RELATIVE INPUT/OUTPUT FILES

. **COMMANDS - ABOUT THE SAME AS INDEXED COMMANDS**
. **LOGIC IS QUITE DIFFERENT**

DEFINITION OF A RELATIVE FILE:

. **ONE FILE ONLY - ONLY A DATA COMPONENT**
. **ACCESS: <u>RANDOMLY</u> BY A RELATIVE RECORD NUMBER.**
(actually use a field within the record (symbolic key) to derive the Relative Key, which is Anormally@ NOT part of the record.)

. **A RELATIVE FILE IS LIKE AN AEXTERNAL ARRAY@**
IT MAY HAVE UNFILLED ELEMENTS.

Relative Record #1
Relative Record #2
Relative Record #3
.....
Relative Record #n

. **KEY (SYMBOLIC) AND RELATIVE RECORD NUMBER (Relative Key) <u>CAN</u> BE THE SAME.**
.. **MAY BE FIELD W/I RECORD WITH VALUES 1,2,...,N.**
.. **SYMBOLIC KEY IS THEN RELATIVE KEY.**

CONSIDER:

A RECORD OF DATA GATHERED FROM A LAB EXPERIMENT.
LAB EXPERIMENT #1 RESULTED IN ..... DATA
LAB EXPERIMENT #2 RESULTED IN .... DATA

001	**LAB DATA FIELDS**

002	**LAB DATA FIELDS**

ETC.

. CONSECUTIVE RELEATIVE RECORD NUMBERS CAN OCCUR QUITE NATURALLY IN PHYSICAL PHENOMENON

. NORMALLY, MUST MODIFY A SYMBOLIC KEY AND CONVERT INTO A RELATIVE RECORD NUMBER PRIOR TO ANY I/O.

. DISCUSS: PRACTICALITY OF USING SSAN AS RELATIVE KEY?

. DISCUSS: INPUT SYMBOLIC KEY WITH VALUES 100, 200, ETC.?

. DISCUSS SYMBOLIC KEY W/RANGE 40,000 TO 50,000?
DIVIDE EACH KEY BY 40,000 AND TAKE REMAINDER
.. USE REMAINDER AS RELATIVE KEY.

.. ALLOCATE SPACE FOR 1000 RECORDS W/RELATIVE KEYS:
1 -> 1000.

. THIS LEADS US TO AN IMPORTANT CONCEPT:

ENTER: AN AALGORITHM@ FOR CONVERTING A SYMBOLIC KEY TO A RELATIVE KEY (RELATIVE RECORD NUMBER)

. THE PROCESS OF AKEY TO ADDRESS@ TRANSFORMATION IS REFERRED TO AS AHASHING.@
. OBJECTIVE OF HASHING: AHASH TO ARRIVE AT A RELATIVE

RECORD NUMBER - PLACE WHERE RECORD WILL BE STORED IN THE FILE.

. AHASHING@ REFERS TO APPLYING AN ALGORITHM TO A SYMBOLIC KEY TO ARRIVE AT A RELATIVE KEY.

N.B. RELATIVE RECORD NUMBER IS
. ANORMALLY@ NOT FOUND WITHIN THE RECORD,
. IS NOT STORED IN THE RECORD AFTER HASHING, AND
. MUST BE COMPUTED FOR EVERY INPUT OR OUTPUT ACTION ATTEMPTING TO ACCESS THE TARGET RECORD.

RELATIVE FILE PROCESSING IS VERY FAST - MUCH FASTER THAN INDEXED SEQUENTIAL.....WHY??

NOT AS FAST AS A TABLE RESIDENT W/I PROGRAM, BUT SIMILAR.

THERE ARE MANY ALGORITHMS FOR HASHING TO AN ADDRESS. (HASHING ALGORITHMS)

LET=S CONSIDER A PRACTICAL EXAMPLE:

CONSIDER A KEY OF RANGE 40000 TO 50000.

WE WILL AHASH@ TO AN AADDRESS@ USING THE DIVISION/REMAINDER METHOD.

WE WILL DIVIDE BY 40,000 AND ADD 1 TO PRODUCE KEYS WITH A RANGE OF 1 TO 10,000

. THIS => EVERY RECORD HAS ITS OWN, UNIQUE SLOT IN FILE.

. NOT TOO PRACTICAL, IF FILE CONTAINS 1000 RECORDS.

. EXTREMELY WASTEFUL TO ALLOCATE SPACE FOR 10,000