**LAB:12**

**NAME: DHRUV PATEL**

**ROLL NO: CE111**

**SUBJECT: NIS**

**Aim:** Write a program to demonstrate Image Steganography operations: Embed and Extract Hide 1 bit per pixel. Compute MSE (Mean Squared Error) and PSNR (Peak Signal to Noise Ratio) values**.**

* **Source Code**

#include <bits/stdc++.h>

#include <vector>

using namespace std;

int convertBintoDec(string bin)

{

    int dec = 0;

    for (int i = bin.length() - 1, j = 0; i >= 0; i--, j++)

    {

        if (bin[i] == '1')

        {

            dec += pow(double(2), double(j));

        }

    }

    return dec;

}

string convertDectoBin(int dec)

{

    string bin;

    while (dec != 0)

    {

        bin += to\_string(dec % 2);

        dec /= 2;

    }

    reverse(bin.begin(), bin.end());

    return bin;

}

vector<vector<int>> embeded(vector<vector<int>> cover, string msg)

{

    vector<vector<int>> stego\_obj;

    int index = 0;

    for (int i = 0; i < cover.size(); i++)

    {

        vector<int> row;

        for (int j = 0; j < cover[0].size(); j++)

        {

            string bin = convertDectoBin(cover[i][j]);

            bin[bin.length() - 1] = msg[index];

            cover[i][j] = convertBintoDec(bin);

            row.push\_back(cover[i][j]);

            index++;

        }

        stego\_obj.push\_back(row);

    }

    return stego\_obj;

}

string extraction(vector<vector<int>> embeded\_cover)

{

    string extracted\_msg = "";

    for (int i = 0; i < embeded\_cover.size(); i++)

    {

        for (int j = 0; j < embeded\_cover[0].size(); j++)

        {

            string bin = convertDectoBin(embeded\_cover[i][j]);

            extracted\_msg += bin[bin.length() - 1];

        }

    }

    return extracted\_msg;

}

double MSE(vector<vector<int>> i1, vector<vector<int>> i2)

{

    double mse = 0;

    for (int i = 0; i < i1.size(); i++)

    {

        for (int j = 0; j < i1[0].size(); j++)

        {

            mse += pow(double(i1[i][j] - i2[i][j]), double(2));

        }

    }

    mse /= i1.size() \* i1[0].size();

    return mse;

}

double PSNR(double MSE)

{

    double r = 255;

    double psnr = 10 \* (log10(r \* r / MSE));

    return psnr;

}

int isBinary(string input)

{

    for (int i = 0; input[i]; ++i)

        if (input[i] != '0' && input[i] != '1')

            return 0;

    return 1;

}

int main()

{

    string msg;

    cout << "Enter Message (in binary) :" << endl;

    cin >> msg;

    if (isBinary(msg) == 0)

    {

        cout << "Enter valid binary number" << endl;

        return 0;

    }

    int n = sqrt(msg.length());

    if (n \* n != msg.length())

    {

        cout << "Enter Binary Mesage with legth whose whole square root is possible " << endl;

        return 0;

    }

    cout << "----------------------------------------------------- " << endl;

    cout<< "Enter Cover (Dimensions " << n << " \* " << n << " ) : " << endl;

    vector<vector<int>>cover;

    for (int i = 0; i < n; i++)

    {

        vector<int> row;

        for (int j = 0; j < n; j++)

        {

            int val;

            cin >> val;

            row.push\_back(val);

        }

        cover.push\_back(row);

    }

    cout << "----------------------------------------------------- " << endl;

    cout<< "Stego Object : " << endl;

    vector<vector<int>> stego\_obj = embeded(cover, msg);

    for (int i = 0; i < stego\_obj.size(); i++)

    {

        for (int j = 0; j < stego\_obj[0].size(); j++)

        {

            cout << stego\_obj[i][j] << " ";

        }

        cout << endl;

    }

    cout << "----------------------------------------------------- " << endl;

    string extraction\_msg = extraction(stego\_obj);

    cout << "Extracted message : " << extraction\_msg << endl;

    cout << "----------------------------------------------------- " << endl;

    double mse = MSE(cover, stego\_obj);

    cout << "Min Square Error :" << mse << endl;

    cout << "----------------------------------------------------- " << endl;

    cout<< "Peak Signal to Noise Ratio (PSNR) : " << PSNR(mse)<< endl;

    return 0;

}

* **Input Output**



