CHAPTER 18 (DAY 3 After noon)

Chapters and Topics page

CHAPTER 20 (DAY 4 Fore noon)

STACKS

CHAPTER – 20

STACKS

INTRODUCTION

Stacks are a group of ordered elements where the items are put one above the other so that always an item can be placed on the **top** of a **stack** or an item can be removed from the **top** of the **stack**. There is only one end is open always and access to remove items or place the items from this end.

Trays in a cafeteria are a good example when the serving trays are **stacked**. Customers can pick up a tray from the top of the **stack**. It is not allowed to remove a tray from the middle or bottom of the tray **stack**. Trays can be replenished or placed always on the top of the **stack** of trays. Similar example can be found in the libraries where books are **stacked** one over the other. We can remove only the top book from the books **stack** or place new books on the top of the stack. Books are not allowed to be taken away in the middle or bottom nor placed. Many such real life example can be found, the elements of the **stack** may change but the basic operations remain the same.

STACK IMPLEMENTATION

Stack can be implemented using an array of **stacked** element in which case we need to know how many items we can **stack** at a time. That is the **stack** can have at the most the defined maximum number of elements which is implementation dependent. Once we declare the maximum elements, it stays for the entire duration until we compile the code with the new maximum set value. The items that can be **stacked** are integers, characters, floats, or any user defined data types like structures. When no items are placed on the **stack** or starting of the **stack** is the empty condition. While the **stack** is empty no items can be removed from the stack, but items can be added to an **empty stack**.