

Language Paradigms

A. Universal

1. Procedural ‘third generation.’ E.g., C, Pascal, Fortran

Advantages:

Disadvantages:

2. Functional. E.g., Lisp (Emacs, AutoCAD), APL, parts of C++ (, or ? : or void)

Advantages:

Disadvantages:

3. Constraint

a. Relational

Logic programming. E.g. Prolog

b. Special-purpose

Electrical circuits

Graphing/CAD

 snap to grid

 tangent to circle

Advantages:

Disadvantages:

4. Object-oriented Trades having many procedures applied to few data items for having few procedures applied to many data items

E.g., Smalltalk 80, C++, Java

Advantages:

Disadvantages:

B. Special-case (sampler only)

1. Database

SQL ("fourth generation," a constraint language with persistence)

2. Mixed

Access VBA: Database & Object Oriented

Q'Nial: APL (itself a functional language) on nested data types

3. Page-description languages

Postscript

4. Structured data languages

Hypercard (for Apple Macintosh)

UIL [user interface language, now incorporated into User Modeling Language, UML]

SGML [Standard Generalized Markup Language], XML, HTML

5. Operating system languages, 'shells'

DOS .BAT

Unix/Linux csh, bash, korn (support for procedure parallelism)

VMS .COM (support for procedure parallelism)

A new computer language is being invented every day. Here! I'll invent one for you right on the spot. I'll call it My-Language. I'm using Basic as the language in which to write my programming language, just because when it comes to string handling, Basic is clean. I've written an interpreter rather than a compiler. There are only four commands in my language. The syntax of the four commands is as follows:

ADD, <number1>, <number2> SUBTRACT, <number1>, <number2>
MULTIPLY, <number1>, <number2> DIVIDE, <number1>, <number2>

```
' MY-LANGUAGE interpreter, QuickBasic version
' Gene B. Chase, August 25, 2001

forever = (1 = 1)

WHILE forever
  INPUT operation$, arg1, arg2
  IF      operation$ = "ADD" THEN
                                PRINT arg1 + arg2
  ELSEIF operation$ = "SUBTRACT" THEN
                                PRINT arg1 - arg2
  ELSEIF operation$ = "MULTIPLY" THEN
                                PRINT arg1 * arg2
  ELSEIF operation$ = "DIVIDE" THEN
                                PRINT arg1 / arg2
  ELSE PRINT "Error in function on previous line. Reenter."
  END IF
WEND
END
```

Objects

Definition of a (pure) object-oriented language

A language in which all data structures know everything about themselves—including the procedures used on them and their relationship to other structures.

The Copernican Revolution of Computer Science

A data-centered view rather than a procedure-centered view

The Industrial Revolution of Computer Science

Reusability via “interchangeable parts”

The “Software IC™” approach (Brad Cox's term)

How close are things that we already know to being objects fully?

Why isn't Java like Smalltalk?

In Smalltalk, everything including integers is an object; in Java, there are some primitive types like int, boolean, float, and double that are not objects.

Are files objects in most operating systems?

Can you copy all files whose protection is read-only? No! Why? Because the information about a file's protection is not available as information about the file unless the operating system chooses to give it to you (in the VMS operating system on the VAX computer, so-called lexical functions can get this information), or unless you use a very low-level language (in assembly language you could find the location on the disk where the file's attributes are stored and decode them).

Are abstract data types (ADTs) objects?

In Data Structures and Algorithms we learned that an ADT is a data type and all the allowable operations on it. Objects are one way to implement an ADT in a programming language, but there are other ways. All other ways, though, require the programmer to supply a level of discipline in the use of the ADTs that can be enforced automatically in a pure object language.

Are units (sometimes called “modules”—separately compiled files containing both data and code) a kind of object?

Here data and code are stored together, but variables are only **global** (only one copy exists everywhere more permanently) or **local** (only one copy exists temporarily). A pure object-oriented system has **state** variables (one variable for each instance of the object, local to the object, but with a lifetime equal to the lifetime of the object, or longer if we use persistent objects).

Do databases contain objects?

Databases also combine data with procedures that check the validity of the data, and their variables—being disk-based rather than memory-based—have persistence. But adding other objects to a database is not easy, as it would be in a pure object-oriented system. The range of data types in a database is far too limited. Ask questions like this: Can a picture or a list be the value of a variable in your database? A new breed of object-oriented databases is changing that. As early as September, 1993, *Communications of the ACM* devoted a theme issue to this subject.