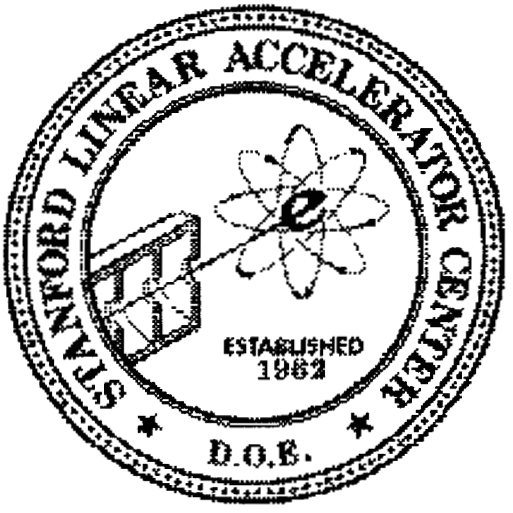
# USING REXX TO TEACH PROGRAMMING

BEBO WHITE
SLAC

# Using REXX

# to Teach

# Programming




**Bebo White**
**SLAC**
**2nd REXX Symposium**
**Asilomar Conference Center**
**May 8, 1991**

# Why? (IMHO)

- The programming education community needs:
  - a flexible, interactive, powerful language with emphasis on basic programming concepts
  - to separate programming instruction from "the language wars"

- Programming students need :
  - a meaningful first exposure to the elements and art of programming
  - the positive feedback of being able to write code quickly and "watch it work"
  - not to be intimidated or bored by concepts couched in language specifics

# Typical(?) Goals of a Beginning Programming Course

- To teach that programming can be fun and something to take pride in
- To provide experience with systematic design processes
- To provide early experience with program documentation development
- To provide a knowledge of general principles applicable to many programming languages
- To provide experience with software tools
- To teach attention to style

# Typical Beginning Programming Course Curriculum

- Smooth transition from everyday planning experiences to formal design of programs
- Early use of sufficiently complex problems where algorithmic solutions are not immediately obvious - motivates PDM
- Early treatment of issues arising from "large" problems
- Logical introduction to control structures ("structured design")
- "Gentle" introduction to data types, variables and parameters
- Discussion of data structures and data abstraction

# What REXX Has To Offer

- An "algorithmic" language "close" to pseudocode
- Allows "self-documenting" code
- Macro capability allows "getting something done fast"
- Modern control structures which are customizable; exceptions allowed in well-defined cases
- Generalized data types; undefined variables
- Generalized/simplistic data structures; user-defined data structures

# What REXX Has To Offer (cont.)

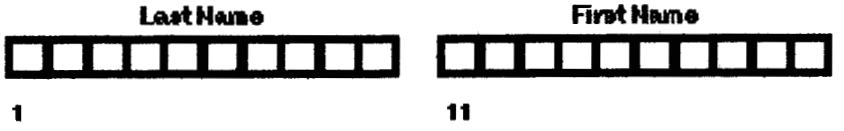
- Generalized/simplistic data abstractions
- Generalized I/O
- Function libraries
- Trace - for debugging and a learning aid
- Sophisticated features/capabilities "under the covers" (e.g., hex manipulation, recursion)

# Teaching Data Structures

- Data Structure concepts should go from the most generalized (i.e., a familar analog) to the most specific
- Data Structures should be perceived as a viable entity which can be easily taken apart and manipulated
- "Algorithms + Data Structures = Programs"

# Records

- To the "layperson" records look like lines in an application form:




- To the "computer_person" records are "values of various datatypes of differing lengths appended to one another in a specific order"

# REXX Knows Both Records

## "Layperson" Records as:

**parse var NameInfo.1 LastName 11 FirstName 21**

## "Computer_person" Records as:

**NameInfo.1.LastName = .....**
**NameInfo.1.FirstName = .....**

# Data Abstraction

- **There is a Share requirement to:**

  **Allow an expression/variable to be the target of an assignment statement**

- **To the "layperson" this is a _____?**

# We Need To...

- Continue to develop the REXX language following its philosophical tradition
- Develop major applications using REXX
- Promote REXX as a mainstream programming language
- Insure the availability of REXX on as many computing platforms as possible