

Name: Morgan jones

Student number:

1706863

Module : (ACCA4017)

Course: Electric and
Electronic Engineering

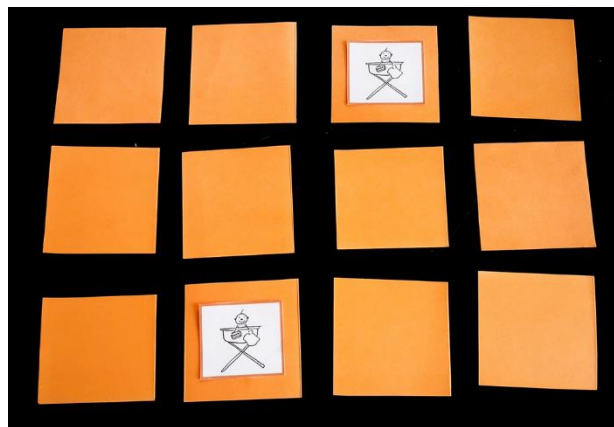
Contents

Name: Morgan jones	1
IMPLEMENTATION – FUNCTION BASED	2
1. Plan	3
2. Annotated Code	3
3. Testing	6
Bibliography	9

IMPLEMENTATION – FUNCTION BASED

Pairs

Develop a C program to allow the user to play a version of a popular card matching game, such as that shown below:



(Note: the game need not be visually complex, and can be implemented with numbers or single letters rather than pictures)

- Identify specific problems in completing the exercise.
 - You may also constrain a problem (simplify it) to produce a simpler solution that may act as a stepping stone to solving the real problem
- Write a clear analysis of the solutions (the thought process behind them) – what logical stages are required. You could use a flowchart or pseudocode
- Show a code listing with annotations
- Show testing (screenshots) that clearly demonstrate that the solutions work under all circumstances

The matching game has various rules which you must identify. For example the player may turn over two “cards” at a time and see their content. Then they must turn the cards back to the blank side before their next turn. The player must attempt to identify matching cards. The goal of the game is to turn over the largest number of matching pairs in the lowest number of moves.

You may implement the rules simply or in a more complex way. You should use at least one function other than the main function. (It is OK to make any arrays used global, to simplify the program. Any variables should be passed as a parameter to a function with a return value if required) Extra credit will be given for more complex implementations.

Note: Remember, there are many marks available for design etc. It is better to have a version of the game that is partially functional, even if it does not incorporate the more complex requirements.

1. Plan

- 1.1 shuffle numbers into rows and columns
- 1.2 ask user for a row and column
- 1.3 print and store chosen first row and column
- 1.4 ask user for a row and column
- 1.5 print and store chosen row and column
- 1.6 print the numbers from the chosen positions
- 1.7 if match shown "cards match" and don't clear board
- 1.8 If no match shown clear board but keep correct pairs
- 1.9 once done repeat process until board all pairs found

2. Annotated Code

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
void shuffle(int[][4]);
int main()
{
    char comma, ans = 'y';
    int rw1, cm1, rw2, cm2, cards[4][4], i(0);
    char cardstatus[4][4]; // array to store whether card is flipped or not
    char gameover = false;
    int moves;
    do
    {
        moves = 0;
        //shuffle
        shuffle(cards); // I think shuffling cards is better idea than shuffling
base start array.
        //display board
        cout << "    1 2 3 4\n";
        cout << " ";
        for (int i = 0; i <= 8; i++)
        {
            cout << "- ";
        }
        cout << endl;
        for (int r = 0; r < 4; r++)
        {
            cout << r + 1 << " | ";
            for (int c = 0; c < 4; c++)
            {
                cout << "* ";
            }
        }
    }
}
```

```

        cardstatus[r][c] = false; // use 0 for false and 1 for
true if true/false is not supported.
    }
    cout << endl;
}
cout << endl;
do // game starts from here as user will do moves here onwards.
{
    do
    {
        //selection
        cout << "Please insert the first card row and column
seperated by a comma.\n";
        cin >> rw1 >> comma >> cm1;
        if (cardstatus[rw1 - 1][cm1 - 1] == true) // check if the
card is flipped.
        {
            cout << "This card is already flipped! Select
another card.";
        }
    } while (cardstatus[rw1 - 1][cm1 - 1] != false);
    do
    {
        cout << "Please insert the second card row and column
seperated by a comma.\n";
        cin >> rw2 >> comma >> cm2;
        if (cardstatus[rw2 - 1][cm2 - 1] == true) // check if the
card is flipped.
        {
            cout << "This card is already flipped! Select
another card.";
        }
        if ((rw1 == rw2) && (cm1 == cm2))
        {
            cout << "You can't select same card twice!";
            continue;
        }
    } while (cardstatus[rw2 - 1][cm2 - 1] != false);
    //fix
    rw1--;
    cm1--;
    rw2--;
    cm2--;
    //reveal
    cout << "    1 2 3 4\n";
    cout << "    ";
    for (int i = 0; i <= 8; i++)
    {
        cout << "-";
    }
    cout << endl;
    for (int r = 0; r < 4; r++)
    {
        cout << r + 1 << " | ";
        for (int c = 0; c < 4; c++)
        {
            if ((r == rw1) && (c == cm1))
            {
                cout << cards[r][c] << " ";
            }
            else if ((r == rw2) && (c == cm2))
            {

```



```

        cout << cards[r][c] << " ";
    }
    else if (cardstatus[r][c] == true)
    {
        cout << cards[r][c] << " ";
    }
    else
    {
        cout << "* ";
    }
    }
    cout << endl;
}
//match?
if (cards[rw1][cm1] == cards[rw2][cm2]) // if cards match, keep
them flipped.
{
    cout << "Cards Matched! Press enter to move on" << endl;

    cardstatus[rw1][cm1] = true;
    cardstatus[rw2][cm2] = true;
}
cin.get();
cin.get();
//this pushes the next board onto a blank screen
for (int b = 0; b <= 20; b++)
    cout << endl;

for (int r = 0; r < 4; r++) // reprint the board, makes it easy
for user to put a new guess.
{
    cout << r + 1 << " | ";
    for (int c = 0; c < 4; c++)
    {
        if (cardstatus[r][c] == true)
        {
            cout << cards[r][c] << " ";
        }
        else
        {
            cout << "* ";
        }
    }
    cout << endl;
}
gameover = true;
for (int r = 0; r < 4; r++) // check all card status, they all
should be true/flipped to end the game.
{
    for (int c = 0; c < 4; c++)
    {
        if (cardstatus[r][c] == false)
        {
            gameover = false;
            break;
        }
    }
    if (gameover == false)
    {
        break;
    }
}

```

```

        moves++; // counting moves here.
        //repeat
    } while (gameover != true); // loop, the only way to get out is to
finish the game!
    cout << "Congratulations, You won!!!" << endl;
    cout << "It Required " << moves << " moves to finish it." << endl <<
endl;

    cout << "Would you like to play again? (y=Yes/n=No) : ";
    ans = cin.get();
    } while (ans == 'y' || ans == 'Y'); // If user wants to play game again.
    cin.get();
    return 0;
}
void shuffle(int cards[][4])
{
    int start[16] = { 1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8 };
    for (int s = 0; s <= 20; s++)
    {
        for (int x = 0; x < 16; x++)
        {
            srand((unsigned)time(NULL));
            int i = rand() % 15 + 1;
            int temp = start[x];
            start[x] = start[i];
            start[i] = temp;
        }
    }
    int i = 0;
    for (int r = 0; r < 4; r++) // put values in cards here
    {
        for (int c = 0; c < 4; c++)
        {
            cards[r][c] = start[i];
            cout << cards[r][c];
            i = i + 1;
        }
        cout << endl;
    }
}

```

3. Testing

3.1 Pairs match

```
Please insert the first card row and column seperated by a comma.
1,1
Please insert the second card row and column seperated by a comma.
3,1
      1 2 3 4
      -----
1 | 4 * * *
2 | * * * *
3 | 4 * * *
4 | * * * *
Cards Matched!
```

Program worked ask expected

3.2 pairs don't match

```
Please insert the first card row and column seperated by a comma.
1,1
Please insert the second card row and column seperated by a comma
      1 2 3 4
      -----
1 | 5 * * *
2 | * 2 * *
3 | * * * *
4 | * * * *

1 | * * * *
2 | * * * *
3 | * * * *
4 | * * * *
Please insert the first card row and column seperated by a comma.
```

Program worked ask expected

3.3 chosen already flipped card

```
1 | 5 * * *
2 | * * * *
3 | * 5 * *
4 | * * * *
Please insert the first card row and column seperated by a comma.
1,1
This card is already flipped! Select another card.Please insert the first card row and column seperated by a comma.
```

Program worked ask expected

3.4 completed game show steps

```
1 | 4 6 7 8
2 | 5 1 2 3
3 | 4 5 6 7
4 | 3 8 1 2
Congratulations, You won!!!
It Required 8 moves to finish it.
```

Program worked ask expected

3.5 restart game

```
Would you like to play again? (y=Yes/n=No) : y
5467
8213
4567
8123
      1 2 3 4
      -----
1 | * * * *
2 | * * * *
3 | * * * *
4 | * * * *

Please insert the first card row and column seperated by a comma.
```

Program worked ask expected

Bibliography

There are no sources in the current document.