM[i][j]\*d3 + a3\*Lab2.MC[i][j];

61. }

62. }

63. // cигнал про завершення обчислення MR.

64. outputMonitor.Signal();

65. System.out.println("T3 finished.");

66. }

67. }

T4.java

1. public class T4 extends Thread{

2. private InputMonitor inputMonitor;

3. private CalculationMonitor calcMonitor;

4. private OutputMonitor outputMonitor;

5. T4(InputMonitor inputMonitor, CalculationMonitor calcMonitor, OutputMonitor outputMonitor){

6. this.calcMonitor = calcMonitor;

7. this.inputMonitor = inputMonitor;

8. this.outputMonitor = outputMonitor;

9. }

10. public void run(){

11. int miNres;

12. int start = Lab2.H\*3, end = Lab2.N;

13. System.out.println("T3 started");

14. Lab2.d = 1;

15. Calculations.inputMatrix(Lab2.MB, 1);

16. Calculations.inputMatrix(Lab2.MC, 1);

17. Calculations.inputVector(Lab2.Z,1);

18. Calculations.inputMatrix(Lab2.MR, 1);

19. Calculations.inputMatrix(Lab2.MM, 1);

20. // cигнал Т2, Т3, T4 про завершення введення даних

21. inputMonitor.InputSignal();

22. // oчікування завершення введення даних у інших потоках

23. inputMonitor.WaitforInput();

24. // копія MB, MC, d.

25. int d4 = calcMonitor.copyD();

26. int[][] MB4 = calcMonitor.copyMB();

27. int[][] MC4 = calcMonitor.copyMC();

28. int e = 3;

29. // обчислення a4 = min(BH)

30. miNres = Calculations.vectorMin(Lab2.Z, e);

31. // обчислення a

32. Lab2.a = calcMonitor.calcA(miNres, Lab2.a);

33. // сигнал потокам про завершення обчислення а

34. calcMonitor.SignalCalcA();

35. // очікування сигналу від потоків про завершення обчислення а.

36. calcMonitor.WaitforCalcA();

37. // копія а

38. int a4 = calcMonitor.copyA(Lab2.a);

39. // Обчислення MRН = MB4\*(MC4\*MMH)\*d4 + a4\*MCH

40. int[][] MC\_M = new int[Lab2.N][Lab2.N];

41. for (int j = start; j < end; j++){

42. for (int i = 0; i < Lab2.N; i++){

43. MC\_M[i][j] = 0;

44. for (int k = 0; k < Lab2.N; k++){

45. MC\_M[i][j] += MC4[i][k] \* Lab2.MM[j][k];

46. }

47. }

48. }

49. int[][] MB\_CM = new int[Lab2.N][Lab2.N];

50. for (int j = start; j < end; j++){

51. for (int i = 0; i < Lab2.N; i++){

52. MB\_CM[i][j] = 0;

53. for (int k = 0; k < Lab2.N; k++){

54. MB\_CM[i][j] += MB4[i][k] \* MC\_M[j][k];

55. }

56. }

57. }

58. for (int j = start; j < end; j++){

59. for (int i = 0; i < Lab2.N; i++){

60. Lab2.MR[i][j] = MB\_CM[i][j]\*d4 + a4\*Lab2.MC[i][j];

61. }

62. }

63. // cигнал про завершення обчислення MR.

64. outputMonitor.Signal();

65. System.out.println("T3 finished.");

66. }

67. }