|  |
| --- |
| #include<iostream>  #include<vector>  void swap(int \**xp*, int \**yp*)  {      int temp = \*xp;      \*xp = \*yp;      \*yp = temp;  }  void selectionSort(int *arr*[],int *n*)  {      int i,j,min\_idx;      for(i = 0; i < n-1; i++)      {          min\_idx = i;          for(j = i +1; j < n; j++)              if(arr[j] < arr[min\_idx])                  min\_idx = j;            swap(&arr[min\_idx], &arr[i]);      }  }  void printArray(int *arr*[], int *size*)  {      int i;      for(i = 0; i < size; i++)          std::cout<< arr[i] << " ";      std::cout<<std::endl;  }  int main(int *argc*, char\* *argv*[])  {      int arr[] = {64,25,12,22,11};      int n = sizeof(arr)/sizeof(arr[0]);        selectionSort(arr,n);      std::cout<<"Sorted Array : \n";      printArray(arr,n);      return 0;  } |

Output:

A screenshot of a computer

Description automatically generated

|  |
| --- |
| #pragma once  #include<iostream>  #include<vector>  using namespace std;  class SpaningTree  {  int v,e,w;  int \*\*adjMatrix;  int \* visited;  public :  SpaningTree(int v,int e);  void addEdge(int start,int e,int weight);  void minimumTree();  void printAdjMatrix();  };  SpaningTree::SpaningTree(int v,int e)  {  this->v = v;  this->e = e;  adjMatrix = new int\*[v];  visited = new int[v];  for(int row = 0 ; row < v; row++)  {  visited[row] = 0;  adjMatrix[row] = new int[v];  for(int column = 0;column < v; column++)  {  adjMatrix[row][column] = 999;  }  }  }  void SpaningTree::addEdge(int start,int e,int weight)  {  adjMatrix[start][e] = weight;  adjMatrix[e][start] = weight;  }  void SpaningTree::printAdjMatrix(){  for(int row = 0 ; row < v; row++){  for(int column = 0;column < v; column++)  { cout<<adjMatrix[row][column]<<"\t"; }  cout<<endl;  }}  void SpaningTree::minimumTree()  {  int p=0,q=0,total=0,min;  visited[0] = 1;  for(int count=0;count<(v-1);count++)  {  min=999;  for(int i=0;i<v;i++)  {  if(visited[i]==1)  {  for(int j=0;j<v;j++)  {  if(visited[j]!=1)  {  if(min > adjMatrix[i][j])  {  min=adjMatrix[i][j];  p=i;  q=j;}}}}}  visited[p]=1;  visited[q]=1;  total=total+min;  cout<<"Minimum cost connection is "<<(p)<<" -> "<<(q)<<" with charge : "<<min<< endl;  }  cout<<"The minimum total cost of connections of all branches is: "<<total<<endl;  }  #include"header.h"  int main(void){  SpaningTree st = SpaningTree(5,7);  st.printAdjMatrix();  st.addEdge(0,1,200);  st.addEdge(0,3,600);  st.addEdge(1,2,300);  st.addEdge(1,4,500);  st.addEdge(2,4,700);  st.addEdge(3,4,900);  cout<<endl<<endl;  st.printAdjMatrix();  st.minimumTree();} |

A screenshot of a computer screen

Description automatically generated