1. Лістинг програми ПРГ2.

1 --kursova robota part 3

2 --MA=MB\*MC(Z\*T)

3 --Zaiats Yulia IO-01

4 --13.04.2013

5

6 with Ada.Integer\_Text\_IO, Ada.Text\_IO; use Ada.Integer\_Text\_IO, Ada.Text\_IO;

7 with Ada.Unchecked\_Deallocation;

8 with Interfaces; use Interfaces;

9 package body Data is

10 --create, input

11 procedure Free\_Vector(A: Vector\_Ptr) is

12 procedure Free\_Vector\_Private is new Ada.Unchecked\_Deallocation(Object => Vector,Name => Vector\_Ptr);

13 tmp: Vector\_Ptr;

14 begin

15 tmp:=A;

16 --Free\_Vector\_Private(tmp);

17 end Free\_Vector;

18 procedure Matrix\_Print(MA:in Matrix\_Ptr) is

19 begin

20 for I in 1..N loop

21 for J in 1..N loop

22 Put(MatGet(MA,i,j));

23 end loop;

24 Put\_Line("");

25 end loop;

26 end Matrix\_Print;

27 function Matrix\_Create(Number: in Integer) return Matrix\_Ptr is

28 Result : Matrix\_Ptr;

29 begin

30 Result := new Matrix(0..N\*N);

31 for i in 1..N loop

32 for j in 1..N loop

33 MatSet(Result,i,j,Number);

34 end loop;

35 end loop;

36 return Result;

37 end Matrix\_Create;

38 function Matrix\_Clone(Source: in Matrix\_Ptr) return Matrix\_Ptr is

39 Result : Matrix\_Ptr;

40 begin

41 Result := new Matrix(0..N\*N);

42 for i in 1..N loop

43 for j in 1..N loop

44 MatSet(Result,i,j,MatGet(Source,i,j));

45 end loop;

46 end loop;

47 return Result;

48 end Matrix\_Clone;

49 function Vector\_Create(Number: in Integer) return Vector\_Ptr is

50 Result : Vector\_Ptr;

51 begin

52 Result := new Vector(1..N);

53 for j in 1..N loop

54 Result(j):=Number;

55 end loop;

56 return Result;

57 end Vector\_Create;

58 function Vector\_Clone(Source: in Vector\_Ptr) return Vector\_Ptr is

59 Result : Vector\_Ptr;

60 begin

61 Result := new Vector(1..N);

62 for j in 1..N loop

63 Result(j):=Source(j);

64 end loop;

65 return Result;

66 end Vector\_Clone;

67 --pack, unpack

68 function Matrix\_Pack(MA:in Matrix\_Ptr; PartMask:in Integer) return Vector\_Ptr is

69 packSize: Integer;

70 C: Vector\_Ptr;

71 from,to,counter:Integer;

72 begin

73 packSize := 0;

74 for I in 0..5 loop

75 if Contain\_Part(Mod\_Type(I),Mod\_Type(PartMask)) then

76 packSize:=packSize+Calc\_Part\_Size(I)\*N;

77 end if;

78 end loop;

79 C := new Vector(0..packSize) ;

80 counter:=1;

81 for I in 0..5 loop

82 if Contain\_Part(Mod\_Type(I),Mod\_Type(PartMask)) then

83 --packSize:=packSize+Calc\_Part\_Size(I);

84 from:= I\*(N/P)+1;

85 to:=from+Calc\_Part\_Size(I)-1;

86 for J in from..to loop

87 for K in 1..N loop

88 C(counter):= MatGet(MA,K,J);

89 counter := counter+1;

90 end loop;

91 end loop;

92 end if;

93 end loop;

94 C(0):=PartMask;

95 return C;

96 end Matrix\_Pack;

97 procedure Matrix\_UnPack(MA:in out Matrix\_Ptr; Data: in Vector\_Ptr; PartMask:in Integer) is

98 from,to,counter:Integer;

99 begin

100 counter:= 1;

101 for I in 0..5 loop

102 if Contain\_Part(Mod\_Type(I),Mod\_Type(PartMask)) and Contain\_Part(Mod\_Type(I),Mod\_Type(Data(0))) then

103 --packSize:=packSize+Calc\_Part\_Size(I);

104 from:= I\*(N/P)+1;

105 to:=from+Calc\_Part\_Size(I)-1;

106 for J in from..to loop

107 for K in 1..N loop

108 MatSet(MA,K,J,Data(counter));

109 --C(counter):= MatGet(MA,K,J);

110 counter := counter+1;

111 end loop;

112 end loop;

113 end if;

114 end loop;

115 end Matrix\_UnPack;

116

117 --get,set

118 procedure MatSet(MA: in out Matrix\_Ptr; i,j,Num: in Integer) is

119 begin

120 MA((j-1)\*N+(i-1)) := Num;

121 end MatSet;

122 function MatGet(MA: in Matrix\_Ptr; i,j: in Integer) return Integer is

123 begin

124 return MA((j-1)\*N+(i-1));

125 end MatGet;

126 --misc

127 function Calc\_Part\_Size(Part: in Integer) return Integer is

128 Part\_Size: Integer;

129 begin

130 Part\_Size := N / P;

131 if (Part+1)=P then

132 Part\_Size := Part\_Size+ N mod P;

133 end if;

134 return Part\_Size;

135 end Calc\_Part\_Size;

136 --misc-packing

137 function Contain\_Part(Part, PartMask: in Mod\_Type) return Boolean is

138 begin

139 if Part = 0 then

140 return (PartMask and 1) = 1;

141 end if;

142 if Part = 1 then

143 return (PartMask and 2) = 2;

144 end if;

145 if Part = 2 then

146 return (PartMask and 4) = 4;

147 end if;

148 if Part = 3 then

149 return (PartMask and 8) = 8;

150 end if;

151 if Part = 4 then

152 return (PartMask and 16) = 16;

153 end if;

154 if Part = 5 then

155 return (PartMask and 32) = 32;

156 end if;

157 return false;

158 end Contain\_Part;

159 --math

160 procedure matrix\_mul\_P(MA,MB:in Matrix\_Ptr; Result:out Matrix\_Ptr; part: in Integer) is

161 startIndex,taskSize:Integer;

162 begin

163 startIndex := part\*(N/P)+1;

164 taskSize := Calc\_Part\_Size(part);

165 for I in startIndex..(startIndex+taskSize-1) loop

166 for J in 1..N loop

167 for K in 1..N loop

168 MatSet(Result,J,I,MatGet(Result,J,I)+MatGet(MA,K,I)\*MatGet(MB,J,K));

169 end loop;

170 end loop;

171 end loop;

172 end matrix\_mul\_P;

173 procedure matrix\_add\_P(MA,MB:in Matrix\_Ptr; Result:out Matrix\_Ptr; part: in Integer) is

174 startIndex,taskSize:Integer;

175 begin

176 startIndex := part\*(N/P)+1;

177 taskSize := Calc\_Part\_Size(part);

178 for I in startIndex..(startIndex+taskSize-1) loop

179 for J in 1..N loop

180 MatSet(Result,J,I,MatGet(MA,J,I)+MatGet(MB,J,I));

181 end loop;

182 end loop;

183 end matrix\_add\_P;

184 procedure matrix\_const\_mul\_P(MA:in out Matrix\_Ptr; Const,Part:in Integer) is

185 startIndex,taskSize:Integer;

186 begin

187 startIndex := part\*(N/P)+1;

188 taskSize := Calc\_Part\_Size(part);

189 for I in startIndex..(startIndex+taskSize-1) loop

190 for J in 1..N loop

191 MatSet(MA,J,I,MatGet(MA,J,I)\*Const);

192 end loop;

193 end loop;

194 end matrix\_const\_mul\_P;

195 function matrix\_max\_P(MA:in Matrix\_Ptr; Part:in Integer) return Integer is

196 startIndex,taskSize,max:Integer;

197 begin

198 max:= -32000;

199 startIndex := part\*(N/P)+1;

200 taskSize := Calc\_Part\_Size(part);

201 for I in startIndex..(startIndex+taskSize-1) loop

202 for J in 1..N loop

203 if MatGet(MA,J,I) > max then

204 max:= MatGet(MA,J,I);

205 end if;

206 end loop;

207 end loop;

208 return max;

209 end matrix\_max\_P;

210 function vector\_mul(A,B: Vector\_Ptr) return Integer is

211 Result:Integer;

212 begin

213 Result:=0;

214 for J in 1..N loop

215 Result:= Result+A(J)\*B(J);

216 end loop;

217 return Result;

218 end vector\_mul;

219 end Data;

1 --kursova robota part 3

2 --MA=MB\*MC(Z\*T)

3 --Zaiats Yulia IO-01

4 --13.04.2013

5

6 with Ada.Unchecked\_Deallocation;

7 with Interfaces; use Interfaces;

8 generic

9 N: Integer;

10 P: Integer;

11 package Data is

12 --types

13 type Vector is array(Integer range <>) of Integer;

14 type Matrix is array(Integer range <>) of Integer;

15 type Vector\_Ptr is access all Vector;

16 type Matrix\_Ptr is access all Matrix;

17 type Mod\_Type is mod 2\*\*32;

18 procedure Free\_Vector(A: Vector\_Ptr);

19 --create, input

20 procedure Matrix\_Print(MA:in Matrix\_Ptr);

21 function Matrix\_Create(Number: in Integer) return Matrix\_Ptr;

22 function Matrix\_Clone(Source: in Matrix\_Ptr) return Matrix\_Ptr;

23 function Vector\_Create(Number: in Integer) return Vector\_Ptr;

24 function Vector\_Clone(Source: in Vector\_Ptr) return Vector\_Ptr;

25 --pack, unpack

26 function Matrix\_Pack(MA:in Matrix\_Ptr; PartMask:in Integer) return Vector\_Ptr;

27 procedure Matrix\_UnPack(MA:in out Matrix\_Ptr; Data: in Vector\_Ptr; PartMask:in Integer);

28 --get,set

29 procedure MatSet(MA: in out Matrix\_Ptr; i,j,Num: in Integer);

30 function MatGet(MA: in Matrix\_Ptr; i,j: in Integer) return Integer;

31 --misc

32 function Calc\_Part\_Size(Part: in Integer) return Integer;

33 --misc-packing

34 function Contain\_Part(Part, PartMask: in Mod\_Type) return Boolean;

35 --math

36 procedure matrix\_mul\_P(MA,MB:in Matrix\_Ptr; Result:out Matrix\_Ptr; part: in Integer);

37 procedure matrix\_add\_P(MA,MB:in Matrix\_Ptr; Result:out Matrix\_Ptr; part: in Integer);

38 procedure matrix\_const\_mul\_P(MA:in out Matrix\_Ptr; Const,Part:in Integer);

39 function matrix\_max\_P(MA:in Matrix\_Ptr; Part:in Integer) return Integer;

40 function vector\_mul(A,B: Vector\_Ptr) return Integer;

41 end Data;

1 --kursova robota part 3

2 --MA=MB\*MC(Z\*T)

3 --Zaiats Yulia IO-01

4 --13.04.2013

5

6 with Ada.Integer\_Text\_IO, Ada.Text\_IO; use Ada.Integer\_Text\_IO, Ada.Text\_IO;

7 with Ada.Command\_line; use Ada.Command\_Line;

8 with Data;

9 procedure Main is

10 N,P:Integer;

11 begin

12 if Argument\_Count /= 1 then

13 Put\_Line("Usage: cw.exe [data\_size]");

14 return;

15 end if;

16 N := Integer'Value(Argument(1));

17 --N:=5;

18 P := 5;

19 declare

20 package DataN is new Data (N,P);

21 use DataN;

22 task T1 is

23 entry passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr);

24 entry passZ(Zx: in Vector\_Ptr);

25 end T1;

26 task T2 is

27 entry PassMB(MBx: in Vector\_Ptr);

28 entry passZ(Zx: in Vector\_Ptr);

29 entry passMA(MAx: in Vector\_Ptr);

30 end T2;

31 task T3 is

32 entry PassMB(MBx: in Vector\_Ptr);

33 entry passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr);

34 entry passMA(MAx: in Vector\_Ptr);

35 end T3;

36 task T4 is

37 entry PassMB(MBx: in Vector\_Ptr);

38 entry passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr);

39 entry passZ(Zx: in Vector\_Ptr);

40 entry passMA(MAx: in Vector\_Ptr);

41 end T4;

42 task T5 is

43 entry PassMB(MBx: in Vector\_Ptr);

44 entry passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr);

45 entry passZ(Zx: in Vector\_Ptr);

46 end T5;

47

48 task body T1 is

49 MA,MB,MC: Matrix\_Ptr;

50 Z,T: Vector\_Ptr;

51 zt: Integer;

52 begin

53 MA:= matrix\_create(0);

54 MB:= Matrix\_Create(1);

55 T2.PassMB(Matrix\_Pack(MB,30));

56 accept passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr) do

57 MC:= Matrix\_Clone(MCx);

58 T:= Vector\_Clone(Tx);

59 end passMC\_T;

60 accept passZ(Zx: in Vector\_Ptr) do

61 Z:= Vector\_Clone(Zx);

62 end passZ;

63 matrix\_mul\_P(MB,MC,MA,0);

64 matrix\_const\_mul\_P(MA,vector\_mul(Z,T),0);

65 T2.passMA(Matrix\_Pack(MA,1));

66 --Matrix\_Print(MA);

67 end T1;

68

69 task body T2 is

70 MA,MB,MC: Matrix\_Ptr;

71 Z,T: Vector\_Ptr;

72 zt: Integer;

73 begin

74 MA:= matrix\_create(0);

75 MB:= Matrix\_Create(0);

76 MC:= Matrix\_Create(1);

77 T:= Vector\_Create(1);

78 accept PassMB(MBx: in Vector\_Ptr) do

79 Matrix\_UnPack(MB,MBx,30);

80 T3.PassMB(Matrix\_Pack(MB,28));

81 end PassMB;

82 T1.passMC\_T(MC,T);

83 T3.passMC\_T(MC,T);

84 accept passZ(Zx: in Vector\_Ptr) do

85 Z:= Vector\_Clone(Zx);

86 T1.passZ(Z);

87 end passZ;

88 matrix\_mul\_P(MB,MC,MA,1);

89 matrix\_const\_mul\_P(MA,vector\_mul(Z,T),1);

90 accept passMA(MAx: in Vector\_Ptr) do

91 Matrix\_UnPack(MA,MAx,1);

92 T3.passMA(Matrix\_Pack(MA,3));

93 end passMA;

94 --Matrix\_Print(MA);

95 end T2;

96

97 task body T3 is

98 MA,MB,MC: Matrix\_Ptr;

99 Z,T: Vector\_Ptr;

100 zt: Integer;

101 begin

102 MA:= matrix\_create(0);

103 MB:= Matrix\_Create(0);

104 Z:= Vector\_Create(1);

105 accept PassMB(MBx: in Vector\_Ptr) do

106 Matrix\_UnPack(MB,MBx,28);

107 T4.PassMB(Matrix\_Pack(MB,24));

108 end PassMB;

109 accept passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr) do

110 MC:= Matrix\_Clone(MCx);

111 T:= Vector\_Clone(Tx);

112 T4.passMC\_T(MC,T);

113 end passMC\_T;

114 T2.passZ(Z);

115 T4.passZ(Z);

116 matrix\_mul\_P(MB,MC,MA,2);

117 matrix\_const\_mul\_P(MA,vector\_mul(Z,T),2);

118 accept passMA(MAx: in Vector\_Ptr) do

119 Matrix\_UnPack(MA,MAx,3);

120 end passMA;

121 accept passMA(MAx: in Vector\_Ptr) do

122 Matrix\_UnPack(MA,MAx,24);

123 end passMA;

124 if N < 10 then

125 Matrix\_Print(MA);

126 end if;

127 end T3;

128

129 task body T4 is

130 MA,MB,MC: Matrix\_Ptr;

131 Z,T: Vector\_Ptr;

132 zt: Integer;

133 begin

134 MA:= matrix\_create(0);

135 MB:= Matrix\_Create(0);

136 accept PassMB(MBx: in Vector\_Ptr) do

137 Matrix\_UnPack(MB,MBx,24);

138 T5.PassMB(Matrix\_Pack(MB,16));

139 end PassMB;

140 accept passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr) do

141 MC:= Matrix\_Clone(MCx);

142 T:= Vector\_Clone(Tx);

143 T5.passMC\_T(MC,T);

144 end passMC\_T;

145 accept passZ(Zx: in Vector\_Ptr) do

146 Z:= Vector\_Clone(Zx);

147 T5.passZ(Z);

148 end passZ;

149 matrix\_mul\_P(MB,MC,MA,3);

150 matrix\_const\_mul\_P(MA,vector\_mul(Z,T),3);

151 accept passMA(MAx: in Vector\_Ptr) do

152 Matrix\_UnPack(MA,MAx,16);

153 T3.passMA(Matrix\_Pack(MA,24));

154 end passMA;

155

156 end T4;

157

158 task body T5 is

159 MA,MB,MC: Matrix\_Ptr;

160 Z,T: Vector\_Ptr;

161 zt: Integer;

162 begin

163 MA:= matrix\_create(0);

164 MB:= Matrix\_Create(0);

165 accept PassMB(MBx: in Vector\_Ptr) do

166 Matrix\_UnPack(MB,MBx,16);

167 end PassMB;

168 accept passMC\_T(MCx: in Matrix\_Ptr; Tx: in Vector\_Ptr) do

169 MC:= Matrix\_Clone(MCx);

170 T:= Vector\_Clone(Tx);

171 end passMC\_T;

172 accept passZ(Zx: in Vector\_Ptr) do

173 Z:= Vector\_Clone(Zx);

174 end passZ;

175 matrix\_mul\_P(MB,MC,MA,4);

176 matrix\_const\_mul\_P(MA,vector\_mul(Z,T),4);

177 T4.passMA(Matrix\_Pack(MA,16));

178 --Matrix\_Print(MA);

179 end T5;

180 begin

181 null;

182 --packed:=Matrix\_Pack(MA,32);

183 --Matrix\_UnPack(MB,packed,63);

184 --Free\_Vector(packed);

185

186 end;

187 null;

188 end Main;