UNIT- 1

Introduction

Basic Terminology :—

Data – values or set of values

Information – Meaningful data

Data structure – Data may be organized in many different ways. The logical or mathematical model of a particular org^n of data is called DS.

It is of 2 type –

1. Linear DS – Array, Linked list, Stack, Queue
2. Non-Linear DS – Tree, graph

Linear DS – Processing of data items is possible in linear fashion i.e. data can be processed one by one sequentially

Non-Linear DS – A DS in which insertion & deletion is not possible in a linear fashion

Elementary Data organization

Field ⟶ Record ⟶ File

Field – Single elementary unit of inf^n

Record – Collect^n of fields

File – Collect^n of records

It efficiently process certain collection of data. So, data are also organized into more complex type of structures (Array, linked list, graph, tree etc.)

steps for study of such DS —

→ Logical / Mathematical description of structure

→ Implementation of the structure on a computer

→ Quantitative analysis of the structure (Memory needed to store the structure, time needed to process)

Data may be organized in many different ways as studied above. The logical or mathematical model of a particular org^n of data is called a DS.

Array:-

It is list of a finite no. of similar or homogeneous data elements
→ Elements are stored in successive memory locations.
→ Elements are referenced by an index set

for Ex.

Let Elements of array $A = a_1, a_2, a_3 \dots a_n$: Subscript Notation
$= A(1), A(2), \dots A(N)$: Parenthesis (FORTRAN, PL/1)
$= A[1], A[2] \dots A[N]$: Bracket Notation (PASCAL, C, C++)

Here A[I] a, A — Subscripted variable
I — subscript

Linear Array = 1D array (single dimensional array)
= each element is referenced by one subscript

Ex. student[5] = {A, B, C, D, E}

Multi Dimensional Array = 2D array
= each element is referenced by 2 or more subscripts

Ex. → Matrices in Mathematics
→ Tables in Business appn
→ A[2][2] = $A\begin{bmatrix} - & - \\ - & - \end{bmatrix}_{2\times 2}$

1D:

	Student
1	A
2	B
3	C
4	D
5	E

student[1] = A

2D:

Dept / store	Sales 1	2	3
1	25	50	100
2	200	300	400
3	500	600	700

sales[1,1] = 25

Arrays

(no bound checking concept in 'C' it is upto pgmr to check the UB)

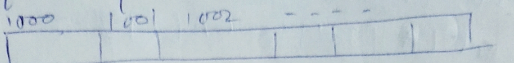
→ Array is a consecutive set of memory locations (not always)

Definition :→ It is a list of finite no. of similar data elements

1 Darray / Linear Array

Representation of linear array in memory :-

Memory = sequence of addressed locations

= 

Address calculation

LOC (A[K]) = addres of element A[K] of the array A

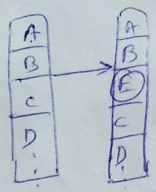
Base (A) = Base address of A (Array name is actually a pointer allocated to the name of the array) to the 1st locat^n of memory block

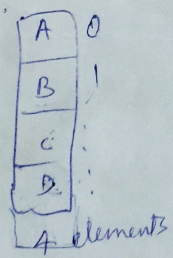
$$LOC(A[K]) = Base(A) + w(K - LB)$$

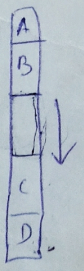
w = no of words per memory cell of array

Analysis regarding Insertion & deletion -

Insertion







Create space to insert E

(Move C, D downward by 1 position)

5 elements

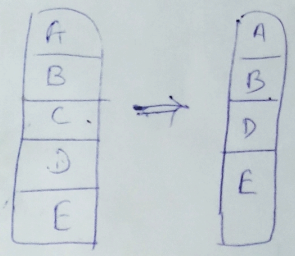
(Insert E & increment the array length by 1)

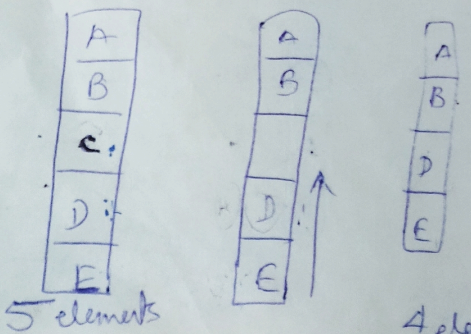
INSERT (LA, N, K, ITEM)

LA – Linear array with N elements
K – Positive integer such that K≤N
ITEM is inserted into K^{th} position in LA

1. set J := N-1 // Initialize counter
2. Repeat 3 & 4 while J ≥ K
3. set LA[J+1] := LA[J] // Move J^{th} element downward
4. set J = J-1 // Decrease counter / end of loop
5. set LA[K] := ITEM // Insert element
6. set N := N+1 // Reset N
7. Exit

Deletion





5 elements

4 elements

DELETE (LA, N, K, ITEM)

1. set ITEM := LA[K]
2. Repeat for J = K to N-1.
3. set LA[J] := LA[J+1] // (Move J+1st element upward) end of loop
4. N := N-1 // Reset the no N of elements in LA
5. Exit

Traversal

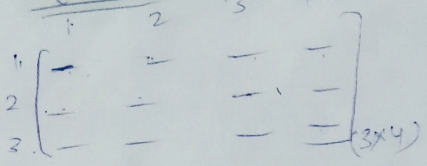
TRAVERSE (LA, LB, UB)

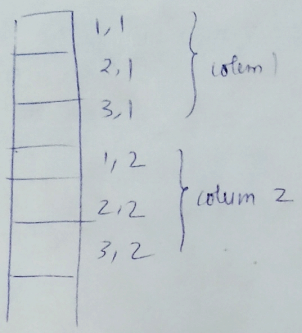
LA - Linear array
LB - lower bound
UB - upper bound

1. Repeat for K = LB to UB
   Apply PROCESS to LA[K]
   (End of loop)
2. Exit

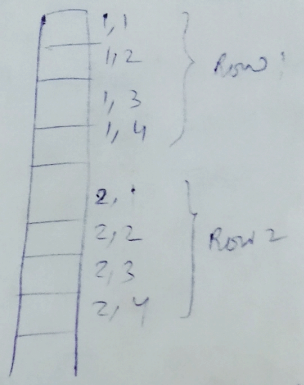
Multidimensional Array

2D Array :





Column-major order



Row-major order

Matrix Programs

Traversal

```
for (i = 1 to n)
  {
    for (j = 1 to m)
      printf ("%A[i][j]);
  }
```

Matrix Addition

```
for (i = 1 to n)
  {
    for (j = 1 to m)
      C[i][j] = A[i][j] + B[i][j]
  }
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

Matrix Multiplication

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times 3} \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}_{3\times 3} = \begin{bmatrix} 1+20+39 & 8+22+42 & \\ & & \\ & & \end{bmatrix}_{4\times 3}$$

m p p n

Mat(A, B, C, M, P, N)

A - M x P Matrix array
B - P x N Matrix array
C - Product of A & B is stored in this M x N matrix

→ ① Repeat ② to ④ for I = 1 to M
→ ② Repeat ③ to ④ for J = 1 to N
→ ③ set C[I,J] := 0 // initialise C[I,J]
→ ④ Repeat for K = 1 to P
C[I,J] := C[I,J] + A[I,K] * B[K,J]

[end of inner loop]

[end of step ② loop]

[end of step ① loop]

→ ⑤ exit

INITIALIZATION OF ARRAYS :-

Initializing an 1D array —

```
int A[5] = {1, 2, 3, 4, 5};
char A[6] = {'h', 'e', 'l', 'l', 'o', '\0'};
char A[] = "Hello";
```

Initializing a 2D array

```
int A[2][2] = {1, 2, 3, 4};
int A[3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

Initializing a 3D array

```
int A[3][3][3] = {
                  ...
                  ...
                  ...
                 };
```

A[1][2][0] = K

A[0][0][0] = 0
A[0][2][2] = 0

Application of arrays –

→ To list some elements of a record.
→ Matrices in Mathematics
→ Tables in Business app^n

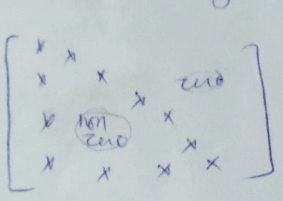
SPARSE Matrices & Vectors –

Matrices with a relatively high proportion of zero entries are called SPARSE Matrices.

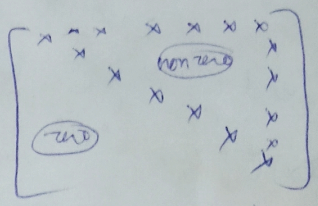
There are 2 types of n square sparse matrix

1) Triangular Matrix –

Non-zero entries can only occur (on) or (below/above) the main diagonal.



lower triangular



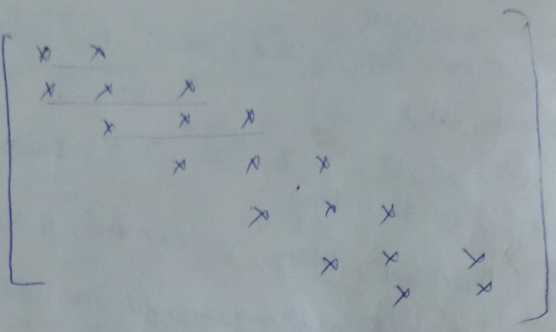
upper triangular

Ex.

$$\begin{bmatrix} 1 & & & \\ 2 & 5 & & \\ 3 & 6 & 8 & \\ 4 & 7 & 9 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 \\ 3 & 6 & 8 & 0 \\ 4 & 7 & 9 & 12 \end{bmatrix}$$

② Tridiagonal Matrix

non-zero entries can only occur on the diagonal or on elements immediately above or below the diagonal.



Ex.

$$\begin{bmatrix} 1 & 2 & & & \\ 3 & 4 & 5 & & \\ & 6 & 7 & 8 & \\ & & 9 & 10 & \end{bmatrix} \equiv \begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 \end{bmatrix}$$

NOTE:- Natural method of representing matrices in memory as 2D array may not be suitable for sparse matrices i.e. one may save space by storing only nonzero entries. (to avoid wastage of memory space which will caused by storing zeros in memory)

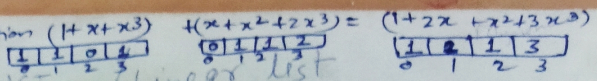
Array Representation of polynomials

$1+x+2x^2+3x^3+4x^4$

1	1	2	3	4	5
0	1	2	3	4	5

5x5

Polynomial Addition $(1+x+x^3)$ $+(x+x^2+2x^3) = (1+2x+x^2+3x^3)$



ORDERED LIST – / Linear List

It is one of the simplest & most commonly found data object.

Ex –

① days of the week
(Mon, Tue, Wed, Thu, Fri, Sat, Sun)

② values in card deck
(2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, Ace)

③ floors of a building
(basement, lobby, mezzanine, first, second, third)

④ Years the United states fought in World war II
(1941, 1942, 1943, 1944, 1945)

It may be empty or can be written as $(a_1, a_2 \cdots a_n)$ where a_i are atoms of some set S.

Difference b/w Array & List :—

Array	List
It is a set consistent of a fixed no. of data items	It is a ordered set consisting of a variable no. of data items
Insertion & deletion operations are performed	can be performed using pointers

Character string in C – (%s) (%c)

"Length of a string = no of characters of the string + 1"

↓ to store the null char ('\0')

A string of char's is stored in successive elements of a char array & terminated by the Null char

char msg[5] = "hello" : means

msg[0] = 'h'
msg[1] = 'e'
msg[2] = 'l'
msg[3] = 'l'
msg[4] = 'o'
msg[5] = '\0' = NULL = 0

ASCII value of NULL is 0

Library functions for String Manipulation –

strcpy – To copy a string, including the null char

string.h

```
char A[10], B[10];
gets(A);
strcpy(B, A);
printf("A: %s and B: %s \n", A, B);
```

strncpy – Similar to strcpy, but it allows the no of char's to be copied.
– If the source is shorter than the destination dest'n is padded with null char upto the length specified.

```
char *strncpy(char *dst, const char *src,
              size_t len);
```

strcat – used to append a src to the end of dest" str

→ char *strcat (char *dst, const char *src)

→ strcat (A, B); // where char A[20] = "; char B[20];

strncat –

char *strncat (char *dst, const char *src, size_t N);

strcmp – To compare 2 strings, it returns no >0 (if string1 > string2), <0 (if string1 < string2), =0

if (strcmp (first, second) == 0)

puts ("The 2 strings are equal");

strncmp – It compares first N char of each string

int strncmp (const char *first, const char *, size_t N);

strlen – It returns the length of the string (not counting the null char)

→ size_t strlen (const char *str);

→ int length;
gets (name);
length = strlen (name);

strrev – To reverse the string

- strrev (str);

strlwr – convert a string into lower case

strlwr (str);

- It can be used by ASCII values

Difference = 32

(lower case ASCII value = upper case ASCII value − 32)

A

WAP

Arrays of strings :-

It is just a 2D array of chars.

```
char A[3][6] = { "Ryan",
                 "Tom",
                 "Jackie" };
```

3 → no of strings

6 → no of char per string.
(If no is not mentioned, compiler will take the longest length of the name as the max length)

WAP to check a string as a palindrome — Page 4 of 25 of Book 2

~~STACK~~

Array as Parameter

Ptr 2 parameters

```
display (int *, int);
main()
{
  int num[] = {1,2,3,4,5,6};
  display (&num[0], 6);
}
display (int *j, int n)
{
  int i;
  for (i=0; i<=(n-1); i++)
  {
    printf ("\n element = %d ", *j);
    j++;
  }
}
```

---

single parameter

```
void display (int *)
main()
{ int n[] = {1,2,3}
  display (&n[0]);
}
void display (int *a)
{
  for (int i=0; i<=2; i++)
  { printf ("%d", *a); a++; }
}
```

Length of string

```
void length (char *);
main()
{
  char a[10] = "computer";
  length (&a[0]);
}
void length (char *a)
{ for (int c=0; *a!='\0'; c++) a++;
  printf ("%d", c);
}
```