**NAME:-VIVEK BULANI**

**ROLL NO:-SECOA115**

**ASSIGNMENT NO:-03**

**AIM** **:-** To write a C++ program for sparse matrix realization and operations on it- Transpose, Fast

Transpose and addition of two matrices.

**PROGRAM:-**

#include <iostream>

using namespace std;

class sparse

{

private:

   int M[10][3];

   int row,col,term,i,j,k;

public:

   void accept()

   {

       cout<<"enter row=";

       cin>>row;

       cout<<"enter column=";

       cin>>col;

       cout<<"enter non zero term=";

       cin>>term;

       M[0][0]=row;

       M[0][1]=col;

       M[0][2]=term;

       for (int i=1;i<=term;i++)

           {

               cout<<"enter row,column,non zero term=";

               cin>>M[i][0]>>M[i][1]>>M[i][2];

           }

   }

   void display()

   {

       for (int i=0;i<=term;i++)

       {

       cout<<M[i][0]<<"\t"<<M[i][1]<<"\t"<<M[i][2]<<endl;

       }

   }

   void add(sparse s1,sparse s2)

       {

           row=s1.row;

           col=s1.col;

           int i=1;

           int j=1;

           int k=1;

           while(i<=s1.term && j<=s2.term)

           {

               if(s1.M[i][0]==s2.M[j][0])

               {

                   if(s1.M[i][1]==s2.M[j][1])

                   {

                       M[k][0]=s2.M[i][0];

                       M[k][1]=s2.M[i][1];

                       M[k][2]=s1.M[i][2]+s2.M[j][2];

                       i++;

                       j++;

                       k++;

                   }

                   else if(s1.M[i][1]<s2.M[j][1])

                   {

                       M[k][0]=s1.M[i][0];

                       M[k][1]=s1.M[i][1];

                       M[k][2]=s1.M[i][2];

                       i++;

                       k++;

                   }

                   else

                   {

                       M[k][0]=s2.M[j][0];

                       M[k][1]=s2.M[j][1];

                       M[k][2]=s2.M[j][2];

                       j++;

                       k++;

                   }

               }

               else if (s1.M[i][0]<s2.M[j][0])

                   {

                       M[k][0]=s1.M[i][0];

                       M[k][1]=s1.M[i][1];

                       M[k][2]=s1.M[i][2];

                       i++;

                       k++;

                   }

               else

                   {

                       M[k][0]=s2.M[j][0];

                       M[k][1]=s2.M[j][1];

                       M[k][2]=s2.M[j][2];

                       j++;

                       k++;

                   }

           }

           while(i<=s1.term)

           {

               M[k][0]=s1.M[i][0];

               M[k][1]=s1.M[i][1];

               M[k][2]=s1.M[i][2];

               i++;

               k++;

           }

           while(j<=s2.term)

           {

               M[k][0]=s2.M[j][0];

               M[k][1]=s2.M[j][1];

               M[k][2]=s2.M[j][2];

               j++;

               k++;

           }

           term=k-1;

           M[0][0]=row;

           M[0][1]=col;

           M[0][2]=term;

       }

   void sim\_trans(sparse s4)

   {

       int n,p;

       p=s4.M[0][0];

       M[0][0]=s4.M[0][1];

       M[0][1]=p;

       M[0][2]=s4.M[0][2];

       term=M[0][2];

       k=1;

       n=s4.M[0][2];

       for(i=0;i<s4.M[0][1];i++)

       {

           for(j=1;j<=n;j++)

           {

               if(i==s4.M[j][1])

               {

                   M[k][0]=s4.M[j][1];

                   M[k][1]=s4.M[j][0];

                   M[k][2]=s4.M[j][2];

                   k++;

               }

           }

       }

   }

   void fast\_trans(sparse s4)

   {

       int m,n,t,col\_no,lo;

       int count[20],index[20];

       m=s4.M[0][0];

       n=s4.M[0][1];

       t=s4.M[0][2];

       term=t;

       M[0][0]=n;

       M[0][1]=m;

       M[0][2]=t;

       for(i=0;i<n;i++)

       {

           count[i]=0;

       }

           for(k=1;k<=t;k++)

           {

               col\_no=s4.M[k][1];

               count[col\_no]=count[col\_no]+1;

           }

       index[0]=1;

       for(i=1;i<n;i++)

       {

           index[i]=index[i-1]+count[i-1];

       }

       for(i=1;i<=t;i++)

       {

           col\_no=s4.M[i][1];

           lo=index[col\_no];

           M[lo][0]=s4.M[i][1];

           M[lo][1]=s4.M[i][0];

           M[lo][2]=s4.M[i][2];

           index[col\_no]++;

       }

   }

};

int main()

{

   sparse s1,s2,s3,s4,s5;

   int ch,in;

   do

   {

       cout<<"1.addition of two matrix\n2.simple transpose\n3.fast transpose";

 cout<<"enter the operation u want to do= ";

       cin>>ch;

       switch (ch)

       {

       case 1:

       {

           cout<<"input for matrix 1\n";

           s1.accept();

           cout<<"input for matrix 2\n";

           s2.accept();

           cout<<"matrix 1:\n";

           cout<<" row\t"<<"column\t"<<"non zero term \n";

           s1.display();

           cout<<"matrix 2:\n";

           cout<<" row\t"<<"column\t"<<"non zero term \n";

           s2.display();

           s3.add(s1,s2);

           cout<<"after addition="<<endl;

           cout<<" row\t"<<"column\t"<<"non zero term \n";

           s3.display();

           break;

       }

       case 2:

       {

           cout<<"input for matrix \n";

           s4.accept();

           s5.sim\_trans(s4);

           cout<<"after simple transpose="<<endl;

           cout<<" row\t"<<"column\t"<<"non zero term \n";

           s5.display();

           break;

       }

       case 3 :

       {

           cout<<"input for matrix \n";

           s4.accept();

           s5.fast\_trans(s4);

           cout<<"after fast transpose="<<endl;

           cout<<" row\t"<<"column\t"<<"non zero term \n";

           s5.display();

           break;

       }

       }

       cout<<"\n do u want to continue?";

       cout<<"\n to continue press 1 else 0"<<endl;

       cin>>in;

   }while(in==1);

   return 0;

   }

**OUTPUT:-**

1.addtion of two matrix

2.simple transpose

3.fast transpose

enter the operation u want to do= 1

input for matrix 1

enter row=3

enter column=3

enter non zero terms=3

enter row,column,non zero terms=2 2 33

enter row,column,non zero terms=2 1 22

enter row,column,non zero terms=1 2 11

input for matrix 2

enter row=3

enter column=3

enter non zero terms=2

enter row,column,non zero terms=1 1 33

enter row,column,non zero terms=2 1 22

matrix 1:

row column non zero term

3 3 3

1 2 11

2 1 22

2 2 33

Matrix 2:

row column non zero term

3 3 3

1 1 33

2 1 22

after addition=

row column non zero term

3 3 4

1 1 33

1 2 11

2 1 44

2 2 33

do u want to continue?

to continue press 1 else 0

1

1.addtion of two matrix

2.simple transpose

3.fast transpose

enter the operation u want to do= 2

input for matrix

enter row=4

enter column=4

enter non zero terms=4

enter row,column,non zero terms=1 1 10

enter row,column,non zero terms=3 4 10

enter row,column,non zero terms=3 3 14

enter row,column,non zero terms=2 1 44

after simple transpose=

row column non zero term

4 4 4

1 1 10

1 2 44

3 3 14

3 4 10

do u want to continue?

to continue press 1 else 0

1

1.addtion of two matrix

2.simple transpose

3.fast transpose

enter the operation u want to do= 3

input for matrix

enter row=3

enter column=4

enter non zero terms=4

enter row,column,non zero terms=2 3 99

enter row,column,non zero terms=3 4 45

enter row,column,non zero terms=1 3 77

enter row,column,non zero terms=2 1 24

after fast transpose=

row column non zero term

3 4 4

1 3 77

2 1 24

2 3 99

3 4 45

do u want to continue?

to continue press 1 else 0

0