Data Structure

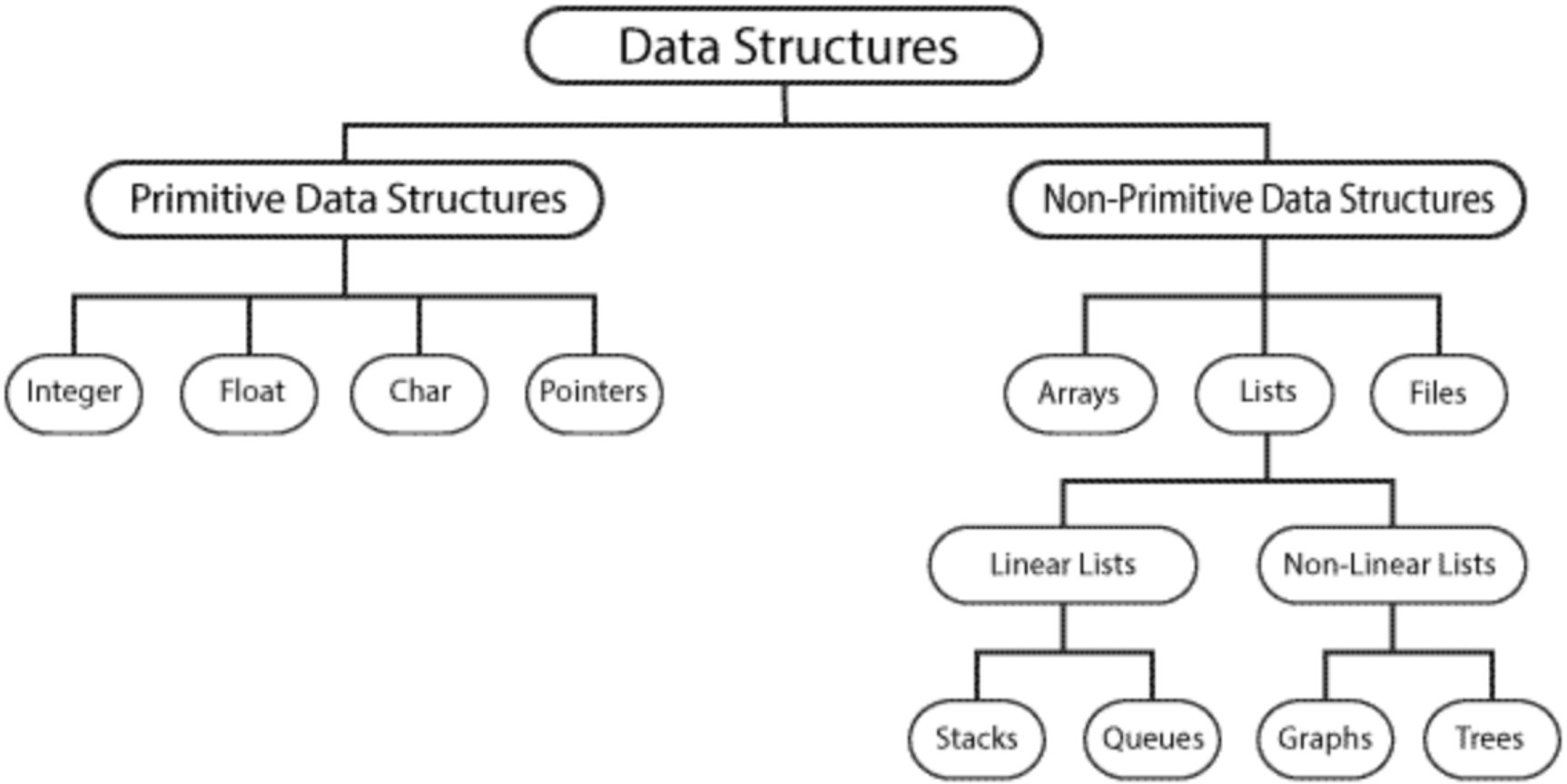
Introduction to Data Structure

- Definition:
  - A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.

- Example :
  - B-tress suits for implementing database, Hast table to look up identifiers used by compilers

- Advantages :
  - Provides a means to manage huge amount of data efficiently.
  - Efficient data structure are a key to designing efficient algorithm.

Type of Data structure



- In Linear Data Structure elements or item form a sequence one after other. It has exactly two neighbours.

- In non Linear Data structure elements or items form a hierarchical relationship between individual data items.

Arithmetic expression and operation

- Arithmetic expression is one which is evaluated by performing a sequence of arithmetic operations to obtain a numeric value with replaces the expression.
- Arithmetic operator : +,-,*,/,Uninary-
- Operand :
  - a numeric constant
  - a numeric variable which may be preceded by a unary+ or unary –
  - A arithmetic expression in parentheses

Arithmetic Evaluation

- Two variable integer I and Real R
- Real variable can hold any value
- Integer variable can hold integer value with any fraction being truncated
- Eg.R→real, I→Int
  - R = ¾ will assign .75 to R
  - I = ¾ will assign 0 to I

Strings and String Operations

- String is a finite sequence of symbols that are chosen from alphabets.

- String Operation:
  - String Concatenation
  - String Length
  - String Substring
  - String Compare
  - String Copy

Strings and String Operations

- Strings in C are represented by arrays of character. The end of the string is marked with a special character, the null character having value 0.
- Whenever we write string, enclosed in double quotes, C automatically creates an array of character, terminated by \0 character.

Eg. Char String[] = ”Hello, world”;

If the dimension is not specified, compiler computes as 13(including \0)

String COPY

- Eg.
- Char String1[]= “Hello, world”;
- Char String2[20];
- Strcpy(String2,String1);

String COMPARE

- This function compares two strings and return 0 if they are identical,
- a negative number if the 1st string is alphabetically “less than” the second string
- a positive number if the first string is “greater”
- Eg.

```
Char string3[] = “this is”;
Char string4[] = “atest”;
If (strcmp(string3,string4) == 0
    Printf(“string are equal\n”);
    Else
    Printf(”strings are different\n);

    Ans: Strings are different
```

Strcmp does not return a Boolean t/f or zero/nonzero

String Concatenation

- Strcat appends one string onto the end of another.

- Eg.
- Char String5[20] = “Hello”;
- Char String6[20] = “world”;
- Strcat(String5, String6);
- Printf(String5);

- Ans: String5   = Helloworld

Relations and Relational Operators

- The relational operator compares two operands and determine the valid of a relationship.

- Relational Operators:
  - <, >, <=, >=, =, ≠

  In C the result of of the relational operator is *int* and has the value 1, if the specified relationship is true and 0 if false

  Eg.

  Z = 10, MT = “Good”

  Z <= 9/3+Z →False

  Z ≠ Z + 5 → True

  MT < “Good morning” → True

Logical Operations and Expressions

- Logical Operator : not, and, or

Higher Precedence 

- Operator Precedence
  - Parantheses
  - Arithmetic
  - Relational
  - Logical

  - Eg. (x < Z + 3) and (X < 0))
  - (X < 2) or (X <0)
  - Not (one < 2)
  - We can have logical variable as character variable.

Operations on data Structure

- Create a data structure
- Destroying the data structure
- Access data with a data structure(selection)
- Change in the data structure(update by assignment statement)

Integer

- A quantity representing objects which are discrete in nature can be represented by an integer.
- Eg. No. of books
- The set of integer I is
- (…….-(n+1),-n,……-2,-1,0,1,2,……n,n+1,….)
- Conventional method for writing negative number is to place a sign symbol in front of a number. This method called sign and magnitude method.

Cont..

- Sign is represented as the left most bit of the binary number.
- Eg.

Numbe	signbit	Magnitude
+7	0	000…111
-6	1	000…010

Adding +7 and -6 requires subtraction operation +7+(-6)

Subtracting -6 from +7 involves addition +7- (-6)

Radix Complement Representation

- In Radix Complement representation all arithmetic operation are performed modulo M, for M= R^N where R is the radix in which the integer expressed and N is the maximum number of digits required to represent an integer modulo N.

2’s complement numbers using modulo 16

- N = 4, 2^4 =16

Integer	2’complement	integer	2’s complement	
0	0000		-1(16-1=15)	1111
1	0001		-2(16-2=14)	1110
2	0010		-3(16-3=13)	1101
3	0011		-4(16-4=12)	1100
4	0100		-5(16-5=11)	1011
5	0101		-6(16-6=10)	1010
6	0110		-7(16-7=9)	1001
7	0111		-8(16-8=15)	1000

2’s complement modulo 16 system are related to integers from -8 to 7

What is the 2,s complement representation of -38 expressed as a modulo 32 number?

- Express -38 as modulo 32 number
  - -38 mod 32 = -6
- To find 2,s complement representation
  - 32 – 6 = 26

- Major advantage of radix complement notation is that we can perform addition and subtraction operation using only addition and complementation.

Example:

A. $3 + 4 = (0011)_2 + (0100)_2 = (0111)_2 = 7$

B. $3 - 4 = (0011)_2 + 2\text{'s Comp}(0100)_2$
$= (0011)_2 + (1100)_2$
$= (1111)_2 = -1$

C. $-3 + 4 = 2\text{'s comp}(0011)_2 + (0100)_2$
$= (1101)_2 + (0100)_2$
$= (0001)_2 = 1$

D. $7 + 7 = (0111)_2 + (0111)_2 = (1110)_2 = -2$

E. $-7-7 = 2\text{'s comp}(0111)_2 + 2\text{'s comp}(0111)_2$
$= (1001)_2 + (1001)_2 = (0010)_2 = 2$

The range of 2's complement form assuming a modulo 16 system is -8 to +7

So 7 + 7 and -7-7 will cause overflow.

1s complement form

a. $3 + 4 = (0011)_2 + (0100)_2 = (0111)_2 = 7$

b. $3 - 4 = (0011)_2 + 1\text{'s comp}(0100)_2 = (0011)_2 + (1011)_2$

$= (1110)_2 = -1$

c. $-3 +4 = (1\text{'s comp}(0011)_2) + (0100)_2$

$= 1100 + 0100 = 0001$

d. $7 + 7 = (0111)_2 + (0111)_2 = (1110)_2 = -1$

e. $-7-7 = 1\text{s'comp}(0111)_2 + 1\text{'scomp}(0111)_2$

$(1000)_2 + (1000)_2 = (0000)_2 = 0$

(d,e) it is a overflow

2's comp form I preferred than 1'scomp form because +0, -0 controversy is avoided.

Integer

- Integer values that can be expressed using an n-bit storage representation

  $2^0 + 2^1 + 2^2 +$ -------- + 2n-1 is $\sum_{i=0}^{n-1} 2^n$-1

In a n-bit 2's comp storage representation, the first bit expresses the sign of the integer. Therefore positive integer in the range

0 to $2^{n-1} - 1$

In general, we find that an integer N can be accommodated using a n bit 2's complement storage representation if

$$-2^{n-1} <= N <= 2^{n-1} - 1$$

Because both +0 and -0 can be represented, the range of integer representation in I's complement is one less than the range of 2'scomplement.

$$-2^{n-1} + 1 <= N <= 2^{n-1} - 1$$

Real Numbers

Fixed- point representation for a real number A_R in radix system R as

$A_R = \pm a_{N-1}\, a_{N-2} \ldots\ldots\ A_1\, a_0.\, a_{-1}\, a_{-2} \ldots\ldots\ A_{-(M-1)}\, a_{-M})_R$

which has a literal expansion of

$A = \pm \sum_{i=-M}^{N-1} a_i R^i$

Fixed point notation is sufficient to represent most of real numbers.

Limitation in fixed point representation

- Distance 16,800,000,000,000 km in celestial bodies
- .000000000082 meters energy emission in bubble chamber.
- Cost of computer is directly affected by Precision of the number that can be handled in computation.
- Greater the precision allowed the larger and more complex in the arithmetic unit.
- Arithmetic computation of real numbers requires a precision of 27decimal digits, when representing in binary it takes 90 binary digits or bits.
- Computer with word size 90 is not economical.

Floating point or scientific notation

16,800,000,000,000 in short $.168 \times 10^{14}$

.0000000000832 in short $.832 \times 10^{-10}$

Sign	Exponent	Fraction

The Fraction part should be normalized

F lie in the interval $R^{-1} <= F < 1$

$2^{-1} <= F <= 1$ if in binary

$16^{-1} <= F <= 1$ if in hexadecimal

Character Information

- **Two mainly used Character set**

EBCDIC (Extended Binary Coded Decimal Interchange Code)

ASCII (American standard code for Information Interchange)

A character is represented in memory as a sequence of bits where a distinctive bit sequence is assigned to each character in the character set. Fixed length and variable length is available. Fixed length is preferable.

Logical Information

- Only two logic constants exists true and False.
- The storage representation of logical values id dependent upon the language compiler or interpreter. The most obvious storage structure that can be applied to logical data is the single bit.

Pointer Information

Suppose a program contains four occurrences of the real constant 3.14. During the compilation process, four copies of 3.14 is created. But by using pointers one copy + 3 pointers is sufficient.

- Less space is occupied by pointers rather than fixed point representation for real numbers.
- Homogenous method of referring any data structure by having single fixed size data item.
- They permit faster insertion and deletion of elements to and from a data structure.

Non-Primitive Data Structure

Arrays

- An ordered set which consists of a fixed number of objects.

  No deletion or insertion operation are performed on arrays.
- Operations:
  - Elements can be changed to a new value.
  - To delete an element, we can make it to zero.

Lists

- An ordered set consisting of a variable number of elements to which insertions and deletions can be made.
- A list which displays the relationship of adjacency between elements is said to be linear.
- Operations:
  - All the operations that carried out in array.
  - Insertion and deletion of elements in a list in specified position.
  - Split a list into a number of other list.
  - Copy a list
  - No. of elements in a list.
  - Sorting the elements in ascending or decending
  - Searching a list foe an element.

Difference between Arrays and List

- In list the size of a list may be changed by updating.

- Each element in a list is composed of one or more fields. A field can be considered to be the smallest piece of information that can be referenced in a programming Language.

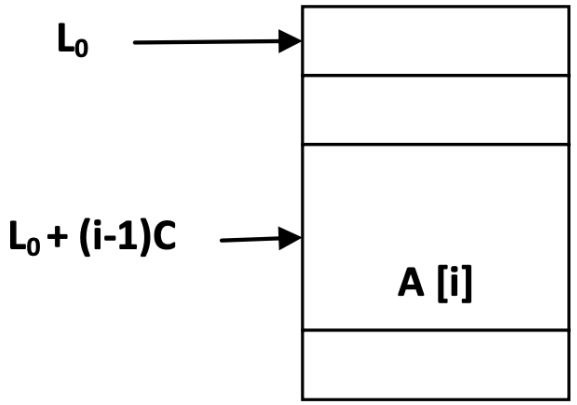
File

- A large list that is stored in the eternal memory of a computer.
- File has records that are accessed in frequently.

Storage Structure for Arrays

One Dimensional Array

- Simplest data structure that makes use of computed address to locate its elements is the one-dimensional array or vector; number of memory locations is sequentially allocated to the vector.
- A vector size is fixed and therefore requires a fixed number of memory locations.
- Vector A with subscript lower bound of "one" is represented as below….



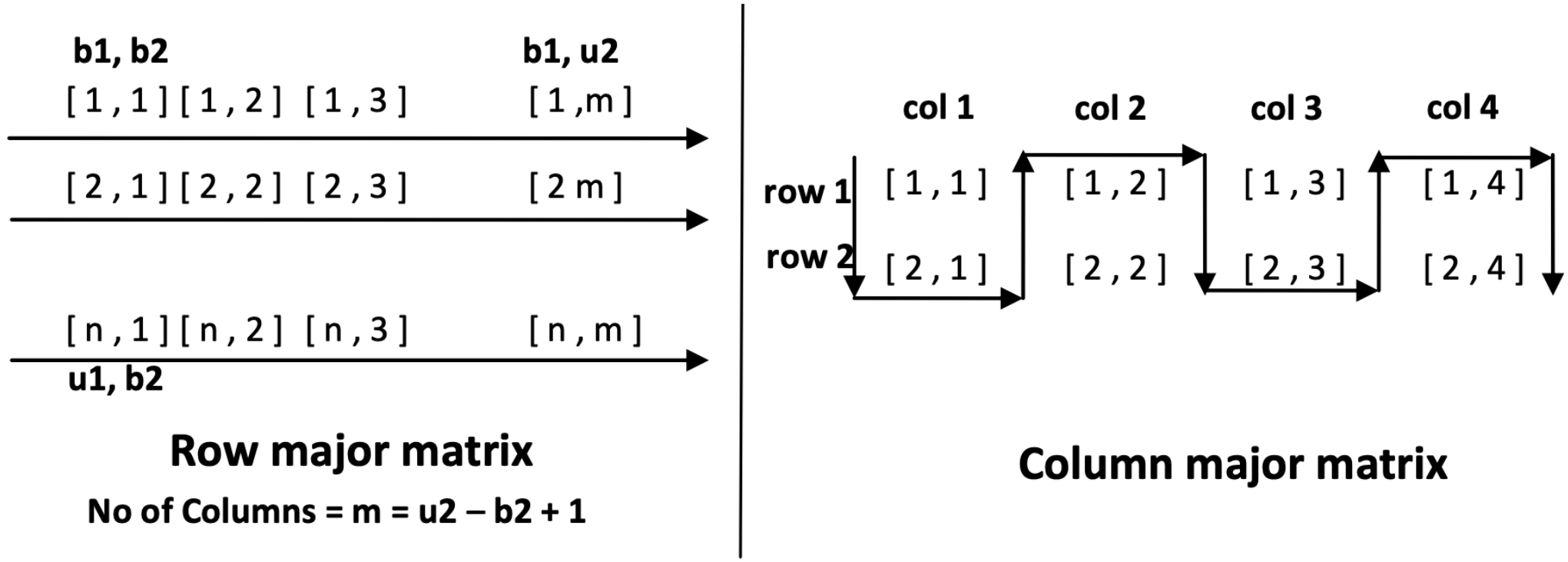
- L_0 is the address of the first word allocated to the first element of vector A.
- C words are allocated for each element or node
- The address of A_i is given equation **Loc (A_i) = L_0 + C (i-1)**
- Let's consider the more general case of representing a vector A whose lower bound for it's subscript is given by some variable b. The location of Ai is then given by **Loc (A_i) = L_0 + C (i-b)**

Storage Structure for Arrays….

Two Dimensional Array

- Two dimensional arrays are also called table or matrix, two dimensional arrays have two subscripts
- Two dimensional array in which elements are stored column by column is called as column major matrix
- Two dimensional array in which elements are stored row by row is called as row major matrix
- First subscript denotes number of rows and second subscript denotes the number of columns
- Two dimensional array consisting of two rows and four columns as above Fig is stored sequentially by columns : A [1, 1], A [2 , 1], A [1 , 2], A [2 , 2], A [1 , 3], A [2 , 3], A [1, 4], A [2, 4]
- The address of element A [i , j] can be obtained by expression **Loc (A [i , j]) = L_0 + (j-1)*2 + i-1**
- In general for two dimensional array consisting of n rows and m columns the address element A [i , j] is given by **Loc (A [i , j]) = L_0 + (j-1)*n + (i – 1)**
- In row major matrix, array can be generalized to arbitrary lower and upper bound in its subscripts, assume that b1 ≤ I ≤ u1 and b2 ≤ j ≤u2

Storage Structure for Arrays….

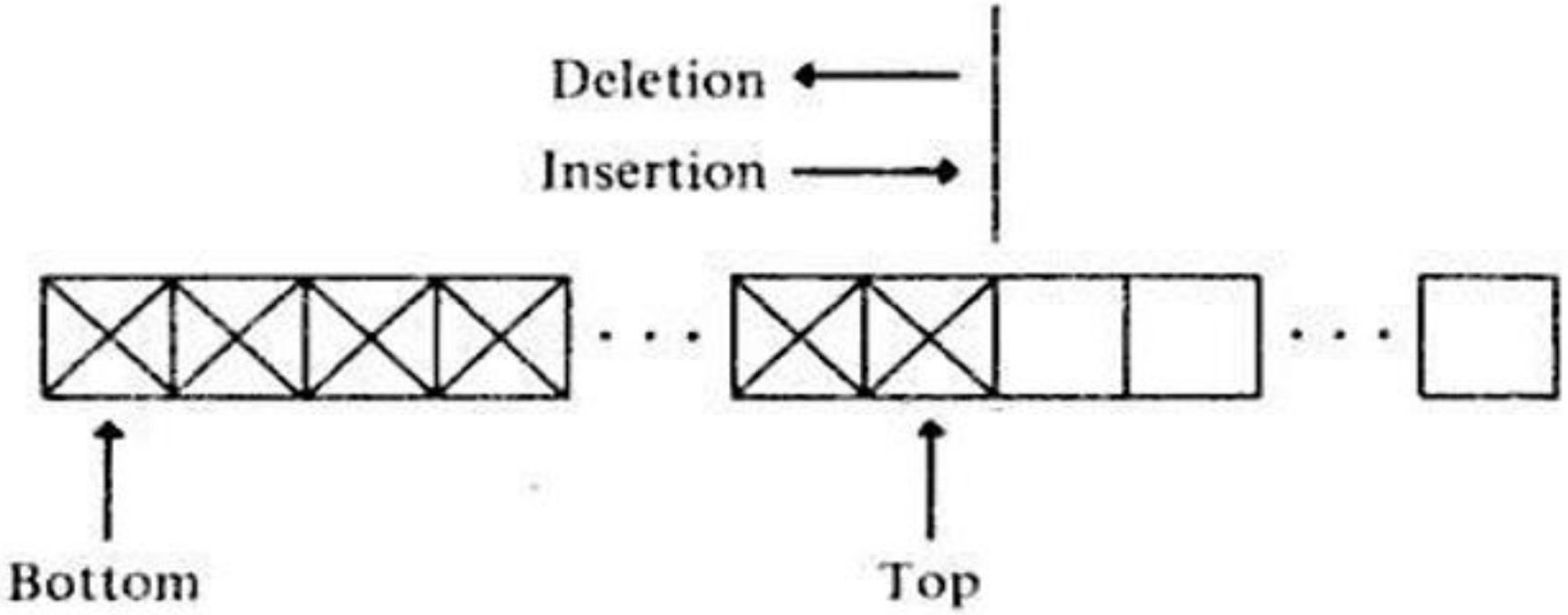


- For row major matrix : **Loc (A [i , j]) = L_0 + (i – b1) *(u2-b2+1) + (j-b2)**

Stack

- A linear list which allows insertion and deletion of an element at one end only is called stack.
- The insertion operation is called as **PUSH** and deletion operation as **POP**.
- The most and least accessible elements in stack are known as top and bottom of the stack respectively.
- Since insertion and deletion operations are performed at one end of a stack, the elements can only be removed in the opposite orders from that in which they were added to the stack; such a linear list is referred to as a LIFO (last in first out) list.

Stack Representation



A pointer TOP keeps track of the top element in the stack. Initially, when the stack is empty, TOP has a value of "one" and so on.

Each time a new element is inserted in the stack, the pointer is incremented by "one" before, the element is placed on the stack. The pointer is decremented by "one" each time a deletion is made from the stack.

Stack- PUSH Operation

Procedure : PUSH (S, TOP, X)

- This procedure inserts an element x to the top of a stack which is represented by a vector S containing N elements with a pointer TOP denoting the top element in the stack.

```
1. [Check for stack overflow]
   If      TOP ≥ N
   Then    write ('STACK OVERFLOW')
           Return
2. [Increment TOP]
   TOP ←TOP + 1
3. [Insert Element]
   S[TOP] ←X
4. [Finished]
   Return
```

Stack- POP Operation

Function : POP (S, TOP)

- This function removes the top element from a stack which is represented by a vector S and returns this element. TOP is a pointer to the top element of the stack.

1. [Check for underflow of stack]

   If TOP = 0

   Then Write ('STACK UNDERFLOW ON POP')

   Take action in response to underflow

   Return

2. [Decrement Pointer]

   TOP ← TOP – 1

3. [Return former top element of stack]

Applications of Stack

- Recursion(When functions are called)
- Polish expression and their compilation
  - To convert infix to postfix
  - To Evaluate a postfix expression

Polish Notation

- Infix Notation
  - Operator symbol is placed between its two operands.
  - A + B, C – D, E * F
- Polish Notation(Prefix)
  - Operator symbol is placed before its two operands.
  - +AB, -CD, *EF, /gh
  - (a+b)*c = *+abc
  - A+(B*c) = +A*BC
- Reverse Polish Notation(postfix)
  - AB+,CD-,EF*,GH/

Infix to Postfix Conversion

- If *Token* is
- (i) a left Parenthesis: Push it onto the stack.
- (ii) A right Parenthesis : Pop and Display stack element until a left parenthesis is popped, but donot display it(*It is an error if the stack becomes empty with no left parenthesis is found)
- (iii) an operator: If the stack is empty or Token has a higher Priority(not even =) than the top stack, Push token onto the stack, otherwise, pop and display the top stack element.

  Then repeat the comparison of Token with the new top stack item.

  A left Parenthesis is in the stack, is assumed to have a lower priority than that of operators.
- (iv) an operand: Display it.

Example 1:(infix to Postfix)
Given expression: A + B * C – D / E

Steps	Infix	Stack	Postfix
a)	A+B*C-D/E	Empty	Empty
b)	+B*C-D/E	Empty	A
c)	B*C-D/E	+	A
d)	*C-D/E	+	AB
e)	C-D/E	+*	AB
f)	-D/E	+*	ABC
g)	D/E	-	ABC*+
h)	/E	-	ABC*+D
i)	E	-/	ABC*+D
j)		-/	ABC*+DE/-

Example 2:(infix to Postfix)
Given expression: A + B – (C +D) +E

Steps	**Infix**	**Stack**	**Postfix**
a)	A*B-(C+D)+E	Empty	Empty
b)	*B-(C+D)+E	Empty	A
c)	B-(C+D)+E	*	A
d)	-(C+D)+E	*	AB
e)	(C+D)+E	-	AB*
f)	C+D)+E	-(	AB*
g)	+D)+E	-(	AB*C
h)	D)+E	-(+	AB*C
i)	)+E	-(+	AB*CD
j)	+E	-	AB*CD+
k)	E	+	AB*CD+-
l)		+	AB*CD+-E
m)			AB*CD+-E+

Postfix Evaluation

- Operand : Push
- Operator : Pop operands, do the maths, push result back onto stack
- Given Postfix Expression : 123+*

Steps	**Postfix**	**Stack**
a)	123+*	Empty
b)	23+*	1
c)	3+*	1 2
d)	+*	1 2 3
e)	*	1 5 //5 from 2 + 3
f)		5 //5 from 1 X 5

Recursion

- A recursive algorithm definition is when something is defined partly in terms of itself
- Mathematical definition of Factorial

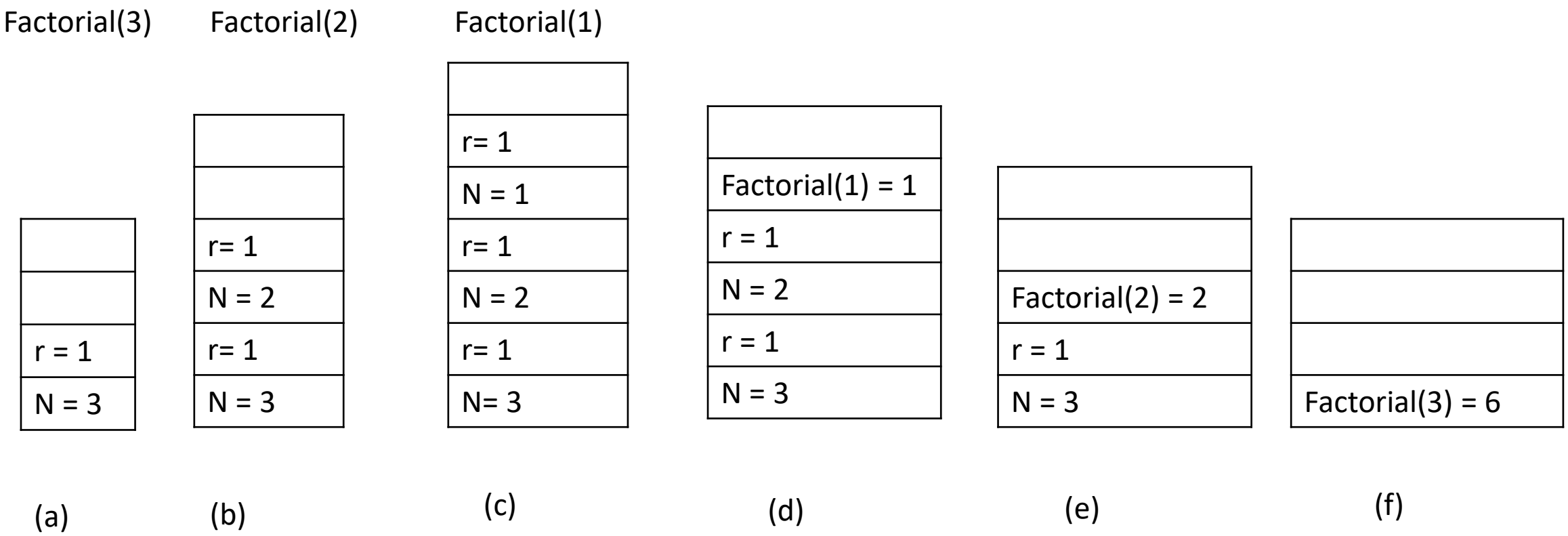
Factorial(n) $= \begin{cases} 1, if\ n \leq 1 \\ n\ *factorial\ (n-1) otherwise \end{cases}$

```
Static int factorial(int n)
Int r = 1;
{ If (n <=1) return 1
        else {r = n* return n * factorial(n-1);
                return r;}
```

Calling the function as

X = factorial(3), this enters the factorial function with n=3 on the stack.

Implementation of Factorial Recursive Function



Queues

- A queue is a linear list of elements in which deletion can take place only at one end called the **FRONT** and insertion can be placed only at the other end called the **REAR.**
- Queue are also called First In First Out(FIFO) lists, since the first element in a queue will be the first element out of the queue.
- Example: Waiting line in the bank.
- Queues may be represented either by list or arrays. Queue maintains two pointers
  - FRONT: contains the location of the front element of the queue.
  - REAR: contains the location of the rear element of the queue.

QUEUE : Example

			1	2	3	4	5
a.	Intially Empty	F=0 R=0					
b.	A,B,C INSERTED	F=1 R=3	A	B	C		
C	A DELETED	F=2 R=3		B	C		
D	D,E INSERTED	F=2 R=5		B	C	D	E
E	B,C DELETED	F=4 R=5				D	E
F	F INSERTED	F=4 R=1	F			D	E
G	D DELETED	F = 5 R = 1	F				E
H	G,H INSERTED	F = 5 R = 3	F	G	H		E
I	E DELETED	F = 1 R = 3	F	G	H		
J	F DELETED	F = 2 R = 3		G	H		
K	K INSERTED	F = 2 R = 4		G	H	K	
L	G AND H DELETED	F = 4 R = 4				K	
M	K DELETED, QUEUE EMPTY	F = 0 R = 0					

Points to remember during queue implementation

- FRONT = NULL will indicate that the queue is empty
- When an element is deleted from the queue FRONT = FRONT + 1
- When an element is inserted from the queue REAR = REAR + 1
- After N insertion , REAR will be pointing to QUEUE[N] or last part of the array. This condition occurs even though the queue itself may not contain many elements.
- When inserting element at REAR = N one way to do this is to simply move the entire queue to the beginning of the array, changing FRONT and REAR accordingly. But this procedure is expensive.
- So we assume the array is circular ie. QUEUE[1] comes after QUEUE[N].
- Similarly, if FRONT = N and an element of queue is deleted, we reset

FRONT = 1 instead of increasing FRONT to N + 1

Queue Insert

QINSERT(QUEUE,N,FRONT,REAR,ITEM)

//This Procedure inserts an element ITEM into a queue

1. If FRONT = 1 and REAR = N, or if
   FRONT = REAR + 1 then Write “Overflow” and Return.
2. [find new value of REAR]
   If FRONT = NULL then [queue initially empty]
   set FRONT = 1 AND REAR = 1
   else if REAR = N then SET REAR = 1
   else set REAR = REAR + 1
3. SET Queue[REAR]= ITEM
4. RETURN.

Queue Delete

QDELETE(QUEUE,N,FRONT,REAR,ITEM)

//This Procedure deletes an element from a queue and assigns it to the variable ITEM

1. [Queue already empty]

If FRONT = NULL, then Write “underflow” and Return.

2. Set ITEM = QUEUE[FRONT]

3. [Find new value of FRONT]

If FRONT = REAR then [queue has only one element to start]

set FRONT = NULL AND REAR = NULL

else if FRONT = N then SET FRONT = 1

else set FRONT= FRONT + 1

4. RETURN.