219A1102

Program 1: A menu driven program to implement Stack Operations.

#include<stdio.h>

 int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{

    top=-1;

    printf("\n Enter the size of STACK[MAX=100]:");

    scanf("%d",&n);

    printf("\n\t STACK OPERATIONS USING ARRAY");

    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

    do

    {

        printf("\n Enter the Choice:");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

            {

                push();

                break;

            }

            case 2:

            {

                pop();

                break;

            }

            case 3:

            {

                display();

                break;

            }

            case 4:

            {

                printf("\n\t EXIT POINT ");

                break;

            }

            default:

            {

                printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");

            }

        }

    }

    while(choice!=4);

    return 0;

}

void push()

{

    if(top>=n-1)

    {

        printf("\n\tSTACK is over flow");

    }

    else

    {

        printf(" Enter a value to be pushed:");

        scanf("%d",&x);

        top++;

        stack[top]=x;

    }

}

void pop()

{

    if(top<=-1)

    {

        printf("\n\t Stack is under flow");

    }

    else

    {

        printf("\n\t The popped elements is %d",stack[top]);

        top--;

    }

}

void display()

{

    if(top>=0)

    {

        printf("\n The elements in STACK \n");

        for(i=top; i>=0; i--)

            printf("\n%d",stack[i]);

        printf("\n Press Next Choice");

    }

    else

    {

        printf("\n The STACK is empty");

    }

}

Program 2 : A program for Infix to Post-fix Conversion.

#include<stdio.h>

char stack[20];

int top = -1;

void push(char x)

{

    stack[++top] = x;

}

char pop()

{

    if(top == -1)

        return -1;

    else

        return stack[top--];

}

int priority(char x)

{

    if(x == '(')

        return 0;

    if(x == '+' || x == '-')

        return 1;

    if(x == '\*' || x == '/')

        return 2;

}

main()

{

    char exp[20];

    char \*e, x;

    printf("Enter the expression :: ");

    scanf("%s",exp);

    e = exp;

    while(\*e != '\0')

    {

        if(isalnum(\*e))

            printf("%c",\*e);

        else if(\*e == '(')

            push(\*e);

        else if(\*e == ')')

        {

            while((x = pop()) != '(')

                printf("%c", x);

        }

        else

        {

            while(priority(stack[top]) >= priority(\*e))

                printf("%c",pop());

            push(\*e);

        }

        e++;

    }

    while(top != -1)

    {

        printf("%c",pop());

    }

}

Program 3 : A program to evaluate Postfix Expression.

#include<stdio.h>

int stack[20];

int top = -1;

void push(int x)

{

        stack[++top] = x;

}

int pop()

{

        return stack[top--];

}

int main()

{

        char exp[20];

        char \*e;

        int n1,n2,n3,num;

        printf("Enter the expression :: ");

        scanf("%s",exp);

        e = exp;

        while(\*e != '\0')

        {

                if(isdigit(\*e))

                {

                        num = \*e - 48;

                        push(num);

                }

                else

                {

                        n1 = pop();

                        n2 = pop();

                        switch(\*e)

                        {

                                case '+':

                                {

                                        n3 = n1 + n2;

                    break;

                                }

                                case '-':

                                {

                                        n3 = n2 - n1;

                                        break;

                                }

                                case '\*':

                                {

                                        n3 = n1 \* n2;

                                        break;

                                }

                                case '/':

                                {

                                        n3 = n2 / n1;

                                        break;

                                }

                        }

                        push(n3);

                }

                e++;

        }

        printf("\nThe result of expression %s  = %d\n\n",exp,pop());

        return 0;

}

Program 4 : A menu driven program to implement Circular Queue

#include<stdio.h>

#include<conio.h>

#define Max 10

int queue[Max];

int rear = -1 , front = -1 ;

void insert ();

void display();

void delete();

void main()

{

int opt;

do

{

printf("\n1.Insert\n2.Delete\n3.Display\n4.exit \n");

scanf("%d",&opt);

switch(opt)

{

case 1 : insert();

break;

case 2 : delete();

break;

case 3 : display();

break;

}

}while(opt!=4);

}

void insert()

{

int val;

printf("\nenter value to be inserted in the queue \n");

scanf("%d",&val);

if(rear==Max-1 && front==0)

{

printf(" overflow");

}

else if(rear == front-1)

{

printf(" overflow");

}

else if(front==-1 && rear==-1)

{

front=0;

rear=0;

queue[rear]=val;

}

else if(rear==Max-1 && front!=0)

{

rear = 0 ;

queue[rear]=val ;

}

else

{

rear++;

queue[rear]=val;

}

display();

}

void delete()

{

if(front == -1)

{

printf("\n underflow");

}

else if(front == rear)

{

printf("\n deleted element is : %d\n",queue[front]);

front = -1 ;

rear = -1 ;

}

else if(front == Max-1)

{

printf("\n deleted element is : %d\n",queue[front]);

front = 0;

}

else

{

printf("\n deleted element is : %d\n",queue[front]);

front ++;

}

display();

}

void display()

{

int i;

if(front>rear)

{

for(i = front ; i < Max ;i++)

{

printf(" \t%d",queue[i]);

}

for(i=0;i<=rear;i++)

{

printf(" \t%d",queue[i]);

}

}

else

{

for(i=front;i<=rear;i++)

{

printf(" \t%d",queue[i]);

}

}

}

**Practical no. 5**

AIM- To write a menu-driven program to implement Singly Linked List Operations

PROGRAM-

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*t,\*x,\*y;

struct node \*s=NULL;

struct node\* insb()

{

int dt;

printf("Enter element data:");

scanf("%d",&dt);

struct node \*t=(struct node\*)malloc(sizeof(struct node\*));

t->data=dt;

t->next=s;

s=t;

return s;

}

void inse()

{

int dt;

printf("Enter element data:");

scanf("%d",&dt);

struct node \*t=(struct node\*)malloc(sizeof(struct node\*));

t->data=dt;

x=s;

while(x->next!=NULL)

x=x->next;

x->next=t;

t->next=NULL;

}

void insp()

{

int dt,p;

printf("Enter element data:");

scanf("%d",&dt);

struct node \*t=(struct node\*)malloc(sizeof(struct node\*));

t->data=dt;

printf("Enter position :");

scanf("%d",&p);

p--;

y=s;

while(p>0)

{

x=y;

y=y->next;

p--;

}

x->next=t;

t->next=y;

}

void delb()

{

if(s==NULL)

printf("No node!");

else

{

t=s;

s=s->next;

free(t);

}

}

void dele()

{

if(s==NULL)

printf("No node!");

else

{

x=s;

while(x->next!=NULL)

{

y=x;

x=x->next;

}

y->next=NULL;

free(x);

}

}

void delp()

{ int p;

if(s==NULL)

printf("No node!");

else

{

printf("Enter position :");

scanf("%d",&p);

p--;

y=s;

while(p>0)

{

x=y;

y=y->next;

p--;

}

x->next=y->next;

free(y);

}

}

void disp()

{

x=s;

while(x->next!=NULL)

{

printf("\n%d",x->data);

x=x->next;

}

printf("\n%d",x->data);

}

void search()

{

int c=0,e;

x=s;

printf("Enter element to be searched:");

scanf("%d",&e);

for(x=s;x->next!=NULL;x=x->next)

{

if(x->data==e)

{

printf("Element found at position:%d",c+1);

break;

}

c++;

}

if(x->next==NULL)

{

if(x->data==e)

printf("Element found at position:%d",c+1);

else

printf("Element not found!");

}

}

void main()

{

int o,ch;

do

{

printf("MENU\n1.Add at beginning\n2.Add at end\n3.Add at position\n4.Delete from beginning\n5.Delete at end\n6.Delete at position\n7.Display\n8.Search");

printf("\n Enter choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: s=insb();

disp();

break;

case 2: inse();

disp();

break;

case 3: insp();

disp();

break;

case 4: delb();

disp();

break;

case 5: dele();

disp();

break;

case 6: delp();

disp();

break;

case 7: disp();

break;

case 8: search();

break;

}

printf("Enter 1 to continue:");

scanf("%d",&o);

}while(o==1);

}

**Practical no. 6**

AIM-To write a program to implement Stack using Linked List

PROGRAM-

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*n,\*top,\*x;

struct node\* push()

{

int dt;

struct node \*t=(struct node\*)malloc(sizeof(struct node\*));

printf("Enter element data:");

scanf("%d",&dt);

t->data=dt;

t->next=top;

top=t;

return top;

}

struct node\* pop()

{

x=top;

top=top->next;

free(x);

return x;

}

void disp()

{

for(x=top;x->next!=NULL;x=x->next)

printf("\n%d",x->data);

printf("\n%d",x->data);

}

void main()

{

int ch,o=1;

do

{

printf("\nMENU\n1.Push\n2.Pop\n3.Display\n");

printf("Enter option:");

scanf("%d",&ch);

switch(ch)

{

case 1: top=push();

disp();

break;

case 2: x=pop();

printf("Element deleted is:%d\n",x->data);

disp();

break;

case 3: disp();

break;

}

printf("\nEnter 1 to continue:");

scanf("%d",&o);

}while(o==1);

getch();

}

**PRACTICAL NO. 7**

AIM: To write a program for menu-driven program to implement Binary Tree Operations.

a. Insert b. Traversal c. Search

PROGRAM:

#include<stdio.h>

#include<stdlib.h>

struct tree

{

int data;

struct tree \*l,\*r;

}\*root,\*q,\*t,\*p;

void insert()

{

int e;

struct tree \*n=(struct tree\*)malloc(sizeof(struct tree\*));

printf("Enter data element:");

scanf("%d",&e);

n->data=e;

n->l=NULL;

n->r=NULL;

if(root==NULL)

root=n;

else

{

p=root;

q=root;

while(p!=NULL)

{

q=p;

if(p->data>e)

p=p->l;

else

p=p->r;

}

if(q->data>e)

q->l=n;

else

q->r=n;

}

}

void ino(struct tree \*p)

{

if(p==NULL)

return;

else

{

ino(p->l);

printf("%5d",p->data);

ino(p->r);

}

}

void preo(struct tree \*p)

{

if(p==NULL)

return;

else

{

printf("%5d",p->data);

preo(p->l);

preo(p->r);

}

}

void posto(struct tree \*p)

{

if(p==NULL)

return;

else

{

posto(p->l);

posto(p->r);

printf("%5d",p->data);

}

}

void search()

{

int ele;

printf("Enter element to be searched:");

scanf("%d",&ele);

p=root;

while(p!=NULL)

{

if(p->data==ele)

{

printf("Element found!");

break;

}

else

{

if(ele>p->data)

p=p->r;

else

p=p->l;

}

}

if(p==NULL)

printf("Element not foumd!");

}

void main()

{

int o,ch;

do

{

printf("\nMENU\n1.Insert\n2.Inorder\n3.Preorder\n4.Postorder\n5.Search");

printf("\nEnter your choice:");

scanf("%d",&o);

switch(o)

{

case 1:insert();

break;

case 2:ino(root);

break;

case 3:preo(root);

break;

case 4:posto(root);

break;

case 5:search();

break;

}

printf("\n Enter 1 to continue:");

scanf("%d",&ch);

}while(ch==1);

}

**Practical no. 8**

**AIM:**To write a program for implementation of Merge Sort.

**PROGRAM:**

#include<stdio.h>

void main()

{

int a[10],b[10],n,m,c[20],i=0,j=0,k=0;

printf("Enter number of elements in array 1 & 2:");

scanf("%d%d",&n,&m);

printf("Enter array 1:");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

printf("Enter array 2:");

for(j=0;j<m;j++)

scanf("%d",&b[j]);

i=0;

j=0;

while((i<n)&&(j<m))

{

if(a[i]<=b[j])

{

c[k]=a[i];

k++;

i++;

}

else

{

c[k]=b[j];

k++;

j++;

}

}

if((i==n)&&(j!=m))

{

for(;j<m;j++,k++)

c[k]=b[j];

}

else if((j==m)&&(i!=n))

{

for(;i<n;i++,k++)

c[k]=a[i];

}

printf("Sorted array is:");

for(k=0;k<(n+m);k++)

printf("\t%d",c[k]);

}

**Practical no. 9**

**AIM:**To write a program for performing Binary Search

**PROGRAM**:

#include<stdio.h>

void main()

{

int a[10],e,p=0,n,i,j,l,h,m;

printf("Enter the number of elements in the array:");

scanf("%d",&n);

printf("Enter elements in array:");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

l=0;

h=n-1;

printf("\nEnter element to be searched:");

scanf("%d",&e);

while(l<=h)

{

m=(l+h)/2;

if(a[m]==e)

{

printf("Element found at position:%d",m+1);

break;

}

else if(a[m]>e)

h=m-1;

else

l=m+1;

}

if(l>h)

printf("Element not found!");

}

**OUTPUT:**

Enter the number of elements in the array:6

Enter elements in array:1

3

5

**Practical No. 10**

AIM: To write a menu-driven program for DFS and BFS.

PROGRAM:

#include<stdio.h>

#include<stdlib.h>

#define MAX 20

typedef struct Q

{

int data[MAX];

int R,F;

}Q;

typedef struct node

{

struct node \*next;

int vertex;

}node;

void enqueue(Q \*,int);

int dequeue(Q\*);

int empty(Q\*);

int full(Q\*);

void BFS(int);

void readgraph();

void insert(int vi, int vj);

void DFS(int i);

int visited[MAX];

node \*G[20];

int n;

void main()

{

int i,op;

do

{

printf("\n\n1.Create\n2.BFS\n3.DFS\n4.Quit");

printf("\nEnter Your Choice:");

scanf("%d",&op);

switch(op)

{

case 1:

readgraph();

break;

case 2:

printf("\nStarting Node No. :");

scanf("%d",&i);

BFS(i);

break;

case 3:

for(i=0;i<n;i++)

visited[i]=0;

printf("\nStarting Node No. :");

scanf("%d",&i);

DFS(i);

break;

}

}while(op!=4);

}

void BFS(int v)

{

int w,i,visited[MAX];

Q q;

node \*p;

q.R=q.F=-1;

for(i=0;i<n;i++)

visited[i]=0;

enqueue(&q,v);

printf("\nVisit\t%d",v);

visited[v]=1;

while(!empty(&q))

{

v=dequeue(&q);

for(p=G[v];p!=NULL;p=p->next)

{

w=p->vertex;

if(visited[w]==0)

{

enqueue(&q,w);

visited[w]=1;

printf("\nvisit\t%d",w);

}

}

}

}

void DFS(int i)

{

node \*p;

printf("\n%d",i);

p=G[i];

visited[i]=1;

while(p!=NULL)

{

i=p->vertex;

if(!visited[i])

DFS(i);

p=p->next;

}

}

int empty(Q \*P)

{

if(P->R==-1)

return(1);

return(0);

}

int full(Q \*P)

{

if(P->R==MAX-1)

return(1);

return(0);

}

void enqueue(Q \*P,int x)

{

if(P->R==-1)

{

P->R=P->F=0;

P->data[P->R]= x;

}

else

{

P->R=P->R+1;

P->data[P->R]=x;

}

}

int dequeue(Q \*P)

{

int x;

x=P->data[P->F];

if(P->R==P->F)

{

P->R=-1;

P->F=-1;

}

else

P->F=P->F+1;

return(x);

}

void readgraph()

{

int i,vi,vj,no\_of\_edges;

printf("\nEnter no. of Vertices:");

scanf("%d",&n);

for(i=0;i<n;i++)

G[i]=NULL;

printf("\nEnter no of edges:");

scanf("%d",&no\_of\_edges);

for(i=0;i<no\_of\_edges;i++)

{

printf("\nEnter an edge (u,v):");

scanf("%d%d",&vi,&vj);

insert(vi,vj);

insert(vj,vi);

}

}

void insert(int vi,int vj)

{

node \*p,\*q;

q=(node \*)malloc(sizeof(node));

q->vertex=vj;

q->next=NULL;

if(G[vi]==NULL)

G[vi]=q;

else

{

p=G[vi];

while(p->next!=NULL)

p=p->next;

p->next=q;

}

}