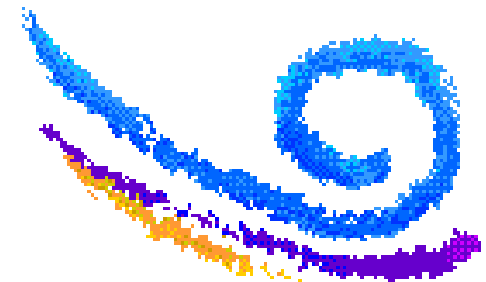


Defining Data in COBOL Programs

COBOL Data Sections

Declaring Data

- ▶ All files, records, and data items you reference in a COBOL program must be declared in the Data Division of the program
 - This is so the compiler knows where to put data items or to find these data items in memory at run time



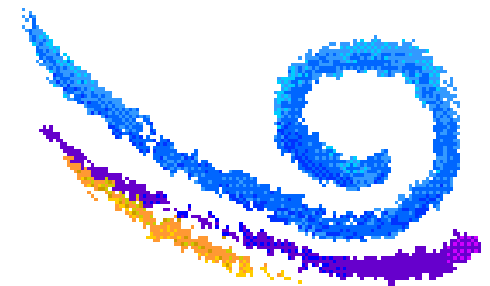
The Data Division

- ▶ The Data Division has four sections - only include the ones you need, but the ones you include must be in this relative order:
 - File section - describes each file with an FD (File Definition) entry and a record layout
 - Working-storage section - describes data items included in the object code for the current program
 - Local-storage section - describes data items created and initialized on program entry at run time
 - Linkage section - describes data items residing outside the current program



File Section

- ▶ Each file named in an *ASSIGN* statement must have an FD entry followed by a record description.
 - The record description may be a detailed record layout or simply an OI-level with a PIC specifying how big the record size is
 - Details are omitted here



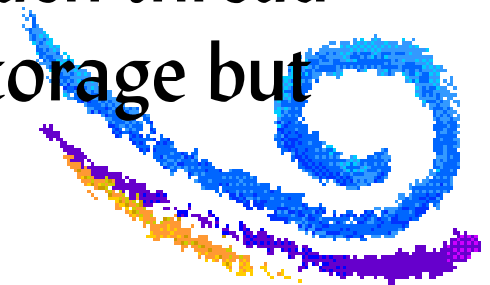
Working-storage Section

- ▶ Here is where most data items are defined
 - That is, named and described
- ▶ The compiler generates room in the object code to hold these items
 - For items with a VALUE clause, the room is initialized at compile time
- ▶ At run-time, the working-storage for a subroutine is found in its last-used state every time the routine is called



Local-storage Section

- ▶ This section is used to hold variables that are created and deleted dynamically
 - Every time the program is entered, the variables are created [and initialized], and on exit the variables are deleted
- ▶ Used for subroutine values that need to have their original values each time the routine is called
- ▶ Also used for multi-threading applications: each thread running the same program shares working-storage but has its own local-storage



Linkage Section

- ▶ Here you declare data items that are either ...
 - Passed to you when you are called as a subroutine
 - Or created by your program in storage dynamically obtained from outside your program using one of these approaches:
 - Direct call to LE routines (CEEGTSTG)
 - Call to Assembler, PL/I, or C routines to obtain storage on your behalf





6790 East Cedar Avenue, Suite 201
Denver, Colorado 80224
USA

<http://www.trainersfriend.com>

800.993.8716

303.393.8716

Sales: kitty@trainersfriend.com

Technical: steve@trainersfriend.com