

Programming in C

Author: Jake Russell

Student ID: 1602835

Lecturer: Tim Bashford

1 CONTENTS

2	Introduction	3
3	Exercise 1	3
3.1	List of Problems	3
3.2	Tests	5
3.2.1	Test 1	5
3.2.2	Test 2	5
3.2.3	Test 3	6
4	Exercise 2	7
4.1	List of problems	7
4.2	Tests	9
4.2.1	Test 1	9
4.2.2	Test 2	9
4.2.3	Test 3	10
4.2.4	Test 4	11
4.2.5	Test 5	13

2 INTRODUCTION

For this assignment there were 3 exercises to be attempted, one to make a game of higher or lower, one in which players hide and find each other's treasures, and finally a game of connect four.

3 EXERCISE 1

3.1 LIST OF PROBLEMS

The first stage to making the program is to find and list the problems to be solved from easiest to hardest. The list of problems are as follows:

- Must ask user to enter 7 values
- Must read the values from the user
- Must store these values
- Must tell player 2 the first value
- Must repeatedly ask player 2 higher or lower
- Must only allow answers between 1 and 10
- Must compare the current number with the next and decide whether player 2's answer was correct
- Must end the game with stating player 1 as the winner if they are incorrect
- If they are correct, must tell them they are correct while telling them the next number

The following image is the authors notes when figuring out how to handle each problem:

- Must ask user to enter 7 values → cout statement
- Must read values entered → getline statement
- Must store these values → array[7]
- Must tell player 2 the first value → cout → array[0]
- Must repeatedly ask Player 2 higher or lower → loop →
↳ for loop → 7 repeats
- Must only allow values between 1 and 10 → if statement
- Must compare current number with next number and
decide whether player 2's ~~answe~~ answer was correct
↳ two if statements, one for comparing, one for checking
- Must end game if incorrect while stating player 1
as the winner → if statement → if incorrect → cout
"incorrect" → return 0
- if correct, tell user and tell them next number → if
statement → if correct → cout "next value" array[i]

Figure 1: Notes 1

3.2 TESTS

3.2.1 Test 1

First test is to make sure that the program will only take answers from the user that are between 1 and 10. To test this the user shall enter value 0, value 10, value 9 and value 1 one below the parameters, one above, one just within the lower echelon and one just within the upper echelon, if the program is working correctly it should reject both the first two values and accept the second two.

```
Player 1 enter 7 values:  
Enter number 1> 0  
Please enter a number between 1 and 10  
Enter number 1> 10  
Please enter a number between 1 and 10  
Enter number 1> 9  
Enter number 2> 1  
Enter number 3>
```

Figure 2:Program 1 test 1

As shown the program has successfully met this criterion.

3.2.2 Test 2

Next test is to make sure the higher or lower commands work as intended, to test this the user shall enter two correct answers, one higher and one lower, and two incorrect answers of higher and lower. This test shall also test whether if the user gets an answer wrong, they lose the game.

```
Player 1 enter 7 values:

Enter number 1> 0
Please enter a number between 1 and 10
Enter number 1> 10
Please enter a number between 1 and 10
Enter number 1> 9
Enter number 2> 1
Enter number 3> 5
Enter number 4> 6
Enter number 5> 8
Enter number 6> 2
Enter number 7> 9
The first Value is 9
Higher (h) or lower (l) > l
Correct, the next value is 1
Higher (h) or lower (l) > h
Correct, the next value is 5
Higher (h) or lower (l) > l
incorrect, Player 1 wins
Press any key to continue . . .
```

Figure 3:Program 1 test 2

```
Player 1 enter 7 values:

Enter number 1> 9
Enter number 2> 1
Enter number 3> 5
Enter number 4> 7
Enter number 5> 4
Enter number 6> 8
Enter number 7> 3
The first Value is 9
Higher (h) or lower (l) > h
incorrect, Player 1 wins
Press any key to continue . . .
```

Figure 4:Program 1 test 2.1

As shown the program has successfully met both criteria.

3.2.3 Test 3

The final test is to check whether if the user gets all the answers correct, they win the game.

```
Player 1 enter 7 values:
Enter number 1> 1
Enter number 2> 2
Enter number 3> 3
Enter number 4> 4
Enter number 5> 5
Enter number 6> 6
Enter number 7> 7
The first Value is 1
Higher (h) or lower (l) > h
Correct, the next value is 2
Higher (h) or lower (l) > h
Correct, the next value is 3
Higher (h) or lower (l) > h
Correct, the next value is 4
Higher (h) or lower (l) > h
Correct, the next value is 5
Higher (h) or lower (l) > h
Correct, the next value is 6
Higher (h) or lower (l) > h
Correct, the next value is 7
Player 2 Wins
Press any key to continue . . .
```

Figure 5:Program 1 test 3

As shown above the program has successfully met this criterion.

4 EXERCISE 2

4.1 LIST OF PROBLEMS

The problems listed for the second exercise are as follows:

- Must ask user to place treasures
- Must take the x and y coordinates from the user and store them
- Must ask the second user for the x and y coordinates for their guess
- Must take their choices and store them
- Must compare their guess with the coordinates of the treasures
- Must have a life count that decreases if they are incorrect
- Must lose the game if lives reach 0
- Must win the game if all three treasures are found
- Must mark the chosen coordinate with an X if they are correct
- Must mark the chosen coordinate with an N if they are incorrect
- Must create an 8x8 grid capable of displaying numbers, letters, and a “-“
- Must display grid

The following images are the notes the author took to handle each problem:

- Must ask user to place treasures → cout statement
- Must take the x and y coordinates from the user and store them → cout "x" → getline → cout "y" → getline
- Must ask the second user for the x and y coordinates for their guess → cout statement
- Must take their choices and store them → cout "x" → getline → cout "y" → getline
- Must compare their guess with the coordinates of the treasures → if statements → one for each treasure
- Must have a life count that decreases if they are incorrect → if statement → integer variable decrements → cout
- Must lose the game if lives reach 0 → leave loop if integer is 0 → cout "Player 1 wins"
- Must win the game if all treasures are found → if statement → integer variable decrements → leave loop if integer is 0 → cout "Player 2 wins"
- Must mark the chosen coordinate with "X" if correct → array[i][j] = "X"
- Must mark the chosen coordinate with ~~"N"~~ if incorrect → array[i][j] = "N"
- Must create an 8x8 grid capable of displaying numbers, letters, and a "-" → string array[9][9] → for loop → nested for loop → if statements → rows and column headers assigned i → must be convert to string → to_string → if not

Figure 6: Notes 2

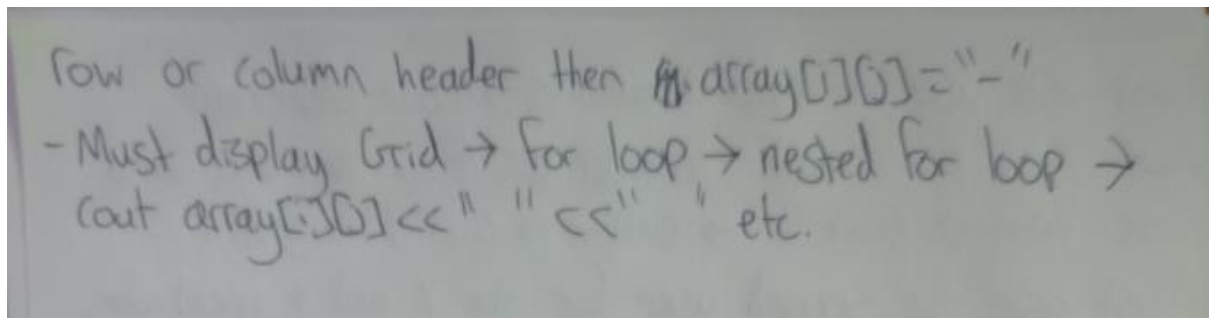


Figure 7:Notes 2.1

4.2 TESTS

4.2.1 Test 1

The first test is to check that the program won't allow the user to enter invalid coordinates, both the higher and lower end will be test by entering a coordinate of 8 and -1. The program should then ask user to enter a valid coordinate instead.

```
Player 1 place treasures now:  
X: -1  
Please enter a coordinate from 0 to 7  
X: 8  
Please enter a coordinate from 0 to 7  
X:
```

Figure 8:Program 2: Test 1

As shown the program was successful in meeting this criterion.

4.2.2 Test 2

The second test is to check whether the grid is updated with an "X" on the coordinate guessed if the user guesses a treasure correctly.

```

Player 1 place treasures now:
X: 1
Y: 1
Treasure 1 placed
X: 2
Y: 2
Treasure 2 placed
X: 3
Y: 3
Treasure 3 placed
  0    1    2    3    4    5    6    7
0    -    -    -    -    -    -    -    -
1    -    -    -    -    -    -    -    -
2    -    -    -    -    -    -    -    -
3    -    -    -    -    -    -    -    -
4    -    -    -    -    -    -    -    -
5    -    -    -    -    -    -    -    -
6    -    -    -    -    -    -    -    -
7    -    -    -    -    -    -    -    -
Player 2 Guess
X: 1
Y: 1
You found a treasure!
  0    1    2    3    4    5    6    7
0    -    -    -    -    -    -    -    -
1    -    X    -    -    -    -    -    -
2    -    -    -    -    -    -    -    -
3    -    -    -    -    -    -    -    -
4    -    -    -    -    -    -    -    -
5    -    -    -    -    -    -    -    -
6    -    -    -    -    -    -    -    -
7    -    -    -    -    -    -    -    -
Player 2 Guess
X:

```

Figure 9:Program 2 test 2

As shown above the program successfully executed the criteria.

4.2.3 Test 3

The third test is to check whether the grid is updated with an “N” on the coordinate chosen if the user guesses incorrectly, the user should then also be updated with their remaining lives.

```

Player 1 place treasures now:
X: 1
Y: 1
Treasure 1 placed
X: 2
Y: 2
Treasure 2 placed
X: 3
Y: 3
Treasure 3 placed
      0    1    2    3    4    5    6    7
0      -    -    -    -    -    -    -    -
1      -    -    -    -    -    -    -    -
2      -    -    -    -    -    -    -    -
3      -    -    -    -    -    -    -    -
4      -    -    -    -    -    -    -    -
5      -    -    -    -    -    -    -    -
6      -    -    -    -    -    -    -    -
7      -    -    -    -    -    -    -    -
Player 2 Guess
X: 1
Y: 1
You found a treasure!
      0    1    2    3    4    5    6    7
0      -    -    -    -    -    -    -    -
1      -    X    -    -    -    -    -    -
2      -    -    -    -    -    -    -    -
3      -    -    -    -    -    -    -    -
4      -    -    -    -    -    -    -    -
5      -    -    -    -    -    -    -    -
6      -    -    -    -    -    -    -    -
7      -    -    -    -    -    -    -    -
Player 2 Guess
X: 2
Y: 1
No treasure found
You have 2 lives left
      0    1    2    3    4    5    6    7
0      -    -    -    -    -    -    -    -
1      -    X    -    -    -    -    -    -
2      -    N    -    -    -    -    -    -
3      -    -    -    -    -    -    -    -
4      -    -    -    -    -    -    -    -
5      -    -    -    -    -    -    -    -
6      -    -    -    -    -    -    -    -
7      -    -    -    -    -    -    -    -
Player 2 Guess
X:

```

Figure 10:Program 2 test 3

As shown above the program successfully executed the criteria.

4.2.4 Test 4

The forth test is to check that if the user correctly guesses all three treasures without losing all 3 lives, they win the game, this shall be test by getting two answers incorrect then 3 correct.

```

Player 1 place treasures now:
X: 1
Y: 1
Treasure 1 placed
X: 2
Y: 2
Treasure 2 placed
X: 3
Y: 3
Treasure 3 placed
  0    1    2    3    4    5    6    7
0    -    -    -    -    -    -    -    -
1    -    -    -    -    -    -    -    -
2    -    -    -    -    -    -    -    -
3    -    -    -    -    -    -    -    -
4    -    -    -    -    -    -    -    -
5    -    -    -    -    -    -    -    -
6    -    -    -    -    -    -    -    -
7    -    -    -    -    -    -    -    -
Player 2 Guess
X: 1
Y: 2
No treasure found
You have 2 lives left
  0    1    2    3    4    5    6    7
0    -    -    -    -    -    -    -    -
1    -    -    N    -    -    -    -    -
2    -    -    -    -    -    -    -    -
3    -    -    -    -    -    -    -    -
4    -    -    -    -    -    -    -    -
5    -    -    -    -    -    -    -    -
6    -    -    -    -    -    -    -    -
7    -    -    -    -    -    -    -    -
Player 2 Guess
X: 2
Y: 3
No treasure found
You have 1 lives left
  0    1    2    3    4    5    6    7
0    -    -    -    -    -    -    -    -
1    -    -    N    -    -    -    -    -
2    -    -    -    N    -    -    -    -
3    -    -    -    -    -    -    -    -
4    -    -    -    -    -    -    -    -
5    -    -    -    -    -    -    -    -
6    -    -    -    -    -    -    -    -
7    -    -    -    -    -    -    -    -
Player 2 Guess
X: 1
Y: 1
You found a treasure!
  0    1    2    3    4    5    6    7
0    -    -    -    -    -    -    -    -
1    -    X    N    -    -    -    -    -
2    -    -    -    N    -    -    -    -
3    -    -    -    -    -    -    -    -
4    -    -    -    -    -    -    -    -
5    -    -    -    -    -    -    -    -
6    -    -    -    -    -    -    -    -
7    -    -    -    -    -    -    -    -
Player 2 Guess

```

Figure 11:Program 2 test 4

```

Player 2 Guess
X: 2
Y: 2
You found a treasure!
  0  1  2  3  4  5  6  7
0  -  -  -  -  -  -  -  -
1  -  X  N  -  -  -  -  -
2  -  -  X  N  -  -  -  -
3  -  -  -  -  -  -  -  -
4  -  -  -  -  -  -  -  -
5  -  -  -  -  -  -  -  -
6  -  -  -  -  -  -  -  -
7  -  -  -  -  -  -  -  -
Player 2 Guess
X: 3
Y: 3
You found a treasure!
  0  1  2  3  4  5  6  7
0  -  -  -  -  -  -  -  -
1  -  X  N  -  -  -  -  -
2  -  -  X  N  -  -  -  -
3  -  -  -  X  -  -  -  -
4  -  -  -  -  -  -  -  -
5  -  -  -  -  -  -  -  -
6  -  -  -  -  -  -  -  -
7  -  -  -  -  -  -  -  -
Player 2 wins
Press any key to continue . . .

```

Figure 12:Program 2 test 4.1

As shown above the program was successful in executing this criterion.

4.2.5 Test 5

The final test is to check that the user loser if they run out of their 3 lives.

```

Player 1 place treasures now:
X: 1
Y: 1
Treasure 1 placed
X: 2
Y: 2
Treasure 2 placed
X: 3
Y: 3
Treasure 3 placed
    0     1     2     3     4     5     6     7
0     -     -     -     -     -     -     -     -
1     -     -     -     -     -     -     -     -
2     -     -     -     -     -     -     -     -
3     -     -     -     -     -     -     -     -
4     -     -     -     -     -     -     -     -
5     -     -     -     -     -     -     -     -
6     -     -     -     -     -     -     -     -
7     -     -     -     -     -     -     -     -
Player 2 Guess
X: 1
Y: 2
No treasure found
You have 2 lives left
    0     1     2     3     4     5     6     7
0     -     -     -     -     -     -     -     -
1     -     -     N     -     -     -     -     -
2     -     -     -     -     -     -     -     -
3     -     -     -     -     -     -     -     -
4     -     -     -     -     -     -     -     -
5     -     -     -     -     -     -     -     -
6     -     -     -     -     -     -     -     -
7     -     -     -     -     -     -     -     -
Player 2 Guess
X: 2
Y: 3
No treasure found
You have 1 lives left
    0     1     2     3     4     5     6     7
0     -     -     -     -     -     -     -     -
1     -     -     N     -     -     -     -     -
2     -     -     -     N     -     -     -     -
3     -     -     -     -     -     -     -     -
4     -     -     -     -     -     -     -     -
5     -     -     -     -     -     -     -     -
6     -     -     -     -     -     -     -     -
7     -     -     -     -     -     -     -     -
Player 2 Guess
X: 3
Y: 4
No treasure found
You have 0 lives left
    0     1     2     3     4     5     6     7
0     -     -     -     -     -     -     -     -
1     -     -     N     -     -     -     -     -
2     -     -     -     N     -     -     -     -
3     -     -     -     -     N     -     -     -
4     -     -     -     -     -     -     -     -
5     -     -     -     -     -     -     -     -
6     -     -     -     -     -     -     -     -
7     -     -     -     -     -     -     -     -
Player 1 wins
Press any key to continue . . .

```

Figure 13:Program 2 test 5

As shown above the program successfully executed the criteria.

5 APPENDIX

5.1 PROGRAM 1

```
#include <sstream>
#include <iostream>
#include <string>

using namespace std;
string input;
stringstream stream;
string checkValue(string answer, int value1, int value2);

int main(int argc, char** argv)
{
    cout << "Player 1 enter 7 values:" << endl << endl;
    int values[7];
    int i = 0;
    int number;
    while (i < 7)
    {
        cout << "Enter number " << i + 1 << "> ";
        getline(cin, input);
        stream.str(input);
        stream >> number;
        stream.clear();
        if (number > 0 && number < 10)
        {
            values[i] = number;
            i++;
        }
        else
        {
            cout << "Please enter a number between 1 and 10" << endl;
        }
    }

    string answer;
    for (i = 0; i < 7; i++)
    {
        if (i == 0)
        {
            cout << "The first Value is " << values[i] << endl;
        }
        else
        {
            cout << "Higher (h) or lower (l) > ";
            getline(cin, answer);
            string verdict = checkValue(answer, values[i-1], values[i]);
            if (verdict == "correct")
            {
                cout << "Correct, the next value is " << values[i] <<
endl;
            }
            else
            {
                cout << "incorrect, Player 1 wins" << endl;
                system("pause");
                return 0;
            }
        }
    }
}
```

```

        }
    }
    cout << "Player 2 Wins" << endl;
    system("pause");
    return 0;
}
string checkValue(string answer, int value1, int value2)
{
    string correctAnswer;
    string verdict;
    if (value1 > value2)
    {
        correctAnswer = "l";
    }
    else
    {
        correctAnswer = "h";
    }
    if (answer == correctAnswer)
    {
        verdict = "correct";
    }
    else
    {
        verdict = "incorrect";
    }
    return verdict;
}

```

5.2 PROGRAM 2

```

#include <sstream>
#include <iostream>
#include <string>

using namespace std;
string input;
stringstream stream;

void displayGrid(string grid[9][9]);
int placeTreasuresX();
int placeTreasuresY();
bool checkFound(int x, int y, int t1X, int t1Y, int t2X, int t2Y, int t3X, int t3Y,
string grid[9][9]);
int main(int argc, char** argv)
{
    string grid[9][9];
    int i;
    for (i = 0; i < 9; i++)
    {
        int j;
        for (j = 0; j < 9; j++)
        {
            grid[i][j] = " ";
            if (i == 0 && j != 0)
            {
                int a = j - 1;
                std::string s = to_string(a);
                grid[i][j] = s;
            }
            if (i != 0 && j == 0)

```

```

        {
            int a = i - 1;
            std::string s = to_string(a);
            grid[i][j] = s;
        }
        if (i && j != 0)
        {
            grid[i][j] = "-";
        }
    }

}

int t1Coord1;
int t1Coord2;
int t2Coord1;
int t2Coord2;
int t3Coord1;
int t3Coord2;
cout << "Player 1 place treasures now:" << endl;
t1Coord1 = placeTreasuresX();
t1Coord2 = placeTreasuresY();
cout << "Treasure 1 placed" << endl;
t2Coord1 = placeTreasuresX();
t2Coord2 = placeTreasuresY();
cout << "Treasure 2 placed" << endl;
t3Coord1 = placeTreasuresX();
t3Coord2 = placeTreasuresY();
cout << "Treasure 3 placed" << endl;
displayGrid(grid);
int lives = 3;
int treasures = 3;
while (lives && treasures != 0)
{
    bool found;
    cout << "Player 2 Guess" << endl;
    int guessX = placeTreasuresX();
    int guessY = placeTreasuresY();
    found = checkFound(guessX, guessY, t1Coord1, t1Coord2, t2Coord1,
t2Coord2, t3Coord1, t3Coord2, grid);
    if (found == false)
    {
        lives = lives - 1;
        cout << "You have " << lives << " lives left" << endl;
    }
    else
    {
        treasures = treasures - 1;
    }
    displayGrid(grid);
}
if (lives == 0)
{
    cout << "Player 1 wins" << endl;
}
else
{
    cout << "Player 2 wins" << endl;
}

system("pause");

```

```

        return 0;
    }
    void displayGrid(string grid[9][9])
    {
        int width = 9, height = 9;

        for (int i = 0; i < height; i++)
        {
            for (int j = 0; j < width; j++)
            {
                cout << grid[i][j] << " " << " " << " " << " " << " ";

            }
            cout << endl;
        }
    }
    int placeTreasuresX()
    {
        int coord;
        int i = 0;
        while (i < 1)
        {
            cout << "X: ";
            getline(cin, input);
            stream.str(input);
            stream >> coord;
            stream.clear();
            if (coord < 0 || coord > 7)
            {
                cout << "Please enter a coordinate from 0 to 7" << endl;
            }
            else
            {
                return coord;
            }
        }
    }
    int placeTreasuresY()
    {
        int coord;
        int i = 0;
        while (i < 1)
        {
            cout << "Y: ";
            getline(cin, input);
            stream.str(input);
            stream >> coord;
            stream.clear();
            if (coord < 0 || coord > 7)
            {
                cout << "Please enter a coordinate from 0 to 7" << endl;
            }
            else
            {
                return coord;
            }
        }
    }
}

```

```

bool checkFound(int x, int y, int t1X, int t1Y, int t2X, int t2Y, int t3X, int t3Y,
string grid[9][9])
{
    if (x == t1X && y == t1Y)
    {
        cout << "You found a treasure!" << endl;
        grid[x+1][y+1] = "X";
        return true;
    }
    if (x == t2X && y == t2Y)
    {
        cout << "You found a treasure!" << endl;
        grid[x+1][y+1] = "X";
        return true;
    }
    if (x == t3X && y == t3Y)
    {
        cout << "You found a treasure!" << endl;
        grid[x+1][y+1] = "X";
        return true;
    }
    grid[x+1][y+1] = "N";
    cout << "No treasure found" << endl;

    return false;
}

```