Friedrich-Schiller-Universität Jena Fakultät für Mathematik und Informatik Institut für Informatik Lehrstuhl für Programmiersprachen und Compiler	Höhere Programmierung SS 2001	Aufgabenblatt 13 Ausgabe: 25.06.2001 Abgabetermin: 02.07.2001 16:00 Uhr

Aufgabe 1: Operatoren

Formulieren Sie für die unteren Beispiele die Operatorspezifikationen.

a) Matrixmultiplikation (n × n Matrizen)

b) Vektorvergleich (n-dimensionale Vektoren)

c) Mit dem Additionszeichen soll das Einfügen eines Listenelements vom Typ `ListenElementType` in eine vorhandene Liste, gegeben durch den Typ `ListenKopfType` (wie z.B. auf Praktikumsblatt 6), realisiert werden.

d) Erg := L + R; Erg und L sind Arrays, R ist vom Elementtyp des Arrays. Das Ergebnisarray Erg enthält alle Elemente von L und zusätzlich das Element R. Beachten Sie, dass dabei ein neues Array entsteht, das nicht mit dem als Parameter übergebenen Array typkompatibel ist. Geben Sie die Definitionen der benötigten Typen für das Array mit an.

3 Punkte

Aufgabe 2: Sichtbarkeit und Lebensdauer von Variablen

Analysieren Sie die folgenden Codefragmente. Kommentieren Sie den Code und geben Sie an, auf welche Definition bei der Benutzung von Variablen zugegriffen wird. Stellen Sie auf einer Zeitachse die Lebensdauer der einzelnen Variablen und formalen Parameter dar. Was gibt das Programm aus?

a)

```
WITH Ada.Integer_Text_Io, Ada.Text_Io;

PROCEDURE Main IS
  X: Integer := 1;

  FUNCTION Sub1(A:IN Integer) RETURN Integer IS
    X: Integer := 0;

  BEGIN
    X := A;
    RETURN (2*X);
  END Sub1;

  PROCEDURE Sub2(X: IN OUT Integer) IS

    FUNCTION Sub3(X: IN Integer) RETURN integer IS
    BEGIN
      X:=5*X;
      RETURN (X/2);
    END Sub3;

  BEGIN
    X:=2*Sub1(Sub3(X));
  END Sub2;

BEGIN
  Sub2;
  X:=Sub1(X);
  Ada.Text_Io.Put_Line(X);
END Main;
```

b)

```
WITH Ada.Integer_Text_Io, Ada.Text_Io;

PROCEDURE Main IS

  X: Integer := 1;
  Y: Integer := 2;
  Z: Integer := 3;

  SubAccessType IS ACCESS FUNCTION(X: IN Integer;
                                   Y: IN Integer) RETURN Integer;

  SubAccess: SubAccessType;

  FUNCTION Sub1(X: IN Integer; Y: IN Integer) RETURN Integer IS

  BEGIN
    RETURN (X*Z+Y);
  END Sub1;

  FUNCTION Sub2 (A: IN Integer; B: IN integer) RETURN Integer IS

  BEGIN
    Y := 7;
    RETURN (X*Y+A*B);
  END Sub2;

BEGIN
  SubAccess := Sub1'Access;
  X := SubAccess(X, Z);

  Ada.Integer_Text_Io.Put(X);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Y);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Z);
  Ada.Text_Io.New_Line;

  SubAccess := Sub2'Access;
  Z := SubAccess(Y, 5);

  Ada.Integer_Text_Io.Put(X);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Y);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Z);
  Ada.Text_Io.New_Line;
END Main;
```

c)

```
WITH Ada.integer_Text_io, Ada.Text_Io, Ada.Io_Exceptions;

PROCEDURE Main IS

  X: Integer := 3;

  SubAccessType IS ACCCESS FUNCTION(X: IN Integer) RETURN Integer;
  SubAccess: SubAccessType;

  FUNCTION Sub2(X: IN integer) Return integer;

  FUNCTION Sub1(A: IN Integer; Prg1: IN SubAccessType) Return Integer;

    y : integer :=0;

  BEGIN
    Y := Prg1(A);
    SubAccess := Sub2'Access;
    RETURN(Y);
  END Sub1;

  FUNCTION Sub2(X: IN Integer) RETURN Integer;

  BEGIN
    IF X>0 THEN
      RETURN (X * Prg1(X-1));
    ELSE
      RETURN 1;
    END IF;
  END Sub2;

  FUNCTION Sub3(X: Integer) RETURN Integer IS

  BEGIN
    IF X>0 THEN
      X := X / 2;
      RETURN ( 1 + SubAccess(X));
    ELSE
      RETURN 0;
    END IF;
  END Sub3;

BEGIN

  SubAccess := Sub3'Access;
  X := Sub2(X, SubAccess);
  Ada.Integer_Text_Io.Put(X);
  Ada.Text_Io.New_Line;

  X := SubAccess(X);
  Ada.Integer_Text_Io.Put(X);
  Ada.Text_Io.New_Line;

END Main;
```

Anmerkung: Die Programme in dieser Aufgabe wurden nur für Übungszwecke erstellt. Die dargestellten Algorithmen selbst haben keine praktische Relevanz.

8 Punkte

Aufgabe 3: Operatordefinition

Entwickeln Sie ein Programm, das von zwei gegebenen 3-dimensionalen Vektoren das Skalar- und Vektorprodukt berechnet. Die Produkte sollen als Operatoren definiert werden.

6 Punkte

Aufgabe 4: Überladen von Operatoren (Zusatzaufgabe für Wirtschaftsinformatiker)

Gegeben seien folgende Spezifikationen für den Operator "+". Dabei sind T1 und T2 unterschiedliche Typen.

```
FUNCTION "+" (left, right : T1) RETURN T1;         --Definition Plus1
FUNCTION "+" (left, right : T1) RETURN T2;         --Definition Plus2
FUNCTION "+" (left : T2; right: T1) RETURN T1;     --Definition Plus3
FUNCTION "+" (left : T1, right: T2) RETURN T1;     --Definition Plus4
```

Geben Sie die verwendeten Operatoren für die folgenden Ausdrücke an. Dabei sind a, b, c vom Typ T1 und x, y, z vom Typ T2.

a) `b := a + x;`

b) `c := a + x + y;`

c) `a := (a + x) + (y + b);`

d) `a := a + b + c;`

e) `x := x + y + z;`

3 Punkte