COURSE PR-02: Data Representation

LENGTH: 2 half-day sessions

PREREQUISITES: Programming experience using a procedural language that supports a wide range of data types and attributes.

DESIGNED FOR:

1. Data administrators, data-base designers, or systems analysts who prepare data definitions, perform data modeling, and design logical data bases.
2. Systems analysts who design reports, source documents, displays, or other application system user interfaces.
3. Experienced programmers whose previous experience hasn't stressed the relationship of data representation to program structure and logic.

DESCRIPTION:

As background we begin by clarifying the distinction among (a) elementary data items, (b) composite data items, and (c) container data items, citing examples of each. For elementary items we examine the four basic *types*: discrete, numeric, logical, and text. We present criteria for choosing appropriate representations for data items of each of those types, emphasizing the distinction between *external* and *internal* representations. We then introduce the concept of generic data *class* and suggest representations for such common classes as money, dates, names of people, names of organizations, street addresses, distance, mass, temperature, and others of particular interest to the group. We stress the need for standards for such representations, and review critically alternative proposed standards for the various generic data classes, with emphasis on laying a foundation for developing a library of reusable program modules or object-oriented classes.

1999 Update: Millennium bug and Mars Crash

Two recent newsworthy events are directly traceable to inappropriate choice of data-representation. The so-called "Y2K crisis" and the September, 1999, crash of NASA's Mars probe. For organizations that standardized their date representations a decade or two ago the cost of Y2K compliance for their internally developed applications was zero. And we just learned that NASA lost a costly unmanned spaceship because of an easily avoided misunderstanding over the unit of measure for rocket thrust!

Many hard-to-fix bugs in business applications also stem from inappropriate data representation or, worse, from multiple incompatible representations of the same type of data. The cost of choosing appropriate data representations is trivial; the payback often enormous.

Object-oriented programming languages give us a way of localizing the impact of the choice of internal data representation, but we still have to make a good choice

In the second session we briefly present different kinds of composite data (records, data-flows, structures) that occur in typical business applications, and discuss techniques for their efficient and flexible representation. We go on to look at container structures that programmers and data-base designers need, such as tables, files, arrays, trees, and lists, and examine the tradeoffs among ease of manipulation, economy of storage, and efficiency of access. We conclude by discussing the concept of *encapsulation* of data representations and its relationship to object-oriented programming.

Information Disciplines, Inc., Chicago
August, 1990

OBJECTIVES:

Upon completing this course the participant will be able to:

1. Classify data according to the basic data types and important subtypes.
2. Simplify the structure and organization of new programs and enhance their reliability by choosing efficient and appropriate internal data representations.
3. Cater to human users by choosing appropriate external data representations for reports, source documents, and other application system outputs and inputs.
4. Contribute to organization or project standards for representation of many classes of data.
5. Identify the need for program modules for validating, converting, or manipulating data items that conform to those standards.
6. Given precise requirements, design work files, master files, and non-shared data bases.
7. Participate effectively in the design and enhancement of integrated data bases.