**GROUP 15**

**GROUP MEMBER 1: NEHAL 22F-3638**

**GROUP MEMBER 2: MALAIKA 22F-3629**

**CS-1002 Programming Fundamentals**

**Semester Final Project**

**(24 Puzzle Game)**

# **CODE:**

# **(MAIN.CPP)**

#include<iostream>

#include"header.h"

#include<iomanip>

#include<windows.h>

#include<fstream>

#include<ctime>

using namespace std;

void initi(int init\_state[5][5]);//initializing initial state

void goal(int goalState[5][5]);//initializing goal state

bool isSolvable(int goalState[5][5], int init\_state[5][5]);//checking that is the initial state is reachable to goal state

bool isGoal(int init\_state[5][5], int goalState[5][5]);//checking is goal state is reached or not

bool legalmoves(int init\_state[5][5], char move);//checking the legal moves for space tile

void make\_move(int init\_state[5][5], char move);//making move by swapping tiles

void change(int goalState[5][5], int init\_state[5][5]);//as the initial state is declared in seperate function so to making change in it we use this function

void file(int init\_state[5][5], int goalState[5][5], int const\_array[5][5], char attempt[5]);

void printpath(); //for printing the path

int init\_state[5][5];

int goalState[5][5];

int f = 0;

int main()

{

srand(time(0));

char move; char choice = '1';

char random;

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

int goalState[5][5];

int init\_state[5][5];

goal(goalState); //randomly initializing

SetConsoleTextAttribute(h, 7);

do

{

cout << "Do you want to initialize the initial state randonmly or want to have it by a file?" <<endl ;

cout << "Press I for initializing randomly."<<endl;

cout << "Press f for taking it from file"<<endl ;

cin >> random;

if (random == 'i' || random == 'I')

{

do

{

initi(init\_state); //randomly initializing

} while (!isSolvable(goalState, init\_state)); //checking solvability

break;

}

else if (random == 'f' || random == 'F')

{

fstream afile;

afile.open("init.txt", ios::in);

if (!(afile.is\_open())) //opening file

{

cout << "Error opening File.." << endl;

system("pause");

}

else //reading array from text file

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

afile >> init\_state[i][j];

}

}

SetConsoleTextAttribute(h, FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

cout << setw(25) << "\*\*\*INITIAL STATE\*\*\*" << endl << endl << setw(3);

for (int i = 0; i < 5; i++) //printing array read from file in console

{

for (int j = 0; j < 5; j++)

{

const\_array[i][j] = init\_state[i][j];

if (const\_array[i][j] == 25)

{

cout << "|" << setw(3) << " " << "|" << setw(3);

}

else

cout << "|" << setw(3) << const\_array[i][j] << "|" << setw(3);

}

cout << endl;

}

cout << endl;

system("pause");

}

break;

}

else

cout << "You entered wrong input.Try again."<<endl;

} while ( random != 'i'|| random != 'I' || random != 'f' || random != 'F');

bool isgoal = isGoal(init\_state, goalState); //declaring a variable for checking that ,Is initial is reached to goal or not.

do

{

system("cls");

change(goalState, init\_state); //printing the goal and initial state (changed).

cout << endl;

SetConsoleTextAttribute(h, 7); //printing in color

cout << "\*\*Enter U for moving the space up.\*\*" << char(24) << endl;

cout << "\*\*Enter D for moving the space down.\*\*" << char(25) << endl;

cout << "\*\*Enter R for moving the space right.\*\*" << char(26) << endl;

cout << "\*\*Enter L for moving the space left.\*\*" << char(27) << endl;

cin >> move;

if (legalmoves(init\_state, move)) //checking the legal moves

{

attempt[f] = move;

f++;

}

bool check2 = legalmoves(init\_state, move);

if (check2 == true)

{

counter++; //taken to count the number of moves

make\_move(init\_state, move);

}

else

{

cout << "Invalid move..." << endl;

system("pause");

}

isgoal = isGoal(init\_state, goalState); //to check wether the initial state has reached to goal or not

SetConsoleTextAttribute(h, 13);

cout << "Want to play more? Press 1" << endl;

cin >> choice;

} while (isgoal == false && choice == '1');

if (isgoal == true) //in case of completing the puzzle

{

SetConsoleTextAttribute(h, 2);

cout << "You have reached your Goal state" << endl;

cout << "Game Over" << endl;

change(goalState, init\_state);

printpath();

file(init\_state, goalState, const\_array, attempt);

}

if (choice != '1')

{ //incase when user exit the game itself

cout << "Game over" << endl;

change(goalState, init\_state);

printpath();

file(init\_state, goalState, const\_array, attempt);

}

system("pause>0");

return 0;

}

# **(HEADER.h)**

#pragma once

#include<iostream>

#include<iomanip>

#include<windows.h>

#include<fstream>

#include<ctime>

using namespace std;

int const\_array[5][5];

char attempt[1000];

int counter = 0;

void initi(int init\_state[5][5]) //for initializing the initial state

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0; i < 5; i++) //initializing the initial state array to 0 for avoiding garbage values

{

for (int j = 0; j < 5; j++)

{

init\_state[i][j] = 0;

}

}

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; )

{

int k, l, count = 0;

init\_state[i][j] = rand() % 25 + 1; //random values

for (k = 0; k < 5; k++)

{

for (l = 0; l < 5; l++)

{

if (init\_state[i][j] == init\_state[k][l])

{

count++;

}

}

}

if (count == 1) //for generating unique values on each index of array

{

j++;

}

}

}

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

const\_array[i][j] = init\_state[i][j]; //here const\_array will remain same throughout the program

//will only change initial array for updating moves

}

cout << endl;

}

}

void change(int goalState[5][5], int init\_state[5][5]) //for printing all states when function is called

{ //initial state(const\_array)

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE); //goal state

SetConsoleTextAttribute(h, FOREGROUND\_BLUE | FOREGROUND\_INTENSITY); //updated state(status)

cout << setw(25) << "\*\*\*INITIAL STATE\*\*\*" << endl << endl << setw(3);

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

if (const\_array[i][j] == 25) //const\_array printing

{

cout << "|" <<setw(3)<< " " << "|"<<setw(3);

}

else

cout << "|" << setw(3)<< const\_array[i][j] << "|" << setw(3);

}

cout << endl;

}

cout << endl;

SetConsoleTextAttribute(h, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY); //giving colours

cout << setw(25) << "\*\*\*\*GOAL STATE\*\*\*\*" << endl << setw(3);

cout << endl;

for (int i = 0; i < 5; i++) //goalstate printing

{

for (int j = 0; j < 5; j++)

{

if (goalState[i][j] == 25)

{

cout << "|" << setw(3)<< " " << "|" << setw(3);

}

else

cout << "|" << setw(3) << goalState[i][j] << "|" << setw(3);

}

cout << endl;

}

cout << endl;

cout << setw(25) << "\*\*\*\*STATUS\*\*\*\*" << endl<<setw(3);

cout << endl;

for (int i = 0; i < 5; i++) //updated array printing

{

for (int j = 0; j < 5; j++)

{

if (init\_state[i][j] == 25)

{

cout << "|" <<setw(3) << " " << "|" << setw(3);

}

else

cout << "|" << setw(3) << init\_state[i][j] << "|" << setw(3);

}

cout << endl;

}

}

void goal(int goalState[5][5])

{

for (int i = 0; i < 5; i++) //randomly initializing goal state with unique value in each index

{

for (int j = 0; j < 5; )

{

int k, l, count = 0;

goalState[i][j] = rand() % 25 + 1;

for (k = 0; k < 5; k++)

{

for (l = 0; l < 5; l++)

{

if (goalState[i][j] == goalState[k][l])

{

count++;

}

}

}

if (count == 1)

{

j++;

}

}

}

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(h, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

cout << setw(25) << "\*\*\*\*GOAL STATE\*\*\*\*" << endl << setw(3);

cout << endl;

for (int i = 0; i < 5; i++) //printing goal state

{

for (int j = 0; j < 5; j++)

{

if (goalState[i][j] == 25)

cout << "|" << setw(3) << " " << "|" << setw(3);

else

cout << "|" << setw(3) << goalState[i][j] << "|" << setw(3);

}

cout << endl;

}

}

bool isSolvable(int goalState[5][5], int init\_state[5][5]) //checking solvability

{

bool check1 = false; //checking the solvabity by taking inversion count of both initial and goal state

bool check2 = false;

bool solve = false;

int count1 = 0, count2 = 0;

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

for (int x = i; x < 5; x++)

{

for (int y = j; y < 5; y++)

{

if (init\_state[i][j] != 25 && init\_state[x][y] != 25)

{

if (init\_state[i][j] < init\_state[x][y])

{

count1++;

}

}

}

}

}

}

if (count1 % 2 == 0)

{

check1 = true;

}

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

for (int x = i; x < 5; x++)

{

for (int y = j; y < 5; y++)

{

if (goalState[i][j] != 25 && goalState[x][y] != 25)

{

if (goalState[i][j] < goalState[x][y])

{

count2++;

}

}

}

}

}

}

if (count2 % 2 == 0)

{

check2 = true;

}

if (check1 == true && check2 == true) //if both counts are even then the puzzle is solvable

{

solve = true;

}

else if (check1 == false && check2 == false) //if both counts are odd then the puzzle is solvable

{

solve = true;

}

return solve; //in any other case the puzzle is not solvable

}

bool isGoal(int init\_state[5][5], int goalState[5][5]) //check is the puzzle is solved or not

{

bool check;

int k = 0, l = 0;

for (int i = 0; i < 5; i++) //comparing both states

{

for (int j = 0; j < 5; j++)

{

if (init\_state[i][j] == goalState[k][l])

{

check = true;

}

else

{

check = false;

break;

}

l++;

}

if (check == true)

k++;

else

break;

}

return check;

}

bool legalmoves(int init\_state[5][5], char move)

{

int i, j;

bool check = false;

for (i = 0; i < 5; i++)

{

for (j = 0; j < 5; j++)

{

if (init\_state[i][j] == 25) //taking index of the space(25) and validating the move as per legal moves that can be taken on

{ //that specific index

if (i == 0 && j == 0)

{

switch (move)

{

case 'R':

case 'r':

case 'd':

case 'D':

check = true;

break;

default:

check = false;

}

}

else if ((i == 0 && j == 1) || (i == 0 && j == 2) || (i == 0 && j == 3))

{

switch (move)

{

case 'R':

case 'r':

case 'l':

case 'L':

case 'd':

case 'D':

check = true;

break;

default:

check = false;

}

}

else if (i == 0 && j == 4)

{

switch (move)

{

case 'L':

case 'l':

case 'd':

case 'D':

check = true;

break;

default:

check = false;

}

}

else if ((i == 1 && j == 0) || (i == 2 && j == 0) || (i == 3 && j == 0))

{

switch (move)

{

case 'R':

case 'r':

case 'u':

case 'U':

case 'd':

case 'D':

check = true;

break;

default:

check = false;

}

}

else if (i == 4 && j == 0)

{

switch (move)

{

case 'R':

case 'r':

case 'u':

case 'U':

check = true;

break;

default:

check = false;

}

}

else if ((i == 4 && j == 1) || (i == 4 && j == 2) || (i == 4 && j == 3))

{

switch (move)

{

case 'R':

case 'r':

case 'u':

case 'U':

case 'l':

case 'L':

check = true;

break;

default:

check = false;

}

}

else if (i == 4 && j == 4)

{

switch (move)

{

case 'L':

case 'l':

case 'u':

case 'U':

check = true;

break;

default:

check = false;

}

}

else if ((i == 3 && j == 4) || (i == 2 && j == 4) || (i == 1 && j == 4))

{

switch (move)

{

case 'L':

case 'l':

case 'u':

case 'U':

case 'd':

case 'D':

check = true;

break;

default:

check = false;

}

}

else

{

check = true;

}

}

}

}

return check;

}

void make\_move(int init\_state[5][5], char move)

{

int temp, row, column;

bool check = legalmoves(init\_state, move); //first check if the move given by user is legal or ot then make move by

//shuffling the value of space with its nearby tile that user want to shuffle

if (check == true)

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

if (init\_state[i][j] == 25)

{

row = i;

column = j;

}

}

}

switch (move) //taking switch case in moves, for shufling with respect to row and column index

{

case 'u':

case 'U':

temp = init\_state[row][column];

init\_state[row][column] = init\_state[row - 1][column];

init\_state[row - 1][column] = temp;

break;

case 'd':

case 'D':

temp = init\_state[row][column];

init\_state[row][column] = init\_state[row + 1][column];

init\_state[row + 1][column] = temp;

break;

case 'l':

case 'L':

temp = init\_state[row][column];

init\_state[row][column] = init\_state[row][column - 1];

init\_state[row][column - 1] = temp;

break;

case 'R':

case 'r':

temp = init\_state[row][column];

init\_state[row][column] = init\_state[row][column + 1];

init\_state[row][column + 1] = temp;

break;

default:

cout << "Invalid move." << endl;

}

}

}

void file(int init\_state[5][5], int goalState[5][5], int const\_array[5][5], char attempt[5])

{ //this is for storing the data in file if the game is over

ofstream outfile("gameOver.txt");

if (outfile.is\_open())

{

outfile << "\n\n Your goal state was: \n"; //storing goal state in .txt file

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

outfile << goalState[i][j] << setw(5);

}

outfile << "\n";

}

outfile << "\n\n Your initial state was: \n"; //storing initial state in .txt file

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

outfile << const\_array[i][j] << setw(5);

}

outfile << "\n";

}

outfile << "\n\n Your ending up state was: \n"; //storing ending up state in .txt file

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

outfile << init\_state[i][j] <<setw(5);

}

outfile << "\n";

}

outfile << "\n\n Your moves are: \n"; //sequence of moves user has taken

for (int i = 0; attempt[i] != '\0'; i++)

{

outfile << attempt[i] << " ";

}

outfile << "\nYour have done " << counter << " moves\n"; //number of moves user has taken

outfile.close();

}

else

{

cout << "Error" << endl;;

}

}

void printpath() //printing the path in console

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(h, 6);

cout << "\n\n Your moves are: \n";

for (int i = 0; attempt[i] != '\0'; i++)

{

cout << attempt[i] << " ";

}

cout << "\nYour have done " << counter << " moves\n";

}

# Screenshots:







