# Stack-Queue-Linked List-Searching

# implement the Stack using an array

#include<iostream.h>

#include<stdlib.h>

class stack

{

int stk[5];

int top;

public:

stack()

{

top=-1;

}

void push(int x)

{

if(top > 4)

{

cout <<"stack over flow";

return;

}

stk[++top]=x;

cout <<"inserted" <<x;

}

void pop()

{

if(top <0)

{

cout <<"stack under flow";

return;

}

cout <<"deleted" <<stk[top--];

}

void display()

{

if(top<0)

{

cout <<" stack empty";

return;

}

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

cout <<stk[i] <<" ";

}

};

main()

{

int ch;

stack st;

while(1)

{

cout <<"**\n**1.push 2.pop 3.display 4.exit**\n**Enter ur choice";

cin >> ch;

switch(ch)

{

case 1: cout <<"enter the element";

cin >> ch;

st.push(ch);

break;

case 2: st.pop(); break;

case 3: st.display();break;

case 4: exit(0);

}

}

return (0);

}

# implement the Queue using an array

#include<iostream.h>

#include<stdlib.h>

class queue

{

int queue1[5];

int rear,front;

public:

queue()

{

rear=-1;

front=-1;

}

void insert(int x)

{

if(rear > 4)

{

cout <<"queue over flow";

front=rear=-1;

return;

}

queue1[++rear]=x;

cout <<"inserted" <<x;

}

void delet()

{

if(front==rear)

{

cout <<"queue under flow";

return;

}

cout <<"deleted" <<queue1[++front];

}

void display()

{

if(rear==front)

{

cout <<" queue empty";

return;

}

for(int i=front+1;i<=rear;i++)

cout <<queue1[i]<<" ";

}

};

main()

{

int ch;

queue qu;

while(1)

{

cout <<"**\n**1.insert 2.delet 3.display 4.exit**\n**Enter ur choice";

cin >> ch;

switch(ch)

{

case 1: cout <<"enter the element";

cin >> ch;

qu.insert(ch);

break;

case 2: qu.delet(); break;

case 3: qu.display();break;

case 4: exit(0);

}

}

return (0);

}

# implement the Queue using linked list

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class node

{

public:

class node \*next;

int data;

};

class queue : public node

{

node \*head;

int front,rare;

public:

queue()

{

front=-1;

rare=-1;

}

void push(int x)

{

if (rare < 0 )

{

head =new node;

head->next=NULL;

head->data=x;

rare ++;

}

else

{

node \*temp,\*temp1;

temp=head;

if(rare >= 4)

{

cout <<"queue over flow";

return;

}

rare++;

while(temp->next != NULL)

temp=temp->next;

temp1=new node;

temp->next=temp1;

temp1->next=NULL;

temp1->data=x;

} }

void display()

{

node \*temp;

temp=head;

if (rare < 0)

{

cout <<" queue under flow";

return;

}

while(temp != NULL)

{

cout <<temp->data<< " ";

temp=temp->next;

}

}

void pop()

{

node \*temp;

temp=head;

if( rare < 0)

{

cout <<"queue under flow";

return;

}

if(front == rare)

{

front = rare =-1;

head=NULL;

return;

}

front++;

head=head->next;

}

};

main()

{

queue s1;

int ch;

while(1)

{

cout <<"**\n**1.PUSH**\n**2.POP**\n**3.DISPLAY**\n**4.EXIT**\n** enter ru choice:";

cin >> ch;

switch(ch)

{

case 1:

cout <<"**\n** enter a element";

cin >> ch;

s1.push(ch); break;

case 2: s1.pop();break;

case 3: s1.display();break;

case 4: exit(0);

}

}

return (0);

}

**implement the Stack using linked list**

#include<iostream.h>

// Creating a NODE Structure

struct node

{

int data;

struct node \*next;

};

// Creating a class STACK

class stack

{

struct node \*top;

public:

stack() // constructure

{

top=NULL;

}

void push(); // to insert an element

void pop(); // to delete an element

void show(); // to show the stack

};

// PUSH Operation

void stack::push()

{

int value;

struct node \*ptr;

cout<<"\nPUSH Operationn";

cout<<"Enter a number to insert: ";

cin>>value;

ptr=new node;

ptr->data=value;

ptr->next=NULL;

if(top!=NULL)

ptr->next=top;

top=ptr;

cout<<"\nNew item is inserted to the stack!!!";

}

// POP Operation

void stack::pop()

{

struct node \*temp;

if(top==NULL)

{

cout<<"\nThe stack is empty!!!";

}

temp=top;

top=top->next;

cout<<"\nPOP Operation........\nPoped value is "<<temp->data;

delete temp;

}

// Show stack

void stack::show()

{

struct node \*ptr1=top;

cout<<"\nThe stack is\n";

while(ptr1!=NULL)

{

cout<<ptr1->data<<" ->";

ptr1=ptr1->next;

}

cout<<"NULL\n";

}

// Main function

int main()

{

stack s;

int choice;

while(1)

{

cout<<"\n-----------------------------------------------------------";

cout<<"\n\t\tSTACK USING LINKED LIST\n\n";

cout<<"1:PUSH\n2:POP\n3:DISPLAY STACK\n4:EXIT";

cout<<"\nEnter your choice(1-4): ";

cin>>choice;

switch(choice)

{

case 1:

s.push();

break;

case 2:

s.pop();

break;

case 3:

s.show();

break;

case 4:

return 0;

break;

default:

cout<<"\nPlease enter correct choice(1-4)!!";

break;

}

}

return 0;

}

**Insertion Sorting in C++**

#include<iostream.h>

int main()

{

    int i,j,n,temp,a[30];

    cout<<"Enter the number of elements:";

    cin>>n;

    cout<<"\nEnter the elements\n";

    for(i=0;i<n;i++)

    {

        cin>>a[i];

    }

    for(i=1;i<=n-1;i++)

    {

        temp=a[i];

        j=i-1;

        while((temp<a[j])&&(j>=0))

        {

            a[j+1]=a[j];    //moves element forward

            j=j-1;

        }

        a[j+1]=temp;    //insert element in proper place

    }

    cout<<"\nSorted list is as follows\n";

    for(i=0;i<n;i++)

    {

        cout<<a[i]<<" ";

    }

    return 0;

}

## Algorithm for Insertion Sort in C & C++

Let ARR is an array with N elements

1. Read ARR

2. Repeat step 3 to 8 for I=1 to N-1

3. Set Temp=ARR[I]

4. Set J=I-1

5. Repeat step 6 and 7 while Temp<ARR[J] AND J>=0

6. Set ARR[J+1]=ARR[J] [Moves element forward]

7. Set J=J-1

 [End of step 5 inner  
loop]

8. Set ARR[J+1]=Temp [Insert element in proper place]

 [End of step 2 outer  
loop]

9. Exit

**Selection Sorting in C++**

#include <iostream.h>

// Sort arr[] of size n using Selection Sort.

void SelectionSort (int arr[], int n)

{

int i, j;

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

{

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

{

// Comparing consecutive data and switching values if value at i > j.

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

{

arr[i] = arr[i]+arr[j];

arr[j] = arr[i]-arr[j];

arr[i] = arr[i]-arr[j];

}

}

// Value at i will be minimum of all the values above this index.

}

}

int main()

{

int n, i;

cout<<"**\n**Enter the number of data element to be sorted: ";

cin>>n;

int arr[n];

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

{

cout<<"Enter element "<<i+1<<": ";

cin>>arr[i];

}

SelectionSort(arr, n);

// Display the sorted data.

cout<<"**\n**Sorted Data ";

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

cout<<"->"<<arr[i];

return 0;

}

Algo:

1. Take input of data.  
2. Call SelectionSort() function with ‘arr’ the array of data and ‘n’ the number of values, in the argument list.  
3. Implement Sorting algorithm using nested for loop.  
4. The first loop will run on ‘i’ from 0 to n-1.  
5. The second loop will run on ‘j’ from i+1 to n-1.  
6. Compare value at i with value at j.  
7. Switch the values if arr[j+1] < arr[j].

8. Return to main and display the result.

9. Exit.

**Binary Search in C++**

#include<iostream.h>

int main()

{

    int search(int [],int,int);

    int n,i,a[100],e,res;

    cout<<"How Many Elements:";

    cin>>n;

    cout<<"\nEnter Elements of Array in Ascending order\n";

    for(i=0;i<n;++i)

    {

        cin>>a[i];

    }

    cout<<"\nEnter element to search:";

    cin>>e;

    res=search(a,n,e);

    if(res!=-1)

        cout<<"\nElement found at position "<<res+1;

    else

        cout<<"\nElement is not found....!!!";

    return 0;

}

int search(int a[],int n,int e)

{

    int f,l,m;

    f=0;

    l=n-1;

    while(f<=l)

    {

        m=(f+l)/2;

        if(e==a[m])

            return(m);

        else

            if(e>a[m])

                f=m+1;

            else

                l=m-1;

    }

    return -1;

}

**Linear Search in C++**

#include<iostream.h>

int main()

{  
cout<<"Enter The Size Of Array:   ";  
int size;  
cin>>size;  
  
  
int array[size], key,i;  
  
// Taking Input In Array  
 for(int j=0;j<size;j++){  
 cout<<"Enter "<<j<<" Element: ";  
 cin>>array[j];  
 }  
  
//Your Entered Array Is  
 for(int a=0;a<size;a++){  
    cout<<"array[ "<<a<<" ]  =  ";  
    cout<<array[a]<<endl;  
 }  
  
 cout<<"Enter Key To Search  in Array";  
 cin>>key;  
  
   for(i=0;i<size;i++){  
    if(key==array[i]){  
 cout<<"Key Found At Index Number :  "<<i<<endl;  
 break;  
    }  
 }  
  
  
if(i != size){  
cout<<"KEY FOUND at index :  "<<i;  
}  
else{  
cout<<"KEY NOT FOUND in Array  ";  
}  
   return 0;  
}