|  |
| --- |
| Practical no :-01  Aim : Write a program in C++ to implement Array.  Solution :-  #include <iostream>  using namespace std;  int main(){     float percentage[] = {56.4 , 99.0, 12.20, 67.2};     cout<<"printing all values of the array :\n";     for(int i = 0; i<4 ; i++){        cout<<"element "<<i+1<<" = "<<percentage[i]<<endl;     }     return 0;  }  Practical No :- 02  Aim : Write a program to accept the elements in 2D array and perform |
| all the matrix operations i.e. addition, multiplication, substraction etc. |

A)

Solution :-

#include <iostream>

using namespace std;

int main() {

   int r=2, c=4, sum[2][4], i, j;

   int a[2][4] = {{1,5,9,4} , {3,2,8,3}};

   int b[2][4] = {{6,3,8,2} , {1,5,2,9}};

   cout<<"The first matrix is: "<<endl;

   for(i=0; i<r; ++i) {

      for(j=0; j<c; ++j)

      cout<<a[i][j]<<" ";

      cout<<endl;

   }

   cout<<endl;

   cout<<"The second matrix is: "<<endl;

   for(i=0; i<r; ++i) {

      for(j=0; j<c; ++j)

      cout<<b[i][j]<<" ";

      cout<<endl;

   }

   cout<<endl;

   for(i=0;i<r;++i)

   for(j=0;j<c;++j)

   sum[i][j]=a[i][j]+b[i][j];

   cout<<"Sum of the two matrices is:"<<endl;

   for(i=0; i<r; ++i) {

      for(j=0; j<c; ++j)

      cout<<sum[i][j]<<" ";

      cout<<endl;

   }

   return 0;

}

B)

Solution :-

#include <iostream>

using namespace std;

int main() {

int r=2, c=4, sub[2][4], i, j;

int a[2][4] = {{1,5,9,4} , {3,2,8,3}};

int b[2][4] = {{6,3,8,2} , {1,5,2,9}};

cout<<"The first matrix is: "<<endl;

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

for(j=0; j<c; ++j)

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

cout<<endl;

}

cout<<endl;

cout<<"The second matrix is: "<<endl;

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

for(j=0; j<c; ++j)

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

cout<<endl;

}

cout<<endl;

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

for(j=0;j<c;++j)

sub[i][j]=a[i][j]-b[i][j];

cout<<"Subtraction of the two matrices is:"<<endl;

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

for(j=0; j<c; ++j)

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

cout<<endl;

}

return 0;

}

C)

Solution :-

#include <iostream>

using namespace std;

int main() {

int r=2, c=4, mul[2][4], i, j;

int a[2][4] = {{1,5,9,4} , {3,2,8,3}};

int b[2][4] = {{6,3,8,2} , {1,5,2,9}};

cout<<"The first matrix is: "<<endl;

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

for(j=0; j<c; ++j)

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

cout<<endl;

}

cout<<endl;

cout<<"The second matrix is: "<<endl;

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

for(j=0; j<c; ++j)

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

cout<<endl;

}

cout<<endl;

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

for(j=0;j<c;++j)

mul[i][j]=a[i][j]\*b[i][j];

cout<<"mul of the two matrices is:"<<endl;

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

for(j=0; j<c; ++j)

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

cout<<endl;

}

return 0;

}

|  |  |  |
| --- | --- | --- |
| Practical No :-03  Aim :- Explain following techniques | | |
| Technique |  | Bubble sort |

Solution :-

#include<iostream>

using namespace std;

void swapping(int &a, int &b) {      //swap the content of a and b

   int temp;

   temp = a;

   a = b;

   b = temp;

}

void display(int \*array, int size) {

   for(int i = 0; i<size; i++)

      cout << array[i] << " ";

   cout << endl;

}

void bubbleSort(int \*array, int size) {

   for(int i = 0; i<size; i++) {

      int swaps = 0;         //flag to detect any swap is there or not

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

         if(array[j] > array[j+1]) {       //when the current item is bigger than next

            swapping(array[j], array[j+1]);

            swaps = 1;    //set swap flag

         }

      }

      if(!swaps)

         break;       // No swap in this pass, so array is sorted

   }

}

int main() {

int n;

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

cin >> n;

int arr[n]; //create an array with given number of elements

cout << "Enter elements:" << endl;

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

cin >> arr[i];

}

cout << "Array before Sorting: ";

display(arr, n);

bubbleSort(arr, n);

cout << "Array after Sorting: ";

display(arr, n);

}

|  |
| --- |
| Practical No :- 04  Aim : Write a program in C++ to delete particular element from an array |
| of 10 integers. |

Solution :-

#include<iostream>

using namespace std;

int main()

{

int arr[10], tot=10, i, elem, j, found=0;

cout<<"Enter 10 Array Elements: ";

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

cin>>arr[i];

cout<<"\nEnter Element to Delete: ";

cin>>elem;

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

{

if(arr[i]==elem)

{

for(j=i; j<(tot-1); j++)

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

found++;

i--;

tot--;

}

}

if(found==0)

cout<<"\nElement doesn't found in the Array!";

else

cout<<"\nElement Deleted Successfully!";

cout<<endl;

return 0;

}

|  |
| --- |
| Practical No :- 05  Aim :-  Consider two single dimensional array of size 20 and 3 |
| respectively. Write a program in C++ to display all the elements |
| which are common in both arrays. |

Solution :-

#include <bits/stdc++.h>

using namespace std;

int main(){

   //defining the array

   int arr1[] = { 1, 45, 54, 71, 76, 12 };

   int arr2[] = { 1, 7, 5, 4, 6, 12 };

   int n1 = sizeof(arr1) / sizeof(arr1[0]);

   int n2 = sizeof(arr2) / sizeof(arr2[0]);

   sort(arr1, arr1 + n1);

   sort(arr2, arr2 + n2);

   cout << "First Array: ";

   for (int i = 0; i < n1; i++)

      cout << arr1[i] << " ";

   cout << endl;

   cout << "Second Array: ";

   for (int i = 0; i < n2; i++)

      cout << arr2[i] << " ";

   cout << endl;

   vector<int> v(n1 + n2);

   vector<int>::iterator it, st;

   //finding the common elements

   it = set\_intersection(arr1, arr1 + n1, arr2, arr2 + n2, v.begin());

   cout << "\nCommon elements:\n";

   for (st = v.begin(); st != it; ++st)

      cout << \*st << ", ";

   cout << '\n';

   return 0;

}

Practical No :- 06

Aim :- Write a program in C++ to implement queue using Array.

Solution :-

#include <iostream>

using namespace std;

int queue[100], n = 100, front = - 1, rear = - 1;

void Insert() {

   int val;

   if (rear == n - 1)

   cout<<"Queue Overflow"<<endl;

   else {

      if (front == - 1)

      front = 0;

      cout<<"Insert the element in queue : "<<endl;

      cin>>val;

      rear++;

      queue[rear] = val;

   }

}

void Delete() {

   if (front == - 1 || front > rear) {

      cout<<"Queue Underflow ";

      return ;

   } else {

      cout<<"Element deleted from queue is : "<< queue[front] <<endl;

      front++;;

   }

}

void Display() {

   if (front == - 1)

   cout<<"Queue is empty"<<endl;

   else {

      cout<<"Queue elements are : ";

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

      cout<<queue[i]<<" ";

         cout<<endl;

   }

}

int main() {

   int ch;

   cout<<"1) Insert element to queue"<<endl;

   cout<<"2) Delete element from queue"<<endl;

   cout<<"3) Display all the elements of queue"<<endl;

   cout<<"4) Exit"<<endl;

   do {

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

      cin>>ch;

      switch (ch) {

         case 1: Insert();

         break;

         case 2: Delete();

         break;

         case 3: Display();

         break;

         case 4: cout<<"Exit"<<endl;

         break;

         default: cout<<"Invalid choice"<<endl;

      }

   } while(ch!=4);

   return 0;

}

|  |
| --- |
| Practical No :- 07  Aim :-  Write menu driven program which create and display the circular |
| linked list. |

Solution :-

#include <bits/stdc++.h>

using namespace std;

struct Node

{

    int data;

    struct Node \*next;

}\*Head=NULL;

void create(int a[],int n)

{

    struct Node \*t,\*last;

    Head=new Node;

    Head->data=a[0];

    Head->next=Head;

    last=Head;

        for(int i=1;i<n;i++)

        {

           t=new Node;

    t->data=a[i];

    t->next=last->next;

    last->next=t;

    last=t;

        }

}

int count(struct Node \*p)

{

    int c=0;

    do

    {

        c++;

     p=p->next;

    }while(p!=Head);

    return c;

}

void insert(int pos, int x)

{

    if(pos>count(Head) || pos<0)

    return;

    struct Node \*t,\*p=Head;

    t=new Node;

    t->data=x;

    if(pos==0)

    {

        if(Head==NULL)

        {

            Head=t;

            Head->next=t;

        }

        while(p->next!=Head)

        p=p->next;

        t->next=Head;

        p->next=t;

        Head=t;

    }

    else{

        for(int i=0;i<pos-1;i++)

            p=p->next;

            t->next=p->next;

            p->next=t;

    }

}

int Delete(struct Node \*p,int index)

{

    struct Node \*q;

    int x=-1;

    if(index<1 || index>count(Head))

    return x;

     if(index==1)

    {

        x=p->data;

         while(p->next!=Head)

        p=p->next;

        if(Head==p)

        {

            delete Head;

            Head=NULL;

        }

        else

        {

        p->next=Head->next;

        delete Head;

        Head=p->next;

        }

        return x;

    }

    else

    {

    for(int i=0;i<index-2;i++)

    {

        p=p->next;

    }

    q=p->next;

    p->next=q->next;

    x=q->data;

    delete q;

    return x;

    }

}

void display(struct Node \*p)

{

    do

    {

        cout<<p->data<<" ";

        p=p->next;

    }while(p!=Head);

}

int main()

{

    int a[500];

    int option,n,pos,x,index,t;

    do

    {

        cout<<"1. Create Circular Linked list"<<endl<<"2. Insert in Circular Linked list"<<endl<<"3. Delete "<<endl<<"4. Display"<<endl<<"5. Exit"<<endl;;

        cout<<"Enter an option :"<<endl;

        cin>>option;

        switch(option)

        {

        case 1 :

        {

            cout<<"Enter no of integers : "<<endl;

            cin>>n;

            cout<<"Enter the numbers"<<endl;

            for(int i=0;i<n;i++)

            cin>>a[i];

            create(a,n);

            cout<<endl;

            break;

        }

        case 2:

        {

            cout<<"Enter position to insert an element : "<<endl;

            cin>>pos;

            cout<<"Enter element : "<<endl;

            cin>>x;

            insert(pos,x);

            cout<<endl;

            break;

        }

        case 3:

        {

            cout<<"Enter position to delete element : "<<endl;

            cin>>index;

            cout<<"Deleted element is: "<<Delete(Head,index);

            cout<<endl;

            break;

        }

        case 4:

        {

            cout<<"Displaying elements :";

            display(Head);

            cout<<endl;

            break;

        }

        default:

        cout<<"Exiting program......"<<endl;

        }

    }while(option<=4);

    return 0;

}

Practical No :- 08

Aim :- Create binary search tree 15, 2, 25, 45, 35, 23, 100, 5

Solution :-

# include <iostream>

# include <cstdlib>

using namespace std;

struct nod//node declaration

{

   int info;

   struct nod \*l;

   struct nod \*r;

}\*r;

class BST

{

   public://functions declaration

   void search(nod \*, int);

   void find(int, nod \*\*, nod \*\*);

   void insert(nod \*, nod \*);

   void del(int);

   void casea(nod \*,nod \*);

   void caseb(nod \*,nod \*);

   void casec(nod \*,nod \*);

   void preorder(nod \*);

   void inorder(nod \*);

   void postorder(nod \*);

   void show(nod \*, int);

   BST()

   {

      r = NULL;

   }

};

void BST::find(int i, nod \*\*par, nod \*\*loc)//find the position of the item

{

   nod \*ptr, \*ptrsave;

   if (r == NULL)

   {

      \*loc = NULL;

      \*par = NULL;

      return;

   }

   if (i == r→info)

   {

      \*loc = r;

      \*par = NULL;

      return;

   }

   if (i < r→info)

   ptr = r→l;

   else

   ptr = r→r;

   ptrsave = r;

   while (ptr != NULL)

   {

      if (i == ptr→info)

      {

         \*loc = ptr;

         \*par = ptrsave;

         return;

      }

      ptrsave = ptr;

      if (i < ptr→info)

      ptr = ptr→l;

      else

      ptr = ptr→r;

   }

   \*loc = NULL;

   \*par = ptrsave;

}

void BST::search(nod \*root, int data) //searching

{

   int depth = 0;

   nod \*temp = new nod;

   temp = root;

   while(temp != NULL)

   {

      depth++;

      if(temp→info == data)

      {

         cout<<"\nData found at depth: "<<depth<<endl;

         return;

      }

      else if(temp→info > data)

      temp = temp→l;

      else

      temp = temp→r;

   }

   cout<<"\n Data not found"<<endl;

   return;

}

void BST::insert(nod \*tree, nod \*newnode)

{

   if (r == NULL)

   {

      r = new nod;

      r→info = newnode→info;

      r→l= NULL;

      r→r= NULL;

      cout<<"Root Node is Added"<<endl;

      return;

   }

   if (tree→info == newnode→info)

   {

      cout<<"Element already in the tree"<<endl;

      return;

   }

   if (tree→info > newnode→info)

   {

      if (tree→l != NULL)

      {

         insert(tree→l, newnode);

      }

      else

      {

         tree→l= newnode;

         (tree→l)→l = NULL;

         (tree→l)→r= NULL;

         cout<<"Node Added To Left"<<endl;

         return;

      }

   }

   else

   {

      if (tree→r != NULL)

      {

         insert(tree→r, newnode);

      }

      else

      {

         tree→r = newnode;

         (tree→r)→l= NULL;

         (tree→r)→r = NULL;

         cout<<"Node Added To Right"<<endl;

         return;

      }

   }

}

void BST::del(int i)

{

   nod \*par, \*loc;

   if (r == NULL)

   {

      cout<<"Tree empty"<<endl;

      return;

   }

   find(i, &par, &loc);

   if (loc == NULL)

   {

      cout<<"Item not present in tree"<<endl;

      return;

   }

   if (loc→l == NULL && loc→r == NULL)

   {

      casea(par, loc);

      cout<<"item deleted"<<endl;

   }

   if (loc→l!= NULL && loc→r == NULL)

   {

      caseb(par, loc);

      cout<<"item deleted"<<endl;

   }

   if (loc→l== NULL && loc→r != NULL)

   {

      caseb(par, loc);

      cout<<"item deleted"<<endl;

   }

   if (loc→l != NULL && loc→r != NULL)

   {

      casec(par, loc);

      cout<<"item deleted"<<endl;

   }

   free(loc);

}

void BST::casea(nod \*par, nod \*loc )

{

   if (par == NULL)

{

   r= NULL;

}

else

{

   if (loc == par→l)

   par→l = NULL;

   else

   par→r = NULL;

   }

}

void BST::caseb(nod \*par, nod \*loc)

{

   nod \*child;

   if (loc→l!= NULL)

      child = loc→l;

   else

      child = loc→r;

   if (par == NULL)

   {

      r = child;

   }

   else

   {

      if (loc == par→l)

         par→l = child;

      else

         par→r = child;

   }

}

void BST::casec(nod \*par, nod \*loc)

{

   nod \*ptr, \*ptrsave, \*suc, \*parsuc;

   ptrsave = loc;

   ptr = loc→r;

   while (ptr→l!= NULL)

   {

      ptrsave = ptr;

      ptr = ptr→l;

   }

   suc = ptr;

   parsuc = ptrsave;

   if (suc→l == NULL && suc→r == NULL)

      casea(parsuc, suc);

   else

      caseb(parsuc, suc);

   if (par == NULL)

   {

      r = suc;

   }

   else

   {

      if (loc == par→l)

         par→l = suc;

      else

         par→r= suc;

   }

   suc→l = loc→l;

   suc→r= loc→r;

}

void BST::preorder(nod \*ptr)

{

   if (r == NULL)

   {

      cout<<"Tree is empty"<<endl;

      return;

   }

   if (ptr != NULL)

   {

      cout<<ptr→info<<" ";

      preorder(ptr→l);

      preorder(ptr→r);

   }

}

void BST::inorder(nod \*ptr)//inorder traversal

{

   if (r == NULL)

   {

      cout<<"Tree is empty"<<endl;

      return;

   }

   if (ptr != NULL)

   {

      inorder(ptr→l);

      cout<<ptr→info<<" ";

      inorder(ptr→r);

   }

}

void BST::postorder(nod \*ptr)//postorder traversal

{

   if (r == NULL)

   {

      cout<<"Tree is empty"<<endl;

      return;

   }

   if (ptr != NULL)

   {

      postorder(ptr→l);

      postorder(ptr→r);

      cout<<ptr→info<<" ";

   }

}

void BST::show(nod \*ptr, int level)//print the tree

{

   int i;

   if (ptr != NULL)

   {

      show(ptr→r, level+1);

      cout<<endl;

      if (ptr == r)

         cout<<"Root→: ";

      else

      {

         for (i = 0;i < level;i++)

         cout<<" ";

      }

      cout<<ptr→info;

      show(ptr→l, level+1);

   }

}

int main()

{

   int c, n,item;

   BST bst;

   nod \*t;

   while (1)

   {

      cout<<"1.Insert Element "<<endl;

      cout<<"2.Delete Element "<<endl;

      cout<<"3.Search Element"<<endl;

      cout<<"4.Inorder Traversal"<<endl;

      cout<<"5.Preorder Traversal"<<endl;

      cout<<"6.Postorder Traversal"<<endl;

      cout<<"7.Display the tree"<<endl;

      cout<<"8.Quit"<<endl;

      cout<<"Enter your choice : ";

      cin>>c;

      switch(c)

      {

         case 1:

            t = new nod;

            cout<<"Enter the number to be inserted : ";

            cin>>t→info;

            bst.insert(r, t);

            break;

         case 2:

            if (r == NULL)

            {

               cout<<"Tree is empty, nothing to delete"<<endl;

               continue;

            }

            cout<<"Enter the number to be deleted : ";

            cin>>n;

            bst.del(n);

            break;

         case 3:

            cout<<"Search:"<<endl;

            cin>>item;

            bst.search(r,item);

            break;

         case 4:

            cout<<"Inorder Traversal of BST:"<<endl;

            bst.inorder(r);

            cout<<endl;

            break;

         case 5:

            cout<<"Preorder Traversal of BST:"<<endl;

            bst.preorder(r);

            cout<<endl;

            break;

         case 6:

            cout<<"Postorder Traversal of BST:"<<endl;

            bst.postorder(r);

            cout<<endl;

            break;

         case 7:

            cout<<"Display BST:"<<endl;

            bst.show(r,1);

            cout<<endl;

            break;

         case 8:

            exit(1);

         default:

            cout<<"Wrong choice"<<endl;

      }

   }

}