**BCA 407: Practical on Data Structures**

**Objective: To practically train students in Data structure using C++.**

Implement following data structures and its applications using C++

1. Matrix

2. Stack

3. Queue

4. Single Linked List

5. Bubble Sort

6. Recursion

7. Linear Search

8. Binary Search

9. Tower of Hanoi

10. Adjacency matrix representation of graph

**1. Matrix**

#include<iostream.h>

#include<conio.h>

class Matrix

{

int A[3][3],B[3][3],C[3][3];

public:

void Get();

void Add();

void Sub();

void Mult();

void Div();

};

void Matrix :: Get()

{

cout << "\n Enter matrix 1 elements:";

for(int i=1;i<=3;i++)

{

for(int j=1;j<=3;j++)

{

cin >> A[i][j];

}

}

cout << "\n Enter matrix 2 elements:";

for(i=1;i<=3;i++)

{

for(int j=1;j<=3;j++)

{

cin >> B[i][j];

}

}

}

void Matrix :: Add()

{

cout << "\n Addition : \n";

for(int i=1;i<=3;i++)

{

for(int j=1;j<=3;j++)

{

C[i][j] = A[i][j] + B[i][j];

cout << C[i][j] << "\t";

}

cout << "\n";

}

}

void Matrix :: Sub()

{

cout << "\n Subtraction : \n";

for(int i=1;i<=3;i++)

{

for(int j=1;j<=3;j++)

{

C[i][j] = A[i][j] - B[i][j];

cout << C[i][j] << "\t";

}

cout << "\n";

}

}

void Matrix :: Mult()

{

cout << "\n Multiplication : \n";

for(int i=1;i<=3;i++)

{

for(int j=1;j<=3;j++)

{

C[i][j]=0;

for(int k=1;k<=3;k++)

{

C[i][j] = C[i][j] + A[i][k] \* B[k][j];

cout << C[i][j] << "\t";

}

cout << "\n";

}

}

}

void Matrix :: Div()

{

cout << "\n Division : \n";

for(int i=1;i<=3;i++)

{

for(int j=1;j<=3;j++)

{

C[i][j] = A[i][j] / B[i][j];

cout << C[i][j] << "\t";

}

cout << "\n";

}

}

void main()

{

Matrix a;

int ch;

clrscr();

a.Get();

do

{

cout<<"Enter choice: "<<endl;

cin>>ch;

switch(ch)

{

case 1:

a.Add();

break;

case 2:

a.Sub();

break;

case 3:

a.Mult();

break;

case 4:

a.Div();

break;

case 5:

cout<<"Exit"<<endl;

break;

default:

cout<<"Invalid Choice"<<endl;

}

}while(ch!=5);

getch();

}

**2. Stack**

#include <iostream.h>

#include<conio.h>

class stack

{

int stack[5],val, top;

public:

stack()

{

top=-1;

}

void push();

void pop();

void show();

};

void stack :: push()

{

if(top>=2)

cout<<"Stack Overflow"<<endl;

else

{

cout<<"\n Enter the value:";

cin>>val;

top++;

stack[top]=val;

}

}

void stack :: pop()

{

if(top<=-1)

cout<<"Stack Underflow"<<endl;

else

{

cout<<"The popped element is "<< stack[top] <<endl;

top--;

}

}

void stack :: show()

{

if(top>=0)

{

cout<<"Stack elements are:";

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

cout<<stack[i]<<" ";

cout<<endl;

}

else

cout<<"Stack is empty";

}

void main()

{

int ch;

stack s;

clrscr();

cout<<"1) Push in stack"<<endl;

cout<<"2) Pop from stack"<<endl;

cout<<"3) Display stack"<<endl;

cout<<"4) Exit"<<endl;

do

{

cout<<"Enter choice: "<<endl;

cin>>ch;

switch(ch)

{

case 1:

s.push();

break;

case 2:

s.pop();

break;

case 3:

s.show();

break;

case 4:

cout<<"Exit"<<endl;

break;

default:

cout<<"Invalid Choice"<<endl;

}

}while(ch!=4);

getch();

}

**3. Queue**

#include<iostream.h>

#include<conio.h>

class Queue

{

int a[4],R,F;

public:

Queue()

{

F=R=0;

}

void Insert();

void Delete();

void Show();

};

void Queue :: Insert()

{

int item;

if(R>=3)

{

cout << "\n Queue is full.";

}

else

{

cout << "\n enter item :";

cin >> item;

R++;

a[R]=item;

if(F==0)

{

F=1;

}

}

}

void Queue :: Delete()

{

if(F==0)

{

cout << "\n Empty";

}

else

{

cout << "\n Deleted element is :" << a[F];

if(F==R)

{

F=R=0;

}

else

{

F++;

}

}

}

void Queue :: Show()

{

if(F==0)

{

cout << "\n Empty.";

}

else

{

cout << "\n Elements are : \n";

for(int i=F;i<=R;i++)

{

cout << "\n" << a[i];

}

}

}

void main()

{

Queue s;

int ch;

clrscr();

do

{

cout<<"\n 1: Insert";

cout<<"\n 2: Delete";

cout<<"\n 3: Show";

cout<<"\n 4: Exit";

cout << "\n Enter choice : ";

cin >> ch;

switch(ch)

{

case 1: s.Insert();

break;

case 2: s.Delete();

break;

case 3: s.Show();

break;

default: cout << "\n Wrong Choice.";

}

}while(ch<=3);

}

**4. Single Linked List**

#include<iostream.h>

#include<conio.h>

int cnt=1;

class LinkList

{

int Info,loc;

LinkList \*Link;

public:

void Insert();

void Delete();

void Sort();

void Reverse();

void Display();

};

LinkList \*Start,\*Node,\*Temp1,\*Temp2;

void LinkList :: Insert()

{

int item;

cout << "\n At which location u want to insert item:";

cin >> loc;

if(loc > cnt)

{

cout << "\n Invalid Position.";

getch();

return;

}

Node = new LinkList;

cout << "\n Enter the item";

cin >> item;

Node->Info = item;

Node->Link = NULL;

if(loc == 1)

{

if(Start == NULL)

{

Start=Node;

cnt++;

return;

}

else

{

Node->Link=Start;

Start=Node;

cnt++;

return;

}

}

else

{

Temp1=Start;

Temp2=Start;

for(int i=1; i<loc-1; i++)

Temp1=Temp1->Link;

if(Temp1->Link==NULL)

{

Temp1->Link=Node;

cnt++;

return;

}

else

{

for(int i=1; i<loc; i++)

Temp2=Temp2->Link;

Temp1->Link=Node;

Node->Link=Temp2;

cnt++;

return;

}

}

}

void LinkList :: Delete()

{

int loc;

if(Start == NULL)

{

cout << "\n Link list is empty.";

return;

}

cout << endl << "Enter the Location of Node to be deleted : ";

cin >> loc;

Temp1=Start;

Temp2=Start;

if(loc >= cnt)

{

cout << "Invalid Position.";

return;

}

if(loc==1)

{

Start=Start->Link;

cnt--;

return;

}

for(int i=1;i<loc-1;i++)

Temp1=Temp1->Link;

if(Temp1->Link->Link==NULL)

{

Temp1->Link=NULL;

cnt--;

return;

}

else

{

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

Temp2=Temp2->Link;

Temp1->Link=Temp2;

cnt--;

return;

}

}

void LinkList :: Sort()

{

LinkList \*Current;

for(Current=Start;Current!=NULL;Current=Current->Link)

for(Node=Current->Link;Node!=NULL;Node=Node->Link)

{

if (Current->Info > Node->Info)

{

int t;

t = Current->Info;

Current->Info = Node->Info;

Node->Info = t;

}

}

cout << endl << "Linked List is Sorted !! ";

return;

}

void LinkList :: Reverse()

{

Node=Start->Link;

Temp2=Start;

Temp2->Link=NULL;

while(Node!=NULL)

{

Temp1=Node->Link;

Node->Link=Temp2;

Temp2=Node;

Node=Temp1;

}

Start=Temp2;

cout << endl << "Linked List is Reversed !! ";

}

void LinkList :: Display()

{

if(Start==NULL)

{

cout << "\n There are no elements in list.";

}

else

{

for(Node=Start;Node!=NULL;Node=Node->Link)

cout << Node->Info << endl;

}

}

void main()

{

LinkList s;

int ch;

clrscr();

cout << "\n CHOICES FOR DOUBLY LINKED LIST.....";

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

cout << "\n 1. INSERT";

cout << "\n 2. DELETE";

cout << "\n 3. DISPLAY";

cout << "\n 4. SORT";

cout << "\n 5. REVERSE";

cout << "\n 6. Exit";

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

do

{

cout << "\n Enter choice : ";

cin >> ch;

switch(ch)

{

case 1: s.Insert();

break;

case 2: s.Delete();

break;

case 3: s.Display();

break;

case 4: s.Sort();

break;

case 5: s.Reverse();

break;

default: cout << "\n Wrong Choice.";

}

}while(ch<=5);

}

**5. Bubble Sort**

#include<iostream.h>

#include<conio.h>

class BSort

{

int a[10],N;

public:

void Get();

void Sort();

void Put();

};

void BSort :: Get()

{

cout << "\n Enter the how many elements in array: ";

cin >> N;

cout << "\n Enter the elements : ";

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

{

cin >> a[i];

}

}

void BSort :: Sort()

{

int temp;

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

{

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

{

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

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

}

void BSort :: Put()

{

cout << "\n Elements after sorting are : \n";

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

{

cout << a[i] << "\n";

}

}

void main()

{

BSort b;

clrscr();

b.Get();

b.Sort();

b.Put();

getch();

}

**6. Recursion**

**Compute Factorial of given Number using recursion.**

#include<iostream.h>

#include<conio.h>

class FactRecursion

{

long n;

public:

long fact(int);

};

long FactRecursion :: fact(int n)

{

if ((n==0)||(n==1))

return 1;

else

return n\*fact(n-1);

}

void main()

{

FactRecursion f;

clrscr();

int n;

cout<<"\n Enter the number:";

cin >> n;

cout<<"Factorial of "<<n<<" is "<<f.fact(n);

getch();

}

**Prime Number:**

#include<iostream.h>

#include<conio.h>

class Rprime

{

public:

int checkPrime(int,int);

};

int Rprime :: checkPrime(int num1, int index)

{

if(num1<2)

{

return 0;

}

if (num1 == 2 || num1==3){

return 1;

}

if (num1 % index == 0)

{

return 0;

}

if (index <= num1/2)

{

return 1;

}

int result=checkPrime(num1, index+1);

return (result);

}

void main()

{

Rprime r;

int Num;

clrscr();

cout<<"\n Enter the number:";

cin>>Num;

if (r.checkPrime(Num,2)==1)

{

cout <<Num<<" is a Prime number !";

}

else

{

cout <<Num<<" is non Prime!";

}

getch();

}

**7. Linear Search**

#include<iostream.h>

#include<conio.h>

class Array

{

int A[10],N;

int k,item;

public:

void Get();

void LSearch();

};

void Array :: Get()

{

cout << "Enter the size of array :";

cin >> N;

cout << "Enter elements :";

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

{

cin >> A[i];

}

}

void Array :: LSearch()

{

cout << "\n Enter item:";

cin >> item;

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

{

if(A[i]==item)

{

cout << "Element " <<item << " is found at " << i+1

<<" location\n\n";

}

}

}

void main()

{

clrscr();

Array a;

a.Get();

a.LSearch();

getch();

}

**8. Binary Search**

#include<iostream.h>

#include<conio.h>

class Array

{

int A[10],N;

int k,item;

public:

void Get();

void Sort();

void BSearch();

};

void Array :: Get()

{

cout << "Enetr N:";

cin >> N;

cout << "Enter element :";

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

{

cin >> A[i];

}

}

void Array :: Sort()

{

int temp;

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

{

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

{

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

{

temp=A[i];

A[i]=A[j];

A[j]=temp;

}

}

}

cout << "\n Elements after sorting :\n";

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

{

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

}

}

void Array :: BSearch()

{

int Beg=0, End=N;

int Mid = (Beg+End)/2;

int Loc;

cout << "\n Enter element to search : ";

cin >> item;

while(Beg <= End && A[Mid] != item)

{

if(item < A[Mid])

{

End = Mid-1;

}

else

{

Beg = Mid+1;

}

Mid = (Beg+End)/2;

}

if(A[Mid] == item)

{

Loc=Mid;

cout << "Item is on " << Loc+1 << " Location\n\n";

}

else

{

Loc=NULL;

cout << "Element is not found";

}

}

void main()

{

clrscr();

Array a;

a.Get();

a.Sort();

a.BSearch();

getch();

}

**9. Tower of Hanoi**

#include <iostream.h>

#include <conio.h>

class Tower

{

int n;

public:

int GetData()

{

cout <<"\n Enter how many disks : ";

cin >> n;

return n;

}

void tower(int n,char A,char B,char C)

{

if(n==1)

cout << A << "->" << C << endl;

else

{

tower(n-1,A,C,B);

cout << A << "->" << C <<endl;

tower(n-1,B,A,C);

}

}

};

void main()

{

Tower t;

clrscr();

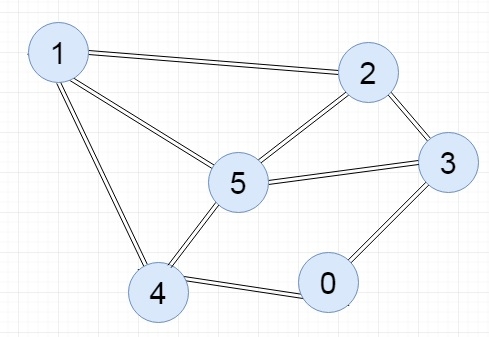
int x=t.GetData();

t.tower(x,'A','B','C');

getch();

}

**10. Adjacency matrix representation of graph**



**Output:**

|  | **0** | **1** | **2** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 0 | 0 | 1 | 1 | 0 |
| **1** | 0 | 0 | 1 | 0 | 1 | 1 |
| **2** | 0 | 1 | 0 | 1 | 0 | 1 |
| **3** | 1 | 0 | 1 | 0 | 0 | 1 |
| **4** | 1 | 1 | 0 | 0 | 0 | 1 |
| **5** | 0 | 1 | 1 | 1 | 1 | 0 |

#include<iostream.h>

#include<conio.h>

class AdMatrix

{

public:

int vertArr[20][20]; //the adjacency matrix initially 0

void displayMatrix(int);

void addEdge(int,int);

};

void AdMatrix :: displayMatrix(int v)

{

int i, j;

for(i = 0; i < v; i++) {

for(j = 0; j < v; j++) {

cout << vertArr[i][j] << " ";

}

cout << endl;

}

}

void AdMatrix :: addEdge(int u, int v) { //function to add edge into the matrix

vertArr[u][v] = 1;

vertArr[v][u] = 1;

}

void main()

{

AdMatrix a;

int v = 6; //there are 6 vertices in the graph

clrscr();

a.addEdge(0, 4);

a.addEdge(0, 3);

a.addEdge(1, 2);

a.addEdge(1, 4);

a.addEdge(1, 5);

a.addEdge(2, 3);

a.addEdge(2, 5);

a.addEdge(5, 3);

a.addEdge(5, 4);

a.displayMatrix(v);

getch();

}