**REPORT**:

The code comprises of class PATH\_FINDER which has following function

1. void getting\_values\_from\_file(int\*\* a, int&, int&);

Read the Value That from file

1. void Printing\_matrix(int\*\* list, int& R\_limit, int& C\_limit);

print matrix given in the question

1. bool is\_valid\_path(int\*\* a, int x, int y);

checks if the value/weight is 1

1. bool shortest\_path(int\*\* a, int x, int y, int\*\* matrix);

gives shortest path in another array i.e 1 in array shows shortest path only

1. void printing (int\*\* a, int x, int y, int\*\* matrix, int R\_limit, int C\_limit ,int & );

print matrix with shortest path

CODE:

|  |
| --- |
| //Assignmnent #4 |
|  |
| #include <iostream> |
| #include <cstring> |
| #include <string> |
| #include <fstream> |
| #include <cstdlib> |
| #include <iomanip> |
| using namespace std; |
|  |
|  |
|  |
| class PATH\_FINDER |
| { |
| private: |
|  |
| int row\_no; |
|  |
| int column\_no; |
| public: |
| PATH\_FINDER() |
| { |
| row\_no = column\_no = 0; |
|  |
| } |
| PATH\_FINDER(int n, int m) |
| { |
| row\_no = n; column\_no = m; |
| } |
| //\_\_\_\_\_SET AND GET FOR ROW |
| int get\_rows\_no() |
| { |
| return row\_no; |
| } |
|  |
| int set\_rows\_no(int input) |
| { |
| row\_no = input; |
| return row\_no; |
| } |
| //\_\_\_\_\_SET AND GET FOR COLUMN |
| int get\_column\_no() |
| { |
| return column\_no; |
| } |
|  |
| int set\_column\_no(int input1) |
| { |
| column\_no = input1; |
| return column\_no; |
| } |
| //\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |
|  |
| void getting\_values\_from\_file(int\*\* a, int&, int&); |
|  |
| void Printing\_matrix(int\*\* list, int& R\_limit, int& C\_limit); |
|  |
| bool is\_valid\_path(int\*\* a, int x, int y); |
|  |
| bool shortest\_path(int\*\* a, int x, int y, int\*\* matrix); |
|  |
| void printing(int\*\* a, int x, int y, int\*\* matrix, int R\_limit, int C\_limit ,int & ); |
|  |
| //\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |
|  |
| }; |
|  |
| /\* Function To Check The Path if Its 1 or 0\*/ |
|  |
| bool PATH\_FINDER::is\_valid\_path(int\*\* a, int x, int y) |
| { |
| if ( a[x][y] == 1) |
| { |
| return true; |
| } |
| return false; |
| } |
|  |
| bool PATH\_FINDER::shortest\_path(int\*\* a, int x, int y, int\*\* matrix) |
| { |
| PATH\_FINDER s; |
|  |
| if (x == 6 && y == 4) // According to the Condition 6 Row And 4 Column Is The Destination |
| { |
| matrix[x][y] = 1; |
| return true; |
| } |
| if (s.is\_valid\_path(a, x, y)) |
| { |
| matrix[x][y] = 1; |
|  |
| if (shortest\_path(a, x + 1, y, matrix)) |
| { |
| return true; |
| } |
| if (shortest\_path(a, x, y + 1, matrix)) |
| { |
| return true; |
| } |
| if (shortest\_path(a, x, y - 1, matrix)) |
| { |
| return true; |
| } |
| if (shortest\_path(a, x - 1, y, matrix)) |
| { |
| return true; |
| } |
| matrix[x][y] = 0; |
| return false; |
| } |
| return false; |
| } |
|  |
| void PATH\_FINDER::printing(int\*\* a, int x, int y, int\*\* matrix,int R\_limit,int C\_limit, int & limit1) |
| { |
| PATH\_FINDER obj; |
| if (obj.shortest\_path(a, x, y, matrix)) |
| { |
| for (int loop = 0; loop < R\_limit; loop++) |
| { |
| for (int loop1 = 0; loop1 < C\_limit; loop1++) |
| { |
| { |
| cout << matrix[loop][loop1] << " "; |
| if (matrix[loop][loop1] == 1) |
| { |
| limit1++; |
| } |
|  |
| } |
| } |
| cout << endl; |
| } |
| } |
| } |
|  |
| //Function to read values from the file |
| void PATH\_FINDER::getting\_values\_from\_file(int\*\* list, int& R\_limit, int& C\_limit) |
| { |
|  |
| char File\_name[20]; |
| cout << "\nENTER FILE NAME FOR GENERATING MATRIX:"; |
| cin.getline(File\_name, 20); |
| cout << "\nMATRIX CONSTRUCTED FROM USERS FILE:\n"; |
| ifstream output; |
| output.open(File\_name); //FILE NAME FROM MAIN GIVEN BY USER |
| for (int loop = 0; loop < R\_limit; loop++) |
| { |
| for (int loop1 = 0; loop1 < C\_limit; loop1++) |
| { |
| output >> (list[loop][loop1]); |
| } |
| } |
| Printing\_matrix(list, R\_limit, C\_limit); |
| } |
|  |
| void PATH\_FINDER::Printing\_matrix(int\*\* list, int& R\_limit, int& C\_limit) |
| { |
| cout << "\n MATRIX :\n"; |
| for (int loop = 0; loop < R\_limit; loop++) |
| { |
| for (int loop1 = 0; loop1 < C\_limit; loop1++) |
| { |
| cout << list[loop][loop1] << " "; |
| } |
| cout << endl; |
| } |
| } |
|  |
|  |
|  |
| int main() |
| { |
| PATH\_FINDER s; |
|  |
| s.set\_rows\_no(9); //9...........as according to the matrix given |
| s.set\_column\_no(10); //10......as according to the matrix given |
| cout << endl; |
| cout << "THE ROW NUMBER IS" << s.get\_rows\_no(); |
| cout << endl; |
| cout << "THE COLUMN NUMBER IS" << s.get\_column\_no(); |
| cout << endl; |
|  |
| //\_\_\_\_\_\_\_\_DYNAMIC\_ALLOCATION\_OF\_TWO-D\_ARRAY\_\_\_\_\_\_\_\_\_ |
| //INTIALIZING MATRIX #1 |
| int\*\* matrix\_array1 = new int\* [s.get\_rows\_no()]; |
| for (int loop = 0; loop < s.get\_rows\_no(); loop++) |
| { |
| matrix\_array1[loop]= new int[s.get\_column\_no()]; |
| } |
| //INTIALIZING MATRIX #2 |
| int\*\* matrix\_array2 =NULL; |
| matrix\_array2 = new int\* [s.get\_rows\_no()]; |
| for (int loop2 = 0; loop2 < s.get\_rows\_no(); loop2++) |
| { |
| matrix\_array2[loop2]= new int [s.get\_column\_no()]; |
| } |
| int R\_limit = s.get\_rows\_no(); |
| int C\_limit = s.get\_column\_no(); |
|  |
| cout << "matrix\n"; |
| for (int loop = 0; loop < R\_limit; loop++) |
| { |
| for (int loop1 = 0; loop1 < C\_limit; loop1++) |
| { |
| matrix\_array2[loop][loop1] =0; |
| } |
| } |
|  |
| //FOR MATRIX 1.....as given in question |
| s.getting\_values\_from\_file(matrix\_array1, R\_limit, C\_limit);//Num\_file.txt |
|  |
| //\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |
| int a = 0; int bb = 0;//source point |
| int limit =0 ; |
| cout << endl; |
| //Function displaying shortest path |
| s.printing(matrix\_array1, a, bb, matrix\_array2, R\_limit, C\_limit,limit); |
| //Path will be -1 as we want path length from source till destination |
| cout << "The Path Length is :" << limit -1; |
| cout << endl << endl; |
|  |
|  |
| delete[]matrix\_array1; |
| delete[]matrix\_array2; |
|  |
| return 0; |
| } |

OUTPUT:

