Data Structure Using C

Ch. 03.02

Elementary Data Structure

03.02 Elementary Data Structure

Syllabus

- Introduction
- Stack
  - Definition
  - Operations on stack
  - Implementation of stacks using arrays
  - Function to insert an element into the stack
  - Function to delete an element from the stack
  - Function to display the items

Ch.5 Syllabus

- Recursion and stacks
- Evaluation of expressions using stacks
  - Postfix expressions
  - Prefix expression
- Queue
  - Introduction
  - Array implementation of queues
  - Function to insert an element into the queue
  - Function to delete an element from the queue

Ch.5 Syllabus

- Circular queue
  - Function to insert an element into the queue
  - Function for deletion from circular queue
  - Circular queue with array implementation
- Deques
- Priority queues

Stack :

- A stack is a linear Data structure where information is based on **Last In First Out(Lifo)**.
- All the deletion and insertion can take place at one end is called as TOP OF STACK(TOS).
- Static stack is implemented using array .
- Dynamic stack is implemented using nodes.

Example of Stack :

- Consider a stack of plates placed on the counter in café.
- During the dinner time customers take plates from the top of the stack and waiter puts plates on the top of the stack.
- Therefore stack operation are possible only in lifo.

Operation on stack

1. Push
2. Pop
3. Peep
4. Update

Stack : Push (Insertion)

- When we want to insert a new value to the stack, it is known as push operation.
- As in stack all the insertion are performed at the end of the stack which is known as TOS.
- Stack is changed for overflow and when it is full appropriate message is given

Pop (Deletion)

- When we want to delete any value or node from the stack it is known as POP operation.
- In stack all the deletion operation are performed at one end only which is known as TOS.
- After pop operation ,stack is checked whether it is empty or not.
- If no more value or node in stack ,it gives message.

Peep (View particular information)

- When we want to view information 0 value of any node it is known as peep operation.
- Once value or node is inserted in stack we can peep any information from stack.
- All the deletion are completed at one end which is known as TOS.
- After view operation there are no value or nodes in stack.

Update

- When we have inserted any value or node and in future if we want to change /edit any information associated at some location in the stack then it is known as update information.
- In this first of the entire node which is to be updated is searched in stack and if node is found.

Implementation of stack using array

- Stack array
  - Stack array is implemented by using array.
  - The array implementation technique is very simple and easy to implement.
  - As we have to use array in stack So in stack size is determined in the beginning only.

Function to **insert** an Element into the stack

- Insert an element into the stack using PUSH.

```
void push()
{ int item;
  if(top==MAXSIZE-1)
  {    printf("\nThe Stack is FULL");
       getch();
       exit(0);
  }else
  {    printf("\nEnter Element to be Inserted :");
       scanf("%d",&item);
       top=top+1;
       stack[top]=item;
  }
}
```

Function to **delete** an Element into the stack

- Delete an element from the stack using POP.

```
int pop()
{ int item;
  if(top==-1)
  {     printf("The stack is EMPTY");
        getch();
        exit(0);
  }
  else
  {     item=stack[top];
        top=top-1;
  }
  return(item);
}
```

Function to **display** Elements of the stack

- Display elements from the stack.

```
void traverse()
{  int i;
   if(top==-1)
   {     printf("\The Stack is EMPTY");
         getch();
         exit(0);
   }
   else
   {     for(i=top;i>=0;i--)
         {     printf("Traverse the element");
               printf("%d\n",stack[i]);
         }
   }
}
```

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAXSIZE 5
void push();
int pop();
void traverse();
int stack[MAXSIZE];
int top=-1;
void main()
{   int choice; char ch;
    do
    {       printf("\n1. PUSH");
            printf("\n2. POP");
            printf("\n3. TRAVERSE");
            printf("\nEnter Your Choice :");
            scanf("%d",&choice);
            switch(choice)
            {           case 1: push(); break;
                        case 2: printf("\nThe deleted element is %d",pop());break;
                        case 3: traverse();break;
                        default: printf("\nYou Entered Wrong Choice ");
            }
            printf("\nDo you Wish to continue(Y/N) ");
            fflush(stdin);
            scanf("%c",&ch);
    }while(ch=='y'||ch=='Y');
}
```

```
void push()
{
   int item;
   if(top==MAXSIZE-1)
   {   printf("\nThe Stack is FULL");
       getch();
       exit(0);
   }
   else
   {   printf("\nEnter The element to be inserted :");
       scanf("%d",&item);
       top=top+1;
       stack[top]=item;
   }
}
```

```
int pop()
{
  int item;
  if(top==-1)
  {
     printf("The stack is EMPTY");
     getch();
     exit(0);
  }
  else
  {  item=stack[top];
     top=top-1;
  }
  return(item);
}
```

```
void traverse()
{
   int i;
   if(top==-1)
   {   printf("\The Stack is EMPTY");
       getch();
       exit(0);
   }
   else
   {   for(i=top;i>=0;i--)
       {      printf("Traverse the element");
              printf("%d\n",stack[i]);
       }
   }
}
```

Stack Recursion

- Stacks are also important to supports nested or recursive function calls.
- A function is called recursively if a statement within body of function calls same function.
- Recursive function is based on LIFO mechanism.

Postfix Evaluation :

evaluation.exe

- The virtue of postfix notation is that it enables easy evaluation of expressions.
- To begin with, the need for parentheses is eliminated.
- Secondly, the priority of the operators is no longer relevant.
- The expression can be evaluated by making a left to right scan, stacking operands, and evaluating operators using as operands the correct elements from the stack and finally placing the result onto the stack.

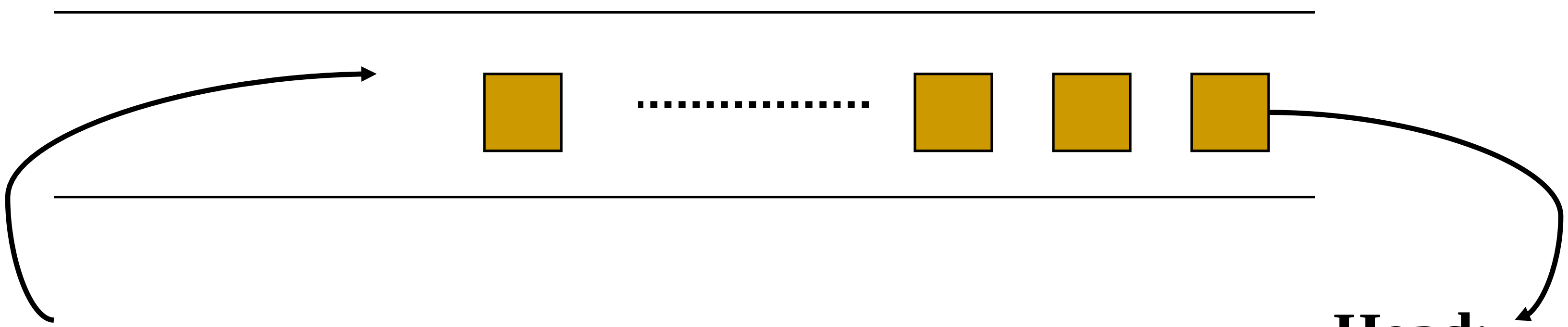
Prefix Evaluation :

- Prefix notation has seen wide application in Lisp s-expressions, where the brackets are required due to the arithmetic operators having variable arity.
- The Ambi programming postfix reverse polish notation is used in many stack-based programming language like PostScript and Forth, and the operating principle of certain calculators, notably from HP.

QUEUE :

- A queue is a data structure that enforces the **first-come first-serve** order, or equivalently the **first-in first-out** (FIFO) order.
- That is, the element that is inserted first into the queue will be the element that will deleted first, and the element that is inserted last is deleted last.
- Same as an array and stack ,queue is also a linear data structure.
- It can be used to represent a linear list.
- Insertion operation is known as INSERT(enqueue)
- Deletion operation is known as DELETE(dequeue).

A Graphic Model of a Queue



Tail:
All new items are added on this end

Head:
All items are deleted from this end

Types of queue

- 1. Static Queue
- 2. Dynamic Queue

- **Static Queue:**
  - Static queue is implemented by using arrays.
  - The array implementation technique is very simple and easy to implement.
  - But there is a problem while implementing static queue.
  - As we have use array in static queue.
  - The name itself suggest that it is static.
  - Size of array is fixed before starting the program.

- **Dynamic queue**
  - Dynamic queue is implemented by using pointers.
  - The link list implementation technique is very simple.
  - It is same as singly link list.
  - Here there is no problem of defining the size of the beginning as we are using pointers.
  - So dynamic queue is any size.

Operations on Queues

- **Insert(**item**)**: (also called enqueue)
  - It adds a new item to the tail of the queue
- **Remove( )**: (also called delete or dequeue)
  - It deletes the head item of the queue, and returns to the caller. If the queue is already empty, this operation returns NULL
- **getHead( ):**
  - Returns the value in the head element of the queue
- **getTail( ):**
  - Returns the value in the tail element of the queue
- **isEmpty( )**
  - Returns **true** if the queue has no items
- **size**( )
  - Returns the number of items in the queue

Circular Queue

- When a new item is inserted at the rear, the pointer to rear moves upwards.
- Similarly, when an item is deleted from the queue the front arrow moves downwards.
- After a few insert and delete operations the rear might reach the end of the queue and no more items can be inserted although the items from the front of the queue have been deleted and there is space in the queue.

Deques :

- It is a double-ended queue.
- Items can be inserted and deleted from either ends.
- More versatile data structure than stack or queue.
- E.g. policy-based application (e.g. low priority go to the end, high go to the front)

Priority Queues

- More specialized data structure.
- Similar to Queue, having front and rear.
- Items are removed from the front.
- Items are ordered by key value so that the item with the lowest key (or highest) is always at the front.
- Items are inserted in proper position to maintain the order.