

Richard Hankinson p137603

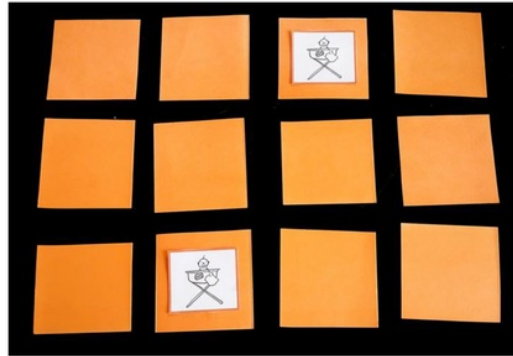
by Richard Peter Hankinson

FILE	12382_RICHARD_PETER_HANKINSON_RICHARD_HANKINSON_P137603_173926_228704158.DOCX (1.08M)		
TIME SUBMITTED	01-MAY-2019 11:45PM (UTC+0100)	WORD COUNT	1309
SUBMISSION ID	106032782	CHARACTER COUNT	6069

PART C – (IMPLEMENTATION – FUNCTION BASED) (due date May 2nd 2019 3pm)

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)

The specific problems in completing the exercise are creating an array with 6 pairs of numbers ranging from 1 to 6. The user needs to be able to select two positions in the array and if the number in these positions are the same, the programme will interpret this as a 'match'. Turns and scores need to be counted and displayed to the user to track progression. To do this, a way of prompting the user to input a position value and then have the programme determine if the stated position has two matching values. If it does, the programme needs to tell the user that they have a match. If it doesn't, the programme needs to tell the user that they have not got a match and then to guess again.

The values in the array need to be swapped around randomly to stop the user being able to memorise the positions in future game plays. To do this a way of randomly swapping the positions needs to be implemented.

The programme must end then all the pairs have been found.

Randomise game

In order to make the positions of the values in the array random, there needs to be a random number generated between 0 and 11. The value in that position then needs to be swapped with a value in another position. Which will also be chosen randomly by the random number generator code. To make a random number, 'srand' needs to be used in conjunction with a way of ending up with a remainder of a value between 0 and 11.

In order to swap the positions of two values in the array a storage variable needs to be used. This is because if a value is added into the array, the value becomes that of what is entered and there is no memory to know what was originally inside that

position. For example, if there was a 1 in position one and a 2 in position two. If the 1 is placed into position two. The value 2 becomes a 1. The array will then show a 1 in position one and a 1 in position two. Rendering the value of 2 lost. This would be useless. So, in order to do accomplish a swap these are the logical steps:

1. Value one – moved to – Value temp
2. Value two – moved to - Value one
3. Value temp – moved to – value two

This will be implemented at the beginning of the game so that the user cannot remember card positions from previous games.

User input and score counting

Input from the user with be two numerical values between 0 and 11. These two numbers must fall in-between these parameters in order to play. The program must count the number or turns the user has had and if any of the pairs have been found. When a pair is found, the values in those positions in the array must be turned into a unused numerical value (pair numbers are 1-6 so they will become 0 if found). To accomplish this an if, else if and else will be used to determine whether a match has been found or not. One around the loop will be counted as one turn using a counter.

Game end

Once the score counter reaches 6 and all pairs have been found the programme will end. Finishing the game. To accomplish this the whole game will be inside a do while loop:

Do – play the game

While – the score is less than 6.

```

9 //Sample code Part C - Pairs game
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <time.h>
13
14 void shuffle(void); //prototype
15
16 int pairs[12] = {6,4,2,3,1,2,5,1,3,4,5,6 }; //array values (6 pairs of 1-6)
17
18 int main()
19 {
20     int card1,card2,turnstaken=0, pairsfound=0, i;
21
22     shuffle();// function to randomise value positions in array
23
24     printf("You are about to play a pairs card game. There are 6 pairs ranging from position 0 - 11. Lets
        play!! \n"); // Explain the game
25
26     do
27     {
28         for (i=0; i <=11; i++)
29         {
30             printf("%d ",pairs[i]); //This shows the array contents for test purposes
31         }
32     }
33     printf("\nNext turn!\n");
34     printf("Enter position of first card");
35     scanf("%d", &card1); // Enter position of card one
36     printf("Enter position of second card");
37     scanf("%d", &card2); //Entre position for card two
38     printf("\nYou turned cards %d and %d. ", card1,card2);
39
40     if (card1 >12) //If user enters a value outside of the parameters
41     {
42         printf("You entered an invalid card possition ");
43     }
44
45     else if (pairs[card1] == 0 || pairs[card2] == 0)//stops user entering possition of previously won
        pairs
46     {
47         printf("\nAt least one of these cards have been taken! Pick again!\n");
48     }
49
50     else if (card1 == card2)//stops user entering same number and getting a match
51     {
52         printf("\nYou can not pick the same card!\n ");
53     }
54
55     else if (pairs[card1] == pairs[card2])
56     {
57         pairsfound+=1; //Add 1 to pairs found score
58         printf(" It's a match!\n");
59         pairs[card1] = 0; //Change found pair value to zero
60         pairs[card2] = 0; //Change found pair value to zero
61     }
62
63     else
64     {
65         printf(" Sorry no luck \n");
66     }
67
68     turnstaken +=1; //add 1 to turns taken variable
69     printf("Turns taken = %d\n", turnstaken); // display value stored in turnstaken variable
70     printf("Pairs found = %d out of 6 \n", pairsfound); // display value stored in pairsfound
71 }

```

```

71 while (pairsfound < 6); // End when all pairs found
72 printf("All pairs found!! \n");
73 return 0;
74 }
75
76 void shuffle() //Function to randomly change the position of the numbers stored in the array
77 {
78     int pairone, pairtwo, temp, i;
79     for (i=0; i <=100; i++) //Counter. Repeat loop 100 times to guarantee randomness
80     {
81         srand ((unsigned int)time(0));
82         pairone = rand() % 12;
83         pairtwo = rand() % 12;
84         temp = pairs[pairtwo];
85         pairs[pairtwo] = pairs[pairone];
86         pairs[pairone] = temp;
87     }
88 }
89
90 }
91

```

Tests

Array position random:

Below shows random array value positions at the beginning of different game plays.

```

You are about to play a pairs card game. There are 6 pairs
ranging from position 0 - 11. Lets play!!
6 2 4 3 1 2 5 1 3 4 5 6
Next turn!
Enter position of first card
You are about to play a pairs card game. There are 6 pairs
ranging from position 0 - 11. Lets play!!
4 4 2 3 1 2 5 1 3 6 5 6
Next turn!
Enter position of first card
You are about to play a pairs card game. There are 6 pairs
ranging from position 0 - 11. Lets play!!
6 4 2 3 1 2 5 6 3 4 5 1
Next turn!
Enter position of first card

```

Winning a pair:

Below example showing a correct guess. Notice positions 0 and 6 in the array have been zeros. Also turns taken and pairs found counters working correctly.

```

You are about to play a pairs card game. There are 6
pairs ranging from position 0 - 11. Lets play!!
5 4 2 3 1 2 5 1 3 4 6 6
Next turn!
Enter position of first card 0
Enter position of second card 6

You turned cards 0 and 6. It's a match!
Turns taken = 1
Pairs found = 1 out of 6
0 4 2 3 1 2 0 1 3 4 6 6
Next turn!
Enter position of first card |

```

Game completed:

Below the game has been terminated when all 6 pairs have been found. Also, there is a little prompt to tell the user all the pairs have been found.

```

You turned cards 2 and 5. It's a match!
Turns taken = 5
Pairs found = 4 out of 6
0 0 0 3 1 0 0 1 3 0 0 0
Next turn!
Enter position of first card3
Enter position of second card8

You turned cards 3 and 8. It's a match!
Turns taken = 6
Pairs found = 5 out of 6
0 0 0 0 1 0 0 1 0 0 0 0
Next turn!
Enter position of first card4
Enter position of second card7

You turned cards 4 and 7. It's a match!
Turns taken = 7
Pairs found = 6 out of 6
All pairs found!!
Program ended with exit code: 0

```

Same position twice:

Below the user has entered the same number for each card position. The code to stop this has worked and has displayed an error message to the user. Not that the turn counter is still counting because this is being considered as a turn.

```

You are about to play a pairs card game. There are 6 pairs
ranging from position 0 - 11. Lets play!!
6 4 2 4 1 2 5 1 3 3 5 6
Next turn!
Enter position of first card 2
Enter position of second card 2

You turned cards 2 and 2.
You can not pick the same card!
Turns taken = 1
Pairs found = 0 out of 6
6 4 2 4 1 2 5 1 3 3 5 6
Next turn!
Enter position of first card

```

Below the same test but with different numbers:

```

Next turn!
Enter position of first card 0
Enter position of second card0

You turned cards 0 and 0.
You can not pick the same card!
Turns taken = 2
Pairs found = 0 out of 6
6 4 2 4 1 2 5 1 3 3 5 6
Next turn!
Enter position of first card

```

```

You turned cards 7 and 7.
You can not pick the same card!
Turns taken = 5
Pairs found = 0 out of 6
6 4 2 4 1 2 5 1 3 3 5 6
Next turn!
Enter position of first card

```

This was originally a problem before the code was amended to deal with this problem because by selecting the same card position twice, the programme saw the same number twice and thought this was a match.

Incorrect values:

Below is an example of the user entering a value outside the stated value range. Resulting in the program displaying an error message.

```
You are about to play a pairs card game. There are 6 pairs
ranging from position 0 - 11. Lets play!!
6 4 2 3 1 2 4 1 3 5 5 6
Next turn!
Enter position of first card 100
Enter position of second card 4

You turned cards 100 and 4. You entered an invalid card
position Turns taken = 1
Pairs found = 0 out of 6
6 4 2 3 1 2 4 1 3 5 5 6
Next turn!
Enter position of first card
```

Already won cards:

Below the user entered a card position that had already been found to be a pair. The program has displayed an error message telling the user about this. This is still considered a turn as the user should be using their memory to remember what cards have been taken.

```
You turned cards 0 and 11. It's a match!
Turns taken = 1
Pairs found = 1 out of 6
0 4 2 3 1 2 5 1 3 4 5 0
Next turn!
Enter position of first card 0
Enter position of second card 2

You turned cards 0 and 2.
At least one of these cards have been taken! Pick again!
Turns taken = 2
Pairs found = 1 out of 6
0 4 2 3 1 2 5 1 3 4 5 0
Next turn!
Enter position of first card
```

Faults:

Below the user has inserted unused values and a error has been displayed but it is the wrong error. Invalid card position should have been displayed but instead the 'at least one of these cards has been taken' message has been displayed.

```
You turned cards -1 and 13.
At least one of these cards have been taken! Pick again!
Turns taken = 2
Pairs found = 0 out of 6
6 4 2 3 1 2 4 1 3 5 5 6
Next turn!
Enter position of first card|
```

Code:

```
//
// main.c
// Pairs Game Richard Hankinson
//
// Created by Richard Hankinson on 25/03/2019.
// Copyright © 2019 Richard Hankinson. All rights reserved.
//

//Sample code Part C - Pairs game
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void shuffle(void); //prototype

int pairs[12] = {6,4,2,3,1,2,5,1,3,4,5,6}; //array values (6 pairs of 1-6)

int main()
{
    int card1, card2, turnstaken=0, pairsfound=0, i;

    shuffle(); // function to randomise value positions in array

    printf("You are about to play a pairs card game. There are 6 pairs ranging from position 0 - 11. Lets play!! \n"); // Explain the
    game

    do
    {
        for (i=0; i <= 11; i++)
        {
            printf("%d ", pairs[i]); //This shows the array contents for test purposes
        }

        printf("\nNext turn!\n");
        printf("Enter position of first card");
        scanf("%d", &card1); // Enter position of card one
        printf("Enter position of second card");
        scanf("%d", &card2); //Entre position for card two
        printf("\nYou turned cards %d and %d. ", card1, card2);

        if (card1 > 12) //If user enters a value outside of the parameters
        {
            printf("You entered an invalid card position ");
        }

        else if (pairs[card1] == 0 || pairs[card2] == 0) //stops user entering position of previously won pairs
        {
            printf("\nAt least one of these cards have been taken! Pick again!\n");
        }
    }
```

```

    else if (card1 == card2) //stops user entering same number and getting a match
    {
        printf("\nYou can not pick the same card!\n ");
    }

    else if (pairs[card1] == pairs[card2])
    {
        pairsfound+=1; //Add 1 to pairs found score
        printf(" It's a match!\n");
        pairs[card1] = 0; //Change found pair value to zero
        pairs[card2] = 0; //Change found pair value to zero
    }
    else
    {
        printf(" Sorry no luck \n");
    }
    turnstaken +=1; //add 1 to turns taken variable
    printf("Turns taken = %d\n", turnstaken); // display value stored in turnstaken variable
    printf("Pairs found = %d out of 6 \n", pairsfound); // display value stored in pairsfound
}
while (pairsfound < 6); // End when all pairs found
printf("All pairs found!! \n");
return 0;
}

void shuffle() //Function to randomly change the position of the numbers stored in the array
{
    int pairone, pairtwo, temp, i;
    for (i=0; i <=100; i++) //Counter. Repeat loop 100 times to garentee randomness
    {
        srand ((unsigned int)time(0));
        pairone = rand() % 12;
        pairtwo = rand() % 12;
        temp = pairs[pairtwo];
        pairs[pairtwo] = pairs[pairone];
        pairs[pairone] = temp;
    }
}

```

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

PART C

Coding/Annotation- A working games with limited function work. You have thought about validating cards >12 - this is a good thought, but really this should be >11(the array is 0-11) and you should also validate for negative numbers too. Good effort overall 13/20

Documentation - I would like more of a logical breakdown in numbered format (as shown in my sample document on Moodle) However, you have clearly though through the stages. You should talk from a logic point of view - there's a bit too much coding creeping in - remember, the logical breakdown should be written before you start coding. Testing is quite good, but don't forget to test the boundaries of your if statements eg the user entering 11 (or 12 in your case!) Also negative numbers as explained above 9/15

Total 22/35

Overall total for assignment: 42.5+22

64.5%

Test mark 78%

Overall mark for module 69 %

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8
