# Data Organization(Lists and Arrays)

# Objectives

- To understand how indexing is used to organize data.

- To understand concepts of single and multi indexing.

# Organizing data

- Humans hate disorder.

- We try and organize our lives, our cities, our homes and even sometimes our desks and our bedrooms.

- There are many ways of organizing things, and we naturally organize different things in different ways.

- The organization we use depends on the things we are trying to organize and often more importantly, what we actually wish to do with them once they are organized.

  - Addresses are organized into postal areas so that it is easier for letters to be delivered.

  - A bookshop organizes its novels on the shelves usually alphabetically by author.

- The way we organize the things we must manipulate to achieve a task can make a massive difference to how quickly, easily or successfully the task can be done.

*Department of CSE*

# Game of *Spit Not So (Berlekamp et al, 1982)*

The game is played as follows by two players:

- *Write the nine words: SPIT NOT SO FAT FOP AS IF IN PAN* on separate pieces of cards.

- The cards are placed face up.

- Each player takes it in turns to take a card.

- To win you must get all the cards that have the same letter.

  - For example if you had picked up the cards SPIT, IF and IN, you would have all the cards containing the letter I so would win.

- If all the cards are taken with no one having all of a particular letter then the game is a draw.

- *Play a few games to get then idea, then turn the page to see how a suitable organization will help you play better.*

*Department of CSE*

- Before you start to play draw up the following grid, with the words that are on the cards written in the squares shown.

- The position of each word is important.

- They have been organized so that there is a single letter in common in each row, column and diagonal.

- All the F-words run along the bottom, for example.

| NOT | IN | PAN |
|-----|------|-----|
| SO | SPIT | AS |
| FOP | IF | FAT |

- Keep this hidden from your opponent – otherwise they may see the secret.

- Play the game exactly as before, except, now each time you pick up a card, put a cross in the square that corresponds to that word.

- Put a nought in the squares corresponding to the cards your opponent takes.

- For example, the above game will end with your table looking like the following:

| NOT | IN | PAN |
|---|---|---|
| **O** | | **X** |
| SO | SPIT | AS |
| **O** | **X** | |
| FOP | IF | FAT |
| **X** | **O** | **X** |

*Department of CSE*

# Is there a better way to store and access related items?

- *How should I store the details of all the student names of my class?*

  - *If 60 students in a class can I have 60 different variable names?*
    - E.g. name1, name2, name3 ......, name60.

    - ✓ *Is it organized for efficient access?*

    - ✓ *Is it easy to be remembered?*

    - ✓ *What happens if number of students increase from 60 to 80?*

# *Is there a better way to store and access related items?... Answer*

- *How should I store the details of all the student names of my class?*

  - *If 60 students in a class can I have 60 different variable names?*
    - E.g. name1,name2, name3 ......., name60.

    - ✓ *Is it organized for efficient access?* <span style="color:red">No</span>

    - ✓ *Is it easy to be remembered?*<span style="color:red">No</span>

    - ✓ *What happens if number of students increase from 60 to 80?*

      <span style="color:red">*It becomes even more difficult to be handled and remembered*</span>

# Lists

- A list is a sequence of items that are arranged in a particular order.

- Eg. list of your all time favorite records

  1. Bohemisn Rhspdody – Queen
  2. Stairway to Heaven – Led Zeppelin
  3. Your Song – Elton John
  4. Who made who – AC/DC
  5. Sweet Child of Mine – Guns 'n Roses

- Any item on the list can be identified by its position in the list.

- We denote the first item in the list as list [first_location] and the second item in the list as list[second_location].



Capacity = 15

| 2 | 7 | 1 | 3 | 7 | 2 | 1 | 0 | 8 | 2 | 4 | | | | |

used                                    unused
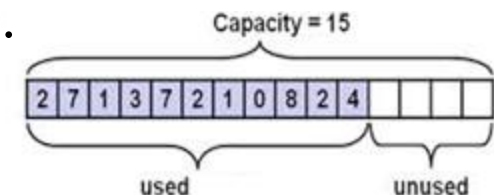
# Lists – Storage organization

- Data is stored at some location and each location has an address ( Figure: Storage Organization).
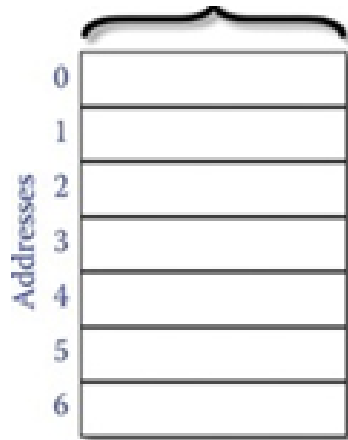


Figure : Storage Organization



Figure: List of Five painting records

- Each piece of data is located at an address ( Figure: List of Five painting records).

# Lists - Indices

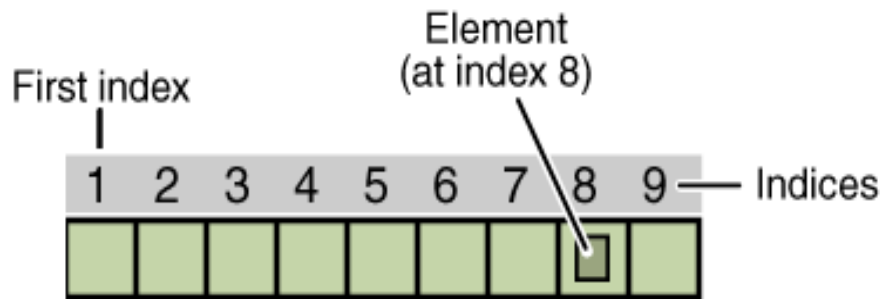- Addresses in lists are numbers (indices) as illustrated in Figure: List indices

.



Figure: List indices

# Adding Elements to list

- Lists can have new entries added anywhere, including in the middle, and their length can increase as needed.

   *Original List*
1. *Bohemisn Rhspdody – Queen*
2. *Stairway to Heaven – Led Zeppelin*
3. *Your Song – Elton John*
4. *Who made who – AC/DC*
5. *Sweet Child of Mine – Guns'n Roses*

   *List after adding a new element*
1. *Bohemisn Rhspdody – Queen*
2. *Every Breath you take - Police*
3. *Stairway to Heaven – Led Zeppelin*
4. *Your Song – Elton John*
5. *Who made who – AC/DC*
6. *Sweet Child of Mine – Guns 'n Roses*
7. *The Ghost of Tom Jord – Bruce Springsteen*

# Removing Elements from List

- Entries can be removed from lists.

### Original List
1. *Bohemisn Rhspdody – Queen*
2. *Every Breath you take - Police*
3. *Stairway to Heaven – Led Zeppelin*
4. *Your Song – Elton John*
5. *Who made who – AC/DC*
6. *Sweet Child of Mine – Guns 'n Roses*
7. *The Ghost of Tom Jord – Bruce Springsteen*

### List after deletion of two elements
1. *Bohemisn Rhspdody – Queen*
2. *Stairway to Heaven – Led Zeppelin*
3. *Your Song – Elton John*
4. *Who made who – AC/DC*
5. *Sweet Child of Mine – Guns 'n Roses*

# Real Time Applications of lists

We use lists all the time…

✓ Whenever I go shopping.

✓ The "To-Do" list is one of the most common ways that people use to organize their work or studies.
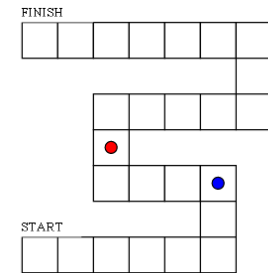
✓ Many simple children's board games are based on list-like structures.

# Arrays

- Array is a fixed sized list comprising of collections of variables, which we call elements.

- An array's elements are numbered using indices (Figure: Organization of array)

- The total number of elements in a given array is called as the length of an array.
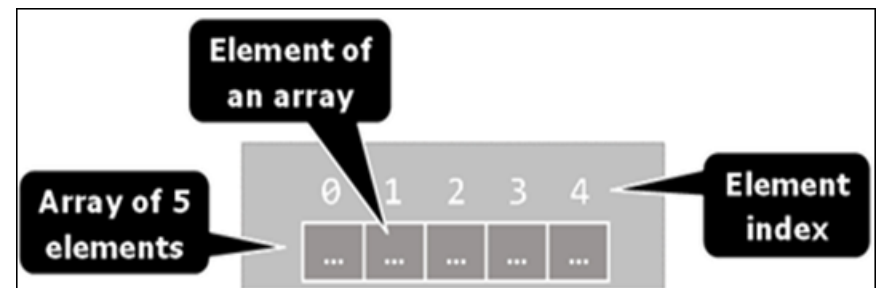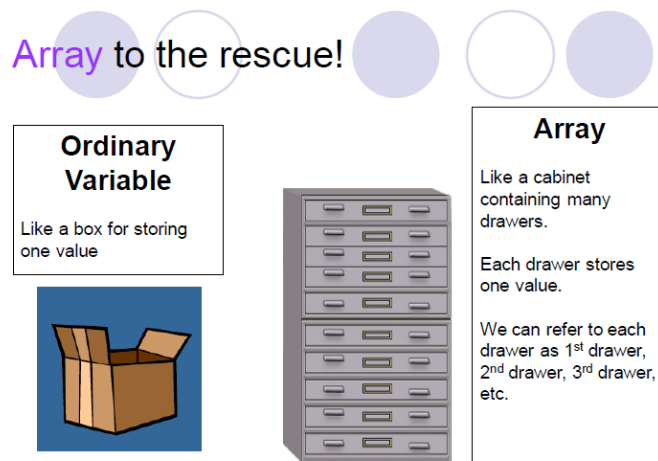


Figure: Organization of array

# Arrays *contd---*

- All elements of a given array are of the same type.

- This allows one to represent a group of similar elements as an ordered sequence and work on them as a whole.

- Arrays can be in different dimensions (Figure: A 1-D and 2-D array), but the most used are the one-dimensional and the two-dimensional arrays.

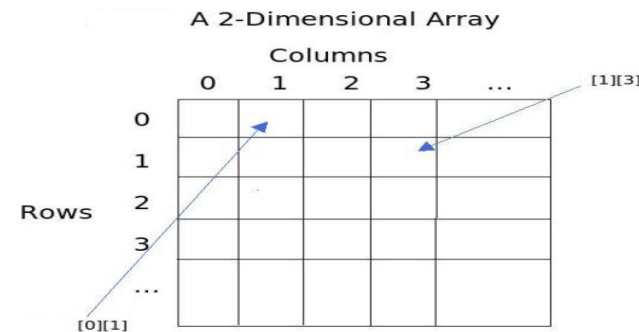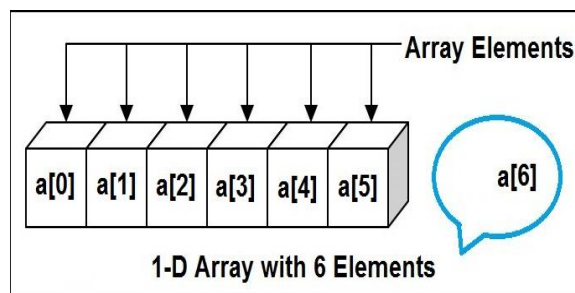- One-dimensional arrays are also called vectors.

Figure: A 1-D and 2-D array

# Array Properties

- *Arrays have three important properties.*

    - Arrays represent a group of related data.

      (e.g. -- temperature for the last five days, or stock prices for the last 30 days.)

    - All data within a single array must share the same data type.

    - The size of an array is fixed  once it is created.

# Declaration/Creation of an Array

- Allocate an array with length of specified elements of mentioned type.

- During the creation/declaration of the array one must set the total number of the elements in the brackets (a non-negative integer number), defining its length.

    Eg. number mark[6], string name[5]

- The array cannot change its location nor can it change its length.

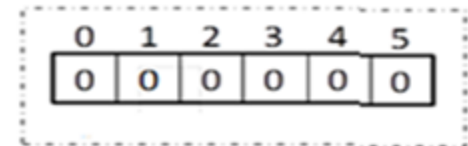- The array cannot expand to fill more nor can it shrink to take up less storage.



Figure: mark[6]

# Boundaries of an Array

- Arrays are by default zero-based, which means the enumeration of the elements starts from 0.

- The index of the first element is 0, and that of the second element is 1, and so on.

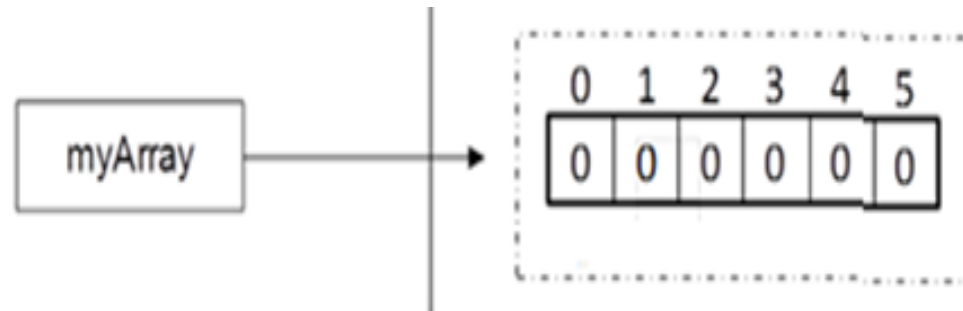- In an array of N elements, the last element has the index N-1.



Figure: Boundary of myArray is 5

# Array element access

- We access the array elements directly using their indices.

- Each element can be accessed through the name of the array and the element's index (consecutive number) placed in the brackets.

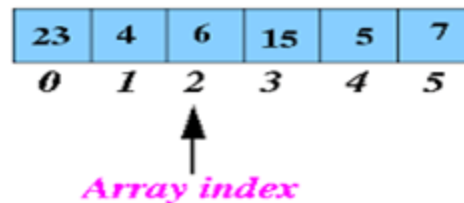- For Eg. numberArray[2] produces the 3$^{rd}$ element, 6.

| 23 | 4 | 6 | 15 | 5 | 7 |
|----|---|---|----|---|---|
| *0* | *1* | *2* | *3* | *4* | *5* |

↑
**Array index**

Figure: Elements of numberArray

# Array element access *contd---*

- As long as we keep track of the location of the array, we will be able to access every item in the array.

- The location of the array is known as the base address or the anchor, and is the memory address of the first item in the array.

- From a computer's perspective, the name of an array corresponds to the anchor of the array.

  - If the name of the array is known, then the anchor is known and all of the array items can be accessed by their index.

  - Access elements by….(anchor of array A)+(i-1)

✓ *Can you show how can the elements (4,1, and 6) be accessed from the array Arr=[1,2,3,4,5,6] ?*

# Answer......

4----------arr[3]

1----------arr[0]

6----------arr[5]

1.  *Given the array scores,*

| scores | 79 | 87 | 94 | 82 | 67 | 98 | 87 | 81 | 74 | 91 |

*Can you find*

✓ *How many elements are there?*

✓ *What is the fifth element in the array?*

✓ *What is the result of mean = (scores[0]+scores[1])/2*

2.  *Workout the average of Quiz1 and student0 with the following data?*

|          | Quiz1 | Quiz2 |
|----------|-------|-------|
| Student0 | 95    | 85    |
| Student1 | 89    | 80    |

# Answers

*How many elements are there?* <span style="color:red">*10*</span>

*What is the fifth element in the array?* <span style="color:red">*67*</span>

*What is the result of*

$$mean = (scores[0] + scores[1]) / 2;$$

<span style="color:red">*(79+87)/2 = 83*</span>

*Can you workout the average of Quiz1 and student0 with the following data???* <span style="color:red">*92,90*</span>

# Real time Applications of arrays

- A row of houses down one side of a street is organized like an array.



- Photo albums are similar to an array. They are "flip albums", with leaves for holding each photo.

- *To the array of animals given below show*


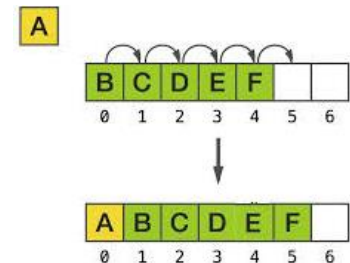
✓ *How to add*



✓ *How to delete*

# Answer….

Create an empty location by first performing delete
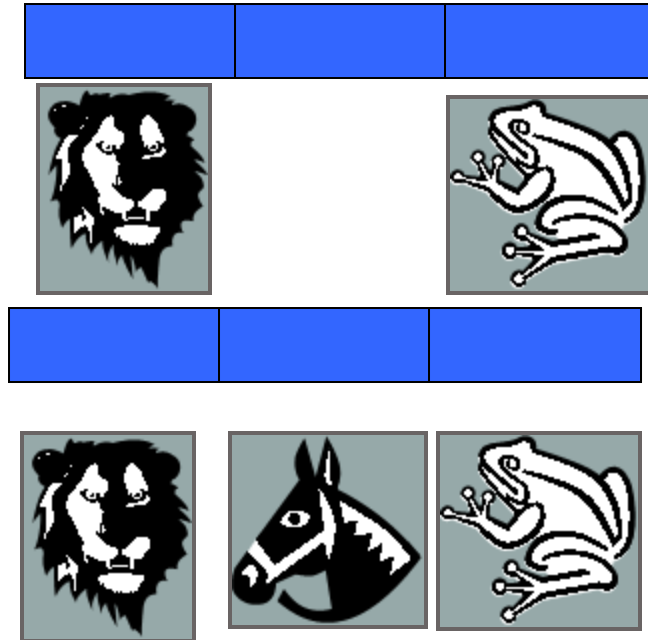


Insert the new element in the desired location
by moving the other elements

- *Consider the payroll task, that requires several pieces of information (hours worked, rate of pay, and regular and over time pay) for the employees of an organization.*

✓ *Can we organize such data as individual arrays, one for each of the piece of information?*

✓ *Is this an efficient approach?*

# Answer….

- *Consider the payroll task, that requires several pieces of information (hours worked, rate of pay, and regular and over time pay) for the employees of an organization.*

- ✓ *Can we organize such data as individual arrays, one for each of the piece of information?* <span style="color:red">*Yes*</span>

- ✓ *Is this an efficient approach?* <span style="color:red">*No because individual single D Arrays of say… 2 attributes do not capture the relationship between the two attributes… A Multi D array is a better choice*</span>

# Multi Dimensional Arrays

- Although it is useful to have data in one-dimensional arrays, it is sometimes useful to arrange data into rows and columns (two dimensional arrays), rows columns and ranks (three-dimensional), or even higher dimensionality.
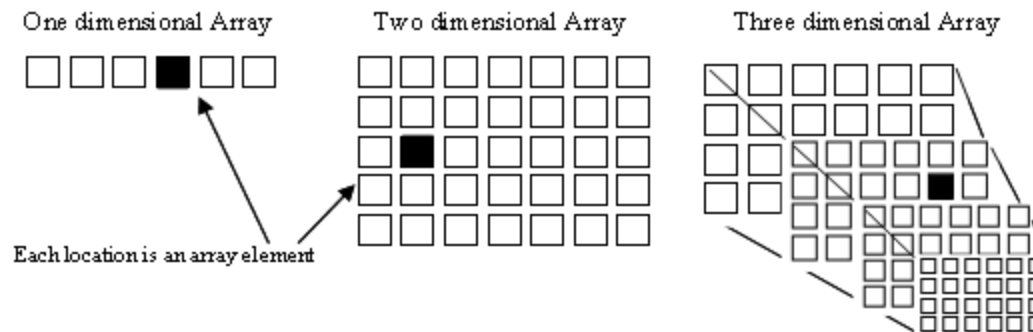


- A particular cell in a five-dimensional array may be accessed as follows:

$$Arr\_name[dim1][dim2][dim3][dim4][dim5].$$

# Multi Dimensional Arrays *contd---*

- For eg., if there are eight students and ten questions, and the answers can be stored in a two-dimensional array as given.

```
Students' Answers to the Questions:
          0 1 2 3 4 5 6 7 8 9
Student 0 A B A C C D E E A D
Student 1 D B A B C A E E A D
Student 2 E D D A C B E E A D
Student 3 C B A E D C E E A D
Student 4 A B D C C D E E A D
Student 5 B B E C C D E E A D
Student 6 B B A C C D E E A D
Student 7 E B E C C D E E A D
```

- Similarly, the distances between the cities can be represented using a two-dimensional array as given.

| Distance Table (in miles) | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Chicago | Boston | New York | Atlanta | Miami | Dallas | Houston |
| Chicago | 0 | 983 | 787 | 714 | 1375 | 967 | 1087 |
| Boston | 983 | 0 | 214 | 1102 | 1763 | 1723 | 1842 |
| New York | 787 | 214 | 0 | 888 | 1549 | 1548 | 1627 |
| Atlanta | 714 | 1102 | 888 | 0 | 661 | 781 | 810 |
| Miami | 1375 | 1763 | 1549 | 661 | 0 | 1426 | 1187 |
| Dallas | 967 | 1723 | 1548 | 781 | 1426 | 0 | 239 |
| Houston | 1087 | 1842 | 1627 | 810 | 1187 | 239 | 0 |

# Multi Dimensional Arrays *contd*---

- An array can be declared with multiple dimensions.

2 Dimensional

3 Dimensional



Three-dimensional array with twenty four elements
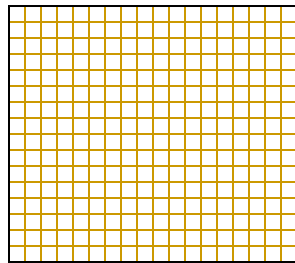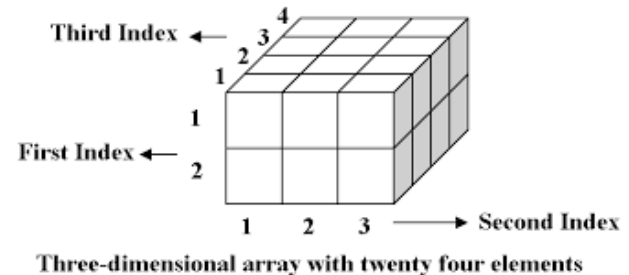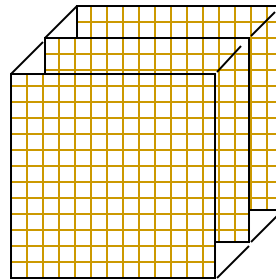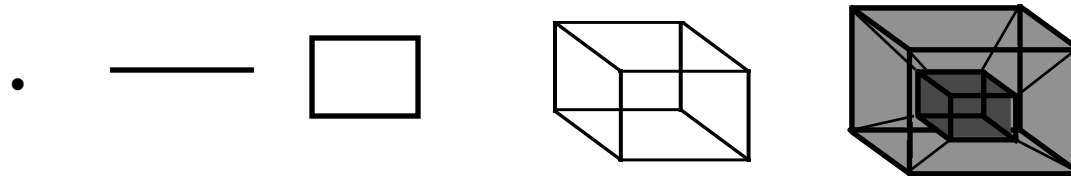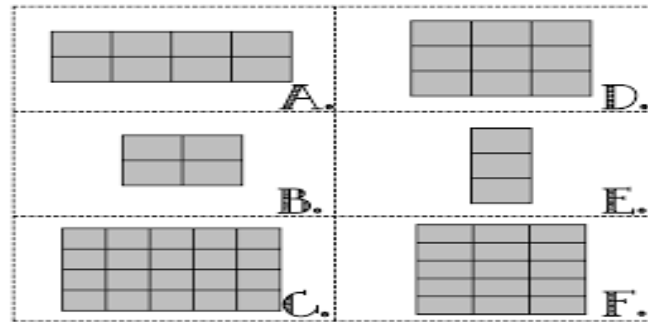
- Multiple dimensions get difficult to visualize graphically ( say, from a dot to a cube within a cube) .

✓ *Can you specify the dimension of the array and show how individual elements can be accessed?*



✓ *Can you show the pictorial representation of the memory associated with an array named a containing 10 integers?*

✓ *Can you model an array to store the datebook information (specify the dimension, number of indices, show how to access, modify a particular entry of the datebook)?*

# Answer….

A. 2X4   D. 3X3
B. 2X2   E. 1X3
C. 4X5   F. 5X3



One Dimensional array

Initialization    `arr a[] = new int [10];`

Value   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Index   a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9]

Model using one 2D array for storing days of a particular month
Locate an entry using 3 dimensions and update ……
E.g. cal[2015][10][3]

# Organizing data

- Humans hate disorder.

- We try and organise our lives, our cities, our homes and even sometimes our desks and our bedrooms.

- There are many ways of organising things, and we naturally organise different things in different ways.

- The organisation we use depends on the things we are trying to organise and often more importantly, what we actually wish to do with them once they are organised.

  - Addresses are organised into postal areas so that it is easier for letters to be delivered.

  - A bookshop organises its novels on the shelves usually alphabetically by author.

- The way we organise the things we must manipulate to achieve a task can make a massive difference to how quickly, easily or successfully the task can be done.

*Department of CSE*

# Game of *Spit Not So (Berlekamp et al, 1982)*

The game is played as follows by two players:

- *Write the nine words: SPIT NOT SO FAT FOP AS IF IN PAN* on separate pieces of cards.

- The cards are placed face up.

-  Each player takes it in turns to take a card.

- To win you must get all the cards that have the same letter.

    - For example if you had picked up the cards SPIT, IF and IN, you would

  have all the cards containing the letter I so would win.

-  If all the cards are taken with no one having all of a particular letter then the game is a draw.

*Department of CSE*

- *Play a few games to get then idea, then turn the page to see how a suitable organisation will help you play better.*

*Department of CSE*

- Before you start to play draw up the following grid, with the words that are on the cards written in the squares shown.

- The position of each word is important.

- They have been organised so that there is a single letter in common in each row, column and diagonal.

- All the F-words run along the bottom, for example.

| NOT | IN | PAN |
|-----|------|-----|
| SO | SPIT | AS |
| FOP | IF | FAT |

- Keep this hidden from your opponent – otherwise they may see the secret.

- Play the game exactly as before, except, now each time you pick up a card, put a cross in the square that corresponds to that word.

- Put a nought in the squares corresponding to the cards your opponent takes.

- For example, the above game will end with your table looking like the following:

| NOT | IN | PAN |
|---|---|---|
| **O** | | **X** |
| SO | SPIT | AS |
| **O** | **X** | |
| FOP | IF | FAT |
| **X** | **O** | **X** |

*Department of CSE*

# *What has been described?*

- Arrays and their importance.

- Operations on arrays(create, add, delete, locate).

- The properties of arrays.

- Multi-dimensional arrays.

*Credits*

- *Software Solution, Foundation for program design, Lewis and Loftus*

- *Computing Without Computers, A Gentle Introduction to Computer Programming, Data Structures and Algorithms, Version 0.15, Paul Curzon*

- *Google images*