Exercise 1.1: Type equivalence

We have the following declarations:

type Array1 = array[1..10] of BOOLEAN;
    Array2 = array[1..10] of BOOLEAN;
var A    : Array1;
    B, C : array[0..9] of BOOLEAN;
    D, E : Array2;
    F    : Array2;
    X    : array[1..10] of Array1;
    Z, Y : array[2..11] of Array2;

• Which variables are structurally equivalent?

• Which variables are name equivalent? (Some cases are treated differently in different languages. Which cases are that and what possibilities exist?)

• Name advantages and disadvantages of the two concepts of type equivalence!

Exercise 1.2: Type conversions

We have the following code snippet:

a, b: real;
c: int;
c := a + b;

Assume that the variables have the values a = 3.6 and b = 3.8. Which value has c after the assignment in a language with implicit type conversion (from real to int) if

• the conversion is done by mathematical rounding and is done as early as possible,

• the conversion is done by mathematical rounding and is done as late as possible,

• the conversion is done by truncating decimal places and is done as early as possible,

• the conversion is done by truncating decimal places and is done as late as possible?
Exercise 1.3: Record and array layout

procedure Test (A: Integer; B: Integer) is
  I: Integer;
  type Rec is record
    Flags: array (1..3) of Boolean;
    Arr: array (1..A) of Integer;
    Arr2: array (1..B) of Integer;
    Status: Boolean;
    Count: Integer;
  end record;
  RObj: Rec;
  K: Integer;
begin ...
end Test;

• At what time is the size of the variable RObj determined?

• Describe the layout of the record RObj. You are allowed to use different representations for each of the arrays inside RObj.

• Describe the layout of the activation record (on the stack) in detail.

Exercise 1.4: Multi-dimensional array

The following code operates on 2-dimensional arrays of arbitrary size:

type Matrix is array (Integer range <>, Integer range <> ) of Real;
type Matrix_Access is access all Matrix;

procedure Main is
  M1 : Matrix (1 .. 10, 5 .. 17);
  M2 : Matrix_Access := new Matrix (3 .. 6, 1 .. 2);
begin
  -- M2 := M1'Access; -- (1) let M2 point to M1
  ... = M2.all (5, 1);
end Main;

• Where are the contents of M1 stored in procedure Main? On the stack or on the heap? Where are the contents of M2 stored?

• For the access to M2 the array dope is needed. Why? Where could it be stored?

• What would happen if the line marked with (1) would be active and not commented out? And why?

• Show the contents of the dope. Show how the elements of M2.all are arranged in memory, assuming row-major array layout. What would be the difference in column-major?
Exercise 1.5: Recursive procedure

procedure Recursive (Condition : Boolean) is
    X : Integer;
begin
    if Condition then
        X := 3;  -- Assignment 1
        Recursive (False);  -- Call
        ... := X;  -- Assignment 2
    else
        X := 7;  -- Assignment 3
    end if;
end Recursive;

-- main program:
begin
    Recursive (False);
    Recursive (True);
end;

• How many times is the procedure Recursive entered for one execution of the main program? In what order are the assignment statements executed during one execution of the whole program?
• What value of X is used when assignment 2 is executed?
• List all the changes of the stack while the program executes. Include changes to variables and changes of the stack pointer.
• What would happen if the call of the procedure Recursive to itself would pass the value True instead of False?
• Show the call stack when assignment 2 is executed for the first time.