From: Ed Greengrass

To: WG11

Subject: Response to Turba's Comments on CLIDT WD 5 (WG11/N262), part 3 of 3

Date: Wed, 17 Jul 91 13:45:27 -0400

****

(8) Turba: This [Complex] datatype assumes an implementation strategy based on Cartesian coordinates, which is not the only means by which complex numbers can be implemented. This is unnecessary and conflicts with the definition of the complex datatype in languages such as Extended Pascal.

****

Answer:

Ed Barkmeyer tells me he has already reponded directly to Turba on this matter so my own comment may be superfluous. However, Turba's objection raises a difficulty that arises so often, a source of confusion to which we are all subject, that another comment may not be out of place.

CLIDT WD 5 actually defines Complex in two ways: first, as "the solution space of all polynomial equations having real coefficients" and second, in terms of a real-part and a complex-part, each interpreted as a Real value. This latter definition involves a characterizing operation Promote which (if I understand correctly) promotes a real number into the corresponding complex number. The resulting 2nd definition is:

Add(Promote (real-part), Square Root(Negate(Multiply
(Promote(imaginary-part), promote(imaginary part)))))

where Add and SquareRoot are also characterizing operations of Complex.

It is this definition that evidently concerns Turba since it involves the concepts of real and imaginary parts and hence is "based on Cartesian coordinates." But does such a definition imply any specific implementation or representation? No. The confusion arises because a term can be meaningful at both the conceptual (abstract) level AND the implementation level.

The crucial question with regard to a datatype T is whether the term in question is meaningful to USERS of T (as opposed to implementors of T). Are "real-part" and "imaginary-Part" meaningful terms to users of type Complex? Obviously. A user, e.g., an application, can ask for the "real-part" of a complex number V. The request is meaningful whether V is represented in cartesian coordinates or polar coordinates. Of course, the implementation of the operation "get real-part" is quite different (and in this case simpler) if based on the cartesian representation than if based on the polar. But the user doesn't have to know the representation of V, and hence doesn't have to know the implementation of "get real-part".

Similar considerations apply to a type such as Date-and-Time. It is meaningful for a user to ask for the month, day, and year of a Date-and Time value, D1. The user doesn't know or care whether D1 is represented as a Record with month, day, and, year fields or as an ordinal date (an Integer).

To take an extreme case, consider the term "CPU word length". With regard to type Complex or indeed ANY CLIDT type, this is plainly an implementation level detail. But one might define a set of application specific "computer architecture" types in terms of CLIDT. In such a set, "CPU word length" might be a legitimate "conceptual" datatype or a conceptual attribute of some other datatype.