Data Structured

Lab Manual

Week 1	Functions
Week 2	Arrays one dimension
Week 3	Array multidimensional
Week 4	Pointers
Week 5	Structures
Week 6	Linear Lists
Week 7	Stacks using array
Week 8	Queues using array
Week 9	Linked List
Week 10	Stacks using Linked List
Week 11	Queues using Linked List
Week 12	Trees

Lab#1

Functions

Example1

Write c++ program to add two integer numbers

```
# include <iostream>
using namespace std;

int add(int, int);              //Function prototype(declaration)

int main() {
    int num1, num2, sum;
    cout<<"Enters two numbers to add: ";
    cin>>num1>>num2;
    sum = add(num1,num2);          //Function call
    cout<<"Sum = "<<sum;
    return 0;
}

int add(int a,int b) {             //Function declarator
    int add;
    add = a+b;
    return add;                   //Return statement
}
```

Example2

Write c++ program to compare two numbers which bigger

```
#include <iostream>
using namespace std;
void compare(int a,int b)

{
    if(a>b)  cout<<a<<" is bigger than "<<b<<endl;

    else if(a<b) cout<<a<<" is less than "<<b<<endl;

    else

        cout<<a<<" is equal "<<b<<endl;
}
int main()

{

    compare(4,5);
    compare(7,3);
    compare(3,3);
    return 0;

}
```

Example3

Write c++ program to create power function

```
#include <iostream>
using namespace std;

int power(int value, int power) {
    int result = 1;

    for (int i = 0; i < power; ++i) {
        result *= value;
    }

    return (result);
}


int main()
 {
int num, Pow;

cout << "Enter a value: ";
cin >> num;
cout << "Enter a power: ";
cin >> Pow;
int x= power(num, Pow) ;
cout <<x<<endl;

    system("pause");
    return 0;
}
```

Example4

Write c++ program to check if your degree valid or not

```
#include <iostream>
#include <string.h>
using namespace std;

bool validateMark (int mark) {
    if ( mark >= 0 && mark <= 100)
       return true;
    else
       return false;
}

int main () {
   int mark;
   cout << "Enter mark: ";
   cin >> mark;

   if (!validateMark (mark))
      cout << "\nInvalid Mark - Out of Range\n";

   system ("pause");
}
```

Lab#2

Arrays one dimension

Example1:

- **Write a program that asks the user to type 10 integers of an array.**
- **The program must compute and write how many integers are greater than or equal to 10.**

```
#include <iostream>
using namespace std;
int main()
{
    int a[10],i,b=0, N=5;
    cout<<"enter the array\n";
    for(i=0;i<N;i++)
    {
      cin>>a[i];
      if (a[i]>=10)
      b++;
    }

    cout<<"the number of integers greater or equal to 10 is: "<<b;
   system("pause");
 return 0;
}
```

Example2:

- **Write a program that asks the user to type 10 integers of an array and an integer V.**
- **The program must search if V is in the array of 10 integers**
- **The program writes "V is in the array" or "V is not in the array**

```
#include <iostream>
using namespace std;

const int N = 10;
int main ()
{
    int N;
    cin>>N;
    int t[N], i=0, V;

    for (i = 0; i < N; i++)
    {
        cout << "Type an integer: ";
        cin >> t[i];
    }
    cout << "Type the value of V: ";
    cin >> V;
    for (i = 0; i < N; i++)
    {
        if (t[i] == V)
        {
          cout << "V is in the array" << endl;
          return 0;
        }
    }
    cout << "V is not in the array" << endl;
    system("pause");
    return 0;
}
```

Example3

- **Write a program that asks the user to type 10 integers of an array and an integer value V .**
- **The program must search if the value V exists in the array and must remove the first occurrence of V**
- **Shifting each following element left and adding a zero at the end of the array. The program must then write the final array.**

```
#include <iostream>
using namespace std;

const int N=10;
int main()
{
    int t[N],i,j,V;
    bool found;
    for(i=0;i<N;i++)
    {
        cout << "Type an integer: ";
        cin >> t[i];
    }
    cout << "Type the value of V: ";
    cin >> V;

    for (i=0;i<N;i++)
        if (t[i]==V)
        {
            for (j=i;j<N-1;j++)
                t[j]=t[j+1];
            t[N-1]=0;
            break;
        }

    for(i=0;i<N;i++)
        cout << t[i] << endl;
system("pause");
    return 0;
}
```

Example4:

- **Write a program that asks the user to type 5 integers of an array.**
- **The program will then display either "the array is growing", " the array is decreasing", "the array is constant", or "the array is growing and decreasing".**

```
#include <iostream>
using namespace std;
int main()
{
    int N=5;
    int a[N],i;
    int up=0,down=0;

    cout << "Please enter an integer: ";
    cin >> a[0];
    for(i=1;i<N;i++)
    {
        cout << "Please enter an integer: ";
        cin >> a[i];
        if(a[i-1]>a[i]) down++;
        if(a[i-1]<a[i]) up++;
    }

   if(up!=0 && down !=0)
       cout <<"Decresing and incresing" <<endl;
   if(up==0 && down ==0)
       cout <<"Constant" <<endl;

   if(up==N-1)
     cout <<" incresing" <<endl;
   if(down==N-1)
     cout <<" Decresing"<< endl;
            system("pause");
    return 0;
}
```

Example 5

- **Write a program that asks the user to type 10 integers of an array. The program will then sort the array in descending order and display it.**

```
const int N=10;
int main()
{
    int a[N],i,j,min,imin,tmp;

    for (i=0;i<N;i++)
    {
        cout << "Please enter an integer: ";
        cin >> a[i];
    }

    for (i=0;i<N-1;i++)
    {
        imin=i;
        min=a[i];
        for (j=i+1;j<N;j++)
        {
            if (a[j]<min)
            {
                min=a[j];
                imin=j;
            }
         }
        tmp=a[imin];
        a[imin]=a[i];
        a[i]=tmp;
    }
    cout << "The sorted array:" << endl;
    for (i=0;i<N;i++)
    cout << "a[" << i << "] = " << a[i] << endl;
 system("pause");
```

Assignment

- **Write a program which takes 2 arrays of 10 integers each, A and B. C is an array with 20 integers.**
- **The program should put into (C) the appending of B and A, the first 10 integers of C from array A, the latter 10 from B.**
- **Then the program should display c.**

Lab#3

Arrays multidimensional

Example1

- **Define array[2][4] with char values**
- **Update some value of array and display it**

```
#include <iostream>
using namespace std;

int main() {
  string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
  };

letters[0][0] = "Z";

  for(int i = 0; i < 2; i++) {
    for(int j = 0; j < 4; j++) {
      cout << letters[i][j] << "\n";
    }
  }

  system("pause");
  return 0;
}
```

Example2:

- a small game of Battleship

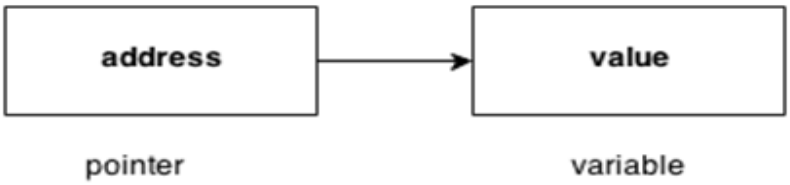
```
#include <iostream>
using namespace std;

int main() {
  // We put "1" to indicate there is a ship.
  bool ships[4][4] = {
    { 0, 1, 1, 0 },
    { 0, 0, 0, 0 },
    { 0, 0, 1, 0 },
    { 0, 0, 1, 0 }
  };

  // Keep track of how many hits the player has and how many turns they have played in these
  //variables
  int hits = 0;
  int numberOfTurns = 0;

  // Allow the player to keep going until they have hit all four ships
  while (hits < 4) {
    int row, column;

    cout << "Selecting coordinates\n";

    // Ask the player for a row
    cout << "Choose a row number between 0 and 3: ";
    cin >> row;

    // Ask the player for a column
    cout << "Choose a column number between 0 and 3: ";
    cin >> column;

    // Check if a ship exists in those coordinates
    if (ships[row][column]) {
      // If the player hit a ship, remove it by setting the value to zero.
      ships[row][column] = 0;

      // Increase the hit counter
      hits++;

      // Tell the player that they have hit a ship and how many ships are left
      cout << "Hit! " << (4-hits) << " left.\n\n";
    } else {
      // Tell the player that they missed
      cout << "Miss\n\n";
    }

    // Count how many turns the player has taken
    numberOfTurns++;
  }

  cout << "Victory!\n";
  cout << "You won in " << numberOfTurns << " turns";

  return 0;
}
```

Lab#4

Pointers

C++ Pointers

The pointer in C++ language is a variable, it is also known as locator or indicator that points to an address of a value.



Usage of pointer

There are many usage of pointers in C++ language.

1) Dynamic memory allocation

In c language, we can dynamically allocate memory using malloc() and calloc() functions where pointer is used.

2) Arrays, Functions and Structures

Pointers in c language are widely used in arrays, functions and structures. It reduces the code and improves the performance.

Symbols used in pointer

Symbol	Name	Description
& (ampersand sign)	Address operator	Determine the address of a variable.
* (asterisk sign)	Indirection operator	Access the value of an address.

Declaring a pointer

The pointer in C++ language can be declared using * (asterisk symbol).

```
int *  a; //pointer to int
char *  c; //pointer to char
```

Example1

```
#include <iostream>
using namespace std;
int main()
{
int number=30;
int *   p;
p=&number;//stores the address of number variable
cout<<"Address of number variable is:"<<&number<<endl;
cout<<"Address of p variable is:"<<p<<endl;
cout<<"Value of p variable is:"<<*p<<endl;
   return 0;
}
```

Example2

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string food = "Pizza";  // A string variable
  string* ptr = &food;  // A pointer variable that stores the address of food

  cout << food << "\n";
  cout << &food << "\n";

  cout << ptr << "\n";
  cout << *ptr << "\n";

  *ptr = "Hamburger";

cout << *ptr << "\n";
cout << food << "\n";
  system("pause");
  return 0;
}
```

Exampe3

Pointer Program to swap 2 numbers without using 3rd variable

```
#include <iostream>
using namespace std;
int main()
{
int a=20,b=10,*p1=&a,*p2=&b;
cout<<"Before swap: *p1="<<*p1<<" *p2="<<*p2<<endl;
*p1=*p1+*p2;
*p2=*p1-*p2;
*p1=*p1-*p2;
cout<<"After swap: *p1="<<*p1<<" *p2="<<*p2<<endl;
  system("pause");
   return 0;
}
```

Call by value and call by reference in C++

Difference between call by value and call by reference in C++

No.	Call by value	Call by reference
1	A copy of value is passed to the function	An address of value is passed to the function
2	Changes made inside the function is not reflected on other functions	Changes made inside the function is reflected outside the function also
3	Actual and formal arguments will be created in different memory location	Actual and formal arguments will be created in same memory location

Example4

```
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
 int swap;
 swap=*x;
 *x=*y;
 *y=swap;
}
int main()
{
 int x=500, y=100;
 swap(&x, &y);  // passing value to function
 cout<<"Value of x is: "<<x<<endl;
 cout<<"Value of y is: "<<y<<endl;
 return 0;
}
```

Lab#5

Structures

Defining a Structure:

To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member, for your program. The format of the struct statement is this:

```
struct [structure tag]
{
   member definition;
   member definition;
   ...
   member definition;
} [one or more structure variables];
```

Here is the way you would declare the Book structure:

```
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
}book;
```

Accessing Structure Members:

To access any member of a structure, we use the **member access operator (.)**. The member access operator is coded as a period between the structure variable name and the structure member that we wish to access. You would use **struct** keyword to define variables of structure type. Following is the example to explain usage of structure:

```
#include <iostream>
#include <cstring>

using namespace std;

struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};

int main( )
{
   struct Books Book1;        // Declare Book1 of type Book
   struct Books Book2;        // Declare Book2 of type Book
```

```
   // book 1 specification
   strcpy( Book1.title, "Learn C++ Programming");
   strcpy( Book1.author, "Chand Miyan");
   strcpy( Book1.subject, "C++ Programming");
   Book1.book_id = 6495407;

   // book 2 specification
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Yakit Singha");
   strcpy( Book2.subject, "Telecom");
   Book2.book_id = 6495700;

   // Print Book1 info
   cout << "Book 1 title : " << Book1.title <<endl;
   cout << "Book 1 author : " << Book1.author <<endl;
   cout << "Book 1 subject : " << Book1.subject <<endl;
   cout << "Book 1 id : " << Book1.book_id <<endl;

   // Print Book2 info
   cout << "Book 2 title : " << Book2.title <<endl;
   cout << "Book 2 author : " << Book2.author <<endl;
   cout << "Book 2 subject : " << Book2.subject <<endl;
   cout << "Book 2 id : " << Book2.book_id <<endl;

   return 0;
}
```

The typedef Keyword

There is an easier way to define structs or you could "alias" types you create. For example:

```
typedef struct
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
}Books;
```

Now, you can use *Books* directly to define variables of *Books* type without using struct keyword. Following is the example:

```
Books Book1, Book2;
```

Lab#6
Linear List

```
#include<iostream>
#include<assert.h>
using namespace std;
const int max_size=5;
      class list
      {
        private:
                int listarray[max_size] ;
                int numberofelement;
                int currentposition;
        public:
                list();
                void insert( const int &elem);
                bool first (  int &elem);
                bool next   (  int &elem);
                int number_of_element ();

      };
      list::list()
      {
           numberofelement=0;
           currentposition=-1;
      }
      void list::insert(const int &elem)
      {
           if(numberofelement<max_size)
           {
               listarray[numberofelement]=elem;
               numberofelement++;
           }
           else
        {
            cout<<"List is Full";
            return;
        }
    }
     bool list::first( int &elem)
    {
         if(numberofelement == 0)
             return false;
         else
         {
             currentposition=0;
             elem=listarray[currentposition];
             return true;
         }
    }
     bool list::next( int &elem)
    {
         assert(numberofelement>0);
         if(currentposition>=numberofelement-1)
             return false;
         else
         {
             currentposition++;
             elem=listarray[currentposition];
             return true;
         }
    }
```

```
      int list::number_of_element()
      {
            return numberofelement;
      }
int main()
{
    list L;
    int i;
    int x=0;
    while( x < max_size-1)
    {
        x=L.number_of_element();
        cout<<"enter element("<<x<<")\n";
        cin>>i;
        L.insert(i) ;

    }
    cout<<"Here are the element of your List\n";
    int elem;
    bool notempty(L.first(elem));
    while(notempty)
    {
            cout<<"element "<<elem<<endl;
            notempty=L.next(elem);
    }

    system("pause");
  return 0;
}
```

Lab Assignment

Search for application used linear list

Lab#7
Stacks using Array

```
#include<iostream>
using namespace std;
const int max_stack=5;
class stack
{
      public:
             int stk[max_stack];
             int top;
             stack()
              {
                top=-1;
              }
    ////////////////////////////////////////////////////////////////////////////////
            void push()
              {
                 int x;
                 cout <<"enter the element  ";
                 cin >>x;
                 if(top > max_stack)
                 {
                         cout <<"stack over flow";

                 }
                 else
                 {
                     ++top;
                     stk[top]=x;
                     cout <<"inserted  " <<x;
                 }
               }
    ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////////////////
             void pop()
               {
                    if(top <0)
                    {
                         cout <<"you cannot, stack is empty";
                         return;
                    }
                    else
                    {
                        int data=stk[top];
                        top--;
                        cout <<"deleted  " <<data;
                    }
                }
    ////////////////////////////////////////////////////////////////////////////////
             void display()
               {
                   if(top<0)
                    {
                          cout <<" stack empty";
                    }
                    else
                    {
                        cout <<"Stack Elements\n";
                        cout <<"---------------\n";
                          for(int i=top;i>=0;i--)
                              cout <<"Element "<<i<<"= "<<stk[i]<<endl;
                    }
                }
};
    ////////////////////////////////////////////////////////////////////////////////
```

```
main()
{
      int ch;
      stack st;
      while(1)
         {
               cout<<"\n*****************************************************";
               cout <<"\n1.push    2.pop    3.display    4.exit\n";
               cout<<"*****************************************************\n";
               cout<<"Enter ur choice   ";
               cin >> ch;
               switch(ch)
                {
                 case 1:  st.push(); break;
                 case 2:  st.pop();  break;
                 case 3:  st.display();break;
                 case 4:  exit(0);
                 }

          }


system("pause");
return (0);
}
```

Lab#8
Queues using arrays

```
#include<stdlib.h>
#include<iostream>
using namespace  std;
int const MAX=5;
int rear=-1;
int front=-1;
int queue[MAX];
//////////////////////////////////////////////////////////////////
void enqueue()
{
  int element;
  if(rear==(MAX-1))
  {
     cout<<"\nQueue Full\n";
  }
  else
  {
    if(front==-1)
        front=0;
    cout<<"\nEnter the new element\n";
    cin>>element;
    rear=rear+1;
    queue[rear]=element;
    cout<<"\nElement "<<element<<" is added\n";
  }
}
//////////////////////////////////////////////////////////////////
void dequeue()
{
  if(front==-1||front>rear)
  {
    cout<<"Queue is empty\n";
  }
  else
  {
    cout<<"Element  "<<queue[front]<< "  is deleted from queue\n";

    for(int i=0;i<MAX-1;i++)
    {
       queue[i]=queue[i+1];
    }
    rear=rear-1;

  }
}
//////////////////////////////////////////////////////////////////
void display()
{

  if(front==-1||front>rear)
  {
    cout<<"Queue is empty\n";
  }
  else
  {
   cout<<"Queue\n";
   for(int i=front;i<=rear;i++)
   {
      cout<<queue[i]<<endl;
   }
  }
}
//////////////////////////////////////////////////////////////////
```

```
int main()
{
  int ch;
  while(1)
  {
  cout<<"\n*****************************************";
  cout<<"\n1.Insert   2.Delete   3.Display   4.Exit\n";
  cout<<"*****************************************";
  cout<<"\nChoose your option    ";
  cin>>ch;
  switch(ch)
  {
   case 1: enqueue();
       break;
   case 2: dequeue();
       break;
   case 3: display();
       break;
   case 4: exit(0);
  }
   }
   system("pause");
return 0;
}
//////////////////////////////////////////////////////////////////
```

Lab#9
Linked List

```
#include<iostream>

using namespace std;

struct node{
    int info;
    node *next;
};
//////////////////////////////////////////////////////////////////
class List{
    public:
        node *last;
        node *first;
        int count;
        List();

        void insert_firstelement();
        void insert_end();
        void insert_middle();

        void display();
};
//////////////////////////////////////////////////////////////////
List::List()
{
    last = NULL;
    first = NULL;
    count=-1;
}
void List::insert_firstelement()
{

    int data;
    node *temp = new node;
    cout<<"Enter the data to insert in begaining of list: ";
    cin>>data;
    temp->info= data;
    if(first==NULL)
    {
        temp->next = NULL;
        first=temp;
        last=temp;
    }
    else
    {
        temp->next = first;
        first=temp;
    }
    count++;
}
//////////////////////////////////////////////////////////////////
void List::insert_end()
{
    int data;
    node *temp = new node;
    cout<<"Enter the data to insert in end of list: ";
    cin>>data;
    temp->info= data;
    temp->next = NULL;
    last->next = temp;
```

```
    last = temp;
    count++;
}

//////////////////////////////////////////////////////////
void List::insert_middle()
{
    int data,pre;
    node *temp = new node;
    cout<<"Enter the data to enqueue: ";
    cin>>data;
    cout<<"where you would to enqueue this value?"<<endl<<"please insert number of node
    cin>>pre;
    if(pre<count)
    {
        temp->info= data;
        node *p1; node *p2;
        p1 = first;
        for(int i=0;i<=pre;i++)
        {
              if(i==pre)
              {
                   p2=p1;
                   p2=p2->next;
                   p1->next = temp;
                   temp->next=p2;
              }
                   else
                        p1=p1->next;
          }
          count++;
     }
    else
          cout<<"you cannot insert in this location";

}
//////////////////////////////////////////////////////////////////
void List::display()
{
     node *p;
     p = first;
     if(first == NULL){
          cout<<"\nNothing to Display\n";
     }
     else
     {
          while(p!=NULL){
               cout<<endl<<p->info;
               p = p->next;
          }
     }
}
//////////////////////////////////////////////////////////////////
int main()
{
```

```
List list1;
int choice;
while(true){
    cout<<"\n_____";
    cout<<"\n.1.insert_firstelement   2.insert in middle  3.insert_end   4.display  5.quit";
    cout<<"\n_____\n";
    cout<<"\nEnter your choice:    ";
    cin>>choice;
    switch(choice){
        case 1:
            list1.insert_firstelement();
            break;
        case 2:
            list1.insert_middle();
            break;
        case 3:
             list1.insert_end();
             break;
        case 4:
            list1.display();
            break;
        case 5:
            exit(0);
            break;
        default:
            cout<<"\nInvalid Input. Try again! \n";
            break;
    }
}
return 0;
```

Lab Assignment

Search for application used Linked list

Lab#10

Stacks Using Linked List

```
#include<iostream>
#include<cstdlib>
#include<malloc.h>
#include<conio.h>
using namespace std;
struct node{
     int info;
     node *next;
};
class stack{
     node *top;
     public:
          stack();
          void push();
          void pop();
          void display();
};
stack::stack(){
     top = NULL;
}
void stack::push(){
     int data;
     struct node *p;
     cout<<"Enter a Number to insert:";
     cin>>data;
     p = new node;
     p->info = data;

     if(top!=NULL)
          p->next = top;
     else
          p->next = NULL;
    top = p;
    cout<<"\nNew item inserted"<<endl;
}
void stack::pop(){
     node *temp;
    if(top==NULL){
        cout<<"\nThe stack is Empty"<<endl;
    }else{
        temp = top;
        top = top->next;
        cout<<"\nThe value popped is "<<temp->info<<endl;
        delete temp;
    }
}
void stack::display(){
     node *p = top;
    if(top==NULL){
        cout<<"\nNothing to Display\n";
    }else{
        cout<<"\nThe contents of Stack\n";
        while(p!=NULL){
            cout<<p->info<<endl;
            p = p->next;
        }
    }
}
int main(){
    stack s;
    int choice;
```

```
    do{
        cout<<"\nEnter your choice:";
        cout<<"\n1. PUSH\n2. POP\n3. DISPLAY\n4. EXIT\n";
        cin>>choice;
        switch(choice){
            case 1:
                s.push();
                break;
            case 2:
                s.pop();
                break;
            case 3:
                s.display();
                break;
            case 4:
                exit(0);
                break;
            default:
                cout<<"Invalid Choice";
                break;
        }
    }while(choice);
    getch();
    return 0;
}
```

Lab Assignment

Search for application used stacks

Lab#11
Queues Using Linked List

```
#include<iostream>
using namespace std;
struct node{
    int info;
    node *next;
};
//////////////////////////////////////////////////////////////////
class Queue{
   private:
        node *rear;
        node *front;
    public:
        Queue();
        void enqueue();
        void dequeue();
        void display();
};
//////////////////////////////////////////////////////////////////
Queue::Queue()
{
    rear = NULL;
    front = NULL;
}
//////////////////////////////////////////////////////////////////
void Queue::enqueue()
{
    int data;
    node *temp = new node;
    cout<<"Enter the data to enqueue: ";
    cin>>data;
    temp->info= data;
    temp->next = NULL;
     if(front == NULL)
     {
         front = temp;
     }
     else
     {
         rear->next = temp;
     }
     rear = temp;
 }
 //////////////////////////////////////////////////////////////////
```

```
void Queue::dequeue()
{
    node *temp;
    if(front == NULL)
    {
        cout<<"\nQueue is Emtpty\n";
    }
    else
    {
        temp = front;
        front = front->next;
        cout<<"The data Dequeued is "<<temp->info;
        delete temp;
    }
}
////////////////////////////////////////////////////////////////////
void Queue::display()
{
    node *p;
    p = front;
    if(front == NULL){
        cout<<"\nNothing to Display\n";
    }
    else
    {
        while(p!=NULL){
            cout<<endl<<p->info;
            p = p->next;
        }
    }
}
int main()
{
    Queue queue;

    int choice;
    while(true){
        cout<<"\n_____________________________________________";
        cout<<"\n1.Enqueue   2.Dequeue   3.Display   4.Quit";
        cout<<"\n_____________________________________________\n";
        cout<<"\nEnter your choice:    ";
        cin>>choice;
        switch(choice){
            case 1:
                queue.enqueue();
                break;
            case 2:
                queue.dequeue();
                break;
            case 3:
                queue.display();
                break;
            case 4:
                exit(0);
                break;
            default:
                cout<<"\nInvalid Input. Try again! \n";
                break;
        }
    }
    return 0;
}
```

Lab Assignment

Search for application used Queue

Lab#12
Trees

```
#include <iostream>
using namespace std;

struct tree_node
{
     tree_node *left;
     tree_node *right;
     int data;
} ;

class bst
{
     public:
     tree_node *root;
     bst()
     {
          root=NULL;
     }
     int isempty()
     {
          return(root==NULL);
     }
     void insert(int item);
     void inorder(tree_node *);
     void postorder(tree_node *);
     void preorder(tree_node *);
};
void bst::insert(int item)
{

     tree_node *temp=new tree_node;
     tree_node *parent;
     temp->data=item;
     temp->left=NULL;
     temp->right=NULL;
     parent=NULL;
     if(isempty())
          root=temp;
     else
     {
          tree_node *ptr;
          ptr=root;
          while(ptr!=NULL)
          {
               parent=ptr;
               if(item>ptr->data)
                    ptr=ptr->right;
               else
                    ptr=ptr->left;
          }
          if(item<parent->data)
               parent->left=temp;
          else
               parent->right=temp;
     }
}
```

```
void bst::inorder(tree_node *ptr)
{
     if(ptr!=NULL)
     {
          inorder(ptr->left);
          cout<<"   "<<ptr->data<<"      ";
          inorder(ptr->right);
     }
}

void bst::postorder(tree_node *ptr)
{
     if(ptr!=NULL)
     {
          postorder(ptr->left);
          postorder(ptr->right);
          cout<<"   "<<ptr->data<<"      ";
     }
}

void bst::preorder(tree_node *ptr)
{
     if(ptr!=NULL)
     {
          cout<<"   "<<ptr->data<<"      ";
          preorder(ptr->left);
          preorder(ptr->right);
     }
}
```

```
int main()
{
    bst b;
    int item;
    for(int i=0;i<8;i++)
    {
            cout<<"please insert value of node("<<i<<")";
            cin>>item;
            b.insert(item);
    }

    cout<<"inorder"<<endl;
    b.inorder(b.root);
    cout<<endl<<"postorder"<<endl;
    b.postorder(b.root);
    cout<<endl<<"preorder"<<endl;
    b.preorder(b.root);
    cout<<endl;
system("pause");
return 0;
}
```

Lab Assignment

Search for application used trees