**DSA PRACTICAL**

**NAME : Khushi Panwar**

1. **Write a program to search an element form a list. Give user the option to perform Linear o Binary Search.**
2. **Without using template functions.**

#include<iostream>

using namespace std;

const int size=50;

class search{

int arr[size]; //array for searching

int n; //size of array

public:

int key; //search element

void getData();

int linearSearch();

int binarySearch();

void display(int);

};

void search::getData(){

cout<<"\n -> Enter the number of elements you want to enter : ";

cin>>n;

if (n<=100){

cout<<endl<<"-> Input "<<n<<" elements : ";

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

cin>>arr[i];

}

}else{

cout<<endl<<"\t !! INVALID INPUT , try again !!"<<endl;

getData();

}

return ;

}//input the array for search operation

int search::linearSearch(){

cout<<endl<<"\t \_ PERFORMING LINEAR SEARCH \_"<<endl;

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

if (key==arr[i]){

return i;

}

}

return -1;

}//linear search function

int search::binarySearch(){

cout<<endl<<"\t \_ PERFORMING BINARY SEARCH \_"<<endl;

cout<<endl<<"\t \_ PERFORMING BINARY SEARCH \_"<<endl;

int low=0;

int high=n-1;

while (low <= high){

int mid = low + (high - low)/2;

if (arr[mid] == key) return mid;

else if (arr[mid] < key) low = mid + 1;

else high = mid - 1;

}

return -1;

}//Binary search function

void search::display(int result){

if (result>=0){

cout<<endl<<"~~ The element has been found at "<<result<<" index!"<<endl<<endl;

}else{

cout<<endl<<"~~ Search Unsuccessful! "<<endl<<endl;

}

return;

}// displaying the search result function

void operationList(){

cout<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\t SEARCH OPERATIONS ON ARRAY "<<endl;

cout<<"\t1. Linear Search "<<endl;

cout<<"\t2. Binary Search "<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<endl;

return ;

}

int main(){

search ob;

char ch='y';

int result, ele;

int choice;

do{

operationList();

cout<<endl<<"-> Select the Operation you want to perform : ";

cin>>choice;

cout<<endl;

switch(choice){

case 1: ob.getData();

cout<<"-> Enter the element you want to search : ";

cin>>ele;

ob.key=ele;

result=ob.linearSearch();

ob.display(result);

break;

case 2: ob.getData();

cout<<"-> Enter the element you want to search : ";

cin>>ob.key;

result=ob.binarySearch();

ob.display(result);

break;

default: cout<<"\t !! INVALID CHOICE !!"<<endl<<endl;

}

cout<<"Do you want to continue (y/n)?";

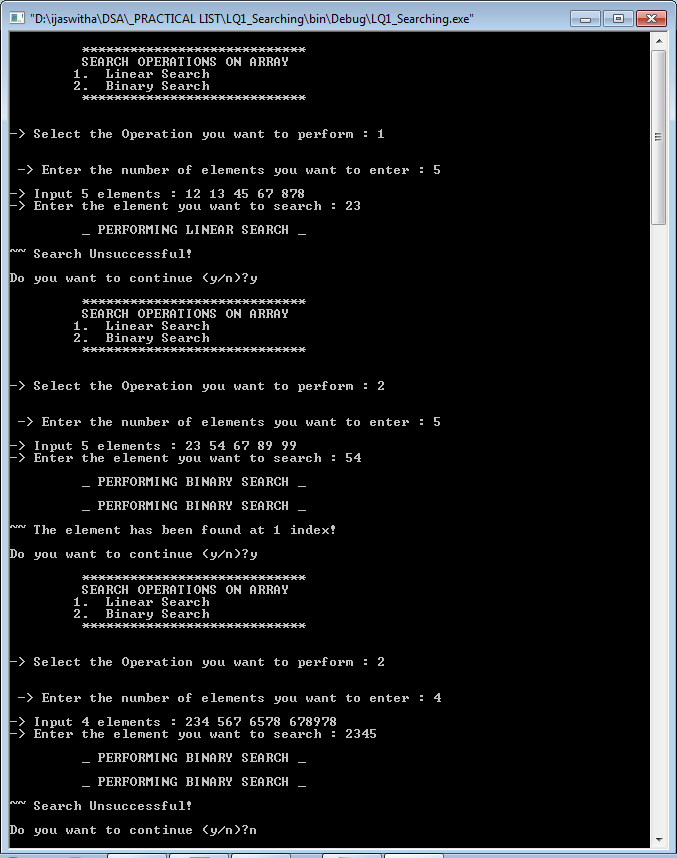
cin>>ch;

}while(ch=='y' || ch=='Y');

cout<<endl<<endl<<"\t PROGRAM ENDS HERE! "<<endl;

return 0;

}



1. **Using Template functions :**

#include<iostream>

using namespace std;

const int size=50;

template <class t>

class search{

t arr[size]; //array for searching

int n; //size of array

public:

t key; //search element

void getData();

int linearSearch();

int binarySearch();

void display(int);

search(){

}

};

template <class t>

void search<t>::getData(){

cout<<"\n -> Enter the number of elements you want to enter : ";

cin>>n;

if (n<=100){

cout<<endl<<"-> Input "<<n<<" elements : ";

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

cin>>arr[i];

}

}else{

cout<<endl<<"\t !! INVALID INPUT , try again !!"<<endl;

getData();

}

cout<<endl<<"-> Which element do you want to search : ";

cin>>key;

return ;

}//input the array for search operation

template <class t>

int search<t>::linearSearch(){

cout<<endl<<"\t \_ PERFORMING LINEAR SEARCH \_"<<endl;

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

if (key==arr[i]){

return i;

}

}

return -1;

}//linear search function

template <class t>

int search<t>::binarySearch(){

cout<<endl<<"\t \_ PERFORMING BINARY SEARCH \_"<<endl;

int low=0;

int high=n-1;

while (low <= high){

int mid = low + (high - low)/2;

if (arr[mid] == key) return mid;

else if (arr[mid] < key) low = mid + 1;

else high = mid - 1;

}

return -1;

}//Binary search function

template <class t>

void search<t>::display(int result){

if (result>=0){

cout<<endl<<"~~ The element has been found at "<<result<<" index!"<<endl<<endl;

}else{

cout<<endl<<"~~ Search Unsuccessful! "<<endl<<endl;

}

return;

}// displaying the search result function

void operationList(){

cout<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\t SEARCH OPERATIONS ON ARRAY USING TEMPLATES"<<endl;

cout<<"\t1. Linear Search "<<endl;

cout<<"\t2. Binary Search "<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<endl;

return ;

}

template <class t>

void operation(search<t> ob){

int result, choice;

operationList();

cout<<endl<<"-> Select the Operation you want to perform : ";

cin>>choice;

cout<<endl;

switch(choice){

case 1: ob.getData();

result=ob.linearSearch();

ob.display(result);

break;

case 2: ob.getData();

result=ob.binarySearch();

ob.display(result);

break;

default: cout<<"\t !! INVALID CHOICE !!"<<endl<<endl;

}

return;

}

int main(){

char ch='y';

do{

string type;

cout<<"\n-> Which type of array do you want to create (int/string/float)? : ";

cin>>type;

if (type=="int" || type=="INT"){

search<int> ob1;

operation(ob1);

}

else if (type=="string" || type=="STRING"){

search<string> ob2;

operation(ob2);

}

else if (type=="float" || type=="FLOAT"){

search<float> ob3;

operation(ob3);

}

else cout<<" \*\* INVALID CHOICE \*\* " <<endl;

cout<<"Do you want to continue (y/n)?";

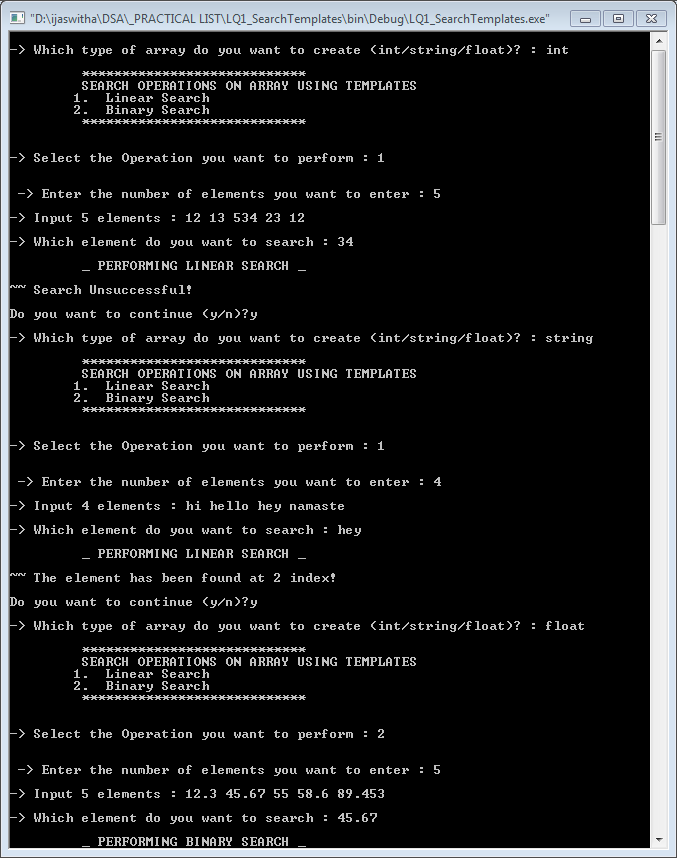
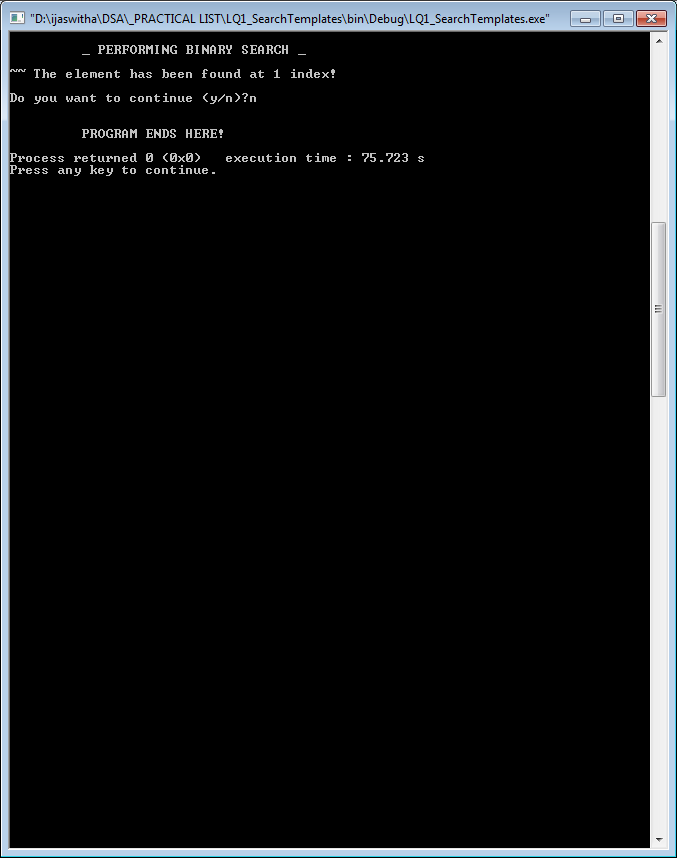
cin>>ch;

}while(ch=='y' || ch=='Y');

cout<<endl<<endl<<"\t PROGRAM ENDS HERE! "<<endl;

return 0;

}

** **

1. **WAP to sort a list of elements. Give user the option to perform sorting using Insertion Sort, Bubble sort or selection sort.**
2. **using classes and objects**

#include<iostream>

using namespace std;

const int size=50;

class sorting{

int arr[size]; //array for sorting

int n; //size of array

public:

void getData();

void bubbleSort();

void insertionSort();

void selectionSort();

void display();

};

void sorting::getData(){

cout<<"\n -> Enter the number of elements you want to enter : ";

cin>>n;

if (n<=100){

cout<<endl<<"-> Input "<<n<<" elements : ";

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

cin>>arr[i];

}

}else{

cout<<endl<<"\t !! INVALID INPUT , try again !!"<<endl;

getData();

}

return ;

}//input the array for search operation

void sorting::insertionSort(){

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

int key=arr[i];

int j=i-1;

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

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

j-=1;

}

arr[j+1]=key;

}

}

void sorting::selectionSort(){

int temp;

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

int min=i;

for (int j=min+1; j<n; j++){

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

min=j;

}

}

if (i!=min){

temp=arr[i];

arr[i]=arr[min];

arr[min]=temp;

}

}

}

void sorting::bubbleSort(){

int temp;

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

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

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

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

}

void sorting:: display(){

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

cout<<"\t"<<arr[i];

}

cout<<endl;

}

void operationList(){

cout<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\t SORTING ON ARRAY "<<endl;

cout<<"\t1. Bubble Sort "<<endl;

cout<<"\t2. Insertion Sort "<<endl;

cout<<"\t3. Selection Sort "<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<endl;

return ;

}

int main(){

sorting ob;

char ch='y';

int result, ele;

int choice;

do{

operationList();

cout<<endl<<"-> Select the sort you want to perform : ";

cin>>choice;

cout<<endl;

switch(choice){

case 1: ob.getData();

cout<<"\n-> Array before sorting : ";

ob.display();

ob.bubbleSort();

cout<<"\n-> Array after BUBBLE SORT : ";

ob.display();

break;

case 2: ob.getData();

cout<<"-> Array before sorting : ";

ob.display();

ob.insertionSort();

cout<<"\n-> Array after INSERTION SORT : ";

ob.display();

break;

case 3: ob.getData();

cout<<"\n-> Array before sorting : ";

ob.display();

ob.selectionSort();

cout<<"\n-> Array after SELECTION SORT : ";

ob.display();

break;

default: cout<<"\t !! INVALID CHOICE !!"<<endl<<endl;

}

cout<<endl<<"Do you want to continue (y/n)? ";

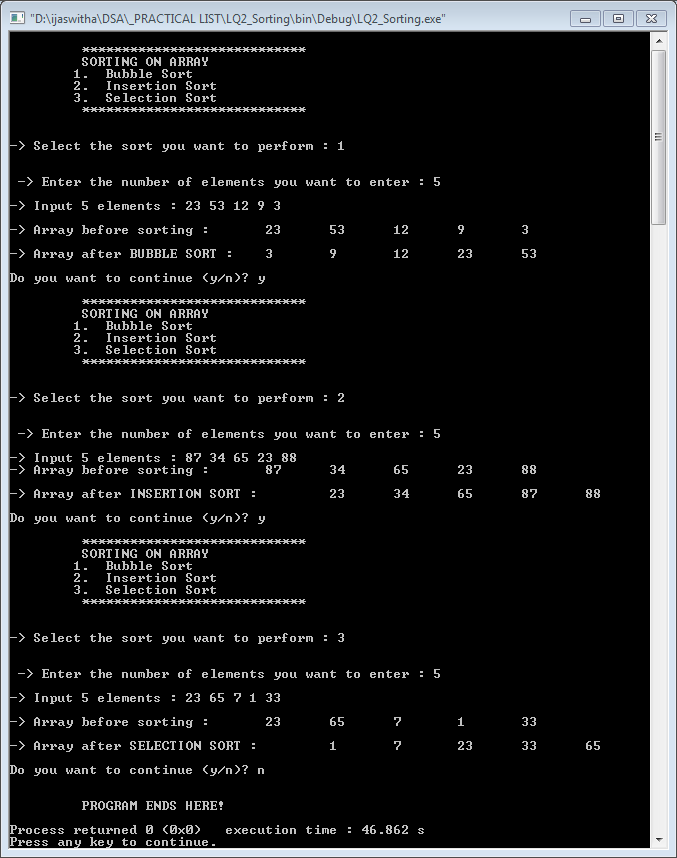
cin>>ch;

}while(ch=='y' || ch=='Y');

cout<<endl<<endl<<"\t PROGRAM ENDS HERE! "<<endl;

return 0;

}



1. **using templates**

#include<iostream>

using namespace std;

const int size=50;

template<class t>

class sorting{

t arr[size]; //array for sorting

int n; //size of array

public:

void getData();

void bubbleSort();

void insertionSort();

void selectionSort();

void display();

};

template<class t>

void sorting<t>::getData(){

cout<<"\n -> Enter the number of elements you want to enter : ";

cin>>n;

if (n<=100){

cout<<endl<<"-> Input "<<n<<" elements : ";

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

cin>>arr[i];

}

}else{

cout<<endl<<"\t !! INVALID INPUT , try again !!"<<endl;

getData();

}

return ;

}//input the array for sorting operation

template<class t>

void sorting<t>::insertionSort(){

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

t key=arr[i];

int j=i-1;

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

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

j-=1;

}

arr[j+1]=key;

}

}

template<class t>

void sorting<t>::selectionSort(){

t temp;

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

int min=i;

for (int j=min+1; j<n; j++){

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

min=j;

}

}

if (i!=min){

temp=arr[i];

arr[i]=arr[min];

arr[min]=temp;

}

}

}

template<class t>

void sorting<t>::bubbleSort(){

t temp;

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

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

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

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

}

template<class t>

void sorting<t>:: display(){

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

cout<<"\t"<<arr[i];

}

cout<<endl;

}

void operationList(){

cout<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\t SORTING ON ARRAY "<<endl;

cout<<"\t1. Bubble Sort "<<endl;

cout<<"\t2. Insertion Sort "<<endl;

cout<<"\t3. Selection Sort "<<endl;

cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<endl;

return ;

}

template<class t>

void operation(sorting<t> ob){

int choice;

operationList();

cout<<endl<<"-> Select the sort you want to perform : ";

cin>>choice;

cout<<endl;

switch(choice){

case 1: ob.getData();

cout<<"\n-> Array before sorting : ";

ob.display();

ob.bubbleSort();

cout<<"\n-> Array after BUBBLE SORT : ";

ob.display();

break;

case 2: ob.getData();

cout<<"-> Array before sorting : ";

ob.display();

ob.insertionSort();

cout<<"\n-> Array after INSERTION SORT : ";

ob.display();

break;

case 3: ob.getData();

cout<<"\n-> Array before sorting : ";

ob.display();

ob.selectionSort();

cout<<"\n-> Array after SELECTION SORT : ";

ob.display();

break;

default: cout<<"\t !! INVALID CHOICE !!"<<endl<<endl;

}

}

int main(){

cout<<"\n\t \*\* SORTING ON ARRAY \*\* "<<endl;

char ch='y';

int choice;

do{

string type;

cout<<"\n-> Which type of array do you want to create (int/string/float)? : ";

cin>>type;

if (type=="int" || type=="INT"){

sorting<int> ob1;

operation(ob1);

}

else if (type=="string" || type=="STRING"){

sorting<string> ob2;

operation(ob2);

}

else if (type=="float" || type=="FLOAT"){

sorting<float> ob3;

operation(ob3);

}

else cout<<" \*\* INVALID CHOICE \*\* " <<endl;

cout<<endl<<"Do you want to continue (y/n)? ";

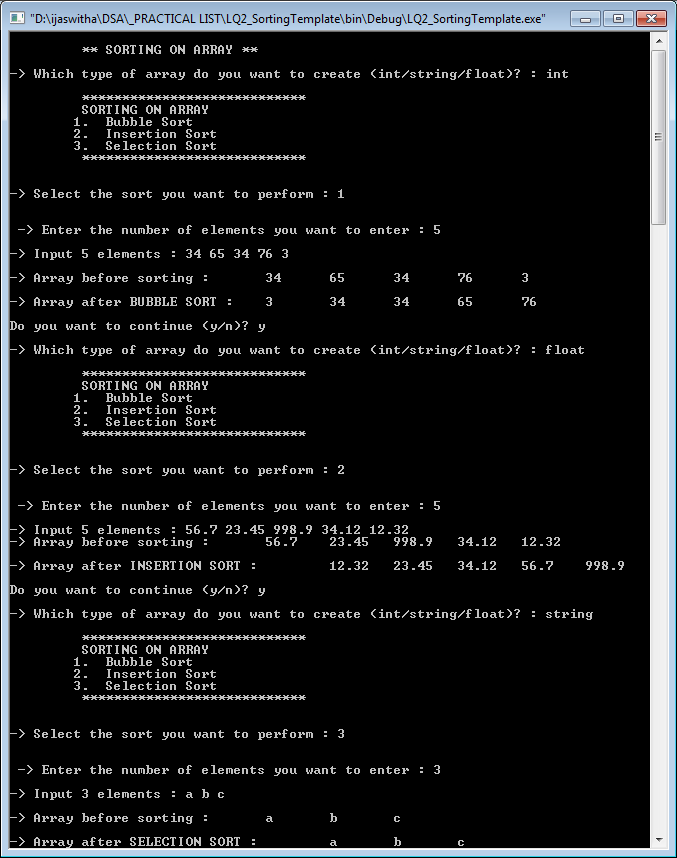
cin>>ch;

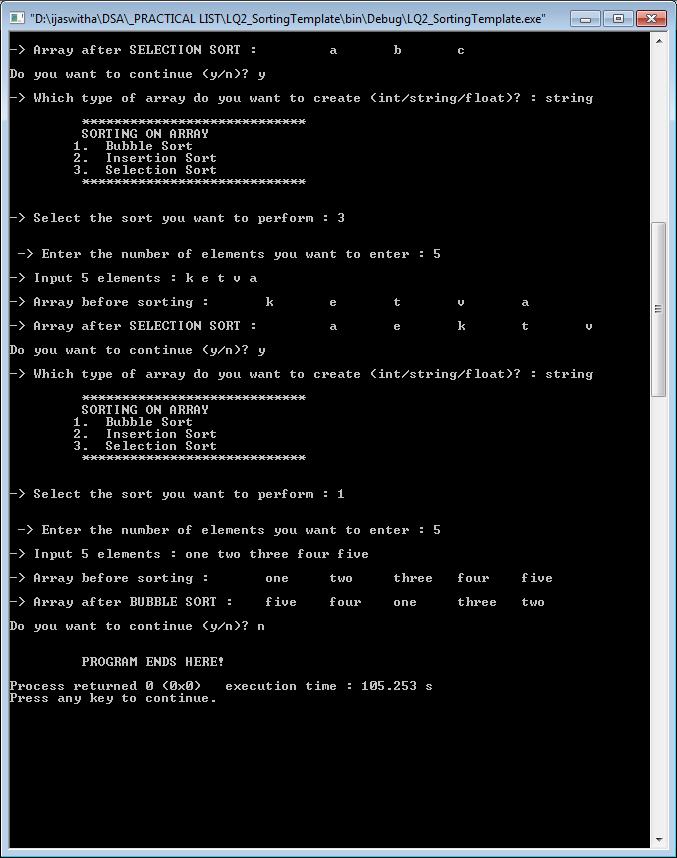
}while(ch=='y' || ch=='Y');

cout<<endl<<endl<<"\t PROGRAM ENDS HERE! "<<endl;

return 0;

}

****

****

**7 . WAP TO IMPLEMENT QUEUE USING LINKED LIST**

//implementing stacks using linked lists : "->" is used to refer to the member belonging to the struct "node" type objects.

#include<iostream>

#include <iomanip>

using namespace std;

struct node

{

int info;

node\*next;

};

class queueType

{

node \*front;

node \*rear;

public:

queueType()

{

front=rear=NULL;

}

void addition(node \*);

node\* createNewNode(int);

int deletion();

void display();

int isEmpty();

};

node \*queueType::createNewNode(int x){

node \*newptr;

newptr=new node();

newptr->info=x;

newptr->next=NULL;

return newptr;

}

void queueType::addition(node \*newptr){

int e;

e=isEmpty();

if (e==1){

front=rear=newptr;

}//empty queue

else{

rear->next=newptr;

rear=newptr;

}//non empty queue

}

int queueType::deletion(){

int e,rem;

node\*temp;

if (front==rear){

rem=front->info;

temp=front;

front=rear=NULL;

return temp->info;

delete temp;

}//queue is empty

else{

rem=front->info;

temp=front;

front=front->next;

return temp->info;

delete temp;

}//non empty queue

}

void queueType::display()

{

int e;

e=isEmpty();

if (e==1){

cout<<"-- NOTHING TO DISPLAY :: QUEUE IS EMPTY "<<endl;

}else{

cout<<endl<<"-> CONTENTS OF THE QUEUE : from the front most to the rearmost end : "<<endl;

node \*temp;

temp=front;

for (temp=front; temp!=NULL; temp=temp->next){

cout<<setw(3)<<temp->info;

}

}

cout<<endl<<endl;

}

int queueType::isEmpty()

{

if (front==NULL) return 1;

else return -1;

}

void operationList()

{

cout<<endl<<endl;

cout<<setw(40)<<"======================="<<endl;

cout<<"\t \t QUEUE OPERATIONS"<<endl;

cout<<"\t 1. Add elements into Queue"<<endl;

cout<<"\t 2. Delete element from Queue"<<endl;

cout<<"\t 3. Display the elements of the queue"<<endl;

cout<<"\t 4. Check if Queue is empty"<<endl;

cout<<setw(40)<<"======================="<<endl<<endl;

return;

}

int main()

{

cout<<setw(50)<<endl<<"\*\* QUEUE IMPLEMENTATION USING LINKED LIST \*\*"<<endl;

int choice, ele, val,rem;

char ch='y';

node \*temp1;

queueType queue;

do{

operationList();

cout<<"-> Select the operation you want to perform : ";

cin>>choice;

switch(choice){

case 1: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : Add the elements \*\* "<<endl;

cout<<"-> Enter a value : ";

cin>>val;

temp1=queue.createNewNode(val);

queue.addition(temp1);

queue.display();

break;

case 2: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : Delete the elements \*\* "<<endl;

rem=queue.isEmpty();

if (rem==1){

cout<<"-- QUEUE is empty! Deletion not possible ! "<<endl;

}else{

ele=queue.deletion();

cout<<"-> Element Deleted : "<<ele<<endl;

queue.display();

}

break;

case 3: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : DISPLAY the elements \*\* "<<endl;

queue.display();

break;

case 4: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : Check emptiness \*\* "<<endl;

ele=queue.isEmpty();

if (ele==1) cout<<endl<<"-> The Queue is Empty! "<<endl;

else cout<<endl<<"-> The Queue is not Empty! "<<endl;

break;

default : cout<<endl<<"!! Select a valid operation !!"<<endl<<endl;

}

cout<<endl<<"-> Do you want to continue (y/n)? ";

cin>>ch;

}while(ch=='y' || ch=='Y');

}

s

**7 . WAP TO IMPLEMENT QUEUE USING ARRAYS**

#include <iostream>

#include <iomanip>

using namespace std;

void operationList()

{

cout<<endl<<endl;

cout<<setw(40)<<"======================="<<endl;

cout<<"\t \t QUEUE OPERATIONS"<<endl;

cout<<"\t 1. Add elements into Queue"<<endl;

cout<<"\t 2. Delete element from Queue"<<endl;

cout<<"\t 3. Display the elements of the queue"<<endl;

cout<<"\t 4. Check if Queue is empty"<<endl;

cout<<"\t 5. Check if Queue is full"<<endl;

cout<<setw(40)<<"======================="<<endl<<endl;

return;

}

const int size=5;

class queuetype{

int Q[size];

int front, rear;

public:

void addition(int a);

int deletion();

int isFull();

int isEmpty();

void display();

queuetype(){

front=rear=-1;

}

};

void queuetype::addition(int a){

if (front==-1){

front=rear=0;

Q[rear]=a;

}else{

Q[++rear]=a;

}

}

int queuetype::deletion(){

int rem;

if (front==rear){

rem=Q[front];

front=rear=-1;

}else{

rem=Q[front];

front++;

}

return rem;

}

int queuetype:: isEmpty(){

int e;

if (front==-1){

return 1;

}else{

return -1;

}

}

int queuetype::isFull(){

if (rear==size-1){

return 1;

}else{

return -1;

}

}

void queuetype::display(){

cout<<endl<<"-> Displaying the Queue : "<<endl;

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

if (i>=front){

cout<<setw(3)<<Q[i];

}else{

cout<<setw(4)<<"\_\_\_";

}

}

for (int i=rear+1; i<size; i++){

cout<<setw(4)<<"\_\_\_";

}

cout<<endl<<endl;

}

int main()

{

cout<<setw(50)<<endl<<"\*\* QUEUE IMPLEMENTATION USING ARRAY \*\*"<<endl;

int choice,ele,res;

char ch='y';

queuetype queue;

while (ch=='y' || ch=='Y'){

operationList();

cout<<"-> What operation do you want to perform : ";

cin>>choice;

switch(choice){

case 1: cout<<endl<<"\t \*\* OPERATION CHOSEN : Enter Elements \*\* "<<endl<<endl;

cout<<"-> Enter the element in the queue : ";

cin>>ele;

queue.addition(ele);

queue.display();

break;

case 2: cout<<endl<<"\t \*\* OPERATION CHOSEN : Delete Elements \*\* "<<endl<<endl;

res=queue.deletion();

cout<<" -- Element Deleted : "<<res<<endl;

queue.display();

break;

case 3: cout<<endl<<"\t \*\* OPERATION CHOSEN : Display Queue \*\* "<<endl<<endl;

queue.display();

break;

case 4: cout<<endl<<"\t \*\* OPERATION CHOSEN : Checking for Underflow \*\* "<<endl<<endl;

res=queue.isEmpty();

if (res==1) cout<<"Queue is Empty! "<<endl;

else cout<<"Queue is not Empty! " <<endl;

queue.display();

break;

case 5: cout<<endl<<"\t \*\* OPERATION CHOSEN : Checking for Overflow \*\* "<<endl<<endl;

res=queue.isFull();

if (res==1) cout<<"Queue is Full! "<<endl;

else cout<<"Queue is not Full! " <<endl;

queue.display();

break;

}

cout<<endl<<"-> Do you want to continue (y/n) : ";

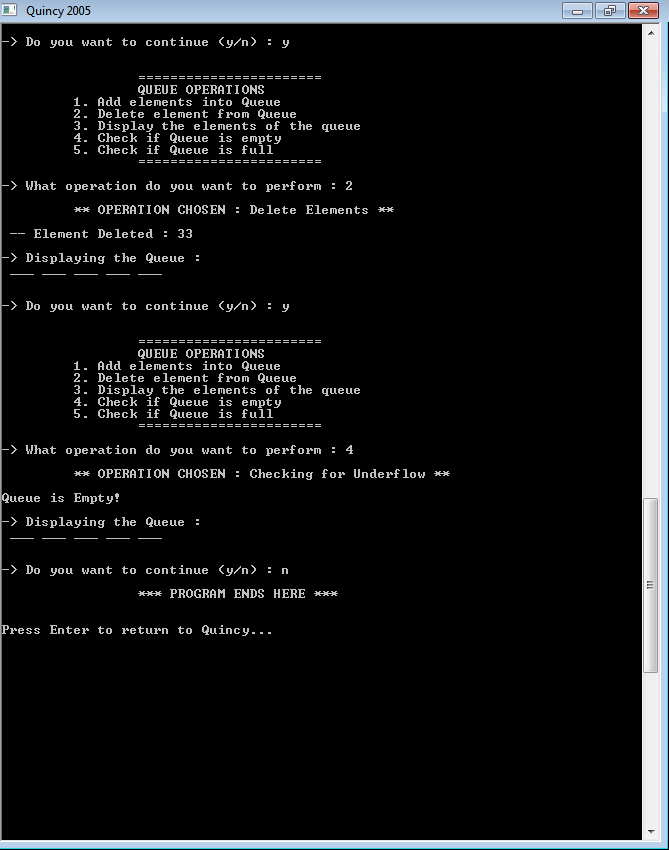
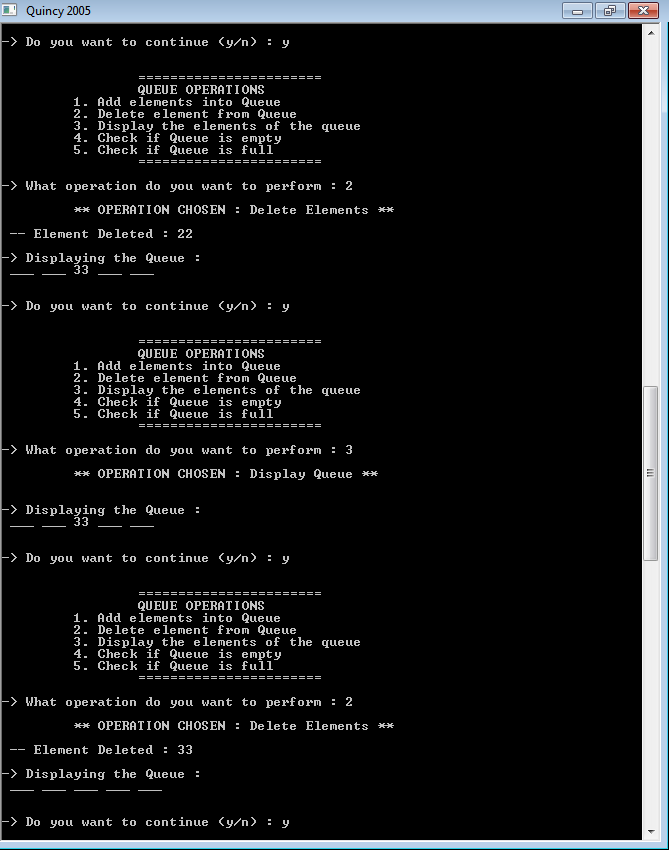
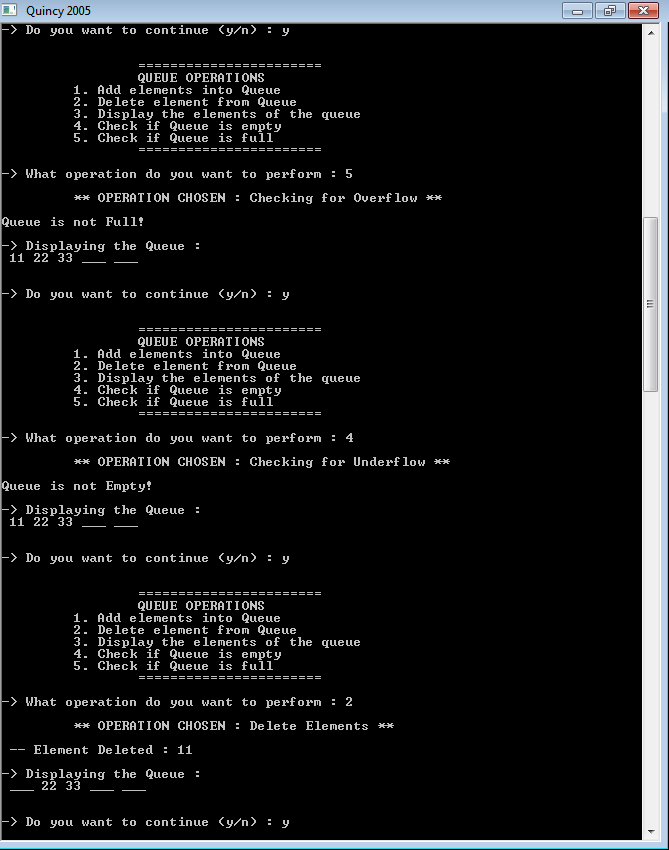
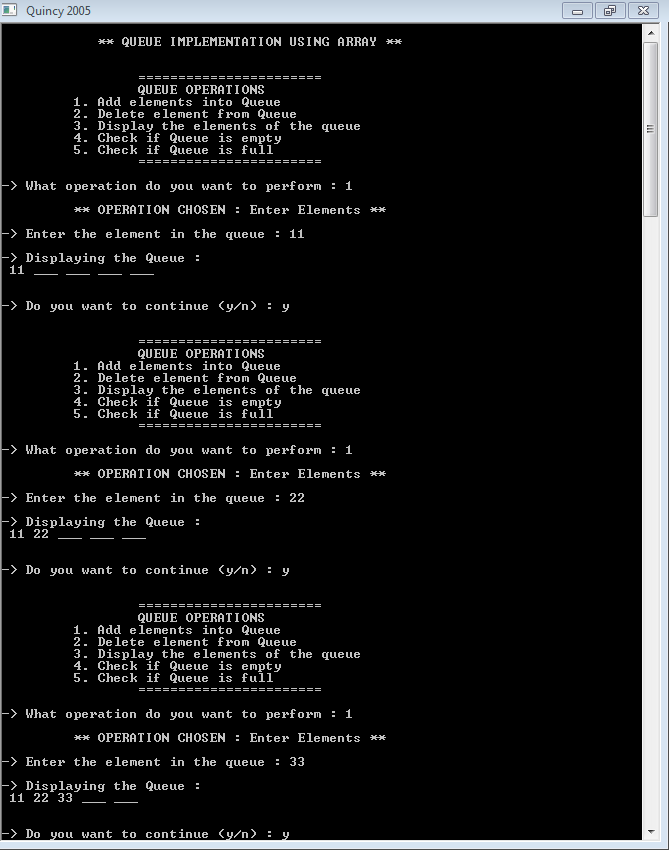
cin>>ch;

}

cout<<endl<<"\t\t \*\*\* PROGRAM ENDS HERE \*\*\*"<<endl<<endl;

return 0;

}



**7 . WAP TO IMPLEMENT STACK USING ARRAYS**

#include<iostream>

#include<iomanip>

using namespace std;

class stacktype{

int top;

int arr[10];

public :

stacktype() {

top=(-1);

}

void push(int num);

int pop();

void display();

int isEmpty();

int isFull();

};

void stacktype::push(int num){

int ele;

for (int i=0; i<num; i++) {

cout<<"Enter the element : ";

cin>>ele;

arr[++top]=ele;

}

return;

}

int stacktype::pop(){

int ele=arr[top--];

return ele;

}

int stacktype::isEmpty(){

int flag;

if (top==-1) flag=1;

else flag=0;

return flag;

}

int stacktype::isFull(){

int flag;

if (top==sizeof(arr)-1) flag=1;

else flag=0;

return flag;

}

void stacktype::display(){

cout<<"-> DISPLAYING THE ARRAY : "<<endl<<endl;

for (int i=0; i<=top; i++)

{

cout<<setw(3)<<arr[i];

}

return;

}

void operationList(){

cout<<setw(20)<<"======================="<<endl;

cout<<"\t STACK OPERATIONS"<<endl;

cout<<"1. PUSH the element into stack"<<endl;

cout<<"2. POP the element from stack"<<endl;

cout<<"3. DISPLAY the elements of stack"<<endl;

cout<<"4. Check if stack is empty"<<endl;

cout<<"5. Check if stack is full"<<endl;

// cout<<"6. Perform linear search in stack"<<endl;

cout<<setw(20)<<"======================="<<endl<<endl;

return;

}

int main()

{

cout<<setw(50)<<endl<<"\*\* STACK IMPLEMENTATION USING ARRAY \*\*"<<endl<<endl;

char ch='y';

int choice,ele, num,res,sz;

stacktype obj;

while(ch=='y' || ch=='Y')

{

operationList();

cout<<"--> Which operation do you want to perform :";

cin>>choice;

cout<<endl;

switch(choice)

{

case 1:

cout<<"\t OPERATION CHOOSEN :: PUSH "<<endl<<endl;

res=obj.isFull();

if (res==1) cout<<"Error : Stack is full, Cannot push elements!"<<endl<<endl;

else

{

cout<<"How many elements do you want to push: ";

cin>>num;

obj.push(num);

}

break;

case 2:

cout<<"\t OPERATION CHOOSEN :: POP"<<endl<<endl;

res=obj.isEmpty();

if (res==1) cout<<"Error : Stack is empty, Cannot pop elements!"<<endl<<endl;

else

{

ele=obj.pop();

cout<<setw(3)<<ele;

}

break;

case 3:

cout<<"\t OPERATION CHOOSEN :: DISPLAY"<<endl<<endl;

obj.display();

break;

case 4:

cout<<"\t OPERATION CHOOSEN :: isEmpty "<<endl<<endl;

res=obj.isEmpty();

if (res==1) cout<<"The stack is Empty"<<endl;

else cout<<"The stack is not Empty"<<endl;

break;

case 5:

cout<<"\t OPERATION CHOOSEN :: isFull"<<endl<<endl;

res=obj.isFull();

if (res==1) cout<<"The stack is full"<<endl;

else cout<<"The stack is not full"<<endl;

break;

default :

cout<<"\t Enter a valid choice!"<<endl;

}

cout<<"\n\n Do you want to continue? (y/n) :";

cin>>ch;

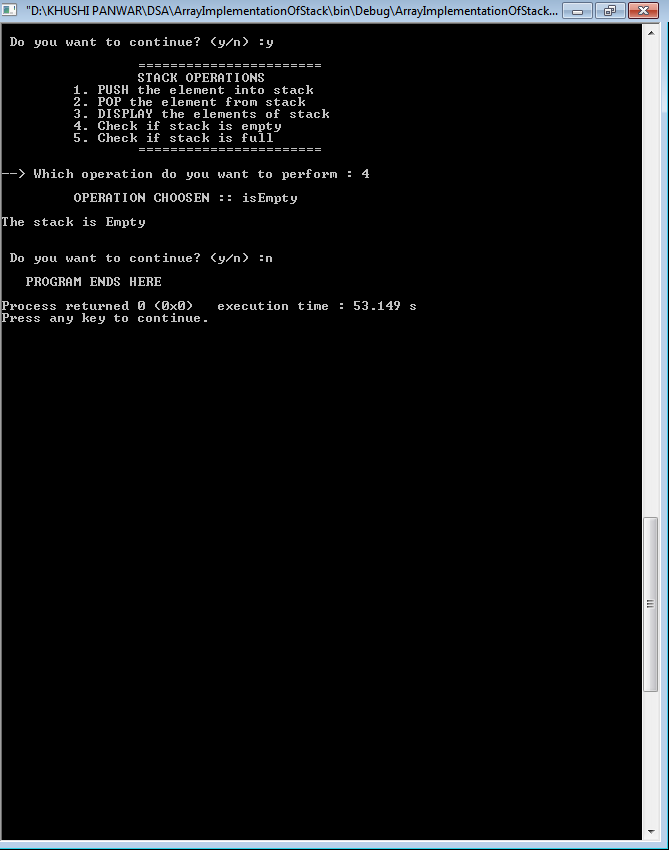
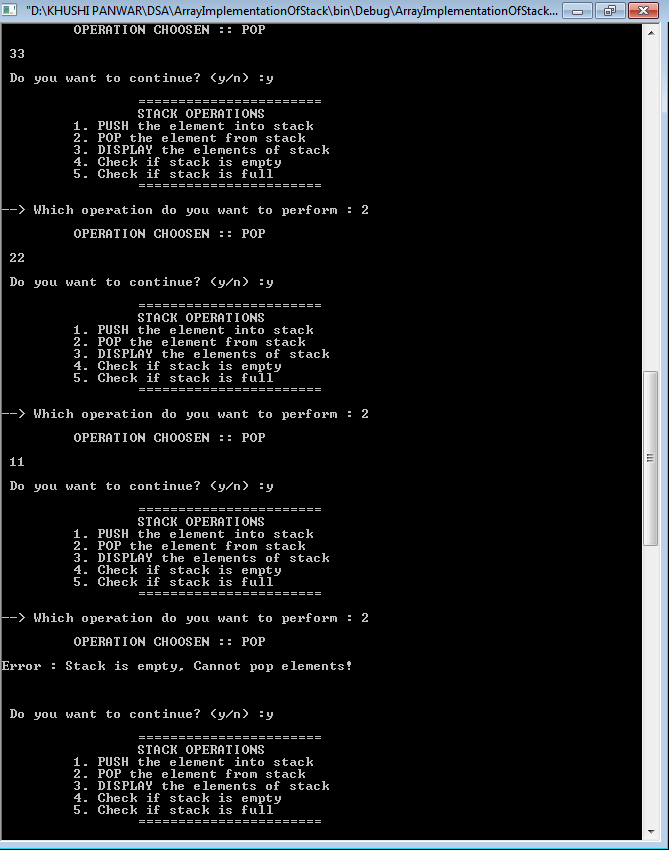
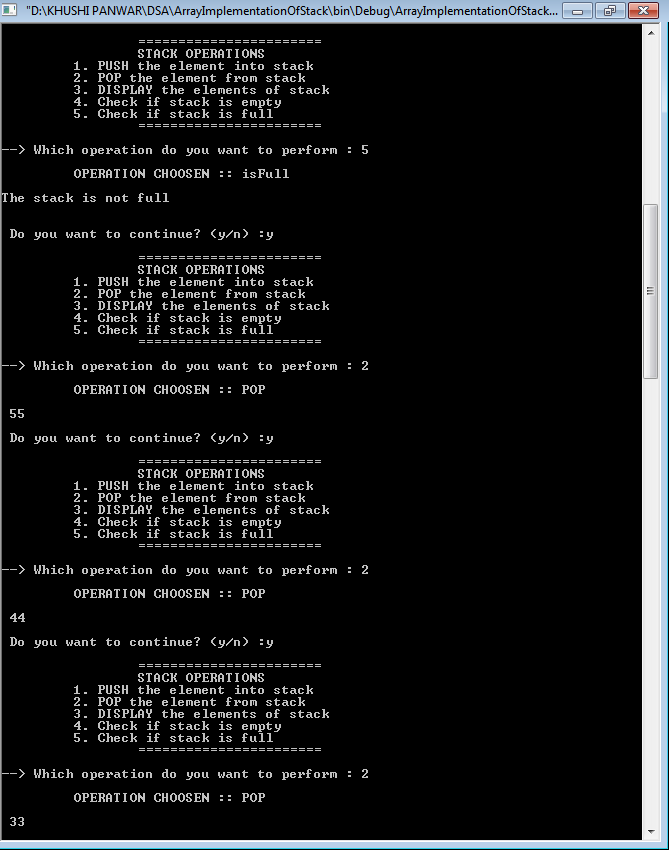
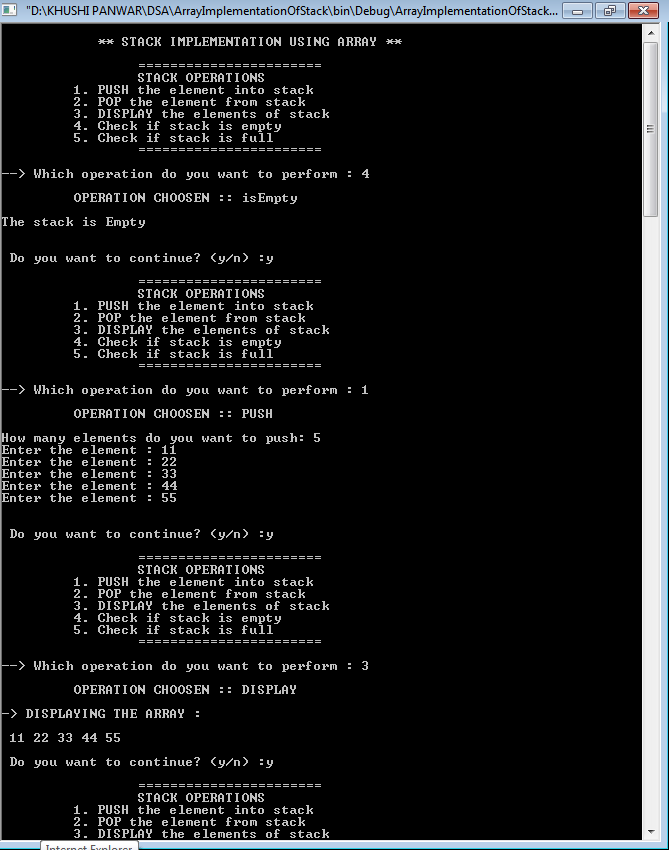
cout<<endl;

}

cout<<setw(20)<<"PROGRAM ENDS HERE"<<endl;

return 0;

}



**6. WAP TO IMPLEMENT STACK USING LINKED LIST**

//implementing stacks using linked lists : "->" is used to refer to the member belonging to the struct "node" type objects.

#include<iostream>

#include <iomanip>

using namespace std;

struct node

{

int info;

node\*next;

};

class stackType

{

node \*top;

public:

stackType()

{

top=NULL;

}

void push(node \*);

int pop();

int isempty();

int display();

node \*createnode(int);

};

int stackType::isempty()

{

if (top==NULL) return 1;

else return -1;

}

node \*stackType::createnode(int x){

node \*temp;

temp=new(node);

temp->info=x;

temp->next=NULL;

return temp;

}

void stackType::push(node \*temp){

if (top==NULL){

top=temp;

}else{

temp->next=top;

top=temp;

}

}

int stackType::display()

{

if(top==NULL) cout<<"-> Stack is empty! "<<endl;

else{

cout<<"-> Contents of stack from top most node are : ";

for (node\* temp=top; temp!=NULL ; temp=temp->next){

cout<<temp->info<<" , ";

}

}

}

int stackType::pop(){

node \*temp;

if (top->next==NULL){

temp=top;

top=NULL;

return temp->info;

delete temp;

}

else{

temp=top;

top=top->next;

return temp->info;

delete temp;

}

}

void menu(){

cout<<setw(30)<<"======================="<<endl;

cout<<"\t\t STACK OPERATIONS"<<endl;

cout<<"\t 1. PUSH the element into stack"<<endl;

cout<<"\t 2. POP the element from stack"<<endl;

cout<<"\t 3. DISPLAY the elements of stack"<<endl;

cout<<"\t 4. Check if stack is empty"<<endl;

cout<<setw(30)<<"\t======================="<<endl<<endl;

return;

}

int main()

{

int choice, ele, val;

char ch='y';

node \*temp1;

stackType st;

do{

menu();

cout<<"-> Select the operation you want to perform : ";

cin>>choice;

switch(choice){

case 1: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : PUSH the elements \*\* "<<endl;

cout<<"-> Enter a value : ";

cin>>val;

temp1=st.createnode(val);

if (temp1!=NULL){

st.push(temp1);

st.display();

}else{

cout<<"\n Compile out of memory";

}

break;

case 2: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : POP the elements \*\* "<<endl;

ele=st.pop();

cout<<"-> Element Popped : "<<ele<<endl;

break;

case 3: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : DISPLAY the elements \*\* "<<endl;

st.display();

break;

case 4: cout<<endl<<setw(40)<<" \*\* OPERATION CHOOSEN : Check emptiness \*\* "<<endl;

ele=st.isempty();

if (ele==1) cout<<endl<<"-> The Stack is Empty! "<<endl;

else cout<<endl<<"-> The Stack is not Empty! "<<endl;

break;

default : cout<<endl<<"!! Select a valid operation !!"<<endl<<endl;

}

cout<<endl<<endl<<"-> Do you want to continue (y/n)?";

cin>>ch;

}while(ch=='y' || ch=='Y');

}

