Presidency University

School of Engineering

Department of Computer Science and Engineering

CSE2006: Data structures

Objectives:

To understand the concept of Linked List

Implementation of insert, delete display operations in Singly Linked List

Implementation of Stack Using Linked List

Topics Include:

Linked List, Defining of Self Referential Structures, Operations on Singly Linked list, Stacks and operations on stack, applications of stack

Program 1: Sinlgy Linked List Program

Program to illustrate singly linked list and operations performed on singly liked list

1. Inserting node at front end
2. Deleting node at front end
3. Display all nodes
4. Insert node at rear end-(at the end)
5. Delete node at rear end-(at the end)
6. Insert a node at particular position
7. Delete a node at particular position
8. Deleting a particular item.

[Operations 1,2,3,4,5 with main function to be written in record]

[Operations 6,7,8 are self-study topics for students]

```
#include<stdio.h>
#include<stdlib.h>
struct list
{
    int item;
    struct list *next;
};
typedef struct list *LIST;
```

```
LIST createlist()
{
    LIST newnode;
    newnode=(LIST)malloc(sizeof(struct list));
    return newnode;
}
LIST insertfront(LIST first, int cost)
{
    LIST temp;
    temp=createlist();
    temp->item=cost;
    temp->next=first;
    first=temp;
    return first;
}
LIST insertrear(LIST first, int cost)
{
    LIST temp,cur;
    temp=createlist();
    temp->item=cost;
    temp->next=NULL;
    if(first==NULL)
    {
            first=temp;
            return first;
    }
    else
    {
            cur=first;
            while(cur->next!=NULL)
            {
                    cur=cur->next;
            }
            cur->next=temp;
            return first;
    }
}

LIST insertpos(LIST first, int cost,int pos)
{
    LIST prev,cur,temp;
    int count=1;
    temp=createlist();
    temp->item=cost;
    if(pos==1)
    {
            temp->next=first;
            first=temp;
            return first;
        }
        prev=NULL;
        cur=first;
        while(cur!=NULL && count!=pos)
        {
            prev=cur;
            cur=cur->next;
            count ++;
        }
        if(count==pos)
        {
            prev->next=temp;
            temp->next=cur;
            return first;
        }
        if(cur==NULL)
            printf("Invalid Position");
        return first;
    }

    void display(LIST first)
    {
        LIST cur;
        if(first==NULL)
        {
            printf("No nodes to display\n");
            return;
        }
        cur=first;
        while(cur!=NULL)
        {
            printf("%d\t",cur->item);
            cur=cur->next;
        }
        printf("\n");
    }
    LIST deletefront(LIST first)
    {
        LIST cur;
        if(first==NULL)
        {
            printf("No nodes to delete\n");
        }
    else if(first->next==NULL)
    {
            printf("Deleted item is %d\n",first->item);
            free(first);
            first=NULL;
    }
    else
    {
            cur=first;
            first=first->next;
            printf("Deleted item is %d\n",cur->item);
            free(cur);
    }
    return first;
}
LIST deleterear(LIST first)//delete at end
{
    LIST cur,prev;
    if(first==NULL)
    {
            printf("No nodes to delete\n");
    }
    else if(first->next==NULL)
    {
            printf("Deleted item is %d\n",first->item);
            free(first);
            first=NULL;
    }
    else
    {
            cur=first;
            while(cur->next!=NULL)
            {
                    prev=cur;
                    cur=cur->next;
            }
            printf("The deleted item is %d\n",cur->item);
            free(cur);
            prev->next=NULL;
    }
    return first;
}
LIST deleteitem(LIST first, int val)
{
    LIST cur,prev;
    if(first==NULL)
    {
        printf("No nodes to delete\n");
        return first;
    }
    else if(first->item==val)
    {
        printf("Deleted item is %d\n",first->item);
        cur=first;
        first=first->next;
        free(cur);
        return first;
    }
    prev=NULL;
    cur=first;
    while(cur!=NULL && cur->item!=val)
    {
        prev=cur;
        cur=cur->next;
    }
    if(cur==NULL)
        printf("Item not found to delete\n");
    else
    {
        printf("Item deleted is %d\n",cur->item);
        prev->next=cur->next;
        free(cur);
    }
    return first;
}

LIST deletepos(LIST first, int pos)
{
    LIST cur,prev;
    int count=1;
    if(first==NULL)
    {
        printf("No nodes to delete\n");
        return first;
    }
    if(pos==1)
    {
        printf("Item deleted is %d\n",first->item);
        cur=first;
        first=first->next;
        free(cur);
        return first;
    }
    prev=NULL;
    cur=first;
    while(cur!=NULL && count!=pos)
    {
        prev=cur;
        cur=cur->next;
        count++;
    }
    if(cur==NULL)
        printf("Invalid position\n");
    else
    {
        printf("Item deleted is %d\n",cur->item);
        prev->next=cur->next;
        free(cur);
    }
    return first;
}

int main()
{
    LIST FIRST=NULL;
    int cost,choice,pos,val;
    for(;;)
    {
        printf("1.Insert Front\t2.Delete front \t3.Display\n");
        printf("4.Insert Rear\t5.Delete Rear\n");
        printf("6.Insert Pos\t7.Delete Pos\t8. Delete item\n");
        printf("\nEnter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
                case 1: printf("enter the cost\n");
                        scanf("%d",&cost);
                        FIRST=insertfront(FIRST,cost);
                        break;
                case 2: FIRST=deletefront(FIRST);
                        break;
                case 3: display(FIRST);
                        break;
                case 4: printf("enter the cost\n");
                        scanf("%d",&cost);
                        FIRST=insertrear(FIRST,cost);
                        break;
                case 5: FIRST=deleterear(FIRST);
                        break;
                case 6:printf("enter the cost\n");
                        scanf("%d",&cost);
                        printf("enter the position where the item to be inserted\n");
                        scanf("%d",&pos);
                        FIRST=insertpos(FIRST,cost,pos);
                        break;
                case 7: printf("enter the position to be deleted\n");
                        scanf("%d",&pos);
                        FIRST=deletepos(FIRST,pos);
                        break;
                case 8: printf("enter the item cost to be deleted\n");
                        scanf("%d",&cost);
                        FIRST=deleteitem(FIRST,cost);
                        break;
                default: return 0;
            }
        }
    }
```

Program 4: Stack Program using Linked List [Self STUDY]

Program to implement stack operations using linked list

```
#include<stdio.h>
#include<stdlib.h>
#define STACK_SIZE 5
struct list
{
        int item;
        struct node *next;
};
typedef struct list *LIST;
int count=0;
LIST createlist()
{
        LIST newnode;
        newnode= (LIST) malloc (sizeof(struct list));
        return (newnode);
}

LIST push(LIST first, int cost)
{
     LIST temp;
        if(count<STACK_SIZE)
        {
                temp = createlist();
                temp->item=cost;
                temp->next=first;
                first=temp;
                count++;
        }
        else
        {
                printf("Stack is full insertion not possible\n");
        }
        return (first);
}

LIST pop(LIST first)
{
        LIST cur;
        if(first==NULL)
        {
                printf("Stack empty\n");
                return(first);
        }
        if(first->next==NULL)
        {
                printf("Item poped is %d\n",first->item);
                first=NULL;
                count--;
                return(first);
        }
        cur=first;
        printf("Item poped is %d\n",cur->item);
        first=first->next;
        free(cur);
        count--;
        return(first);
}

void display(LIST first)
{
        LIST cur;
        if(first == NULL)
        {
                printf("Stack is empty\n");
        }
        else
        {
                cur=first;
                printf("stack contains \n");
                while(cur!=NULL)
                {
                        printf("%d\n",cur->item);
                        cur=cur->next;
                }
        }
}
```

```
int main()
{
 int ch,cost;
 LIST first = NULL, temp;
          for(;;)
          {
            printf("\nEnter the choice \n 1. for PUSH \n 2. POP \n 3. Display\n 4. exit \n");
            scanf("%d", &ch);
               switch(ch)
                    {
                              case 1: printf("Enter the item cost\n");
                                        scanf("%d",&cost);
                                        first = push(first,cost);
                                        break;
                              case 2: first = pop(first);
                                        break;
                              case 3: display(first);
                                        break;
                              case 4: exit(0);
                              default: printf("invalid option \n");
                    }
          }
return(0);
}
```