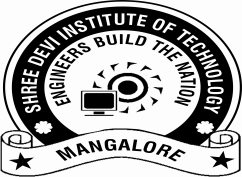
**SHREE DEVI INSTITUTE OF TECHNOLOGY**

Kenjar, Mangalore-574142

****

**DATA STRUCTURE LABORATORY**

**Lab manual- 22mcal18**

**Department of Master of computer Application**

**I Semester**

**2022-2023**

|  |
| --- |
| **Data Structures with Algorithms Lab** |
| 1. Implement a Program in C for converting an Infix Expression to Postfix Expression. |
| 1. Implement a Program in C for evaluating an Postfix Expression. |
| 1. Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations: a. Insert b. Delete c. Display |
| 1. Write a C program to simulate the working of a singly linked list providing the following operations: a. Display & Insert b. Delete from the beginning/end c. Delete a given element |
| 5. Write a C program to Implement the following searching techniques a. Linear Search b. Binary Search. |
| 6. Write a C program to implement the following sorting algorithms using user defined functions: a. Bubble sort (Ascending order) b. Selection sort (Descending order). |
| 7. Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm ( C programming) |
| 8. From a given vertex in a weighted connected graph, find shortest paths to other vertices Using Dijkstra's algorithm (C programming) |

**1.Implement a Program in C for converting an Infix Expression to Postfix Expression**

#include<stdio.h>

#include<ctype.h>

char stack[100];

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;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

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());

}return 0;

}

**2.Implement a program in c for evaluating an postfix expression**

#include<stdio.h>

#include<conio.h>

#define MAX 20

int IsOperator(char c)

{

switch(c)

{

case '$':

case '^':

case '\*':

case '/':

case '+':

case '-':

case '(':

case ')': return 1;

default : return 0;

}

}

int eval(int b, int a,char c)

{

int tmp=1;

switch(c)

{

case'+': return b+a;

case '-': return b-a;

case'/': return b/a;

case'\*': return b\*a;

case'^':

case'$':/\*b^a\*/

for(;a>=1;a--)tmp\*=b;

return tmp;

default: return 0;

}

}

int main()

{

char p[20],c;

int stack[MAX],a,b,l,i,top=-1;

printf("\n Enter a valid postfix expression without spaces:\n");

scanf("%s",p);

for(l=0;p[l]!='\0';l++); //get count of chars

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

{

c=p[i];

if(c>='0' && c<='9')

stack[++top] = (c-48);

else if(IsOperator(c)==1)

{

a=stack[top--];

b=stack[top--];

stack[++top]=eval(b,a,c);

}

else

{

printf("\n Invalid expression….");

getch();

return;

}

}

printf("Answer is %d",stack[top]);

getch();

return 0;

}

**3.** **Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations: a. Insert b. Delete c. Display**

#include <stdio.h>

#define MAX 50

void insert();

void delete();

void display();

int queue\_array[MAX];

int rear = - 1;

int front = - 1;

main()

{

int choice;

while (1)

{

printf("1.Insert element to queue \n");

printf("2.Delete element from queue \n");

printf("3.Display all elements of queue \n");

printf("4.Quit \n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

printf("Wrong choice \n");

} /\* End of switch \*/

} /\* End of while \*/

} /\* End of main() \*/

void insert()

{

int add\_item;

if (rear == MAX - 1)

printf("Queue Overflow \n");

else

{

if (front == - 1)

/\*If queue is initially empty \*/

front = 0;

printf("Inset the element in queue : ");

scanf("%d", &add\_item);

rear = rear + 1;

queue\_array[rear] = add\_item;

}

} /\* End of insert() \*/

void delete()

{

if (front == - 1 || front > rear)

{

printf("Queue Underflow \n");

return ;

}

else

{

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

front = front + 1;

}

} /\* End of delete() \*/

void display()

{

int i;

if (front == - 1)

printf("Queue is empty \n");

else

{

printf("Queue is : \n");

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

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

printf("\n");

}

} /\* End of display() \*/

**4.Write a C program to simulate the the working of a singly linked list providing the following operations a.Display & Insert b.Delete from the beginning/end c.Delete given element**

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

// A structure to represent a stack

struct StackNode {

int data;

struct StackNode\* next;

};

struct StackNode\* newNode(int data)

{

struct StackNode\* stackNode =

(struct StackNode\*)

malloc(sizeof(struct StackNode));

stackNode->data = data;

stackNode->next = NULL;

return stackNode;

}

int isEmpty(struct StackNode\* root)

{

return !root;

}

void push(struct StackNode\*\* root, int data)

{

struct StackNode\* stackNode = newNode(data);

stackNode->next = \*root;

\*root = stackNode;

printf("%d pushed to stack\n", data);

}

int pop(struct StackNode\*\* root)

{

if (isEmpty(\*root))

return INT\_MIN;

struct StackNode\* temp = \*root;

\*root = (\*root)->next;

int popped = temp->data;

free(temp);

return popped;

}

int peek(struct StackNode\* root)

{

if (isEmpty(root))

return INT\_MIN;

return root->data;

}

int main()

{

struct StackNode\* root = NULL;

push(&root, 10);

push(&root, 20);

push(&root, 30);

printf("%d popped from stack\n", pop(&root));

printf("Top element is %d\n", peek(root));

return 0;

}

**5.Write a C program to Implement the following searching techniques a.Linear Search b.Binay Search**

#include<stdio.h>

#include<conio.h>

int a[25],i,j,n;

void display()

{

printf("Array elements are : \n");

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

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

}

void sort()

{

int temp;

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

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

{

if(a[i]>a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

display();

}

void linear(int item)

{

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

{

if(a[i]==item)

{

printf("\nElement found at position %d\n",i+1);

return;

}

}

printf("\nElement not found\n");

}

void binary(int item)

{

int hb=n-1,lb=0,mid;

sort();

while(lb<=hb)

{ mid=(hb+lb)/2;

if(a[mid]==item)

{

printf("\nElement found at %d\n",mid+1);

return;

}

else if(item<a[mid])

hb=mid-1;

else

lb=mid+1;

}

printf("\nElement not found\n");

}

int main()

{

int ch,item;

printf("Enter the array size : ");

scanf("%d",&n);

printf("Enter the array element\n");

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

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

do

{

printf("\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\n 1.Linear search\n 2.Binary search\n 3.Exit\n");

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("Enter the searching element : \n");

scanf("%d",&item);

display();

linear(item);

break;

case 2: printf("Enter the searching element : \n");

scanf("%d",&item);

binary(item);

break;

case 3: printf("Exit");

break;

default:printf("Invalid");

} }while(ch!=3);return 0;}

**6.Write a C program to implement the following sorting algorithms using user defined functions a.Push an Element on to stack(ASCENDING ORDER) B.Selection sort(Decending order)**

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

/\*

void selectSort(int arr[], int n) //this is regular sort for ascend

{

int i, j, min\_idx;

// One by one move boundary of unsorted subarray

for (i = 0; i < n-1; i++)

{

// Find the minimum element in unsorted array

min\_idx = i;

for (j = i+1; j < n; j++)

if (arr[j] < arr[min\_idx])

min\_idx = j;

// Swap the found minimum element with the first element

swap(&arr[min\_idx], &arr[i]);

}

}

\*/

void selectSort(int arr[], int n)

{

int i, j, max\_idx;

// One by one move boundary of unsorted subarray

for (i = n-1; i > 0; i--)

{

// Find the minimum element in unsorted array

max\_idx = 0;

for (j = 1; j <= i; j++)

if (arr[j] < arr[max\_idx])

max\_idx = j;

// Swap the found minimum element with the first element

swap(&arr[max\_idx], &arr[i]);

}

}

void bubbleSort(int arr[], int n)

{

int i, j;

for (i = 0; i < n-1; i++)

// Last i elements are already in place

for (j = 0; j < n-i-1; j++)

if (arr[j] > arr[j+1])

swap(&arr[j], &arr[j+1]);

}

void printArray(int arr[], int size)

{

int i;

printf("The elements in array are \n");

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

{

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

}

}

void readArray(int arr[], int size)

{

int i;

printf("\n Enter elements is array : \n");

/\* initialize elements of array n to data \*/

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

{

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

}

}

int main ()

{

int n[ 20 ]; /\* n is an single dimension array of 20 integers OR\*/

int i,j, size, ch=0;

printf("\n Enter the max num of elements is array : ");

scanf("%d", &size);

do

{

printf("\n 1. Selection sort(Descend) \n 2. Bubble sort(Ascend) \n 3. Exit \n");

scanf("%d", &ch);

switch(ch)

{

case 1:

readArray(n,size);

selectSort(n, size);

printArray(n, size);

break;

case 2:

readArray(n,size);

bubbleSort(n,size);

printArray(n, size);

break;

}

}while(ch !=3);

return 0;

}

**7.Find Minimum cost spanning Tree of a given undirected graph using kruskal's algorithm(c programing)**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,cost[9][9],parent[9];

int find(int);

int uni(int,int);

void main()

{

clrscr();

printf("\n\n\tImplementation of Kruskal's algorithm\n\n");

printf("\nEnter the no. of vertices\n");

scanf("%d",&n);

printf("\nEnter the cost adjacency matrix\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

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

if(cost[i][j]==0)

cost[i][j]=999;

}

}

printf("\nThe edges of Minimum Cost Spanning Tree are\n\n");

while(ne<n)

{

for(i=1,min=999;i<=n;i++)

{

for(j=1;j<=n;j++)

{

if(cost[i][j]<min)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

u=find(u);

v=find(v);

if(uni(u,v))

{

printf("\n%d edge (%d,%d) =%d\n",ne++,a,b,min);

mincost +=min;

}

cost[a][b]=cost[b][a]=999;

}

printf("\n\tMinimum cost = %d\n",mincost);

getch();

}

int find(int i)

{

while(parent[i])

i=parent[i];

return i;

}

int uni(int i,int j)

{

if(i!=j)

{

parent[j]=i;

return 1;

}

return 0;

**8.From a given vertex in a weighted connected graph,find shortest paths to other vertices using Dijistra's algorithm(cprogramming)**

#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<string.h>

#include<math.h>

#define IN 999

#define N 10

int max;

int dijkstra(int cost[N][N], int source, int target);

int main()

{

int cost[N][N],i,j,w,ch,co;

int source, target,x,y;

printf("Enter num of vertex : ");

scanf("%d", &max);

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

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

cost[i][j] = IN;

for(x=0;x< max;x++)

{

for(y=0;y< max;y++)

{

printf("Enter the weight of the path between nodes %d and %d: ",x,y);

scanf("%d",&w);

cost [x][y] = w;

}

printf("\n");

}

printf("\nEnter the source : ");

scanf("%d", &source);

printf("\nEnter the target : ");

scanf("%d", &target);

co = dijsktra(cost,source,target);

printf("\nThe Shortest Path: %d",co);

}

int dijsktra(int cost[N][N],int source,int target)

{

int dist[N],prev[N],selected[N]={0},i,m,min,start,d,j;

int myTop =-1;

int pt[N];

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

{

dist[i] = IN;

prev[i] = -1;

}

start = source;

selected[start]=1;

dist[start] = 0;

while(selected[target] ==0)

{

min = IN;

m = 0;

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

{

d = dist[start] +cost[start][i];

if(d< dist[i]&&selected[i]==0)

{

dist[i] = d;

prev[i] = start;

}

if(min>dist[i] && selected[i]==0)

{

min = dist[i];

m = i;

}

}

//printf("\nThe value of M is %d\n", m);

pt[++myTop] = m;

start = m;

selected[start] = 1;

}

start = target;

j = 0;

while(start != -1)

{

start = prev[start];

}

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

/\*printf("\n Values of vertex for shortest path is %d \n",/return dist[target];