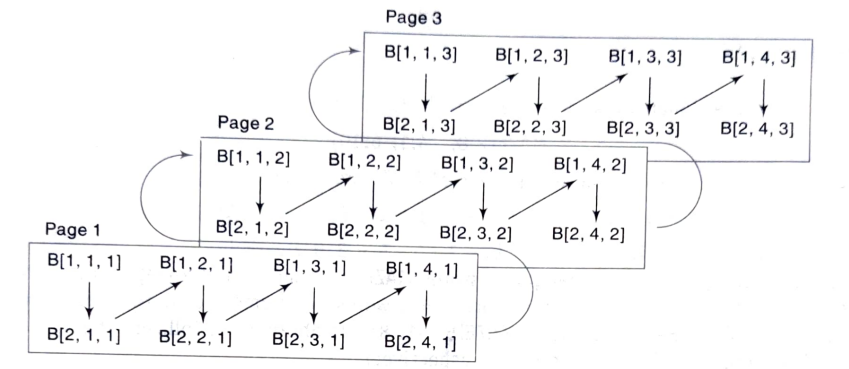
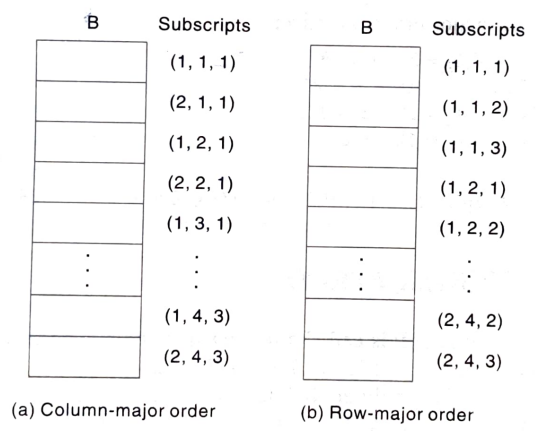
**Multi-dimensional Arrays**

**Eg: for element B [2,4,3] there are three subscripts indicating row, column and page.**

****

**Representation of 2X4X3 array. Array is stored in 3 layers (called pages). Each page contains 2X4 elements.**

****

**Storing of array B in memory**

**Effective Index**

**The effective index Ei is the number of indices preceding Ki (any subscript value) in the index set, and E, can be calculated from Ei = K i - lower bound**

**Then the address LOC(C[K1, K2 ,..., Kn] of an arbitrary element of C can be obtained from the column-major formula**

**Base(C) + w[(((... (En L n-1 + E n-1)L n-2) +…. + E3)L 2 + E 2)L 1 + E 1]**

**or from the row-major formula**

**Base(C) + w[((... (E1 L2 + E2)L3 + E3) L4 +… + En-1)L n + E n]**

**Example**

**Suppose a three-dimensional array MAZE is declared using MAZE(2:8, -4:1, 6:10)**

**Then the lengths of the three dimensions of MAZE are, respectively,**

**L1=8-2+1=7, L2=1-(-4)+1=6, L3=10-6+1=5,**

**Accordingly, MAZE contains L1 . L2 . L3 =7. 6. 5=210 elements.**

**Suppose the programming language stores MAZE in memory in row-major order, and**

**Suppose Base(MAZE) = 200 and there are w = 4 words per memory cell.**

**The address of an element of MAZE-for example, MAZE[5, -1, 8] is obtained as follows.**

**The effective indices of the subscripts are, respectively,**

**E1 = 5 - 2 = 3, E2 = -1 - (-4) = 3, E3 =8 – 6 = 2**

**Using equation for row-major order, we have:**

**E1L2 = 3 . 6 = 18**

**E1L2 + E2 = 18 + 3 = 21**

**(E1L2 + E2)L3 = 21 . 5 = 105**

**(E1L2 + E2)L3 + E3 = 105 + 2 = 107**

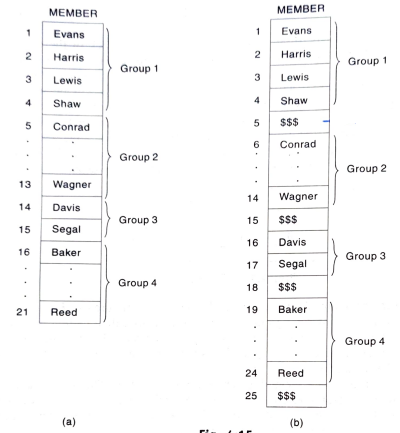
**Therefore,**

**LOC(MAZE[5, -1, 8]) = 200 + 4(107) = 200 + 428 = 628**

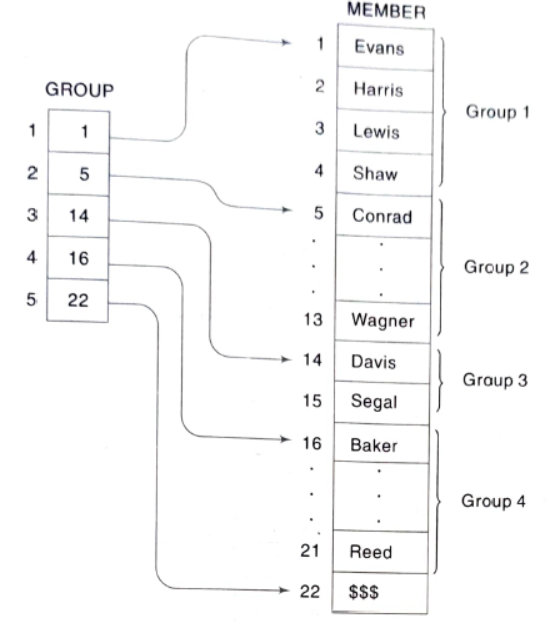
**Pointer Arrays**

****

**Above data is to be stored using pointer arrays**

****

**Data stored using pointer arrays**

****

**Suppose one wants to print only the names in the Lth group , where the value of L is part of the input. Based on the above diagram Since GROUP[L] and GROUP[L + 1] - 1 contain the locations of the first and last name in the Lth group, the following module accomplishes task:**

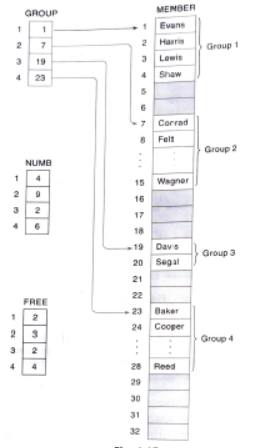
**1. Set FIRST = GROUP[L] and LAST = GROUP[L + 1] - 1.**

**2. Repeat for K = FIRST to LAST:**

**Write: MEMBER[K].**

**[End of loop.]**

**3. Return.**

****

**Suppose, if we want to print only the names in the Lth group, where L is part of the input, but now the groups are stored as in above figure.**

**It is observed that GROUP[L] and GROUP[L] + NUMB[L] – 1 contain, the locations of the first and last names in the Lth group. Thus the following module accomplishes the task:**

**1. Set FIRST= GROUP[L] and LAST = GROUP[L] + NUMB[L] - 1.**

**2. Repeat for K = FIRST to LAST:**

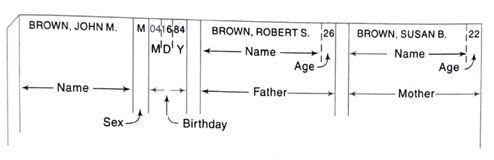
**Write: MEMBER[K].**

**[End of loop.]**

**3. Return.**

**The variables FIRST and LAST are mainly used for notational convenience**

**Records; Record Structures**

****

**Example**

**Suppose a hospital keeps a record on each newborn baby which contains the following data items: Name, Sex, Birthday, Father, Mother. Suppose further that Birthday is a group item with sub-items Month, Day and Year, and Father and Mother are group items, each with sub-items Name and Age. Above figure shows how such a record could appear.**

**The structure of the above record is usually described as follows. (Note that Name**

**appears three times and Age appears twice in the structure.)**

**1 Newborn**

**2 Name**

**2 Sex**

**2 Birthday**

**3 Month**

**3 Day**

**3 Year**

**2 Father**

**3 Name**

**3 Age**

**2 Mother**

**3 Namne**

**3 Age**

**Here, 1, 2, 3 are level number.**

**Example**

**A class of student records may be organized .as follows:**

**1 Student(20)**

**2 Name**

**3 Last**

**3 First**

**3 MI (Middle Initial)**

**2 Test(3)**

**2 Final**

**2 Grade**

**The identifier Student(20) indicates that there are 20 students. The identifier Test (3)**

**indicates that there are three tests per student.**

**Indexing Items in a Record**

**Example**

**(a) Consider the record structure Newborn. Sex and year need no qualification, since each refers to a unique item in the structure. On the other hand, suppose we want to refer to the age of the father. This can be done by writing**

**Newborn.Father.Age or simply Father.Age**

**Example**

**(b) Suppose the first line in the record structure is replaced by**

**1 Newborn(20)**

**That is, Newborn is defined to be a file with 20 records. Then every item automatically becomes a 20-element array.**

**Some languages allow the sex of the sixth newborn to be referenced by writing**

**Newborn.Sex[6] or simply Sex[6]**

**Analogously, the age of the father of the sixth newborn may be referenced by writing**

**Newborn.Father.Age[6] or simply Father.Age[6]**

**(c) Consider the record structure Student. Since Student is declared to be a file with 20 students, all items automatically become 20-element arrays. Furthermore, Test becomes a two-dimensional array. In particular, the second test of the sixth student may be referenced by writing**

**Student.Test[6, 2] or simply Test[6,2]**

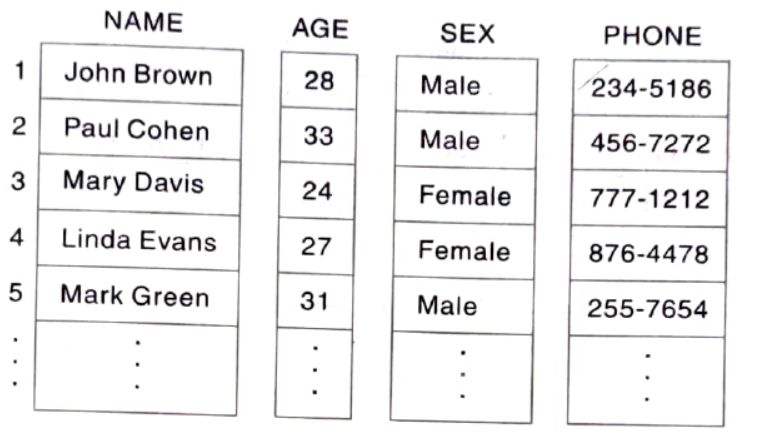
**The order of the subscripts corresponds to the order of the qualifying**

**identifiers. For example,**

**Test[3, 1]**

**does not refer to the third test of the first student, but to the first test of the third student.**

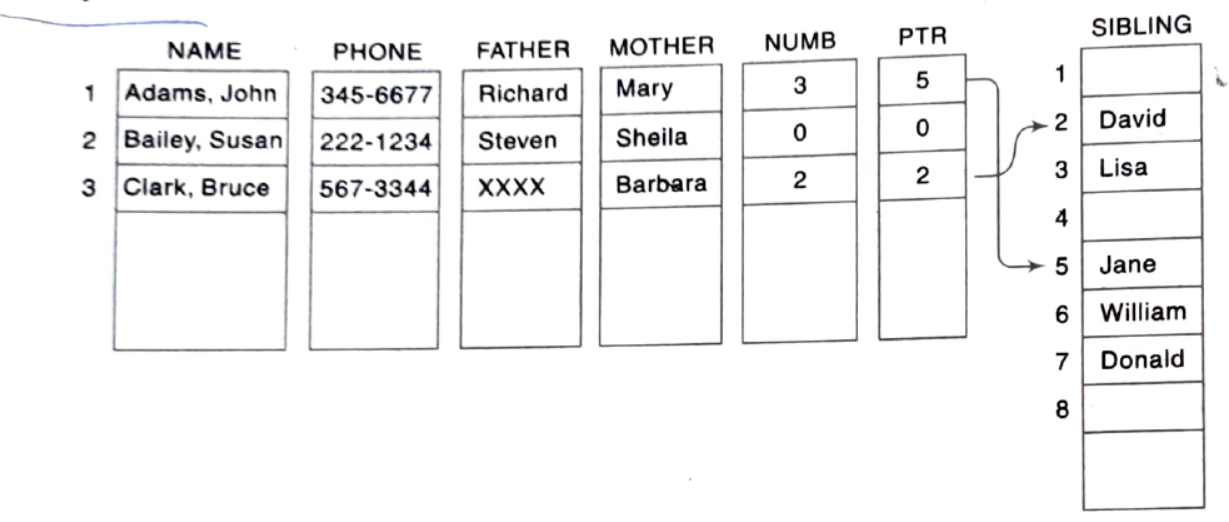
**Parallel Arrays**

****

**Records with Variable Lengths**

**Three such records of student details may be as follows:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NAME** | **FONE NO.** | **FATHER** | **MOTHER** | **SIBLINGS** |
| **Adams. John** | **346677** | **Richard** | **Mary** | **Jane, William, Donald** |
| **Bailey, Süsan** | **222-1234** | **Steven** | **Sheila** | **XXXX** |
| **Clark, Bruce** | **567-3344** | **XXXX** | **Barbara** | **David, Lisa** |

****