**Data Structure**

**In computer science, a data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structures are used in almost every program or software system. Data structures provide a ways to manage huge amounts of data efficiently.**

**Stack**

**A stack is a linear data structure in which all insertions and deletions of elements are done at one end (of a stack) – called the Top Of Stack (TOS). The addition (insertion) of new element or deletion of an existing element always takes place at the top of the stack.**

**A stack is a linear data structure in which items (elements) are added (inserted) or removed (deleted) only at one end – called the top of the stack. In other words, a stack is a list of elements in which an element (item) is inserted or deleted only at one end, called the top of the stack.**

**In our everyday life, we can see many examples of stacks, such as a stack of plates (dishes), a stack of folded towels, a stack of books in our library, a stack of papers in our printer tray, a stack of coins, and so on. In each example of stack, a new item is added only at the top of the stack. Similarly, an existing item is removed only from the top of the stack.**

**Stacks have many important applications (use) in computer science as below:**

* **For example, consider a stack of plates placed on the counter in a busy cafeteria. During the lunch hour, customers take plates off the top of the stack and waiters (employees) put the washed plates back on the top of the stack. The plate most recently put on the stack is the first one to be taken off. The bottom plate is the first one put on, and the last one to be used. This means, that the last plate to be added to the stack is the first place to be taken off. Therefore, we can say that stacks follow Last In First Out (LIFO) method.**
* **For example, consider a stack of books in our library, when we add a new book to a stack of books, we add it to the top of the stack. Similarly, when we remove a book from the stack, we generally remove it from the top of the stack. The book that is add most recently onto the stack is the first one to be removed. This means, that the last item to be added to a stack is the first item to be removed. Therefore, stacks are also called Last In First Out (LIFO) lists.**
* **The main application of stack is the evaluation of expressions in Polish Notation – Infix, Prefix and Postfix.**
* **Towers of Hanoi**
* **Backtracking**
* **Quicksort (instead of recursion)**
* **Runtime memory management.**

**A stack is an ordered collection of homogeneous data elements (data items of the same type) where the insertion and deletion operations take place at one end only. A stack is a restricted data structure, because only a small number of operations are performed on it.**

**Stack Program**

**Write a C program to implement a stack using an array.**

**OR**

**Write a C program that performs basic operations on a stack using an array.**

**OR**

**Write a C program for stack with the use of an array.**

**OR**

**Write a menu driven C program for the stack with the use of array.**

**#include <stdio.h>**

**#include <conio.h>**

**#include <stdlib.h>**

**#define MAXSIZE 5**

**int stack[MAXSIZE], top = -1;**

**void push();**

**void pop();**

**void peep();**

**void update();**

**void display();**

**void main()**

**{**

**int choice = 0;**

**do**

**{**

**clrscr();**

**printf("\n Main Menu (Basic Operations on Stack)");**

**printf("\n 1. Push a New Item at the Top of the Stack");**

**printf("\n 2. Pop an Item from the Top of the Stack");**

**printf("\n 3. Peep a Specified Item in the Stack");**

**printf("\n 4. Update (change) a Specified Item in the Stack");**

**printf("\n 5. Display Items in the Stack");**

**printf("\n 6. Exit");**

**printf("\n Enter your choice (from 1 to 6): ");**

**scanf("%d", &choice);**

**switch(choice)**

**{**

**case 1:**

**push();**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**peep();**

**break;**

**case 4:**

**update();**

**break;**

**case 5:**

**display();**

**break;**

**case 6:**

**exit(0);**

**break;**

**default:**

**printf("\n Invalid choice");**

**}**

**printf("\n Press any key to continue...");**

**getch();**

**} while(choice != 6);**

**}**

**void push()**

**{**

**int item = 0;**

**if(top == MAXSIZE-1)**

**{**

**printf("\n Stack is full (stack overflow).");**

**}**

**else**

**{**

**printf("\n Enter the element to be pushed at TOS: ");**

**scanf("%d", &item);**

**top = top + 1;**

**stack[top] = item;**

**}**

**}**

**void pop()**

**{**

**int item = 0;**

**if(top == -1)**

**{**

**printf("\n Stack is empty (stack underflow).");**

**}**

**else**

**{**

**item = stack[top];**

**top = top - 1;**

**printf("\n The item popped is %d.", item);**

**}**

**}**

**void peep()**

**{**

**int item = 0, pos = 0;**

**if(top == -1)**

**{**

**printf("\n Stack is empty (stack underflow).");**

**}**

**else**

**{**

**printf("\n Enter the position to read the element: " );**

**scanf("%d", &pos);**

**if(pos <= 0 || pos > top+1)**

**{**

**printf("\n Position out of range.");**

**}**

**else**

**{**

**item = stack[top-pos+1];**

**printf("\n The peeped item is %d.", item);**

**}**

**}**

**}**

**void update()**

**{**

**int item = 0, pos = 0, value = 0;**

**if(top == -1)**

**{**

**printf("\n Stack is empty (stack underflow).");**

**}**

**else**

**{**

**printf("\n Enter the position to update the element: " );**

**scanf("%d", &pos);**

**if(pos <= 0 || pos > top+1)**

**{**

**printf("\n Position out of range.");**

**}**

**else**

**{**

**item = stack[top-pos+1];**

**printf("\n Enter the new value: " );**

**scanf("%d", &value);**

**stack[top-pos+1] = value;**

**printf("\n The updated (changed) item is %d.", item);**

**}**

**}**

**}**

**void display()**

**{**

**int i = 0;**

**if(top == -1)**

**{**

**printf("\n Stack is empty (stack underflow).");**

**}**

**else**

**{**

**printf("\n Elements or items in the stack are: ");**

**for(i = top; i >= 0; i--)**

**{**

**printf("\n %d", stack[i]);**

**}**

**}**

**}**