Linked list

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

typedef struct SLL

{

int data;

struct SLL\* next;

} node;

node\* create(node\* head)

{

int val;

node\* new\_node;

printf("Enter the Elements: ");

scanf("%d", &val);

new\_node = (node\*)malloc(sizeof(node));

new\_node->data = val;

new\_node->next = head;

return new\_node;

}

void display(node\* head)

{

node\* temp = head;

while (temp != NULL)

{

printf("%d -> ", temp->data);

temp = temp->next;

}

printf("NULL\n");

}

void insert(node\*\* head)

{

\*head = create(\*head);

}

void delete(node\*\* head)

{

int val;

node\*temp =\*head;

node\*prev = NULL;

printf("Enter the element to delete: ");

scanf("%d", &val);

while (temp != NULL && temp->data != val)

{

prev = temp;

temp = temp->next;

}

if (temp == NULL)

{

printf("Element %d not found in the list.\n", val);

return;

}

if (prev == NULL)

{

\*head = temp->next;

}

else

{

prev->next = temp->next;

}

free(temp);

printf("Element %d deleted from the list.\n", val);

}

int main()

{

int choice;

node\* head = NULL;

clrscr();

do

{

printf("\n1.Create\n2.Display\n3.Insert\n4.Delete\n5.Quit\nEnter choice:");

scanf("%d",&choice);

switch(choice)

{

case 1: insert(&head); break;

case 2: display(head); break;

case 3: insert(&head); break;

case 4: delete(&head); break;

case 5: break;

default: printf("Invalid Choice\n");

}

}while(choice!=5);

getch();

return 0;

}

Stack array

#include <stdio.h>

#include <conio.h>

#define size 5

int stack[size];

int top = -1;

void push()

{

int n;

printf("\nEnter item in stack: ");

scanf("%d", &n);

if (top == size - 1)

{

printf("\nStack is Full");

}

else

{

top = top + 1;

stack[top] = n;

printf("\nItem pushed: %d", n);

}

}

void pop()

{

int item;

if (top == -1)

{

printf("\nStack is Empty");

}

else

{

item = stack[top];

printf("\nItem popped is = %d", item);

top--;

}

}

void display()

{

int i;

printf("\nItems in Stack are:");

if (top == -1)

{

printf("\nStack is Empty");

return;

}

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

{

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

}

}

int main()

{

int choice;

clrscr();

do

{

printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\nEnter Choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1: push(); break;

case 2: pop(); break;

case 3: display(); break;

case 4: break;

default: printf("Invalid Choice");

}

} while (choice != 4);

getch();

return 0;

}

Queue array

#include<stdio.h>

#include<conio.h>

#define size 10

int queue[size];

int front = -1;

int rear = -1;

void enqueue(int item)

{

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

{

front=0;rear=0;

}

else

{

rear=rear+1;

}

queue[rear]=item;

}

void dequeue()

{

if(front==rear)

{

front=-1;rear=-1;

}

else

{

front =front+1;

}

}

int underflow()

{

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

{

printf("\nQueue is Empty");

return 0;

}

else

return 1;

}

int overflow()

{

if(rear==size-1)

{

printf("Queue is Full");

return 0;

}

else

return 1;

}

void display()

{

int i;

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

{

printf("\nElement %d:%d",i+1,queue[i]);

}

}

int main()

{

int choice;

int item;

clrscr();

do

{

printf("\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exist");

scanf("%d",&choice);

switch(choice)

{

case 1:

if(overflow()==1)

{

printf("\nEnter the item to inserted:");

scanf("%d",&item);

enqueue(item);

}break;

case 2:

if(underflow()==1)

{

dequeue();

}break;

case 3:

if(underflow()==1)

{

printf("The Queue is:\n");

display();

}break;

case 4:

break;

default:printf("Invalid Choice:");

}

}while(choice!=4);

getch();

return 0;

}

Tree traversal

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct node

{

int key;

struct node\*left,\*right;

};

struct node\*new\_node(int item)

{

struct node\*new\_node=(struct node\*)malloc(sizeof(struct node));

new\_node->key=item;

new\_node->left=new\_node->right=NULL;

return new\_node;

}

void preordertraversal(struct node\*root)

{

if(root!=NULL)

{

printf("%d", root->key);

printf(" ");

preordertraversal(root->left);

preordertraversal(root->right);

}

}

void inordertraversal(struct node\*root)

{

if(root!=NULL)

{

inordertraversal(root->left);

printf("%d", root->key);

printf(" ");

inordertraversal(root->right);

}

}

void postordertraversal(struct node\*root)

{

if(root!=NULL)

{

postordertraversal(root->left);

postordertraversal(root->right);

printf("%d", root->key);

printf(" ");

}

}

int main()

{

struct node\*root=new\_node(8);

clrscr();

root->left=new\_node(3);

root->right=new\_node(10);

root->left->left=new\_node(1);

root->left->right=new\_node(6);

root->right->left=new\_node(14);

printf("PreOrder Traversal:");

preordertraversal(root);

printf("\n");

printf("InOrder Traversal:");

inordertraversal(root);

printf("\n");

printf("PostOrderTraversal:");

postordertraversal(root);

printf("\n");

getch();

return 0;

}

Bubble sort

#include<stdio.h>

#include<conio.h>

void bubble\_sort(int arr[],int n)

{

int i,j,temp;

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

{

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

{

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

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

}

int main()

{

int n,i;

int arr[100];

clrscr();

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

scanf("%d",&n);

arr[n];

printf("Enter %d elements:",n);

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

{

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

}

printf("Before Sorting:");

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

{

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

printf(" ");

}

printf("\n");

bubble\_sort(arr,n);

printf("After Sorting:");

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

{

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

printf(" ");

}

getch();

return 0;

}

Insertion sort

#include<stdio.h>

#include<conio.h>

void insertion\_sort(int arr[],int n)

{

int i,k,j,key;

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

{

key=arr[i];

j=i-1;

printf("Iteration %d:",i);

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

{

printf("%d",arr[k]);

}

printf("\n");

while(j>=0 && arr[j] >key)

{

arr[j+1]=arr[j];

j=j-1;

}

arr[j+1]=key;

}

}

int main()

{

int i,n;

int arr[100];

clrscr();

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

scanf("%d",&n);

arr[n];

printf("Enter %d elements:",n);

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

{

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

}

printf("Before Sorting:");

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

{

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

}

printf("\n");

insertion\_sort(arr,n);

printf("After Sorting:");

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

{

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

}

getch();

return 0;

}

Binary search tree

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct node

{

int data;

struct node\*left;

struct node\*right;

};

struct node\*create(int data)

{

struct node\*new\_node=(struct node\*)malloc(sizeof(struct node));

new\_node->data=data;

new\_node->left=NULL;

new\_node->right=NULL;

return new\_node;

}

struct node\*insert(struct node\*root,int data)

{

if(root==NULL)

{

return create(data);

}

if(data<root->data)

{

root->left=insert(root->left,data);

}

if(data>root->data)

{

root->right=insert(root->right,data);

}

return root;

}

int search(struct node\*root,int key)

{

while(root!=NULL)

{

if(key==root->data)

{

return 1;

}

if(key<root->data)

{

root=root->left;

}

else

{

root=root->right;

}

}

return 0;

}

int main()

{

int key;

struct node\*root=NULL;

clrscr();

insert(root,50);

insert(root,30);

insert(root,70);

insert(root,20);

insert(root,40);

insert(root,60);

insert(root,80);

printf("Enter the key to search:");

scanf("%d",&key);

if(search(root,key))

{

printf("Key %d found in the binary search tree\n",key);

}

else

{

printf("key %d not found in the binary search tree\n",key);

}

getch();

return 0;

}

Selection sort

#include <stdio.h>

#include <conio.h>

void selection\_sort(int arr[], int n)

{

int i, j, min, temp;

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

{

min = i;

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

{

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

{

min = j;

}

}

temp = arr[min];

arr[min] = arr[i];

arr[i] = temp;

}

}

int main()

{

int n, i;

int arr[100];

clrscr();

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

scanf("%d", &n);

printf("Enter %d elements: ", n);

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

{

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

}

printf("Before Sorting: ");

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

{

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

}

printf("\n");

selection\_sort(arr, n);

printf("After Sorting: ");

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

{

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

}

getch();

return 0;

}

Linear search

#include<stdio.h>

#include<conio.h>

int linear\_sort(int arr[],int n,int key)

{

int i;

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

{

if(arr[i]==key)

{

return i;

}

}

return -1;

}

int main()

{

int i,n,key,arr[100],result;

clrscr();

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

scanf("%d",&n);

printf("Enter %d elements:",n);

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

{

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

}

printf("Enter the key to find:");

scanf("%d",&key);

result=linear\_sort(arr,n,key);

if(result !=-1)

{

printf("Element %d is found..",key);

}

else

{

printf("Element %d is not found...",key);

}

getch();

return 0;

}

Binary search

#include<stdio.h>

#include<conio.h>

int binary\_search(int arr[],int n,int key)

{

int left=0,mid;

int right=n-1;

while(left<=right)

{

mid=left+(right-left)/2;

if(arr[mid]==key)

{

return mid;

}

else if (arr[mid]<key)

{

left=mid+1;

}

else

{

right=mid-1;

}

}

return -1;

}

int main()

{

int n,i,arr[100],result,key;

clrscr();

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

scanf("%d",&n);

arr[n];

printf("Enter %d sorted elements:",n);

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

{

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

}

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

{

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

{

printf("Error:Elements is not sorted..");

getch();

return 0;

}

}

printf("Enter the key to find:");

scanf("%d",&key);

result=binary\_search(arr,n,key);

if(result !=-1)

{

printf("Element %d is found...",key);

}

else

{

printf("Element %d is not found...",key);

}

getch();

return 0;

}

Bfs and dfs

#include <stdio.h>

#include <conio.h>

#define MAX 100

int visited[MAX];

void bfs(int adj[MAX][MAX], int n, int start)

{

int i;

int queue[MAX], front = 0, rear = 0;

queue[rear++] = start;

visited[start] = 1;

while (front < rear)

{

int v = queue[front++];

printf("%d ", v);

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

{

if (adj[v][i] == 1 && !visited[i])

{

queue[rear++] = i;

visited[i] = 1;

}

}

}

}

void dfs(int adj[MAX][MAX], int n, int v)

{

int i;

printf("%d ", v);

visited[v] = 1;

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

{

if (adj[v][i] == 1 && !visited[i])

{

dfs(adj, n, i);

}

}

}

int main()

{

int n,i, e, x, y;

int adj[MAX][MAX] = {0};

clrscr();

printf("Enter the number of vertices: ");

scanf("%d", &n);

printf("Enter the number of edges: ");

scanf("%d", &e);

printf("Enter the edges (format: x y):\n");

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

{

scanf("%d %d", &x, &y);

adj[x][y] = 1;

adj[y][x] = 1;

}

printf("Enter the starting vertex: ");

scanf("%d", &x);

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

{

visited[i] = 0;

}

printf("BFS traversal: ");

bfs(adj, n, x);

printf("\n");

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

{

visited[i] = 0;

}

printf("DFS traversal: ");

dfs(adj, n, x);

printf("\n");

getch();

return 0;

}