

Programming in C

by Daniel Algie

FILE	4108_DANIEL_ALGIE_PROGRAMMING_IN_C_178138_1129390361.DOCX (239.75K)		
TIME SUBMITTED	05-APR-2019 10:55AM (UTC+0100)	WORD COUNT	5880
SUBMISSION ID	104165821	CHARACTER COUNT	28686



UWTSD Swansea
Programming in C
Dr Tim Bashford
Daniel Algie 1706977

Contents

Introduction	3
Game 1 - High and Low Game.....	4
The code.....	4
Testing high or low.....	11
Game 2 - Treasure map game.....	12
The code.....	12
Testing of the Treasure Map Game	18
Game 3 Noughts and Crosses	19
The Code	19
Further programming	23
Testing Noughts and Crosses	24
References	25
Appendix	26
Appendix A – High or Low Game Code	26
Appendix B – Treasure Map Game Code	29
Appendix C – Noughts and Crosses Game Code	35

Table of Figures

Figure 1- Shows the correct inputs for player 1.....	5
Figure 2- Shows incorrect inputs	5
Figure 3- Shows the player input 2 was higher than the player input1.....	6
Figure 4 - Ascii table [2].....	8
Figure 5- no duplicates.....	11
Figure 6-player coordinates	12
Figure 7-coordinates range	13
Figure 8-treasure map duplicates	13
Figure 9-map coordinates and lives	15
Figure 10-player 1 wins.....	16
Figure 11-player 2 wins.....	16
Figure 12-treasure map	17
Figure 13-shows grid and location	20
Figure 14-players taking turns	21
Figure 15-column winner	22
Figure 16-4 in a row	23

Introduction

The report will consist of construction and implementation into 3 different games programmed using C++ these games will consist of a high or low game, treasure map game and connect 4 game. While showing how the games work and run there will be more in-depth on how the programmes were created to make the games work and will be showing screen shots of the games working and explaining any challenges and faults that came from creating these programmes.

Game 1 - High and Low Game

The high and low game consists of two players the first player will input 7 numbers with a range of 1 to 9. The second player is displayed players 1 first input to start of the high or low game once player 2 has the first number player 2 must guess if the number is higher or lower than the players 1 first input using the keypad using "h" for high and "l" for low if player 2 guess is correct he continues on to the 3rd number and so fourth till all numbers are guessed and he wins the game. If player 2 guesses wrong at any point in the game the game is over.

The code

This is where we will show and explain what it is the code does to make the game work.

```
void player1Inputs(int myNumbers[7], char highLow[6])
{
    stringstream stream;
    string input = "";
    int i;

    for (i = 0; i < 7; i++) //loops for player 1 to enter numbers
    {
        cout << "Enter number " << i + 1 << " >" << endl;
        string input = "";
        getline(cin, input);
        stream.str(input);
        stream >> myNumbers[i]; //puts value into numbers in array element
        stream.clear();
    }
}
```

In this part of the code it shows how player 1 inputs the numbers and how the numbers are stored in array to store the 7 inputs the player inputs

```
if (myNumbers[i] > 0 && myNumbers[i] <= 9) //to make sure the numbers inputed is
between 0 & 10
{
    cout << "correct choice " << endl; //display message
}
else
{
    i = i - 1; //if number is entered in wrong it will stay on the same element
until correct
    cout << "numbers must be between 1 to 9 please enter correct number" << endl;
}
}
```

The second part of the input of player 1s numbers is to make sure that they are in range of the values this part of the code makes sure that if a player input is between 1 and 9 it will display correct choice if the numbers are out of the range of the required numbers it will display numbers must be between 1 to 9 please enter a correct number and let the player input another number.

```
Enter number 1 >
1
correct choice
Enter number 2 >
5
correct choice
Enter number 3 >
9
correct choice
Enter number 4 >
7
correct choice
Enter number 5 >
6
correct choice
Enter number 6 >
3
correct choice
Enter number 7 >
5
correct choice
```

Figure 1- Shows the correct inputs for player 1

```
Enter number 1 >
45
numbers must be between 1 to 9 please enter correct number
Enter number 1 >
17
numbers must be between 1 to 9 please enter correct number
Enter number 1 >
92
numbers must be between 1 to 9 please enter correct number
Enter number 1 >
4
correct choice
Enter number 2 >
```

Figure 2- Shows incorrect inputs

Figure 2 Shows incorrect inputs entered and showing numbers must be between 1 and 9 this will keep looping until player 1 enters a correct input value.

```

for (i = 0; i < 6; i++) //loops to figure high or low and into highlow array
{
    if (myNumbers[i] < myNumbers[i + 1])
    {
        highLow[i] = 'H';
    }
    else if (myNumbers[i] > myNumbers[i + 1])
    {
        highLow[i] = 'L';
    }
}
cout << "player 2 the first value is " << myNumbers[0] << endl; //prints 1st
value
}

```

The next part of the code shown is to work out if the numbers inputted is higher or lower than the number inputted before to complete this we used a for loop where I was used in an array to store the numbers i was started at 0 and when i is smaller than 6 (so it will loop from numbers 0 to 6 or as the player sees 1 to 7)

We used an if statement so that if the number the player entered was higher than the previous number entered this is where the “i + 1” comes in to play and checks the next number in the array. This will determine if that number is higher and if so the program sets the value ‘H’ into the high low array ready so when player 2 inputs his choice the programme already knows the outcome. An else if was also used the same way as the if to show the value was lower.

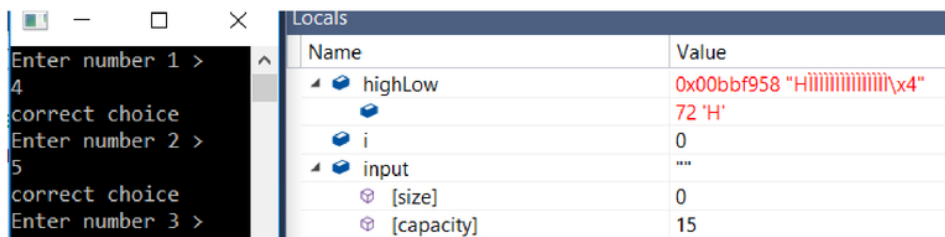


Figure 3- Shows the player input 2 was higher than the player input1

The next part of the programme shows how the player 2 guesses work.

```

for (i = 0; i < 6; i++) //loops for number of guesses
{
    cout << " is number higher or lower ? " << endl;
    getline(cin, input);
    stream.str(input);
    stream >> guess;
    stream.clear();
}

```

This part of the code is used for the player 2 inputs where the player guesses high or low a for loop was used the same way as the number input for player 1 to store the numbers inputted [1]. Where player2 inputs high or lower to guess the number of times it will loop until all the guesses have been reached.

```
int asciiGuess = 0;
    asciiGuess = (int)guess;

    if (asciiGuess == 104 || asciiGuess == 72) //check if guess is lowercase
or capital
    {
        guess = 'H'; //replaces it with capital to match high or low
array
    }
    else if (asciiGuess == 108 || asciiGuess == 76) //check if guess is
lowercase or capital
    {
        guess = 'L'; //replaces it with capital to match high or low array
    }
    else //catch all
    {
        cout << "wrong character used " << endl;
    }
}
```

This part of the code was used to stop player 2 using any other character other than h H and l L to implement this was needed to use the ascii table to set the value of the letters. The ascii guess was created and to equal guess so every player 1 guess would equal the same value as the ascii guess. Using the if statement to set that if ascii guess was equal to 72 or 104 setting that the player could use either a low cases or higher case H for higher and wouldn't have any error when guessing using the correct letter the else if statement was used the same setting either 76 or 108 for player inputting the l for lower, the else statement was used when any other letter was used to guess it would display an error message ("wrong character used") to show that the character used to guess was wrong.

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
\$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c		124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

Figure 4 - Ascii table [2]

```

player 2 the first value is 4
is number higher or lower ?
H
Correct, the next value is 5
is number higher or lower ?
h
Correct, the next value is 7
is number higher or lower ?
l
Correct, the next value is 6
is number higher or lower ?
L
Correct, the next value is 1
is number higher or lower ?
q
wrong charachter used
Incorrect, Sorry Player 1 Wins
Press any key to continue . . .

```

Image 5. Shows the coorect characters and wrong characters used and showing can be in lower case or higher.

```

if (guess == highLow[i]) //checks guess against high low array
{
    count++;
    if (count < 6)
    {
        cout << "Correct, the next value is " << myNumbers[i + 1] << endl;
    }
    else
    {
        cout << "Well done, Player 2 Wins " << endl;
    }
}
else
{
    cout << "Incorrect, Sorry Player 1 Wins " << endl;
    i = 10;
}

```

This part of the code is to work out if the player2 input is equal to the value that was calculated and store in the high and low array. If player 2 enters the correct character equal to high low array it will display to the user "correct, the next value is" and move player 2 on to the next guess if player 2 enters the wrong character and was not equal to high low array it will say "incorrect sorry player 1 wins". If the player 2 guesses all the correct characters it display player 2 is the winner, a count was used to control the displaying of next number or player 2 winning text. To show the game is over.

```

correct choice
Enter number 2 >
6
correct choice
Enter number 3 >
2
correct choice
Enter number 4 >
3
correct choice
Enter number 5 >
8
correct choice
Enter number 6 >
1
correct choice
Enter number 7 >
7
correct choice
player 2 the first value is 4
is number higher or lower ?
h
Correct, the next value is 6
is number higher or lower ?
l
Correct, the next value is 2
is number higher or lower ?
l
Incorrect, Sorry Player 1 Wins
Press any key to continue . . .

```

Image 6. Shows correct guesses or wrong guess.

```

if (myNumbers[i] == myNumbers[i - 1])
{
    cout << "you cannot enter the same nubmer, plese pick another" <<
endl;
    i = i - 1; //if number is entered in wrong it will stay on the
same element until correct
}

```

This part of the code is for stopping player 1 inputting the same numbers in a row so for example if player 1 enters a 4 and then enters in another 4 in the next input it will display an error message and loop back to let the player enter the value again.

```

Enter number 1 >
4
correct choice
Enter number 2 >
4
you cannot enter the same nubmer, plese pick another
Enter number 2 >

```

Figure 5- no duplicates

Testing high or low

Inputs	Values used	Expected results	Pass or fail
player 1 inputs	4,5,6,7,4,6,3	4,5,6,7,4,6,3	pass
Player 2 characters	H,h,h,L,h,l	H,h,h,l,h,l	Pass
Checking out of range numbers	62,43,-1,0	Wont except numbers	pass
Checking wrong character used	T,w,2,*	Wont except characters	pass
Same number used after each other	4,4	Wont except two numbers in row	pass

Game 2 - Treasure map game

The treasure map game is where there is a grid map and a player enters coordinates where they choose to hide their treasure they hide it in three different locations on the grid map by inputting the coordinates they choose. Once the player has placed the treasure in the right coordinates the treasure is hidden behind the map and the second must guess where the first player has hidden the treasure the second player has 3 chances or lives to find the treasure by entering the references on the grid map he thinks the treasure is hidden behind if the player is correct it will show a x to show the treasure has been found if it is the wrong coordinates it will display a n to show the player found no treasure.

The code

This part of the assignment will be showing the treasure map code to show how the game works.

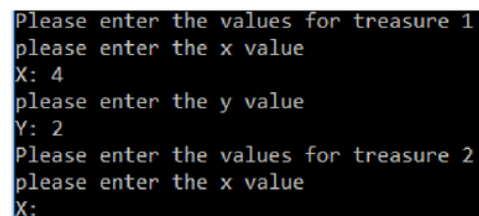
```
cout << "Please enter the values for treasure 1 " << endl;
for (i = 0; i < 2; i++)
{
    if (i == 0)
    {
        cout << "please enter the x value " << endl;
        cout << "X: ";
    }
    else
    {
        cout << "please enter the y value " << endl;
        cout << "Y: ";
    }
    getline(cin, input);
    stream.str(input);
    stream >> player1inputs1[i];
    stream.clear();

    player1inputs1[i] = player1inputs1[i] - 1;
}
```

The code above is used for taking player 1 coordinates for where the player wishes to save the treasure on the map.

To store the coordinates, we have saved them in an array a for loop has been set so the player can input two numbers and save the coordinates.

This has been implemented the same way for the other treasure coordinates with a bit more time and knowledge in the future the player inputs would be all done in one instead of three separate loops.



```
Please enter the values for treasure 1
please enter the x value
X: 4
please enter the y value
Y: 2
Please enter the values for treasure 2
please enter the x value
X:
```

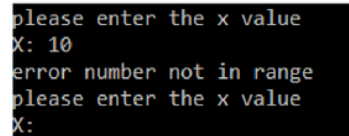
Figure 6-player coordinates

```

if ((player1inputs3[i] > 8) || (player1inputs3[i] < 0))
{
    cout << "error number not in range " << endl;
    i--;
}

```

This was used to make sure the coordinates the player entered were in range of the map if the player coordinates were greater than 8 and smaller than 0 it would display a warning message to let the player know the coordinates used were not in range.



```

please enter the x value
X: 10
error number not in range
please enter the x value
X:

```

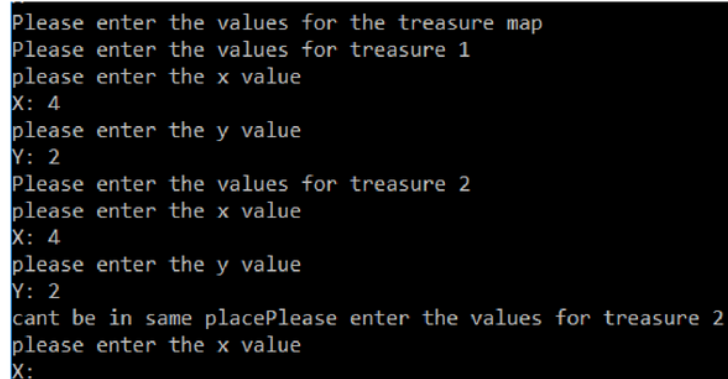
Figure 7-coordinates range

```

//check not in same place as treasure 1
if (player1inputs2[0] == player1inputs1[0])
{
    if (player1inputs2[1] == player1inputs1[1])
    {
        cout << "cant be in same place";
        cout << "Please enter the values for treasure 2 " << endl;
        i = -1;
    }
}

```

This part of the code was used to ensure that player 1 couldn't enter the same coordinates in a row e.g. if the first treasure coordinates are 7,2 and the second coordinates are 7,2 it will display a error message and will make player 1 re enter coordinates for treasure 2 again [3].



```

Please enter the values for the treasure map
Please enter the values for treasure 1
please enter the x value
X: 4
please enter the y value
Y: 2
Please enter the values for treasure 2
please enter the x value
X: 4
please enter the y value
Y: 2
cant be in same placePlease enter the values for treasure 2
please enter the x value
X:

```

Figure 8-treasure map duplicates

```

while (lives != 0 || treasureFound < 3)
{
    DisplayMap(maparray);

    cout << "Player 2 Guess: " << endl;
    for (i = 0; i < 2; i++)
    {
        if (i == 0)
        {
            cout << "X: ";
        }
        else
        {
            cout << "Y: ";
        }

        getline(cin, input);
        stream.str(input);
        stream >> player2Guess[i];
        stream.clear();
    }
}

```

The code above is setting that if lives is not yet reached to 0 it will allow the player to keep getting a chance to enter there coordinates to try and find player 1 treasure all of the guesses from player 2 are stored in the array to compare with player 1 coordinates.

```

if ((player2Guess[0] == player1inputs3[0]))
{
    if (player2Guess[1] == player1inputs3[1])
    {
        maparray[player2Guess[0]][player2Guess[1]] = 'X';
        treasureFound = treasureFound + 1;
        halffail = false;
    }
}

if (halffail == true)
{
    maparray[player2Guess[0]][player2Guess[1]] = 'N';
    lives = lives - 1;
    // how many lives
}
cout << "Your lives left are = " << lives << endl;

```

If player 2 is equal to the coordinates that player 1 entered the programme will display a "X" to show that the correct coordinates has been found it will also add one to the treasure found count to know whether all 3 treasures has been found.

If player 2 coordinates are not that will set the bool to true and will display a "N" to show, there is no treasure in the coordinates that player 2 entered and will take a life from the life count.

```

Player 2 Guess:
X: 8
Y: 2
Your lives left are = 3
  1  2  3  4  5  6  7  8 Y
1 - - - - - - - -
2 - - - - - - - -
3 - - - - - - - -
4 - - - - - - - -
5 - - - - - - - -
6 - - - - - - - -
7 - - - - - - - -
8 - X - - - - - -
X
Player 2 Guess:
X: 4
Y: 8
Your lives left are = 2
  1  2  3  4  5  6  7  8 Y
1 - - - - - - - -
2 - - - - - - - -
3 - - - - - - - -
4 - - - - - - N
5 - - - - - - - -
6 - - - - - - - -
7 - - - - - - - -
8 - X - - - - - -
X

```

Figure 9-map coordinates and lives

```

if (treasureFound == 3)
{
    cout << "congrats player 2 you found all the treasure" << endl;
}
else
{
    cout << "didnt find all the treasure player 1 wins" << endl;
}

```

This statement is used to display which player has won using the if statement if treasure found is equal to 3 (so all treasure has been found) it will display out that player 2 has found all the treasure if player 2 has not found all the treasure it will display didn't find all the treasure player 1 wins.

```

Your lives left are = 0
  1  2  3  4  5  6  7  8  Y
1 - - - - - - - -
2 - - - - N - - -
3 - - - - - - - -
4 - - - - - - - -
5 - - - - - - - -
6 - - - - N - - -
7 - N - - - - - -
8 - - - - - - - -
X
didnt find all the treasure player 1 wins
Press any key to continue . . .

```

Figure 10-player 1 wins

```

  1  2  3  4  5  6  7  8  Y
1 - - - - - - - -
2 - - - - - - - -
3 - - - - - - - -
4 - X - - - - - -
5 - - - - - - - -
6 - - - - - - - -
7 - X - - - - - -
8 - - - - - - X
X
congrats player 2 you found all the treasure

```

Figure 11-player 2 wins

```

void settingarray(char maparray[8][8])
{
    int i, j;
    for (i = 0; i < 8; i++)
    {
        for (j = 0; j < 8; j++)
        {
            maparray[i][j] = '-';
        }
    }
}

```

This part of the code is for setting the grid coordinates for the treasure map it is used in a function and the main programme calls this function when it is needed it was created to display a " - " for every one of the points in the array to make the treasure map visible.

```

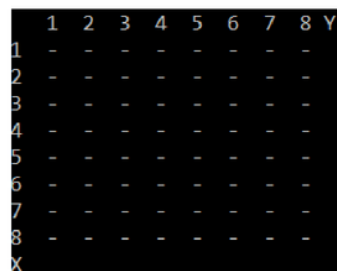
void DisplayMap(char maparray[8][8])
{
    int i, j;

    //displaying the array
    cout << " ";
    for (i = 1; i < 9; i++)
    {
        cout << " " << i;
    }
    cout << " Y";
    cout << endl;

    for (i = 0; i < 8; i++)
    {
        cout << i + 1 << " ";
        for (j = 0; j < 8; j++)
        {
            cout << " " << maparray[i][j] << " ";
        }
        cout << endl;
    };
    cout << "X";
    cout << endl;
}

```

This part of the code was used to set the coordinates for the map. It will display on the map which coordinates are for the column and row, the numbers go from 1 to 8. It also displays which coordinates are x and which coordinates are y making easier for the user to know which coordinates they can pick.



	1	2	3	4	5	6	7	8	Y
1	-	-	-	-	-	-	-	-	
2	-	-	-	-	-	-	-	-	
3	-	-	-	-	-	-	-	-	
4	-	-	-	-	-	-	-	-	
5	-	-	-	-	-	-	-	-	
6	-	-	-	-	-	-	-	-	
7	-	-	-	-	-	-	-	-	
8	-	-	-	-	-	-	-	-	
X									

Figure 12-treasure map

Testing of the Treasure Map Game

Inputs	Values used	Expected results	Pass or fail
player 1 inputs	2,7 4,6 5,2	Accepts inputs and stores coordinates	pass
Player 2 guesses correct	2,7 4,6 5,2	Finds all treasure	Pass
Player 2 guesses wrongly entered	5,6 7,4 8,6	Find no treasure and take all lives away	pass
Checking numbers are in range	10, 20, -7, -9	Wont except numbers	pass
Checking wrong character used	T,w,[,*	Wont except characters	pass
Same number used after each other	2,7 2,7	It won't allow same coordinates twice	pass
Only use one coordinate to find treasure	4	Need both coordinates to fins treasure	pass

Game 3 Noughts and Crosses

Noughts and crosses is a 2 player game where one player is the noughts "O" and the other player is crosses "X". There is a grid and three players take it in turns placing their markers in the coordinates they want their marker to be. But there is a catch it will only go in the first column unless there is a marker already there so it will go above the original marker first person to have four in a row either vertical, horizontally, or diagonally is the winner.

The Code

Here is where it will be explained and showed how the code works to make the game work.

```
void DisplayGrid(char grid[5][6])
{
    int i, j;

    //displaying the array
    for (i = 0; i < 5; i++)
    {
        cout << " " << i;
    }

    cout << endl;

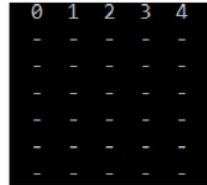
    for (i = 0; i < 6; i++)
    {
        cout << " ";
        for (j = 0; j < 5; j++)
        {
            cout << grid[j][i] << " ";
        }
        cout << endl;
    }
}
```

This part of the code is used to display the grid in which the player will enter their coordinates into. It displays 1 to 4 on the top of the grid to help the player decide where they want to place their marker.

```
void SetGrid(char grid[5][6])
{
    int i, j;

    for (i = 0; i < 6; i++)
    {
        for (j = 0; j < 5; j++)
        {
            grid[j][i] = '-';
        }
    }
}
```

This is setting the grid markings for the grid displaying the “-” to show where the markers can be allocated to when selecting the number shown on top of the grid.



0	1	2	3	4
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-

Figure 13-shows grid and location

```
while (gameEnd == false)
{
    if (playerNum == 1)
    {
        cout << "Player 1 go" << endl;
    }
    else
    {
        cout << "Player 2 go" << endl;
    }

    getline(cin, input);
    stream.str(input);
    stream >> playerGuess;
    stream.clear();

    for (j = 0; j < 6; j++)
    {
        if (grid[playerGuess][j] == '-')
        {
            if (playerNum == 1)
            {
                grid[playerGuess][j] = 'X';
                j = 6;
            }
            else
            {
                grid[playerGuess][j] = 'O';
                j = 6;
            }
        }
    }
}
```

This part of the code is for the players to take turns choosing where they wish to place there markers on the grid. Using a while loop it is set so that gamend == false so as long as no player has matched 4 in a row or column the code will keep looping until the there is a winner.

```

player one goes first you are x's
Player 1 go
4
 0  1  2  3  4
-  -  -  -  X
-  -  -  -  -
-  -  -  -  -
-  -  -  -  -
-  -  -  -  -
-  -  -  -  -
-  -  -  -  -
Player 2 go
4
 0  1  2  3  4
-  -  -  -  X
-  -  -  -  0
-  -  -  -  -
-  -  -  -  -
-  -  -  -  -
-  -  -  -  -
-  -  -  -  -
Player 1 go

```

Figure 14-players taking turns

```

for (i = 0; i < 2; i++)
{
    for (j = 0; j < 5; j++)
    {
        if (grid[j][i] == 'X')
        {
            if (grid[j][i + 1] == 'X')
            {
                if (grid[j][i + 2] == 'X')
                {
                    if (grid[j][i + 3] == 'X')
                    {
                        player1Win = true;
                    }
                }
            }
        }
    }
}

```

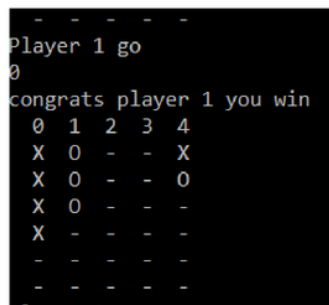
This part of the code is used for working out if 4 of the same markers are in the same column in consecutive order setting the loop to check if the "X" are in the same column it checks for the first X entered then the second and so fourth till there is a match of 4 "X" setting player1Win to =true to set that player 1 is the winner [4]. The same concept has been used for setting player2 "0" and set the bool to player2Win = true.

```

if (player1Win == true)
{
    cout << "congrats player 1 you win " << endl;
    gameEnd = true;
}
if (player2Win == true)
{
    cout << "congrats player 2 you win " << endl;
    gameEnd = true;
}

```

Once established that one player has four in a column it will display the winner to the users.



```

- - - - -
Player 1 go
0
congrats player 1 you win
0 1 2 3 4
X 0 - - X
X 0 - - 0
X 0 - - -
X - - - -
- - - - -
- - - - -

```

Figure 15-column winner

```

for (i = 0; i < 2; i++)
{
    for (j = 0; j < 5; j++)
    {
        if (grid[j+1][i] == '0')
        {
            if (grid[j+1][i] == '0')
            {
                if (grid[j+2][i] == '0')
                {
                    if (grid[j+3][i] == '0')
                    {
                        player2Win = true;
                    }
                }
            }
        }
    }
}

```

This code works the same way as the column but is used for checking if the row has 4 of the same markers instead of the column.

```

congrats player 1 you win
 0 1 2 3 4
0 X 0 X X
X X X X 0
0 - - - 0
0 - - - -
- - - - -
- - - - -

```

Figure 16-4 in a row

Further programming

Thou this is as far as the programme goes in the future it is hoped to have diagonal implemented so can tell when a player has 4 in a row diagonally.

Implement features that will make the game more exciting for the end user where can use character on the keyboard to take out rows and another feature would be the bomb where it is able to delete 4 characters in a square and then the other characters drop down into their place leaving space for more go's and bring more excitement to the game. The major change would be to have the characters start from the bottom not the top. This was not achievable but with more practice and further knowledge this will be achievable and look forward to making this a reality.

Testing Noughts and Crosses

Inputs	Values used	Expected results	Pass or fail
excepts player 1 input and displays correct place and displays correct character	4	Accepts inputs and stores coordinates And displays a x	pass
Excepts player 2 input and displays correct character	2	Accepts inputs and stores coordinates and displays a 0	Pass
Player enters out of range number	10,-2,t,i	Error telling them to enter different number in range	pass
Shows winner when 4 same character in column	4,4,4,4, for x	4 x, in column 4 and display player 1 winner	pass
Show when player has 4 in a row	4,3,2,1	Displays for o in a row	pass

References

- [1 M. Mcgrath, C++ programming in easy steps, Chicago: Southam: In Easy Steps, 2012.
]
- [2 IBM, "IBM," 2nd April 2019. [Online]. Available:
] https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.ioaq100/ascii_table_appendix.htm.
- [3 M. Lee, C++ programming for the absolute beginner, Boston: Mass: Course Technology Cengage Learning, 2009.
- [4 C. p. h. t. programme, Paul J.Deitel, Upper Saddle River, N.J: Pearson Prentice Hall, 2010.
]

Appendix

Appendix A – High or Low Game Code

Here you will find the code for the high or low game

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

using namespace std;
void player2Quess(int myNumbers[7], char highLow[6]);
void player1Inputs(int myNumbers[7], char highLow[6]);

int main(int argc, char** argv)
{
    int myNumbers[7]; //numbers player 1 enters
    char highLow[6];   //array for low and high
    stringstream stream;
    string input = "";
    //int i;
    //char guess; //player 2 guess

    cout << "welcome to the high or low game " << endl;
    cout << "player 1 please start entering 7 numbers ranging from 1 to 9 " << endl;
    player1Inputs(myNumbers, highLow);
    player2Quess(myNumbers, highLow);

    system("Pause");
}

void player1Inputs(int myNumbers[7], char highLow[6])
{
    stringstream stream;
    string input = "";
    int i,j;

    for (i = 0; i < 7; i++) //loops for player 1 to enter numbers
    {
        cout << "Enter number " << i + 1 << " >" << endl;
        string input = "";
        getline(cin, input);
        stream.str(input);
        stream >> myNumbers[i]; //puts value into numbers in array element
        stream.clear();

        if (myNumbers[i] == myNumbers[i - 1])
        {
            cout << "you cannot enter the same nubmer, plese pick another" << endl;
        }
    }
}
```

```

        i = i - 1; //if number is entered in wrong it will stay on the same element until
correct
    }
    else if (myNumbers[i] > 0 && myNumbers[i] <= 9) //to make sure the numbers
inputed is between 0 & 10
    {
        cout << "correct choice " << endl; //display message
    }
    else
    {
        i = i - 1; //if number is entered in wrong it will stay on the same element until
correct
        cout << "numbers must be between 1 to 9 please enter correct number" <<
endl;
    }
}

for (i = 0; i < 6; i++) //loops to figure high or low and into highlow array
{
    if (myNumbers[i] < myNumbers[i + 1])
    {
        highLow[i] = 'H';
    }
    else if (myNumbers[i] > myNumbers[i + 1])
    {
        highLow[i] = 'L';
    }
}
for (j = 0; j < 28; j++) // hide player 1 numbers
{
    cout << endl;
}

cout << "player 2 the first value is " << myNumbers[0] << endl; //prints 1st value
}

void player2Quess(int myNumbers[7], char highLow[6])
{
    int i, count = 0;
    char guess;

    stringstream stream;
    string input = "";

    for (i = 0; i < 6; i++) //loops for number of guesses
    {
        cout << " is number higher or lower ? " << endl;
        getline(cin, input);
        stream.str(input);
        stream >> guess;
        stream.clear();
    }
}

```

```

int asciiGuess = 0;
asciiGuess = (int)guess;

if (asciiGuess == 104 || asciiGuess == 72) //check if guess is lowercase or capital
{
    guess = 'H'; //replaces it with capital to match high or low array
}
else if (asciiGuess == 108 || asciiGuess == 76) //check if guess is lowercase or capital
{
    guess = 'L'; //replaces it with capital to match high or low array
}
else //catch all
{
    cout << "wrong charachter used " << endl;
}

if (guess == highLow[i]) //checks guess against high low array
{
    count++;
    if (count < 6)
    {
        cout << "Correct, the next value is " << myNumbers[i + 1] << endl;
    }
    else
    {
        cout << "Well done, Player 2 Wins " << endl;
    }
}
else
{
    cout << "Incorrect, Sorry Player 1 Wins " << endl;
    i = 10;
}
}
}

```

Appendix B – Treasure Map Game Code

Here you will find the code for the treasure map game

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

using namespace std;
void DisplayMap(char maparray[8][8]);
void settingarray(char maparray[8][8]);
int main(int argc, char** argv)
{
    stringstream stream;
    string input = "";
    int i, j, m;
    int lives = 3;
    char maparray[8][8];
    int player1inputs1[2]; // [x], [z], [1] or 0 for found
    int player1inputs2[2];
    int player1inputs3[2];
    int player2Guess[2]; // [x][y]
    int treasureFound = 0;

    //setting the array
    settingarray(maparray);

    DisplayMap(maparray); //drawing map

    //T1
    cout << "Please enter the values for the treasure map " << endl;
    cout << "Please enter the values for treasure 1 " << endl;
    for (i = 0; i < 2; i++)
    {
        if (i == 0)
        {
            cout << "please enter the x value " << endl;
            cout << "X: ";

        }
        else
        {
            cout << "please enter the y value " << endl;
            cout << "Y: ";

        }
        getline(cin, input);
        stream.str(input);
        stream >> player1inputs1[i];
        stream.clear();

        player1inputs1[i] = player1inputs1[i] - 1;

        if ((player1inputs1[i] > 8) || (player1inputs1[i] < 0))
```

```

        {
            cout << "error number not in range " << endl;
            i--;
        }
    }

//T2
cout << "Please enter the values for treasure 2 " << endl;
for (i = 0; i < 2; i++)
{
    if (i == 0)
    {
        cout << "please enter the x value " << endl;
        cout << "X: ";
    }
    else
    {
        cout << "please enter the y value " << endl;
        cout << "Y: ";
    }

    getline(cin, input);
    stream.str(input);
    stream >> player1inputs2[i];
    stream.clear();

    player1inputs2[i] = player1inputs2[i] - 1;

    if ((player1inputs2[i] > 8) || (player1inputs2[i] < 0))
    {
        cout << "error number not in range " << endl;
        i--;
    }

    //check not in same place as treasure 1
    if (player1inputs2[0] == player1inputs1[0])
    {
        if (player1inputs2[1] == player1inputs1[1])
        {
            cout << "cant be in same place";
            cout << "Please enter the values for treasure 2 " << endl;
            i = -1;
        }
    }
}

//T3
cout << "Please enter the values for treasure 3 " << endl;
for (i = 0; i < 2; i++)
{
    if (i == 0)

```

```

        {
            cout << "please enter the x value " << endl;
            cout << "X: ";
        }
        else
        {
            cout << "please enter the y value " << endl;
            cout << "Y: ";
        }

        getline(cin, input);
        stream.str(input);
        stream >> player1inputs3[i];
        stream.clear();

        player1inputs3[i] = player1inputs3[i] - 1;

        if ((player1inputs3[i] > 8) || (player1inputs3[i] < 0))
        {
            cout << "error number not in range " << endl;
            i--;
        }

        //check not in same place as treasure 2
        if (player1inputs3[0] == player1inputs2[0])
        {
            if (player1inputs3[1] == player1inputs2[1])
            {
                cout << "cant be in same place";
                cout << "Please enter the values for treasure 3 " << endl;
                i = -1;
            }
        }

        //check not in same place as treasure 1
        if (player1inputs3[0] == player1inputs1[0])
        {
            if (player1inputs3[1] == player1inputs1[1])
            {
                cout << "cant be in same place";
                cout << "Please enter the values for treasure 3 " << endl;
                i = -1;
            }
        }
    }
    for (m = 0; m < 36; m++)
    {
        cout << endl;
    }

```

```

while (lives != 0 && treasureFound < 3)
{
    DisplayMap(maparray);

    cout << "Player 2 Guess: " << endl;
    for (i = 0; i < 2; i++)
    {
        if (i == 0)
        {
            cout << "X: ";
        }
        else
        {
            cout << "Y: ";
        }

        getline(cin, input);
        stream.str(input);
        stream >> player2Guess[i];
        stream.clear();

        if (player2Guess[i] > 8 || player2Guess[i] < 1)
        {
            cout << "error number not in range " << endl;
            i--;
        }
        player2Guess[i] = player2Guess[i] - 1;
    }

    bool halffail = true;

    if (player2Guess[0] == player1inputs1[0])    // checking if p2 guess X is same as
    {
        if (player2Guess[1] == player1inputs1[1])
        {
            maparray[player2Guess[0]][player2Guess[1]] = 'X';
            treasureFound = treasureFound + 1;
            halffail = false;
        }
    }

    if ((player2Guess[0] == player1inputs2[0]))
    {
        if (player2Guess[1] == player1inputs2[1])
        {
            maparray[player2Guess[0]][player2Guess[1]] = 'X';
            treasureFound = treasureFound + 1;
            halffail = false;
        }
    }
}

```

T1 X value

```

        if ((player2Guess[0] == player1inputs3[0]))
        {
            if (player2Guess[1] == player1inputs3[1])
            {
                maparray[player2Guess[0]][player2Guess[1]] = 'X';
                treasureFound = treasureFound + 1;
                halffail = false;
            }
        }

        if (halffail == true)
        {
            maparray[player2Guess[0]][player2Guess[1]] = 'N';
            lives = lives - 1;
            // how many lives
        }
        cout << "Your lives left are = " << lives << endl;
    }
    DisplayMap(maparray);

    if (treasureFound == 3)
    {
        cout << "congrats player 2 you found all the treasure" << endl;
    }
    else
    {
        cout << "didnt find all the treasure player 1 wins" << endl;
    }

    system("pause");
}

void DisplayMap(char maparray[8][8])
{
    int i, j;

    //displaying the array
    cout << " ";
    for (i = 1; i < 9; i++)
    {
        cout << " " << i;
    }
    cout << " Y";
    cout << endl;

    for (i = 0; i < 8; i++)
    {
        cout << i + 1 << " ";
        for (j = 0; j < 8; j++)
        {

```

```
        cout << " " << maparray[i][j] << " ";
    }
    cout << endl;
};
cout << "X";
cout << endl;
}
```

```
void settingarray(char maparray[8][8])
{
    int i, j;
    for (i = 0; i < 8; i++)
    {
        for (j = 0; j < 8; j++)
        {
            maparray[i][j] = '-';
        }
    }
}
```

Appendix C – Noughts and Crosses Game Code

Here you will find the code for the noughts and crosses game.

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

using namespace std;
//int playersGo();
void DisplayGrid(char grid[5][6]);
void SetGrid(char grid[5][6]);
bool colCheck(char grid[5][6]);
bool rowCheck(char grid[5][6]);

int main(int argc, char** argv)
{
    stringstream stream;
    string input = "";
    char grid[5][6];
    int playerNum = 1, i, j, n;
    int playerGuess;
    bool gameEnd = false;
    SetGrid(grid);
    DisplayGrid(grid);

    cout << "player one goes first you are x's" << endl;

    while (gameEnd == false)
    {
        if (playerNum == 1)
        {
            cout << "Player 1 go" << endl;
        }
        else
        {
            cout << "Player 2 go" << endl;
        }
        getline(cin, input);
        stream.str(input);
        stream >> playerGuess;
        stream.clear();
        while (playerGuess > 4 || playerGuess < 0)
        {
            cout << "number not in range please chose another number" << endl;
            cout << "please enter another number" << endl;
            getline(cin, input);
            stream.str(input);
            stream >> playerGuess;
            stream.clear();
        }
    }
}
```

```

        for (j = 0; j < 6; j++)
        {
            if (grid[playerGuess][j] == '-')
            {
                if (playerNum == 1)
                {
                    grid[playerGuess][j] = 'X';
                    j = 6;
                }
                else
                {
                    grid[playerGuess][j] = 'O';
                    j = 6;
                }
            }
        }

        gameEnd = colCheck(grid);
        gameEnd = rowCheck(grid);
        DisplayGrid(grid);

        //swaping players
        if (playerNum == 1)
        {
            playerNum = 2;
        }
        else
        {
            playerNum = 1;
        }
    }

    system("pause");
}

bool colCheck(char grid[5][6])
{
    int i, j;
    bool player1Win = false;
    bool player2Win = false;
    bool gameEnd = false;

    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 5; j++)
        {

```

```

        if (grid[j][i] == 'X')
        {
            if (grid[j][i + 1] == 'X')
            {
                if (grid[j][i + 2] == 'X')
                {
                    if (grid[j][i + 3] == 'X')
                    {
                        player1Win = true;
                    }
                }
            }
        }
    }
}

for (i = 0; i < 2; i++)
{
    for (j = 0; j < 5; j++)
    {
        if (grid[j][i] == 'O')
        {
            if (grid[j][i + 1] == 'O')
            {
                if (grid[j][i + 2] == 'O')
                {
                    if (grid[j][i + 3] == 'O')
                    {
                        player2Win = true;
                    }
                }
            }
        }
    }
}

if (player1Win == true)
{
    cout << "congrats player 1 you win " << endl;
    gameEnd = true;
}
if (player2Win == true)
{
    cout << "congrats player 2 you win " << endl;
    gameEnd = true;
}

return gameEnd;
}
bool rowCheck(char grid[5][6])
{

```

```

int i, j;
bool player1Win = false;
bool player2Win = false;
bool gameEnd = false;

for (i = 0; i < 2; i++)
{
    for (j = 0; j < 5; j++)
    {
        if (grid[j][i] == 'X')
        {
            if (grid[j + 1][i] == 'X')
            {
                if (grid[j + 2][i] == 'X')
                {
                    if (grid[j + 3][i] == 'X')
                    {
                        player1Win = true;
                    }
                }
            }
        }
    }
}

for (i = 0; i < 2; i++)
{
    for (j = 0; j < 5; j++)
    {
        if (grid[j + 1][i] == 'O')
        {
            if (grid[j + 1][i] == 'O')
            {
                if (grid[j + 2][i] == 'O')
                {
                    if (grid[j + 3][i] == 'O')
                    {
                        player2Win = true;
                    }
                }
            }
        }
    }
}

if (player1Win == true)
{
    cout << "congrats player 1 you win " << endl;
    gameEnd = true;
}
if (player2Win == true)

```

```

        {
            cout << "congrats player 2 you win " << endl;
            gameEnd = true;
        }

        return gameEnd;
    }

void DisplayGrid(char grid[5][6])
{
    int i, j;

    //displaying the array
    //cout << " ";
    for (i = 0; i < 5; i++)
    {
        cout << " " << i;
    }

    cout << endl;

    for (i = 0; i < 6; i++)
    {
        cout << " ";
        for (j = 0; j < 5; j++)
        {
            cout << grid[j][i] << " ";
        }
        cout << endl;
    }
}

void SetGrid(char grid[5][6])
{
    int i, j;

    for (i = 0; i < 6; i++)
    {
        for (j = 0; j < 5; j++)
        {
            grid[j][i] = '-';
        }
    }
}

```

Programming in C

GRADEMARK REPORT

FINAL GRADE

79/100

GENERAL COMMENTS

Instructor

Exercise 1 (50) 44

Exercise 2 (30) 23

Exercise 3 (20)12

A very good effort made. Tasks 1 and 2 work very nicely. As you mention in the report, task 3 is almost complete and just missing diagonal functionality.

In several places, the code could be a little cleaner (especially the massive chain of nested loops in task 3) and would benefit from a few comments for clarity. Writing more efficient code will, however, come with practice - as a first year assignment, you've completed the tasks well. Good job, Dan!

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39
