Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №7

Паралельне програмування в мові Ada.

Рандеву.

Виконав:

студент групи ІП-22

Сочка О. О.

Перевірив:

Корочкін О. В.

Київ - 2015

**Технічне завдання**



**Етап 1. Побудова паралельного алгоритму**

Пояснення до використовуваних констант:

*  – розмірність векторів і матриць;
*  – кількість ядер;
* .

**Етап 2. Розробка алгоритмів процесів (задач)**

**Задача :**

1. Введення
2. Передати Z, T в .
3. Передати Z, T в .
4. Прийняти від .
5. Прийняти від .
6. Передати в
7. Обчислення

1. Передати в .
2. Прийняти від .
3. Обчислення

1. Прийняти від .
2. Прийняти від .
3. Виведення *A*.

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Передати *Z,T* в .
4. Прийняти  від .
5. Прийняти від .
6. Передати в
7. Обчислення
8. Передати в .
9. Прийняти від .
10. Обчислення
11. Прийняти від .
12. Прийняти  від
13. Передати в .

**Задача :**

1. Введення
2. Прийняти *Z,T* від .
3. Передати *Z,T* в .
4. Прийняти від .
5. в .
6. в .
7. Обчислення
8. Передати в .
9. Прийняти від .
10. Обчислення
11. Прийняти від .
12. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Прийняти від .
4. Передати в
5. Прийняти від .
6. Передати в .
7. Обчислення
8. Прийняти від .
9. Обчислення
10. Передати в .
11. Прийняти від .
12. Передати в
13. Обчислення
14. Прийняти від .
15. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Прийняти від .
4. Передати в
5. Прийняти від .
6. Передати в .
7. Обчислення
8. Прийняти від .
9. Обчислення
10. Прийняти від .
11. Обчислення
12. Передати в
13. Прийняти від .
14. Передати в
15. Передати в
16. Обчислення
17. Прийняти від .
18. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Прийняти від .
4. Передати в
5. Прийняти від .
6. Передати в .
7. Обчислення
8. Прийняти від .
9. Обчислення
10. Прийняти від .
11. Обчислення
12. Прийняти від .
13. Обчислення
14. Передати в
15. Прийняти від .
16. Передати в
17. Передати в
18. Передати в
19. Обчислення
20. Прийняти від .
21. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Прийняти від .
4. Передати в
5. Прийняти від .
6. Передати в .
7. Обчислення
8. Прийняти від .
9. Обчислення
10. Прийняти від .
11. Обчислення
12. Прийняти від .
13. Обчислення
14. Прийняти від .
15. Обчислення
16. Передати в
17. Передати в
18. Передати в
19. Передати в
20. Обчислення
21. Прийняти від .
22. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Прийняти від .
4. Передати в
5. Прийняти від .
6. Передати в .
7. Обчислення
8. Прийняти від .
9. Обчислення
10. Прийняти від .
11. Обчислення
12. Прийняти від .
13. Обчислення
14. Передати в
15. Прийняти від .
16. Передати в
17. Передати в
18. Передати в
19. Обчислення
20. Прийняти від .
21. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Прийняти від .
4. Передати в
5. Прийняти від .
6. Передати в .
7. Обчислення
8. Прийняти від .
9. Обчислення
10. Прийняти від .
11. Обчислення
12. Передати в
13. Прийняти від .
14. Передати в
15. Передати в
16. Обчислення
17. Прийняти від .
18. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Передати *Z,T* в .
3. Прийняти від .
4. Передати в
5. Прийняти від .
6. Передати в .
7. Обчислення
8. Прийняти від .
9. Обчислення
10. Передати в .
11. Прийняти від .
12. Передати в
13. Обчислення
14. Прийняти від .
15. Передати в .

**Задача :**

1. Введення .
2. Прийняти від
3. Передати в .
4. Передати в .
5. Прийняти від
6. Обчислення
7. Передати в .
8. Прийняти від .
9. Обчислення
10. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Прийняти від .
3. Передати в .
4. Передати в .
5. Прийняти від .
6. Обчислення
7. Передати в .
8. Прийняти від .
9. Обчислення
10. Передати в .

**Задача :**

1. Прийняти *Z,T* від .
2. Прийняти  від .
3. Передати в .
4. Прийняти *MK* від .
5. Обчислення
6. Передати в .
7. Прийняти від .
8. Обчислення
9. Передати в .

**Етап 3. Розробка схеми взаємодії задач**



**Етап 4. Розробка програми**

**1 -----------------------------------------------------------**

**2 -- Lab7. Ada. Randevouz**

**3 -- Name: Sochka Oleksandr Oleksandrovich**

**4 -- Group: IP-22**

**5 -- Date: 26.04.2015**

**6 -- MA = a \* B \*(MO\*MX) + (Z\*E) \* R**

**7 -----------------------------------------------------------**

**8**

**9 with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;**

**10 use Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;**

**11**

**12 procedure Lab7 is**

**13 N: Integer := 400;**

**14 D: Integer := 4; -- grid dimension**

**15 P: Integer := D \* D; -- number of processes**

**16 H: Integer := N / P;**

**17 S: Integer := D / 2 + 1;**

**18**

**19 StartTime, FinishTime: Time;**

**20 DiffTime: Duration;**

**21**

**22 type Vector is array (1..N) of Integer;**

**23 type Matrix is array (1..N) of Vector;**

**24**

**25 ---------------------------------------------------------------**

**26 procedure VectorInput(M: out Vector) is**

**27 begin**

**28 for i in 1..N loop**

**29 M(i) := 1;**

**30 end loop;**

**31 end VectorInput;**

**32**

**33 procedure VectorOutput(M: in Vector) is**

**34 begin**

**35 if ( N < 20 ) then**

**36 for i in 1..N loop**

**37 Put(M(i));**

**38 end loop;**

**39 New\_Line;**

**40 end if;**

**41 end VectorOutput;**

**42**

**43 procedure MatrixInput(MA: out Matrix) is**

**44 begin**

**45 for i in 1..N loop**

**46 for j in 1..N loop**

**47 MA(i)(j) := 1;**

**48 end loop;**

**49 end loop;**

**50 end MatrixInput;**

**51**

**52 function Max(x, y: in Integer) return Integer is**

**53 begin**

**54 if ( x > y ) then**

**55 return x;**

**56 end if;**

**57 return y;**

**58 end;**

**59**

**60 function VectorMax(beginning, ending: in Integer; M: in Vector) return Integer is**

**61 a: Integer;**

**62 begin**

**63 a := Integer'First;**

**64 for i in beginning..ending loop**

**65 a := Max(a, M(i));**

**66 end loop;**

**67 return a;**

**68 end VectorMax;**

**69**

**70 procedure CalculateA(beginning, ending, z, alpha: Integer; T, B: in Vector; MO, MK: in Matrix; A: out Vector) is**

**71 tmp: Integer;**

**72 begin**

**73 for i in beginning..ending loop**

**74 A(i) := 0;**

**75 tmp := 0;**

**76 for j in 1..N loop**

**77 for k in 1..N loop**

**78 tmp := tmp + MO(j)(k) \* MK(k)(i);**

**79 end loop;**

**80 end loop;**

**81 A(i) := A(i) + z \* T(i) + alpha \* B(i) \* tmp;**

**82 end loop;**

**83 end CalculateA;**

**84**

**85 ---------------------------------------------------------------**

**86 task type Task\_I(Tiid, Tjid: Integer) is**

**87 entry DataInput1(Z, T: in Vector);**

**88 entry DataInput2(alpha: in Integer; B: in Vector; MO: in Matrix);**

**89 entry DataInput3(MK: in Matrix);**

**90**

**91 entry DataResult\_z(z: in Integer);**

**92 entry DataResult\_zi(zi: in Integer);**

**93**

**94 entry Result\_A(A: in Vector);**

**95 end Task\_I;**

**96 ---------------------------------------------------------------**

**97 type Task\_I\_Ptr is access Task\_I;**

**98**

**99 type Tasks\_Array is array (1..N) of Task\_I\_Ptr;**

**100 type Tasks\_Matrix is array (1..N) of Tasks\_Array;**

**101**

**102 tasksMatrix: Tasks\_Matrix;**

**103**

**104 ---------------------------------------------------------------**

**105 task body Task\_I is**

**106 beginning: Integer := ((Tiid - 1) \* D + Tjid - 1) \* H + 1;**

**107 ending: Integer := ((Tiid - 1) \* D + Tjid) \* H;**

**108**

**109 Ai, Zi, Ti, Bi: Vector;**

**110 MOi, MKi: Matrix;**

**111 alphai, zl, z1: Integer; -- zl - max in task; z1 - max total**

**112 begin**

**113 Put\_Line("task" & Integer'Image((Tiid - 1) \* D + Tjid) & " started...");**

**114**

**115 -- DATA INPUT START**

**116 if Tiid = 1 then**

**117 -- input 1**

**118 if Tjid = 1 then**

**119 VectorInput(Zi);**

**120 VectorInput(Ti);**

**121 else**

**122 accept DataInput1(Z, T: in Vector) do**

**123 Zi := Z;**

**124 Ti := T;**

**125 end DataInput1;**

**126 end if;**

**127 if Tjid < D then**

**128 tasksMatrix(Tiid)(Tjid + 1).DataInput1(Zi, Ti);**

**129 end if;**

**130 tasksMatrix(Tiid + 1)(Tjid).DataInput1(Zi, Ti);**

**131**

**132 -- input 2**

**133 accept DataInput2(alpha: in Integer; B: in Vector; MO: in Matrix) do**

**134 alphai := alpha;**

**135 Bi := B;**

**136 MOi := MO;**

**137 end DataInput2;**

**138**

**139 -- input 3**

**140 if Tjid = D then**

**141 MatrixInput(MKi);**

**142 else**

**143 accept DataInput3(MK: in Matrix) do**

**144 MKi := MK;**

**145 end DataInput3;**

**146 end if;**

**147 if Tjid > 1 then**

**148 tasksMatrix(Tiid)(Tjid - 1).DataInput3(MKi);**

**149 end if;**

**150 tasksMatrix(Tiid + 1)(Tjid).DataInput3(MKi);**

**151 elsif Tiid < D then**

**152 -- input 1**

**153 accept DataInput1(Z, T: in Vector) do**

**154 Zi := Z;**

**155 Ti := T;**

**156 end DataInput1;**

**157 tasksMatrix(Tiid + 1)(Tjid).DataInput1(Zi, Ti);**

**158**

**159 -- input 2**

**160 accept DataInput2(alpha: in Integer; B: in Vector; MO: in Matrix) do**

**161 alphai := alpha;**

**162 Bi := B;**

**163 MOi := MO;**

**164 end DataInput2;**

**165 tasksMatrix(Tiid - 1)(Tjid).DataInput2(alphai, Bi, MOi);**

**166**

**167 -- input 3**

**168 accept DataInput3(MK: in Matrix) do**

**169 MKi := MK;**

**170 end DataInput3;**

**171 tasksMatrix(Tiid + 1)(Tjid).DataInput3(MKi);**

**172 else -- TiiD = D**

**173 -- input 1**

**174 accept DataInput1(Z, T: in Vector) do**

**175 Zi := Z;**

**176 Ti := T;**

**177 end DataInput1;**

**178**

**179 -- input 2**

**180 if Tjid = D then**

**181 alphai := 1;**

**182 VectorInput(Bi);**

**183 MatrixInput(MOi);**

**184 else**

**185 accept DataInput2(alpha: in Integer; B: in Vector; MO: in Matrix) do**

**186 alphai := alpha;**

**187 Bi := B;**

**188 MOi := MO;**

**189 end DataInput2;**

**190 end if;**

**191 if Tjid > 1 then**

**192 tasksMatrix(Tiid)(Tjid - 1).DataInput2(alphai, Bi, MOi);**

**193 end if;**

**194 tasksMatrix(Tiid - 1)(Tjid).DataInput2(alphai, Bi, MOi);**

**195**

**196 -- input 3**

**197 accept DataInput3(MK: in Matrix) do**

**198 MKi := MK;**

**199 end DataInput3;**

**200 end if;**

**201 -- DATA INPUT END**

**202**

**203 -- Put\_Line("task" & Integer'Image((Tiid - 1) \* D + Tjid) & " has all input data...");**

**204**

**205 zl := VectorMax(beginning, ending, Zi);**

**206**

**207 if Tiid = 1 then**

**208 tasksMatrix(Tiid + 1)(Tjid).DataResult\_zi(zl);**

**209 accept DataResult\_z(z: in Integer) do**

**210 z1 := z;**

**211 end DataResult\_z;**

**212 elsif Tiid < S then**

**213 accept DataResult\_zi(zi: in Integer) do**

**214 zl := Max(zl, zi);**

**215 end DataResult\_zi;**

**216 tasksMatrix(Tiid + 1)(Tjid).DataResult\_zi(zl);**

**217 accept DataResult\_z(z: in Integer) do**

**218 z1 := z;**

**219 end DataResult\_z;**

**220 tasksMatrix(Tiid - 1)(Tjid).DataResult\_z(z1);**

**221 elsif Tiid = S then**

**222 accept DataResult\_zi(zi: in Integer) do**

**223 zl := Max(zl, zi);**

**224 end DataResult\_zi;**

**225 accept DataResult\_zi(zi: in Integer) do**

**226 zl := Max(zl, zi);**

**227 end DataResult\_zi;**

**228 if Tjid < S then**

**229 if Tjid > 1 then**

**230 accept DataResult\_zi(zi: in Integer) do**

**231 zl := Max(zl, zi);**

**232 end DataResult\_zi;**

**233 end if;**

**234 tasksMatrix(Tiid)(Tjid + 1).DataResult\_zi(zl);**

**235 accept DataResult\_z(z: in Integer) do**

**236 z1 := z;**

**237 end DataResult\_z;**

**238 if Tjid > 1 then**

**239 tasksMatrix(Tiid)(Tjid - 1).DataResult\_z(z1);**

**240 end if;**

**241 elsif Tjid > S then**

**242 if Tjid < D then**

**243 accept DataResult\_zi(zi: in Integer) do**

**244 zl := Max(zl, zi);**

**245 end DataResult\_zi;**

**246 end if;**

**247 tasksMatrix(Tiid)(Tjid - 1).DataResult\_zi(zl);**

**248 accept DataResult\_z(z: in Integer) do**

**249 z1 := z;**

**250 end DataResult\_z;**

**251 if Tjid < D then**

**252 tasksMatrix(Tiid)(Tjid + 1).DataResult\_z(z1);**

**253 end if;**

**254 else**

**255 accept DataResult\_zi(zi: in Integer) do**

**256 zl := Max(zl, zi);**

**257 end DataResult\_zi;**

**258 accept DataResult\_zi(zi: in Integer) do**

**259 zl := Max(zl, zi);**

**260 end DataResult\_zi;**

**261 z1 := zl;**

**262 tasksMatrix(Tiid)(Tjid - 1).DataResult\_z(z1);**

**263 tasksMatrix(Tiid)(Tjid + 1).DataResult\_z(z1);**

**264 end if;**

**265 tasksMatrix(Tiid - 1)(Tjid).DataResult\_z(z1);**

**266 tasksMatrix(Tiid + 1)(Tjid).DataResult\_z(z1);**

**267 elsif Tiid < D then**

**268 accept DataResult\_zi(zi: in Integer) do**

**269 zl := Max(zl, zi);**

**270 end DataResult\_zi;**

**271 tasksMatrix(Tiid - 1)(Tjid).DataResult\_zi(zl);**

**272 accept DataResult\_z(z: in Integer) do**

**273 z1 := z;**

**274 end DataResult\_z;**

**275 tasksMatrix(Tiid + 1)(Tjid).DataResult\_z(z1);**

**276 else**

**277 tasksMatrix(Tiid - 1)(Tjid).DataResult\_zi(zl);**

**278 accept DataResult\_z(z: in Integer) do**

**279 z1 := z;**

**280 end DataResult\_z;**

**281 end if;**

**282**

**283 -- Put\_Line("task" & Integer'Image((Tiid - 1) \* D + Tjid) & " has z1...");**

**284**

**285 CalculateA(beginning, ending, z1, alphai, Ti, Bi, MOi, MKi, Ai);**

**286**

**287 if Tiid = 1 then**

**288 accept Result\_A(A: in Vector) do**

**289 for i in 1..N loop**

**290 Ai(i) := Ai(i) + A(i);**

**291 end loop;**

**292 end Result\_A;**

**293 if Tjid < D then**

**294 accept Result\_A(A: in Vector) do**

**295 for i in 1..N loop**

**296 Ai(i) := Ai(i) + A(i);**

**297 end loop;**

**298 end Result\_A;**

**299 end if;**

**300 if Tjid > 1 then**

**301 tasksMatrix(Tiid)(Tjid - 1).Result\_A(Ai);**

**302 end if;**

**303 elsif Tiid < D then**

**304 accept Result\_A(A: in Vector) do**

**305 for i in 1..N loop**

**306 Ai(i) := Ai(i) + A(i);**

**307 end loop;**

**308 end Result\_A;**

**309 tasksMatrix(Tiid - 1)(Tjid).Result\_A(Ai);**

**310 else**

**311 tasksMatrix(Tiid - 1)(Tjid).Result\_A(Ai);**

**312 end if;**

**313**

**314 if Tiid = 1 then**

**315 if Tjid = 1 then**

**316 VectorOutput(Ai);**

**317 end if;**

**318 end if;**

**319**

**320 Put\_Line("task" & Integer'Image((Tiid - 1) \* D + Tjid) & " finished!!!");**

**321**

**322 if Tiid = 1 then**

**323 if Tjid = 1 then**

**324 FinishTime := Clock;**

**325 DiffTime := FinishTime - StartTime;**

**326**

**327 Put("Time = ");**

**328 Put(Integer(DiffTime), 1);**

**329 New\_Line;**

**330 end if;**

**331 end if;**

**332 end Task\_I;**

**333**

**334 ---------------------------------------------------------------**

**335 begin**

**336 StartTime := Clock;**

**337**

**338 for i in 1..D loop**

**339 for j in 1..D loop**

**340 if i > 1 or j > 1 then**

**341 tasksMatrix(i)(j) := new Task\_I(i, j);**

**342 end if;**

**343 end loop;**

**344 end loop;**

**345 tasksMatrix(1)(1) := new Task\_I(1, 1);**

**346 end Lab7;**

**347**