**Національний технічний університет України**

**“Київський політехнічний інститут”**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота № 2**

**з дисципліни**

**«Паралельні та розподілені обчислення»**

**Виконав студент групи ІО-83**

**Сидора Сергій**

**Київ, 2010**

**Ієрархічні бібліотеки. Роздільна компіляція. Виключення**

**Ціль роботи**: вивчення механізму пакетів як засобу об’єктно-орієнтованого програмування в Аді

на прикладі побудови ієрархічних бібліотек, вивчення засобів і методів побудови

складних програм, вивчення засобів і методів забезпечення їх надійності виконання

програми.

**Лістинг програми**

Source file: ..\lab2.adb Thu Sep 16 09:25:10 2010

1 -----------------------------

2 -- Лабораторна робота №2

3 -----------------------------

4 -- 1.26: e = ((A + SORT(B)\*(C + SORT(C))

5 -- 2.18: c = MIN(MA\*MB)

6 -- 3.23: k = MIN(MA\*MB\*E)

7 -----------------------------

8 -- Сидора Сергій, група ІО-83

9 -----------------------------

10 ----------------------------PACK PROGRAM------------------------

11 -- Ця програма використовує пакет StartPack

12 ----------------------------------------------------------------

13

14 with Module,

15 module.F2,

16 module.F2.F3,

17 module.F2.FIO,

18 Ada.Text\_IO, Ada.Integer\_Text\_IO;

19 use Ada.Text\_IO, Ada.Integer\_Text\_IO;

20

21 procedure Lab2 is

22 package Modu is new Module(3);

23 use Modu;

24

25 package Modu\_F2 is new Modu.F2;

26 use Modu\_F2;

27

28 package Modu\_F3 is new Modu\_F2.F3;

29 use Modu\_F3;

30

31 package Modu\_IO is new Modu\_F2.FIO;

32 use Modu\_IO;

33

34 --Main program variables

35 A, B, C, E, D: Vector;

36 MA, MB: Matrix;

37 F, G:integer;

38 begin

39 --Func1

40 Put\_Line("Func1");

41 Put\_Line("Input vector A");

42 VectInput(A);

43 Put\_Line("Input vector B");

44 VectInput(B);

45 Put\_Line("Input vector C");

46 VectInput(C);

47 Func1(A, B, C, D);

48 Put\_Line("Func1 result");

49 VectOutput(d);

50 New\_Line;

51

52 --Func2

53 Put\_Line("Func2");

54 Put\_Line("Input matrix MA");

55 MatrixInput(MA);

56 Put\_Line("Input matrix MB");

57 MatrixInput(MB);

58 Func2(MA, MB, F);

59 Put\_Line("Func2 result");

60 Ada.Integer\_Text\_IO.Put(F);

61

62 --Func3

63 New\_Line;

64 Put\_Line("Func3");

65 Put\_Line("Input vector E");

66 VectInput(E);

67 G := Func3(MA, MB, E);

68 Put\_Line("Func3 result");

69 Ada.Integer\_Text\_IO.Put(G);

70 end Lab2;

Source file: ..\module.ads Thu Sep 16 09:25:10 2010

1 -----------------------------

2 -- Лаб2 1.11: e = ((A + SORT(B)\*(C + SORT(C))

3 -- 2.19: c = MIN(MA\*MB)

4 -- 3.23: k = MIN(MA\*MB\*E)

5 -----------------------------

6 -- група ІО-83

7 -- Сидора Сергій

8 -----------------------------

9

10 generic

11 N: integer;

12

13 package Module is

14

15 --N: integer := 3;

16

17 type vector is private;

18

19 ---c=MAX(A)\*(A\*B);

20 procedure Func1(A, B, C: in Vector; E: out Vector);

21

22 private

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

24

25 ---Vector operations

26 ----Multiplication of vectors a\*b

27 function VectMult(a, b: Vector) return integer;

28

29 ----Processing min of vector a

30 function VectMin(a: Vector) return integer;

31

32 ----Sorting vector a

33 function VectSort(a: Vector) return Vector;

34

35 ----Descending summ of vectors a+b

36 function VectSum(a: Vector; B: Vector) return Vector;

37

38 ----Descending difference of vectors a-b

39 function VectScalPlus(a: Vector; B: Integer) return Vector;

40 end Module;

Source file: ..\module.adb Thu Sep 16 09:25:10 2010

1

2 ------------------------BODY OF MAIN PACKAGE--------------------

3 ----------------------------------------------------------------

4 with Ada.Text\_IO; use Ada.Text\_IO;

5 package body Module is

6 --Functions description

7 ---Vector operations

8 ----Multiplication of vectors a\*b

9 function VectMult(a, b: Vector) return integer is

10 sum: integer;

11 begin

12 sum := 0;

13 for i in 1..N loop

14 sum := sum + a(i) \* b(i);

15 end loop;

16 return sum;

17 end VectMult;

18

19 ----Processing min of vector a

20 function VectMin(a: Vector) return integer is

21 Min: integer;

22 begin

23 Min := A(1);

24 for i in 2..N loop

25 if A(i) < Min then

26 Min := A(i);

27 end if;

28 end loop;

29 return Min;

30 end;

31

32 ----Processing min of vector a

33 function VectSort(a: Vector) return Vector is

34 Tmp: Integer;

35 SorA: Vector;

36 begin

37 for J in 1..N loop

38 sorA(j) := A(j);

39 end loop;

40 for i in 1..N loop

41 for j in 1..(N-1) loop

42 if sorA(j) > sorA(j+1) then

43 Tmp := sorA(j);

44 sorA(j) := sorA(j+1);

45 sorA(j+1) := Tmp;

46 end if;

47 end loop;

48 end loop;

49 return sorA;

50 end;

51

52 ----Descending summ of vectors a+b

53 function VectSum(a: Vector; B: Vector) return Vector is

54 c: Vector;

55 begin

56 for i in 1..N loop

57 c(i) := a(i) + b(I);

58 end loop;

59 return c;

60 end VectSum;

61

62 ----Descending summ of vector a and integer b

63 function VectScalPlus(a: Vector; B: Integer) return Vector is separate;

64

65

66 --Main Functions (Func1, Func2, Func3)

67 ---e = ((A + SORT(B)\*(C + SORT(C))

68 procedure Func1(A, B, C: in Vector; E: out Vector) is

69 begin

70 E := VectScalPlus(A, VectMult(VectSort(B), VectSum(C, VectSort(C))));

71 exception

72 when Constraint\_Error =>

73 Put("Error in Func1.");

74 end Func1;

75

76 end Module;

Source file: ..\module-vectplus.adb Thu Sep 16 09:25:10 2010

1 separate (Module)

2 function VectScalPlus(a: Vector; B: Integer) return Vector is

3 c: Vector;

4 begin

5 for i in 1..N loop

6 c(i) := a(i) + b;

7 end loop;

8 return c;

9 end VectScalPlus;

Source file: ..\module-f2.ads Thu Sep 16 09:25:10 2010

1 ----------------------------------------------------------------

2 ----------------------------Module.F2---------------------------

3 ----------------------------------------------------------------

4 generic

5 package Module.F2 is

6 --Types description--

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

8

9 ---c = MIN(MA\*MB)

10 procedure Func2(MA, MB: in Matrix; C: out integer);

11

12 ----Multiplication of matrix on Vector a\*b

13 function MatVecMult(a: Matrix; b:Vector) return Vector;

14

15 ---Matrix operations

16 ----Multiplication of matrix a\*b

17 function MatrixMult(a, b: Matrix) return Matrix;

18

19 ----Searching minimum in matrix m

20 function MatrixMin(m: Matrix) return integer;

21

22 end Module.F2;

Source file: ..\module-f2.adb Thu Sep 16 09:25:10 2010

1 with Ada.Text\_IO; use Ada.Text\_IO;

2 package body Module.F2 is

3 ---c = MIN(MA\*MB)

4 procedure Func2(MA, MB: in Matrix; C: out integer) is

5 begin

6 C := MatrixMin(MatrixMult(MA, MB));

7 exception

8 when Constraint\_Error =>

9 Put("Error in Func2");

10 end Func2;

11

12 ---Matrix operations

13 ----Multiplication of matrix a\*b

14 function MatrixMult(a, b: Matrix) return Matrix is

15 c: Matrix;

16 sum: Integer;

17 begin

18 for i in 1..N loop

19 for j in 1..N loop

20 sum := 0;

21 for z in 1..n loop

22 sum := sum + a(i)(z) \* b(z)(j);

23 end loop;

24 c(i)(j) := sum;

25 end loop;

26 end loop;

27 return c;

28 end MatrixMult;

29

30 ---Matrix operations

31 ----Multiplication of matrix on Vector a\*b

32 function MatVecMult(a: Matrix; b:Vector) return Vector is

33 c: Vector;

34 sum: Integer;

35 begin

36 for i in 1..N loop

37 sum := 0;

38 for z in 1..n loop

39 sum := sum + a(i)(z) \* b(z);

40 end loop;

41 c(i) := sum;

42 end loop;

43 return c;

44 end MatVecMult;

45

46 ----Searching maximum in matrix m

47 function MatrixMin(m: Matrix) return integer is

48 min: integer;

49 begin

50

51 min := m(1)(1);

52 for i in 1..N loop

53 for j in 1..N loop

54 if m(i)(j) < min then

55 min := m(i)(j);

56 end if;

57 end loop;

58 end loop;

59 return min;

60 end;

61 end Module.F2;

Source file: ..\module-f3.ads Thu Sep 16 09:40:31 2010

1 ----------------------------------------------------------------

2 -----------------------------Module.F3--------------------------

3 ----------------------------------------------------------------

4

5 generic

6 package Module.F2.F3 is

7 ---k = MIN(MA\*MB\*E)

8 function Func3(MA, MB: Matrix; E: Vector) return integer;

9 end Module.F2.F3;

Source file: ..\module-f3.adb Thu Sep 16 09:25:10 2010

1 with Ada.Text\_IO;

2 use Ada.Text\_IO;

3 package body Module.F2.F3 is

4 ---k = MIN(MA\*MB\*E)

5 function Func3(MA, MB: Matrix; E: Vector) return integer is

6 begin

7 return VectMin(MatVecMult(MatrixMult(MA, MB), E));

8 exception

9 when Constraint\_Error =>

10 Put("Error in Func3");

11 return 0;

12 end Func3;

13 end Module.F2.F3;

Source file: ..\module-fio.ads Thu Sep 16 09:40:31 2010

1 ----------------------------------------------------------------

2 -- Paralel and distributed computing --

3 -- Laboratory work #2 --

4 -- IO-83 Sydora Sergiy --

5 ----------------------------------------------------------------

6 -- Variant: 1.26, 2.18, 3.23 --

7 ----------------------------------------------------------------

8 ---------------------------Module.FIO---------------------------

9 ----------------------------------------------------------------

10

11 generic

12 package Module.F2.FIO is

13

14 --I/O operations

15 ---Output for Vector type

16 procedure VectOutput(a: Vector);

17

18 ---Matrix output

19 procedure MatrixOutput(a: Matrix);

20

21 ---Vector input

22 procedure VectInput(a: in out Vector);

23

24 ---Matrix input

25 procedure MatrixInput(m: in out Matrix);

26 end Module.F2.FIO;

Source file: ..\module-fio.adb Thu Sep 16 09:40:31 2010

1 with Ada.Text\_IO; use Ada.Text\_IO;

2 with Ada.Integer\_Text\_IO; use Ada.Integer\_Text\_IO;

3 with Ada.IO\_Exceptions; use Ada.IO\_Exceptions;

4

5 package body Module.F2.FIO is

6

7 --I/O operations

8 ---Output for Vector type

9 procedure VectOutput(a: Vector) is

10 begin

11 for i in 1..n loop

12 Put(a(i), 5);

13 end loop;

14 New\_line;

15 end VectOutput;

16

17 ---Matrix output

18 procedure MatrixOutput(a: Matrix) is

19 begin

20 for i in 1..n loop

21 for j in 1..n loop

22 Put(a(i)(j), 4);

23 end loop;

24 New\_Line;

25 end loop;

26 end MatrixOutput;

27

28 ---Vector input

29 procedure VectInput(a: in out Vector) is

30 begin

31 for i in 1..N loop

32 loop

33 begin

34 Get(a(i));

35 exit;

36 exception

37 when Ada.IO\_Exceptions.Data\_Error =>

38 Put("Input error. Please, try again:");

39 Skip\_Line;

40 end;

41 end loop;

42 end loop;

43

44 end VectInput;

45

46 ---Matrix input

47 procedure MatrixInput(m: in out Matrix) is

48 buf: integer;

49 begin

50 for i in 1..N loop

51 for j in 1..N loop

52 Get(M(i)(j));

53 end loop;

54 New\_Line;

55 end loop;

56 end;

57

58 end Module.F2.FIO;